Security

Contents

Security

Introduction

Security requirements are different for each application. This part of the information center covers the factors to consider when determining the scope of your security requirements, enabling you to make an informed choice from the options available.

1. Introduction

Security requirements are different for each application. This part of the information center covers the factors to consider when determining the scope of your security requirements, enabling you to make an informed choice from the options available.

You can use WebSphere® MQ for a wide variety of applications on a range of platforms. The security requirements are likely to be different for each application. For some, security will be a critical consideration.

WebSphere MQ provides a range of link-level security services, including support for the Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

Parent topic: Security

1.1. Security concepts

This collection of topics describes the five security services that are identified in the IBM® Security Architecture.

Security services are the services within a computer system that protect its resources. The five security services that are identified in the IBM Security Architecture are as follows:

- Identification and authentication
- Access control
- Confidentiality
- Data integrity
- Non-repudiation

Security mechanisms are technical tools and techniques that are used to implement security services. A mechanism might operate by itself, or with others, to provide a particular service. Examples of common security mechanisms are as follows:

- Access control lists
- Cryptography
- Digital signatures

When you are planning a WebSphere® MQ implementation, consider which security services and mechanisms you require. For information about what to consider after you have read these topics, see Planning for your security requirements.

For more information about the IBM Security Architecture, see *IBM Security Architecture: Securing the Open Client/Server Distributed Enterprise*, SC28-8135, which is available from the IBM Publications Center at: http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss

Identification and authentication
Identification is the ability to identify uniquely a user of a system or an application that is running in the system. *Authentication* is the ability to prove that a user or application is genuinely who that person or what that application claims to be.

Access control
The *access control* service protects critical resources in a system by limiting access only to authorized users and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner.

Confidentiality
The *confidentiality* service protects sensitive information from unauthorized disclosure.

Data integrity
The *data integrity* service detects whether there has been unauthorized modification of data.

Non-repudiation
The *non-repudiation* service can be viewed as an extension to the identification and authentication service. In general, non-repudiation applies when data is transmitted electronically; for example, an order to a stock broker to buy or sell stock, or an order to a bank to transfer funds from one account to another.

Parent topic: Introduction

Related concepts
Planning for your security requirements
Cryptographic concepts
Cryptographic security protocols: TLS and SSL

1.1.1. Identification and authentication

Identification is the ability to identify uniquely a user of a system or an application that is running in the system. *Authentication* is the ability to prove that a user or application is genuinely who that person or what that application claims to be.

For example, consider a user who logs on to a system by entering a user ID and password. The system uses the user ID to identify the user. The system authenticates the user at the time of logon by checking that the supplied password is correct.

Here are some examples of the identification and authentication service in a WebSphere® MQ environment:

- Every message can contain *message context* information. This information is held in the message descriptor. It can be generated by the queue manager when a message is put on a queue by an application. Alternatively, the application can supply the information if the user ID associated with the application is authorized to do so.
  The context information in a message allows the receiving application to find out about the originator of the message. It contains, for example, the name of the application that put the message and the user ID associated with the application.
- When a message channel starts, it is possible for the message channel agent (MCA) at each end of the channel to authenticate its partner. This technique is known as *mutual authentication*. For the sending MCA, it provides assurance that the partner it is about to send messages to is genuine. For the receiving MCA, there is a similar assurance that it is about to receive messages from a genuine partner.

Parent topic: Security concepts

1.1.2. Access control

The *access control* service protects critical resources in a system by limiting access only to authorized users and their applications. It prevents the unauthorized use of a resource or the use of a resource in an unauthorized manner.

Here are some examples of the access control service in a WebSphere® MQ environment:

- Allowing only an authorized administrator to issue commands to manage WebSphere MQ resources.
- Allowing an application to connect to a queue manager only if the user ID associated with the application is authorized to do so.
- Allowing an application to open only those queues that are necessary for its function.
- Allowing an application to subscribe only to those topics that are necessary for its function.
- Allowing an application to perform only those operations on a queue that are necessary for its function. For example, an application might need only to browse messages on a particular queue, and not to put or get messages.

Parent topic: Security concepts

1.1.3. Confidentiality

The *confidentiality* service protects sensitive information from unauthorized disclosure.

When sensitive data is stored locally, access control mechanisms might be sufficient to protect it on the assumption that the data cannot be read if it cannot be accessed. If a greater level of security is required, the data can be encrypted.

Encrypt Sensitive data when it is transmitted over a communications network, especially over an insecure network such as the Internet. In a networking environment, access control mechanisms are not effective against attempts to intercept the data, such as wiretapping.

Here are some examples of the confidentiality service that can be implemented in a WebSphere® MQ environment:

- After a sending MCA gets a message from a transmission queue, the message is encrypted before it is sent over the network to the receiving MCA. At the other end of the channel, the message is decrypted before the receiving MCA puts it on its destination queue.
- While messages are stored on a local queue, the access control mechanisms provided by WebSphere MQ might be considered sufficient to protect their contents against unauthorized disclosure. However, for a greater level of security, their contents can be encrypted as well.

Parent topic: Security concepts

1.1.4. Data integrity

The *data integrity* service detects whether there has been unauthorized modification of data.

There are two ways in which data might be altered: accidentally, through hardware and transmission errors, or because of a deliberate attack. Many hardware products and transmission protocols have mechanisms to detect and correct hardware and transmission errors. The purpose of the data integrity service is to detect a deliberate attack.

The data integrity service aims only to detect whether data has been modified. It does not aim to restore data to its original state if it has been modified.

Access control mechanisms can contribute to data integrity insofar as data cannot be modified if access is denied. But, as with confidentiality, access control mechanisms are not effective in a networking environment.

Here are some examples of the data integrity service that can be implemented in a WebSphere® MQ environment:

- A data integrity service can be used to detect whether the contents of a message have been deliberately modified while it was being transmitted over a network.
- While messages are stored on a local queue, the access control mechanisms provided by WebSphere MQ might be considered sufficient to prevent deliberate modification of the contents of the messages. However, for a greater level of security, a data integrity service can be used to detect whether the contents of a message have been deliberately modified between the time the message was put on the queue and the time it was retrieved from the queue.

Parent topic: Security concepts

1.1.5. Non-repudiation

The *non-repudiation* service can be viewed as an extension to the identification and authentication service. In general, non-repudiation applies when data is transmitted electronically; for example, an order to a stock broker to buy or sell stock, or an order to a bank to transfer funds from one account to another.

The overall goal of the non-repudiation service is to be able to prove that a particular message is associated with a particular individual.

The non-repudiation service can contain more than one component, where each component provides a different function. If the sender of a message ever denies sending it, the non-repudiation service with *proof of origin* can provide the receiver with undeniable evidence that the message was sent by that particular individual. If the receiver of a message ever denies receiving it, the non-repudiation service with *proof of delivery* can provide the sender with undeniable evidence that the message was received by that particular individual.

In practice, proof with virtually 100% certainty, or undeniable evidence, is a difficult goal. In the real world, nothing is fully secure. Managing security is more concerned with managing risk to a level that is acceptable to the business. In such an environment, a more realistic expectation of the non-repudiation service is to be able to provide evidence that is admissible, and supports your case, in a court of law.

Non-repudiation is a relevant security service in a WebSphere® MQ environment because WebSphere MQ is a means of transmitting data electronically. For example, you might require contemporaneous evidence that a particular message was sent or received by an application associated with a particular individual.

IBM® WebSphere MQ and IBM Tivoli® Access Manager for Business Integration do not provide a non-repudiation service as part of their base function. However, this information center does contain suggestions on how you might provide your own non-repudiation service within a WebSphere MQ environment by writing your own exit programs.

Parent topic: Security concepts

1.2. Planning for your security requirements

This collection of topics explains what you need to consider when planning security in a WebSphere® MQ environment.

The basic considerations are those aspects of security you must consider when implementing WebSphere MQ. On i5/OS®, UNIX systems, and Windows systems, if you ignore these considerations and do nothing, you cannot implement WebSphere MQ. On z/OS®, the effect is that your WebSphere MQ resources are unprotected. That is, all users can access and change all WebSphere MQ resources.

Authority to administer WebSphere MQ

WebSphere MQ administrators need authority to:

- Issue commands to administer WebSphere MQ
- Use the WebSphere MQ Explorer
- Use the operations and control panels on z/OS
- Use the WebSphere MQ utility program, CSQUTIL, on z/OS
- Access the queue manager data sets on z/OS

This is an aspect of access control. For more information, see Authority to administer WebSphere MQ.

Authority to work with WebSphere MQ objects

Applications can access the following WebSphere MQ objects by issuing MQI calls:

- Queue managers
- Queues
- Processes
- Namelists
- Topics

Applications can also use Programmable Command Format (PCF) commands to access these WebSphere MQ objects, and to access channels and authentication information objects as well. These objects are protected by WebSphere MQ and the user IDs associated with the applications need authority to access them.

This is another aspect of access control. For more information, see Authority to work with WebSphere MQ objects.

Channel security

The user IDs associated with message channel agents (MCAs) need authority to access various WebSphere MQ resources. For example, an MCA must be able to connect to a queue manager. If it is a sending MCA, it must be able to open the transmission queue for the channel. If it is a receiving MCA, it must be able to open destination queues. The user IDs associated with applications need authority to use PCF commands to administer channels, channel initiators, and listeners.

This is another aspect of access control. For more information, see Channel security.

Additional considerations

You need to consider the following aspects of security only if you are using certain WebSphere MQ function or base product extensions:

- Security for queue manager clusters
- Security for WebSphere MQ Publish/Subscribe
- Security for WebSphere MQ internet pass-thru

Security for queue manager clusters
Though queue manager clusters can be convenient to use, you must pay special attention to their security.

Security for WebSphere MQ Publish/Subscribe
There are additional security considerations if you are using WebSphere MQ Publish/Subscribe.

Security for WebSphere MQ internet pass-thru
Internet pass-thru can simplify communication through a firewall, but this has security implications.

Link level security and application level security
The remaining security considerations are discussed under two headings: link level security and application level security.

Parent topic: Introduction

Related concepts
Security concepts
Cryptographic concepts
Cryptographic security protocols: TLS and SSL

1.2.1. Security for queue manager clusters

Though queue manager clusters can be convenient to use, you must pay special attention to their security.

A *queue manager cluster* is a network of queue managers that are logically associated in some way. A queue manager that is a member of a cluster is called a *cluster queue manager*.

A queue that belongs to a cluster queue manager can be made known to other queue managers in the cluster. Such a queue is called a *cluster queue*. Any queue manager in a cluster can send messages to cluster queues without needing any of the following:

- An explicit remote queue definition for each cluster queue
- Explicitly defined channels to and from each remote queue manager
- A separate transmission queue for each outbound channel

You can create a cluster in which two or more queue managers are clones. This means that they have instances of the same local queues, including any local queues declared as cluster queues, and can support instances of the same server applications.

When an application connected to a cluster queue manager sends a message to a cluster queue that has an instance on each of the cloned queue managers, WebSphere® MQ decides which queue manager to send it to. When many applications send messages to the cluster queue, WebSphere MQ balances the workload across each of the queue managers that have an instance of the queue. If one of the systems hosting a cloned queue manager fails, WebSphere MQ continues to balance the workload across the remaining queue managers until the system that failed is restarted.

If you are using queue manager clusters, you need to consider the following security issues:

- Allowing only selected queue managers to send messages to your queue manager
- Allowing only selected users of a remote queue manager to send messages to a queue on your queue manager
- Allowing applications connected to your queue manager to send messages only to selected remote queues

These considerations are relevant even if you are not using clusters, but they become more important if you are using clusters.

If an application can send messages to one cluster queue, it can send messages to any other cluster queue without needing additional remote queue definitions, transmission queues, or channels. It therefore becomes more important to consider whether you need to restrict access to the cluster queues on your queue manager, and to restrict the cluster queues to which your applications can send messages.

There are some additional security considerations, which are relevant only if you are using queue manager clusters:

- Allowing only selected queue managers to join a cluster
- Forcing unwanted queue managers to leave a cluster

For more information about all these considerations, see Keeping clusters secure. For considerations specific to WebSphere MQ for z/OS®, see Security in queue manager clusters on z/OS.

Parent topic: Planning for your security requirements

1.2.2. Security for WebSphere MQ Publish/Subscribe

There are additional security considerations if you are using WebSphere® MQ Publish/Subscribe.

In a publish/subscribe system, there are two types of application: publisher and subscriber. *Publishers* supply information in the form of WebSphere MQ messages. When a publisher publishes a message, it specifies a *topic*, which identifies the subject of the information inside the message.

Subscribers are the consumers of the information that is published. A subscriber specifies the topics it is interested in by subscribing to them.

The *queue manager* is an application supplied with WebSphere MQ Publish/Subscribe. It receives published messages from publishers and subscription requests from subscribers, and routes the published messages to the subscribers. A subscriber is sent messages only on those topics to which it has subscribed.

For more information, see Publish/subscribe security.

Parent topic: Planning for your security requirements

1.2.3. Security for WebSphere MQ internet pass-thru

Internet pass-thru can simplify communication through a firewall, but this has security implications.

WebSphere® MQ internet pass-thru is a WebSphere MQ base product extension that is supplied in SupportPac MS81.

WebSphere MQ internet pass-thru enables two queue managers to exchange messages, or a WebSphere MQ client application to connect to a queue manager, over the Internet without requiring a direct TCP/IP connection. This is useful if a firewall prohibits a direct TCP/IP connection between two systems. It makes the passage of WebSphere MQ channel protocol flows into and out of a firewall simpler and more manageable by tunnelling the flows inside HTTP or by acting as a proxy. Using the Secure Sockets Layer (SSL), it can also be used to encrypt and decrypt messages that are sent over the Internet.

For more information about WebSphere MQ internet pass-thru, see *MS81: WebSphere MQ internet pass-thru*, available from the following address: http://www.ibm.com/software/integration/support/supportpacs/

Parent topic: Planning for your security requirements

1.2.4. Link level security and application level security

The remaining security considerations are discussed under two headings: link level security and application level security.

This collection of topics describes link level security and application level security, and compares these two levels of security.

Link level and application level security services are available for you to install, configure, and use. Some services are supplied with WebSphere® MQ and WebSphere MQ base product extensions. The remainder are provided by other IBM® products, vendor products, and the SNA LU 6.2 communications subsystem.

For more information about what is available for link level security, see:

- WebSphere MQ support for SSL and TLS
- Channel exit programs
- The SSPI channel exit program
- SNA LU 6.2 security services

For application level security, see:

- WebSphere MQ Advanced Message Security

You can also provide your own link level and application level security services by writing exit programs. This might involve significant effort in terms of developing and maintaining the exit programs. For more information, see:

- Providing your own link level security
- Providing your own application level security

Link level security
Link level security refers to those security services that are invoked, directly or indirectly, by an MCA, the communications subsystem, or a combination of the two working together.

Application level security
Application level security refers to those security services that are invoked at the interface between an application and a queue manager to which it is connected.

Comparing link level security and application level security
This topic discusses various aspects of link level security and application level security, and compares the two levels of security.

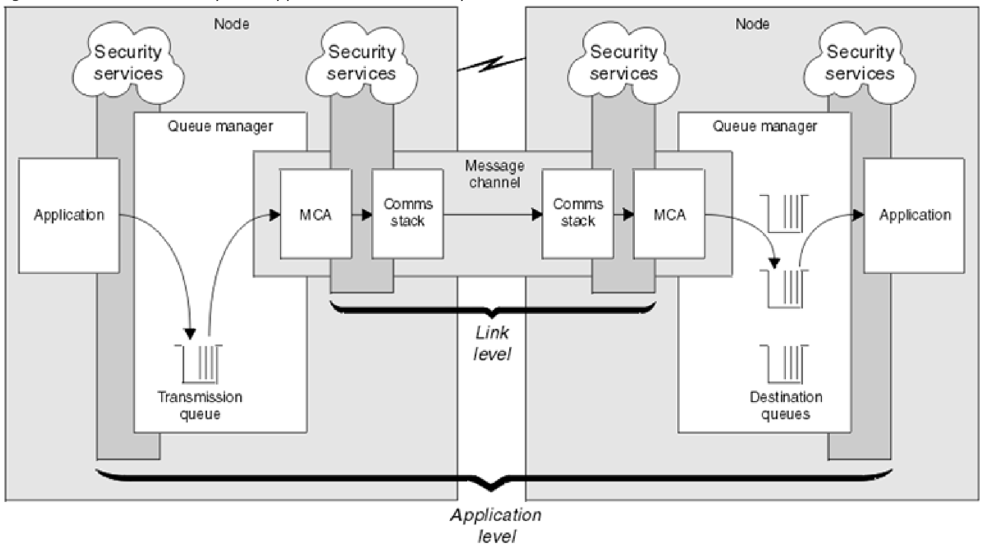
Parent topic: Planning for your security requirements

1.2.4.1. Link level security

Link level security refers to those security services that are invoked, directly or indirectly, by an MCA, the communications subsystem, or a combination of the two working together.

Link level and application level security are illustrated in Figure 1.

Figure 1. Link level security and application level security



Here are some examples of link level security services:

- The MCA at each end of a message channel can authenticate its partner. This is done when the channel starts and a communications connection has been established, but before any messages start to flow. If authentication fails at either end, the channel is closed and no messages are transferred. This is an example of an identification and authentication service.
- A message can be encrypted at the sending end of a channel and decrypted at the receiving end. This is an example of a confidentiality service.
- A message can be checked at the receiving end of a channel to determine whether its contents have been deliberately modified while it was being transmitted over the network. This is an example of a data integrity service.

Parent topic: Link level security and application level security

1.2.4.2. Application level security

Application level security refers to those security services that are invoked at the interface between an application and a queue manager to which it is connected.

These services are invoked when the application issues MQI calls to the queue manager. The services might be invoked, directly or indirectly, by the application, the queue manager, another product that supports WebSphere® MQ, or a combination of any of these working together. Application level security is illustrated in Figure 1.

Application level security is also known as *end-to-end security* or *message level security*.

Here are some examples of application level security services:

- When an application puts a message on a queue, the message descriptor contains a user ID associated with the application. However, there is no data present, such as an encrypted password, that can be used to authenticate the user ID. A security service can add this data. When the message is eventually retrieved by the receiving application, another component of the service can authenticate the user ID using the data that has travelled with the message. This is an example of an identification and authentication service.
- A message can be encrypted when it is put on a queue by an application and decrypted when it is retrieved by the receiving application. This is an example of a confidentiality service.
- A message can be checked when it is retrieved by the receiving application. This check determines whether its contents have been deliberately modified since it was first put on a queue by the sending application. This is an example of a data integrity service.

Parent topic: Link level security and application level security

1.2.4.3. Comparing link level security and application level security

This topic discusses various aspects of link level security and application level security, and compares the two levels of security.

Protecting messages in queues

Link level security can protect messages while they are transferred from one queue manager to another. It is particularly important when messages are transmitted over an insecure network. It cannot, however, protect messages while they are stored in queues at either a source queue manager, a destination queue manager, or an intermediate queue manager.

Application level security, by comparison, can protect messages while they are stored in queues and applies even when distributed queuing is not used. This is the major difference between link level security and application level security and is illustrated in Figure 1.

Queue managers not running in controlled and trusted environments

If a queue manager is running in a controlled and trusted environment, the access control mechanisms provided by WebSphere® MQ might be considered sufficient to protect the messages stored on its queues. This is particularly true if only local queuing is involved and messages never leave the queue manager. Application level security in this case might be considered unnecessary.

Application level security might also be considered unnecessary if messages are transferred to another queue manager that is also running in a controlled and trusted environment, or are received from such a queue manager. The need for application level security becomes greater when messages are transferred to, or received from, a queue manager that is not running in a controlled and trusted environment.

Differences in cost

Application level security might cost more than link level security in terms of administration and performance.

The cost of administration is likely to be greater because there are potentially more constraints to configure and maintain. For example, you might need to ensure that a particular user sends only certain types of message and sends messages only to certain destinations. Conversely, you might need to ensure that a particular user receives only certain types of message and receives messages only from certain sources. Instead of managing the link level security services on a single message channel, you might need to be configuring and maintaining rules for every pair of users who exchange messages across that channel.

There might be an effect on performance if security services are invoked every time an application puts or gets a message.

Organizations tend to consider link level security first because it might be easier to implement. They consider application level security if they discover that link level security does not satisfy all their requirements.

Availability of components

Generally, in a distributed environment, a security service requires a component on at least two systems. For example, a message might be encrypted on one system and decrypted on another. This applies to both link level security and application level security.

In a heterogeneous environment, with different platforms in use, each with different levels of security function, the required components of a security service might not be available for every platform on which they are needed and in a form that is easy to use. This is probably more of an issue for application level security than for link level security, particularly if you intend to provide your own application level security by buying in components from various sources.

Messages in a dead letter queue

If a message is protected by application level security, there might be a problem if, for any reason, the message does not reach its destination and is put on a dead letter queue. If you cannot work out how to process the message from the information in the message descriptor and the dead letter header, you might need to inspect the contents of the application data. You cannot do this if the application data is encrypted and only the intended recipient can decrypt it.

What application level security cannot do

Application level security is not a complete solution. Even if you implement application level security, you might still require some link level security services. For example:

- When a channel starts, the mutual authentication of the two MCAs might still be a requirement. This can be done only by a link level security service.
- Application level security cannot protect the transmission queue header, MQXQH, which includes the embedded message descriptor. Nor can it protect the data in WebSphere MQ channel protocol flows other than message data. Only link level security can provide this protection.
- If application level security services are invoked at the server end of an MQI channel, the services cannot protect the parameters of MQI calls that are sent over the channel. In particular, the application data in an MQPUT, MQPUT1, or MQGET call is unprotected. Only link level security can provide the protection in this case.

Parent topic: Link level security and application level security

This build: January 26, 2011 11:20:42

Notices | Trademarks | Downloads | Library | Support | Feedback

1.3. Cryptographic concepts

This collection of topics describes the concepts of cryptography applicable to WebSphere MQ.

The term *entity* is used to refer to a queue manager, a WebSphere® MQ client, an individual user, or any other system capable of exchanging messages.

Cryptography
Cryptography is the process of converting between readable text, called *plaintext*, and an unreadable form, called *ciphertext*.

Message digests
Message digests are fixed size numeric representations of the contents of messages. A message digest is computed by a hash function.

Digital signatures
A digital signature is formed by encrypting a representation of a message. The encryption uses the private key of the signatory and, for efficiency, usually operates on a message digest rather than the message itself.

Digital certificates
Digital certificates protect against impersonation, certifying that a public key belongs to a specified entity. They are issued by a Certification Authority.

Public Key Infrastructure (PKI)
A Public Key Infrastructure (PKI) is a system of facilities, policies, and services that supports the use of public key cryptography for authenticating the parties involved in a transaction.

Parent topic: Introduction

Related concepts
Security concepts
Planning for your security requirements
Cryptographic security protocols: TLS and SSL

This build: January 26, 2011 11:20:43

Notices | Trademarks | Downloads | Library | Support | Feedback

1.3.1. Cryptography

Cryptography is the process of converting between readable text, called *plaintext*, and an unreadable form, called *ciphertext*.

This occurs as follows:

1. The sender converts the plaintext message to ciphertext. This part of the process is called *encryption* (sometimes *encipherment*).
2. The ciphertext is transmitted to the receiver.
3. The receiver converts the ciphertext message back to its plaintext form. This part of the process is called *decryption* (sometimes *decipherment*).

The conversion involves a sequence of mathematical operations that change the appearance of the message during transmission but do not affect the content. Cryptographic techniques can ensure confidentiality and protect messages against unauthorized viewing (eavesdropping), because an encrypted message is not understandable. Digital signatures, which provide an assurance of message integrity, use encryption techniques. See Digital signatures for more information.

Cryptographic techniques involve a general algorithm, made specific by the use of keys. There are two classes of algorithm:

- Those that require both parties to use the same secret key. Algorithms that use a shared key are known as *symmetric* algorithms. Figure 1 illustrates symmetric key cryptography.
- Those that use one key for encryption and a different key for decryption. One of these must be kept secret but the other can be public. Algorithms that use public and private key pairs are known as *asymmetric* algorithms. Figure 2 illustrates asymmetric key cryptography, which is also known as *public key cryptography*.

The encryption and decryption algorithms used can be public but the shared secret key and the private key must be kept secret.

Figure 1. Symmetric key cryptography

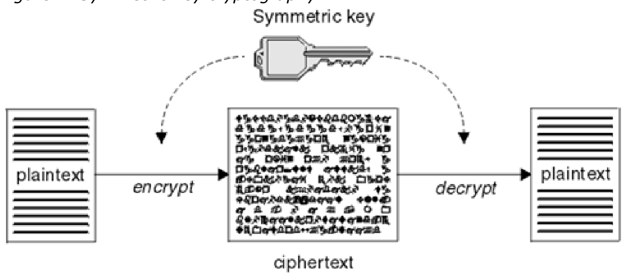


Figure 2. Asymmetric key cryptography

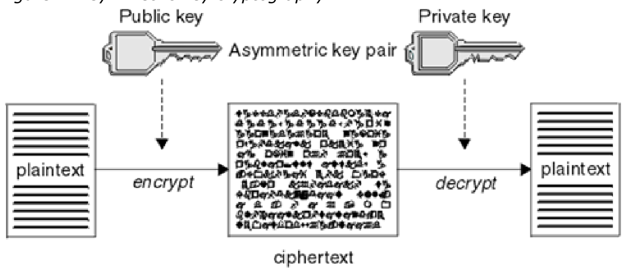


Figure 2 shows plaintext encrypted with the receiver's public key and decrypted with the receiver's private key. Only the intended receiver holds the private key for decrypting the ciphertext. Note that the sender can also encrypt messages with a private key, which allows anyone that holds the sender's public key to decrypt the message, with the assurance that the message must have come from the sender.

With asymmetric algorithms, messages are encrypted with either the public or the private key but can be decrypted only with the other key. Only the private key is secret, the public key can be known by anyone. With symmetric algorithms, the shared key must be known only to the two parties. This is called the *key distribution problem*. Asymmetric algorithms are slower but have the advantage that there is no key distribution problem.

Other terminology associated with cryptography is:

Strength

The strength of encryption is determined by the key size. Asymmetric algorithms require large keys, for example:

768 bits	Low-strength asymmetric key
1024 bits	Medium-strength asymmetric key
2048 bits	High-strength asymmetric key

Symmetric keys are smaller: 256 bit keys give you strong encryption.

Block cipher algorithm

These algorithms encrypt data by blocks. For example, the RC2 algorithm from RCA Data Security Inc. uses blocks 8 bytes long. Block algorithms are typically slower than stream algorithms.

Stream cipher algorithm

These algorithms operate on each byte of data. Stream algorithms are typically faster than block algorithms.

Parent topic: Cryptographic concepts

1.3.2. Message digests

Message digests are fixed size numeric representations of the contents of messages. A message digest is computed by a hash function.

Messages are inherently variable in size. A message digest is a fixed size numeric representations of the contents of a message. A message digest is computed by a hash function, which is a transformation that meets two criteria:

- The hash function must be one-way. It must not be possible to reverse the function to find the message corresponding to a particular message digest, other than by testing all possible messages.
- It must be computationally infeasible to find two messages that hash to the same digest.

A message digest is also known as a Message Authentication Code (MAC), because it can provide assurance that the message has not been modified. The message digest is sent with the message itself. The receiver can generate a digest for the message and compare it with the sender's digest. If the two digests are the same, this verifies the integrity of the message. Any tampering with the message during transmission almost certainly results in a different message digest.

Parent topic: Cryptographic concepts

1.3.3. Digital signatures

A digital signature is formed by encrypting a representation of a message. The encryption uses the private key of the signatory and, for efficiency, usually operates on a message digest rather than the message itself.

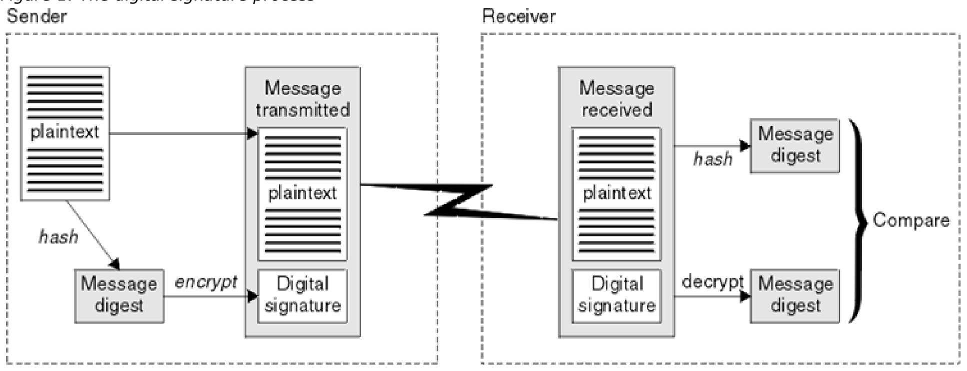
Digital signatures vary with the data being signed, unlike handwritten signatures, which do not depend on the content of the document being signed. If two different messages are signed digitally by the same entity, the two signatures differ, but both signatures can be verified with the same public key, that is, the public key of the entity that signed the messages.

The steps of the digital signature process are as follows:

1. The sender computes a message digest and then encrypts the digest using the sender's private key, forming the digital signature.
2. The sender transmits the digital signature with the message.
3. The receiver decrypts the digital signature using the sender's public key, regenerating the sender's message digest.
4. The receiver computes a message digest from the message data received and verifies that the two digests are the same.

Figure 1 illustrates this process.

Figure 1. The digital signature process



If the digital signature is verified, the receiver knows that:

- The message has not been modified during transmission.
- The message was sent by the entity that claims to have sent it.

Digital signatures are part of integrity and authentication services. Digital signatures also provide proof of origin. Only the sender knows the private key, which provides strong evidence that the sender is the originator of the message.

Note: You can also encrypt the message itself, which protects the confidentiality of the information in the message.

Parent topic: Cryptographic concepts

Related concepts
Message digests

1.3.4. Digital certificates

Digital certificates protect against impersonation, certifying that a public key belongs to a specified entity. They are issued by a Certification Authority.

Digital certificates provide protection against impersonation, because a digital certificate binds a public key to its owner, whether that owner is an individual, a queue manager, or some other entity. Digital certificates are also known as public key certificates, because they give you assurances about the ownership of a public key when you use an asymmetric key scheme. A digital certificate contains the public key for an entity and is a statement that the public key belongs to that entity:

- When the certificate is for an individual entity, the certificate is called a *personal certificate* or *user certificate*.
- When the certificate is for a Certification Authority, the certificate is called a *CA certificate* or *signer certificate*.

If public keys are sent directly by their owner to another entity, there is a risk that the message could be intercepted and the public key substituted by another. This is known as a *man in the middle attack*. The solution to this problem is to exchange public keys through a trusted third party, giving you a strong assurance that the public key really belongs to the entity with which you are communicating. Instead of sending your public key directly, you ask the trusted third party to incorporate it into a digital certificate. The trusted third party that issues digital certificates is called a Certification Authority (CA), as described in Certification Authorities.

What is in a digital certificate
Digital certificates contain specific pieces of information, as determined by the X.509 standard.

Requirements for personal certificates
WebSphere® MQ supports digital certificates that comply with the X.509 standard. It requires the client authentication option.

Certification Authorities
A Certification Authority (CA) is an independent and trusted third party that issues digital certificates to provide you with an assurance that the public key of an entity truly belongs to that entity.

Distinguished Names
The Distinguished Name (DN) uniquely identifies an entity in an X.509 certificate.

Obtaining personal certificates
You can obtain a certificate from a trusted external CA or create a self-signed certificate. Use a CA certificate if your system communicates with other organizations.

How certificate chains work
When you receive the certificate for another entity, you might need to use a *certificate chain* to obtain the *root CA* certificate.

When certificates are no longer valid

Digital certificates can expire or be revoked. Applications can check for revoked certificates using OCSP, or CRLs on LDAP servers.

Parent topic: Cryptographic concepts

1.3.4.1. What is in a digital certificate

Digital certificates contain specific pieces of information, as determined by the X.509 standard.

Digital certificates used by WebSphere® MQ comply with the X.509 standard, which specifies the information that is required and the format for sending it. X.509 is the Authentication framework part of the X.500 series of standards. X.500 is the OSI Directory Standard.

Digital certificates contain at least the following information about the entity being certified:

- The owner's public key
- The owner's Distinguished Name
- The Distinguished Name of the CA that is issuing the certificate
- The date from which the certificate is valid
- The expiry date of the certificate
- A version number
- A serial number

An X.509 V2 certificate also contains an Issuer Identifier and a Subject Identifier, and an X.509 V3 certificate can contain a number of extensions. Some certificate extensions, such as the Basic Constraint extension, are *standard*, but others are implementation-specific. An extension can be *critical*, in which case a system must be able to recognize the field; if it does not recognize the field, it must reject the certificate. If an extension is not critical, the system can ignore it if does not recognize it.

When you receive a certificate from a CA, the certificate is signed by the issuing CA with a digital signature. You verify that signature by using a CA certificate, from which you obtain the public key for the CA. You can use the CA public key to validate other certificates issued by that authority. Recipients of your certificate use the CA public key to check the signature.

Digital certificates do not contain your private key. You must keep your private key secret.

Parent topic: Digital certificates

1.3.4.2. Requirements for personal certificates

WebSphere® MQ supports digital certificates that comply with the X.509 standard. It requires the client authentication option.

Because MQ is a peer to peer system, in SSL terminology this is viewed as client authentication, which means that any personal certificate used for SSL authentication needs to allow a key usage of client authentication. Not all server certificates have this option enabled, so the certificate provider might need to enable client authentication on the root CA for the secure certificate.

Parent topic: Digital certificates

1.3.4.3. Certification Authorities

A Certification Authority (CA) is an independent and trusted third party that issues digital certificates to provide you with an assurance that the public key of an entity truly belongs to that entity.

The roles of a CA are:

- On receiving a request for a digital certificate, to verify the identity of the requestor before building, signing and returning the personal certificate
- To provide the CA's own public key in its CA certificate
- To publish lists of certificates that are no longer trusted in a Certificate Revocation List (CRL). For more information, see Working with revoked certificates

Parent topic: Digital certificates

1.3.4.4. Distinguished Names

The Distinguished Name (DN) uniquely identifies an entity in an X.509 certificate.

The following attribute types are commonly found in the DN:

CN	Common Name
T	Title
O	Organization name
OU	Organizational Unit name
L	Locality name
ST (or SP or S)	State or Province name
C	Country

The X.509 standard defines other attributes that do not usually form part of the DN but can provide optional extensions to the digital certificate.

The X.509 standard provides for a DN to be specified in a string format. For example:

```
CN=John, O=IBM, OU=Test, C=GB
```

Any field within the DN that consists of more than one word requires quotes, either around the field contents or the entire DN. For example:

```
CN="John Smith", O=IBM, OU=Test, C=GB
```

or

```
"CN=John Smith, O=IBM, OU=Test, C=GB".
```

The Common Name (CN) can describe an individual user or any other entity, for example a Web server.

The DN can contain multiple OU attributes, but one instance only of each of the other attributes is permitted. The order of the OU entries is significant: the order specifies a hierarchy of Organizational Unit names, with the highest-level unit first.

Parent topic: Digital certificates

1.3.4.5. Obtaining personal certificates

You can obtain a certificate from a trusted external CA or create a self-signed certificate. Use a CA certificate if your system communicates with other organizations.

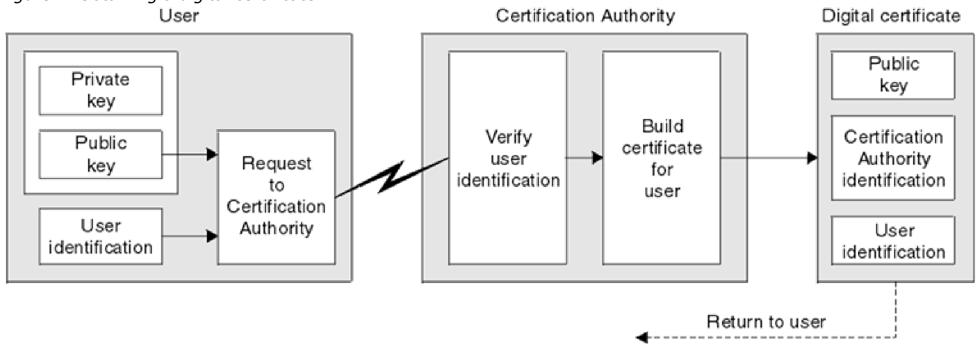
You obtain a digital certificate by sending information to a CA. The X.509 standard defines a format for this information, but some CAs have their own format. Certificate requests are typically generated by the certificate management tool your system uses, for example the iKeyman tool on UNIX systems and RACF® on z/OS®. The information comprises your Distinguished Name and is accompanied by your public key. When your certificate management tool generates your certificate request, it also generates your private key, which you must keep secure. Never distribute your private key.

When the CA receives your request, the authority verifies your identity before building the certificate and returning it to you as a personal certificate.

When you obtain a certificate from a trusted external CA, you pay for the service. When you are testing your system, or you need only to protect internal messages, you can create self-signed certificates. These are created and signed by the certificate management tool your system uses. Self-signed certificates cannot be used to authenticate certificates from outside your organization.

Figure 1 illustrates the process of obtaining a digital certificate from a CA.

Figure 1. Obtaining a digital certificate



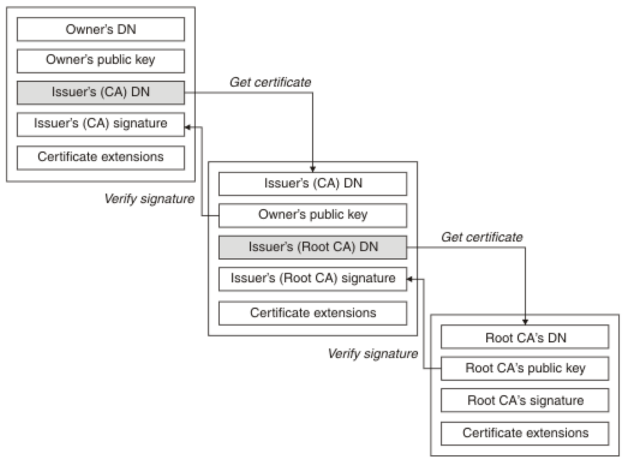
Parent topic: Digital certificates

1.3.4.6. How certificate chains work

When you receive the certificate for another entity, you might need to use a *certificate chain* to obtain the *root CA* certificate.

The certificate chain, also known as the *certification path*, is a list of certificates used to authenticate an entity. The chain, or path, begins with the certificate of that entity, and each certificate in the chain is signed by the entity identified by the next certificate in the chain. The chain terminates with a root CA certificate. The root CA certificate is always signed by the CA itself. The signatures of all certificates in the chain must be verified until the root CA certificate is reached. Figure 1 illustrates a certification path from the certificate owner to the root CA, where the chain of trust begins. Each certificate can contain one or more extensions. A certificate belonging to a CA typically contains a BasicConstraints extension with the isCA flag set to indicate that it is allowed to sign other certificates.

Figure 1. Chain of trust



Parent topic: Digital certificates

1.3.4.7. When certificates are no longer valid

Digital certificates can expire or be revoked. Applications can check for revoked certificates using OCSP, or CRLs on LDAP servers.

Digital certificates are issued for a fixed period and are not valid after their expiry date. Certificates can also become untrustworthy for various reasons, including:

- The owner has moved to a different organization.
- The private key is no longer secret.

Applications can check whether a certificate is revoked by sending a request to an Online Certificate Status Protocol (OCSP) responder (on UNIX systems and Windows only). Alternatively, they can access a CRL on an LDAP server. The OCSP revocation and CRL information is published by a Certification Authority. For more information, see Working with revoked certificates.

Parent topic: Digital certificates

1.3.5. Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) is a system of facilities, policies, and services that supports the use of public key cryptography for authenticating the parties involved in a transaction.

There is no single standard that defines the components of a Public Key Infrastructure, but a PKI typically comprises Certification Authorities and other Registration Authorities (RAs) that provide the following services:

- Issuing digital certificates
- Validating digital certificates
- Revoking digital certificates
- Distributing public keys

The X.509 standard is a Public Key Infrastructure.

Refer to Digital certificates for more information about digital certificates and Certification Authorities (CAs). RAs verify the information provided when digital certificates are requested. If the RA verifies that information, the CA can issue a digital certificate to the requester.

A PKI might also provide tools for managing digital certificates and public keys. A PKI is sometimes described as a *trust hierarchy* for managing digital certificates, but most definitions include additional services. Some definitions include encryption and digital signature services, but these services are not essential to the operation of a PKI.

Parent topic: Cryptographic concepts

1.4. Cryptographic security protocols: TLS and SSL

Cryptographic protocols provide secure connections, enabling two parties to communicate with privacy and data integrity. The Transport Layer Security (TLS) protocol evolved from that of the Secure Sockets Layer (SSL).

Applications use TLS or SSL to establish secure connections between two communicating parties. The primary goal of both protocols is to provide privacy and data integrity. Other goals are as follows:

- Enabling interoperability between applications
- Providing an extensible framework that can readily incorporate new public key and bulk encryption methods
- Ensuring relative computational efficiency

Both TLS and SSL comprise two layers: a Record Protocol and a Handshake Protocol.

Although the two protocols are similar, the differences are sufficiently significant that SSL 3.0 and the various versions of TLS do not interoperate.

Transport Layer Security (TLS) concepts
The Transport Layer Security (TLS) protocol enables two parties to communicate with privacy and data integrity. The TLS protocol evolved from the SSL 3.0 protocol but TLS and SSL do not interoperate.

Secure Sockets Layer (SSL) concepts
Secure Sockets Layer (SSL) protocol enables two parties to communicate with privacy and data integrity.

An overview of the SSL handshake
The SSL handshake enables the SSL client and SSL server to establish the secret keys with which they communicate.

How SSL provides authentication, confidentiality, and integrity
During both client and server authentication there is a step that requires data to be encrypted with one of the keys in an asymmetric key pair and decrypted with the other key of the pair. A message digest is used to provide integrity.

CipherSuites and CipherSpecs
Cryptographic security protocols must agree the algorithms used by a secure connection. CipherSuites and CipherSpecs define specific combinations of algorithms.

Security protocols in WebSphere MQ
WebSphere MQ supports both the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) protocols to provide link level security for message channels and MQI channels.

Parent topic: Introduction

Related concepts
Security concepts
Planning for your security requirements
Cryptographic concepts

1.4.1. Transport Layer Security (TLS) concepts

The Transport Layer Security (TLS) protocol enables two parties to communicate with privacy and data integrity. The TLS protocol evolved from the SSL 3.0 protocol but TLS and SSL do not interoperate.

The TLS protocol provides communications security over the internet, and allows client/server applications to communicate in a way that is private and reliable. The protocol has two layers: the TLS Record Protocol and the TLS Handshake Protocol, and these are layered above a transport protocol such as TCP/IP.

The TLS protocol evolved from the Netscape SSL 3.0 protocol. Although similar, TLS and SSL are not interoperable.

The TLS protocol applies when any of the following CipherSpecs are specified:

- TLS_RSA_WITH_AES_128_CBC_SHA
- ›TLS_RSA_WITH_AES_128_CBC_SHA256‹
- TLS_RSA_WITH_AES_256_CBC_SHA
- ›TLS_RSA_WITH_AES_256_CBC_SHA256‹
- TLS_RSA_WITH_DES_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- TLS_RSA_WITH_NULL_MD5
- TLS_RSA_WITH_NULL_SHA
- ›TLS_RSA_WITH_NULL_SHA256‹
- TLS_RSA_EXPORT_WITH_RC4_40_MD5
- TLS_RSA_WITH_RC4_128_MD5
- TLS_RSA_WITH_RC4_40_MD5

For more information about the TLS protocol, see the information provided by the TLS Working Group on the web site of the Internet Engineering Task Force at http://www.ietf.org.

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Secure Sockets Layer (SSL) concepts
An overview of the SSL handshake
How SSL provides authentication, confidentiality, and integrity
CipherSuites and CipherSpecs
Security protocols in WebSphere MQ
Cryptographic concepts

1.4.2. Secure Sockets Layer (SSL) concepts

Secure Sockets Layer (SSL) protocol enables two parties to communicate with privacy and data integrity.

The Secure Sockets Layer (SSL) provides an industry standard protocol for transmitting data in a secure manner over an insecure network. The SSL protocol is widely deployed in both Internet and intranet applications. SSL defines methods for authentication, data encryption, and message integrity for a reliable transport protocol, typically TCP/IP. SSL uses both asymmetric and symmetric cryptography techniques.

An SSL connection is initiated by the caller application, which becomes the SSL client. The responder application becomes the SSL server. Every new SSL session begins with an SSL handshake, as defined by the SSL protocol.

SSL does not provide any formal access control service, because SSL operates at the link level.

Although SSL and TLS are similar, the two protocols do not interoperate.

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Transport Layer Security (TLS) concepts
An overview of the SSL handshake
How SSL provides authentication, confidentiality, and integrity
CipherSuites and CipherSpecs
Security protocols in WebSphere MQ
Cryptographic concepts

1.4.3. An overview of the SSL handshake

The SSL handshake enables the SSL client and SSL server to establish the secret keys with which they communicate.

This section provides a summary of the steps that enable the SSL client and SSL server to communicate with each other:

- Agree on the version of the SSL protocol to use.
- Select cryptographic algorithms.
- Authenticate each other by exchanging and validating digital certificates.
- Use asymmetric encryption techniques to generate a shared secret key, which avoids the key distribution problem. SSL then uses the shared key for the symmetric encryption of messages, which is faster than asymmetric encryption.

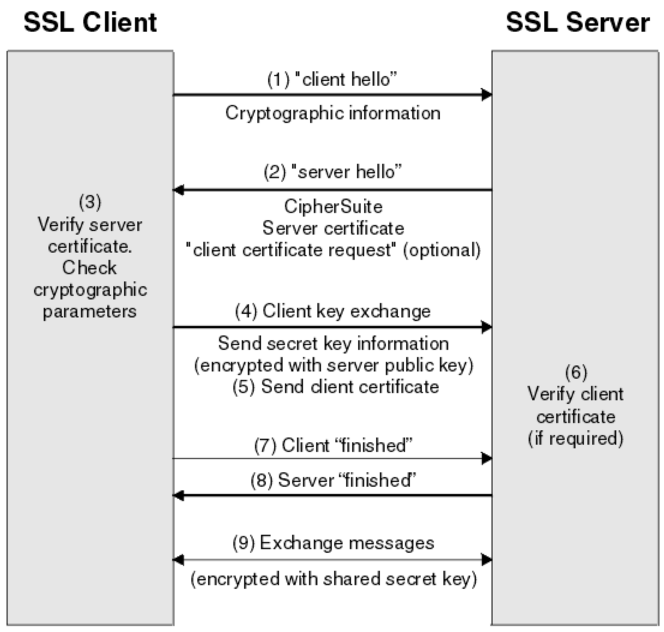
For more information about cryptographic algorithms and digital certificates, refer to the related information.

This section does not attempt to provide full details of the messages exchanged during the SSL handshake. In overview, the steps involved in the SSL handshake are as follows:

1. The SSL client sends a "client hello" message that lists cryptographic information such as the SSL version and, in the client's order of preference, the CipherSuites supported by the client. The message also contains a random byte string that is used in subsequent computations. The SSL protocol allows for the "client hello" to include the data compression methods supported by the client, but current SSL implementations do not typically include this provision.
2. The SSL server responds with a "server hello" message that contains the CipherSuite chosen by the server from the list provided by the SSL client, the session ID, and another random byte string. The SSL server also sends its digital certificate. If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).
3. ›The SSL client verifies the SSL server's digital certificate. For more information, see How SSL provides authentication, confidentiality, and integrity.‹
4. The SSL client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server's public key.
5. If the SSL server sent a "client certificate request", the SSL client sends a random byte string encrypted with the client's private key, together with the client's digital certificate, or a "no digital certificate alert". This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.
6. ›The SSL server verifies the SSL client's certificate. For more information, see How SSL provides authentication, confidentiality, and integrity.‹
7. The SSL client sends the SSL server a "finished" message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.
8. The SSL server sends the SSL client a "finished" message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.
9. For the duration of the SSL session, the SSL server and SSL client can now exchange messages that are symmetrically encrypted with the shared secret key.

Figure 1 illustrates the SSL handshake.

Figure 1. Overview of the SSL handshake


SSL Client
SSL Server
(1) "client hello"
Cryptographic information
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
(3) Verify server certificate. Check cryptographic parameters
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Transport Layer Security (TLS) concepts
Secure Sockets Layer (SSL) concepts
How SSL provides authentication, confidentiality, and integrity
CipherSuites and CipherSpecs
Security protocols in WebSphere MQ
Digital certificates

1.4.4. How SSL provides authentication, confidentiality, and integrity

During both client and server authentication there is a step that requires data to be encrypted with one of the keys in an asymmetric key pair and decrypted with the other key of the pair. A message digest is used to provide integrity.

How SSL provides authentication

For server authentication, the client uses the server's public key to encrypt the data that is used to compute the secret key. The server can generate the secret key only if it can decrypt that data with the correct private key.

For client authentication, the server uses the public key in the client certificate to decrypt the data the client sends during step 5 of the handshake. The exchange of finished messages that are encrypted with the secret key (steps 7 and 8 in the overview) confirms that authentication is complete.

If any of the authentication steps fail, the handshake fails and the session terminates.

The exchange of digital certificates during the SSL handshake is part of the authentication process. For more information about how certificates provide protection against impersonation, refer to the related information. The certificates required are as follows, where CA X issues the certificate to the SSL client, and CA Y issues the certificate to the SSL server:

For server authentication only, the SSL server needs:

- The personal certificate issued to the server by CA Y
- The server's private key

and the SSL client needs:

- The CA certificate for CA Y or the personal certificate issued to the server by CA Y

If the SSL server requires client authentication, the server verifies the client's identity by verifying the client's digital certificate with the public key for the CA that issued the personal certificate to the client, in this case CA X. For both server and client authentication, the SSL server needs:

- The personal certificate issued to the server by CA Y
- The server's private key
- The CA certificate for CA X or the personal certificate issued to the client by CA X

and the SSL client needs:

- The personal certificate issued to the client by CA X
- The client's private key
- The CA certificate for CA Y or the personal certificate issued to the server by CA Y

Both the SSL server and the SSL client might need other CA certificates to form a certificate chain to the root CA certificate. For more information about certificate chains, refer to the related information.

What happens during certificate verification

As noted in steps 3 and 6 of the overview, the SSL client verifies the server's certificate, and the SSL server verifies the client's certificate. There are four aspects to this verification:

1. The digital signature is checked (see Digital signatures).
2. The certificate chain is checked (see How certificate chains work).
3. The expiry and activation dates and the validity period are checked.
4. The revocation status of the certificate is checked (see Working with revoked certificates).

How SSL provides confidentiality

SSL uses a combination of symmetric and asymmetric encryption to ensure message privacy. During the SSL handshake, the SSL client and SSL server agree an encryption algorithm and a shared secret key to be used for one session only. All messages transmitted between the SSL client and SSL server are encrypted using that algorithm and key, ensuring that the message remains private even if it is intercepted. SSL supports a wide range of cryptographic algorithms. Because SSL uses asymmetric encryption when transporting the shared secret key, there is no key distribution problem with SSL. For more information about encryption techniques, refer to Cryptography.

How SSL provides integrity

SSL provides data integrity by calculating a message digest. For more information, refer to Data integrity in message exits.

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Transport Layer Security (TLS) concepts
Secure Sockets Layer (SSL) concepts
An overview of the SSL handshake
CipherSuites and CipherSpecs
Security protocols in WebSphere MQ
Digital certificates
How certificate chains work

1.4.5. CipherSuites and CipherSpecs

Cryptographic security protocols must agree the algorithms used by a secure connection. CipherSuites and CipherSpecs define specific combinations of algorithms.

A CipherSuite is a suite of cryptographic algorithms used by an SSL or TLS connection. A suite comprises three distinct algorithms:

- The key exchange and authentication algorithm, used during the handshake
- The encryption algorithm, used to encipher the data
- The MAC (Message Authentication Code) algorithm, used to generate the message digest

There are several options for each component of the suite, but only certain combinations are valid when specified for an SSL or TLS connection. The name of a valid CipherSuite defines the combination of algorithms used. For example, the CipherSuite SSL_RSA_WITH_RC4_128_MD5 specifies:

- The RSA key exchange and authentication algorithm
- The RC4 encryption algorithm, using a 128–bit key
- The MD5 MAC algorithm

Several algorithms are available for key exchange and authentication, but the RSA algorithm is currently the most widely used. There is more variety in the encryption algorithms and MAC algorithms that are used.

A CipherSpec identifies the combination of the encryption algorithm and MAC algorithm. Both ends of an SSL or TLS connection must agree the same CipherSpec to be able to communicate.

For more information about CipherSpecs, see the related information.

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Transport Layer Security (TLS) concepts
Secure Sockets Layer (SSL) concepts
An overview of the SSL handshake
How SSL provides authentication, confidentiality, and integrity
Security protocols in WebSphere MQ
Working with CipherSpecs

1.4.6. Security protocols in WebSphere MQ

WebSphere MQ supports both the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) protocols to provide link level security for message channels and MQI channels.

Message channels and MQI channels can use the SSL protocol to provide link level security. A caller MCA is an SSL client and a responder MCA is an SSL server. WebSphere® MQ supports Version 3.0 of the SSL protocol. You specify the cryptographic algorithms that are used by the SSL protocol by supplying a CipherSpec as part of the channel definition.

WebSphere MQ also supports Version 1.0 of the Transport Layer Security (TLS) protocol.

At each end of a message channel, and at the server end of an MQI channel, the MCA acts on behalf of the queue manager to which it is connected. During the SSL handshake, the MCA sends the digital certificate of the queue manager to its partner MCA at the other end of the channel. The WebSphere MQ code at the client end of an MQI channel acts on behalf of the user of the WebSphere MQ client application. During the SSL handshake, the WebSphere MQ code sends the user's digital certificate to the MCA at the server end of the MQI channel.

Queue managers and WebSphere MQ client users are not required to have personal digital certificates associated with them when they are acting as SSL clients, unless SSLCAUTH(REQUIRED) is specified at the server side of the channel.

Digital certificates are stored in a *key repository*. The queue manager attribute *SSLKeyRepository* specifies the location of the key repository that holds the queue manager's digital certificate. On a WebSphere MQ client system, the MQSSLKEYR environment variable specifies the location of the key repository that holds the user's digital certificate. Alternatively, a WebSphere MQ client application can specify its location in the *KeyRepository* field of the SSL configuration options structure, MQSCO, on an MQCONNX call. Refer to the WebSphere MQ support for more information about key repositories and how to specify where they are located.

Parent topic: Cryptographic security protocols: TLS and SSL

Related concepts
Transport Layer Security (TLS) concepts
Secure Sockets Layer (SSL) concepts
An overview of the SSL handshake
How SSL provides authentication, confidentiality, and integrity
CipherSuites and CipherSpecs

Related reference
Channel attributes
WebSphere MQ support for SSL and TLS

2. Security tasks for WebSphere MQ

Use this topic to determine what tasks to perform to apply security to your WebSphere® MQ system.

About this task

During this task, you decide what actions are necessary to apply the appropriate level of security to the elements of your WebSphere MQ installation. Each individual task you are referred to gives step-by-step instructions for all platforms.

Procedure

1. **Do you need to limit access to your queue manager to certain users?**

   No: Take no further action.

   Yes: Go to the next question.

2. **Do these users need partial administrative access on a subset of queue manager resources?**

   No: Go to the next question.

   Yes: See Granting partial administrative access on a subset of queue manager resources.

3. **Do these users need full administrative access on a subset of queue manager resources?**

   No: Go to the next question.

   Yes: See Granting full administrative access on a subset of queue manager resources.

4. **Do these users need read only access to all queue manager resources?**

   No: Go to the next question.

   Yes: See Granting read-only access to all resources on a queue manager.

5. **Do these users need full administrative access on all queue manager resources?**

   No: Go to the next question.

   Yes: See Granting full administrative access to all resources on a queue manager.

6. **Do you need user applications to connect to your queue manager?**

   No: Disable connectivity, as described in Removing connectivity to the queue manager

   Yes: See Allowing user applications to connect to your queue manager.

Parent topic: Security

2.1. Granting partial administrative access on a subset of queue manager resources

You need to give certain users partial administrative access to some, but not all, queue manager resources. Use this table to determine the actions you need to take.

Table 1.

The users need to administer objects of this type	Perform this action
Queues	Grant partial administrative access to the required queues, as described in Granting limited administrative access to some queues

Topics	Grant partial administrative access to the required topics, as described in Granting limited administrative access to some topics
Channels	Grant partial administrative access to the required channels, as described in Granting limited administrative access to some channels
The queue manager	Grant partial administrative access to the queue manager, as described in Granting limited administrative access to a queue manager
Processes	Grant partial administrative access to the required processes, as described in Granting limited administrative access to some processes
Namelists	Grant partial administrative access to the required namelists, as described in Granting limited administrative access to some namelists
Services	Grant partial administrative access to the required services, as described in Granting limited administrative access to some services

Parent topic: Security tasks for WebSphere MQ

2.1.1. Granting limited administrative access to some queues

Grant partial administrative access to some queues on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some queues for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName ReqdAction
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.QUEUE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  These commands grant access to the specified queue. To determine which MQSC commands the user can perform on the queue, issue the following commands for each MQSC command:
  ```
  RDEFINE MQCMDS QMgrName.ReqdAction.QType UACC(NONE)
  PERMIT QMgrName.ReqdAction.QType CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
  ```
  To permit the user to use the DISPLAY QUEUE command, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.DISPLAY.QType UACC(NONE)
  PERMIT QMgrName.DISPLAY.QType CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

  ReqdAction

  The action you are allowing the group to take:
  - On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +dsp. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
  - On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP. The authorization *ALLADM is equivalent to all these individual authorizations.
  - On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

  QType

  For the DISPLAY command, one of the values QUEUE, QLOCAL, QALIAS, QMODEL, QREMOTE, or QCLUSTER.
  For other values of *ReqdAction*, one of the values QLOCAL, QALIAS, QMODEL, or QREMOTE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.2. Granting limited administrative access to some topics

Grant partial administrative access to some topics on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some topics for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t topic -g GroupName ReqdAction
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*TOPIC) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.TOPIC.ObjectProfile UACC(NONE)
  PERMIT QMgrName.TOPIC.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  These commands grant access to the specified topic. To determine which MQSC commands the user can perform on the topic, issue the following commands for each MQSC command:
  ```
  RDEFINE MQCMDS QMgrName.ReqdAction.TOPIC UACC(NONE)
  PERMIT QMgrName.ReqdAction.TOPIC CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
  ```
  To permit the user to use the DISPLAY TOPIC command, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.DISPLAY.TOPIC UACC(NONE)
  PERMIT QMgrName.DISPLAY.TOPIC CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

  ReqdAction

  The action you are allowing the group to take:
  - On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +dsp. +ctrl. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
  - On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL. The authorization *ALLADM is equivalent to all these individual authorizations.
  - On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.3. Granting limited administrative access to some channels

Grant partial administrative access to some channels on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some channels for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t channel -g GroupName ReqdAction
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*CHL) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.CHANNEL.ObjectProfile UACC(NONE)
  PERMIT QMgrName.CHANNEL.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  These commands grant access to the specified channel. To determine which MQSC commands the user can perform on the channel, issue the following commands for each MQSC command:
  ```
  RDEFINE MQCMDS QMgrName.ReqdAction.CHANNEL UACC(NONE)
  PERMIT QMgrName.ReqdAction.CHANNEL CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
  ```
  To permit the user to use the DISPLAY CHANNEL command, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.DISPLAY.CHANNEL UACC(NONE)
  PERMIT QMgrName.DISPLAY.CHANNEL CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

The name of the group to be granted access.

ReqdAction

The action you are allowing the group to take:

- On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +dsp. +ctrl, +ctrlx. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
- On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL, *CTRLx. The authorization *ALLADM is equivalent to all these individual authorizations.
- On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.4. Granting limited administrative access to a queue manager

Grant partial administrative access to a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to perform some actions on the queue manager, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t qmgr -g GroupName ReqdAction
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*MQM) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.QMGR.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QMGR.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  These commands grant access to the specified queue manager. To determine which MQSC commands the user can perform on the queue manager, issue the following commands for each MQSC command:
  ```
  RDEFINE MQCMDS QMgrName.ReqdAction.QMGR UACC(NONE)
  PERMIT QMgrName.ReqdAction.QMGR CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
  ```
  To permit the user to use the DISPLAY QMGR command, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.DISPLAY.QMGR UACC(NONE)
  PERMIT QMgrName.DISPLAY.QMGR CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

  ReqdAction

  The action you are allowing the group to take:

  - On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +dsp. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
  - On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP. The authorization *ALLADM is equivalent to all these individual authorizations.
  - On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.5. Granting limited administrative access to some processes

Grant partial administrative access to some processes on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some processes for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -n ObjectProfile -t process -g GroupName ReqdAction
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*PRC) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQADMIN QMgrName.CHANNEL.ObjectProfile UACC(NONE)
PERMIT QMgrName.PROCESS.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```

These commands grant access to the specified channel. To determine which MQSC commands the user can perform on the channel, issue the following commands for each MQSC command:

```
RDEFINE MQCMDS QMgrName.ReqdAction.PROCESS UACC(NONE)
PERMIT QMgrName.ReqdAction.PROCESS CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
```

To permit the user to use the DISPLAY PROCESS command, issue the following commands:

```
RDEFINE MQCMDS QMgrName.DISPLAY.PROCESS UACC(NONE)
PERMIT QMgrName.DISPLAY.PROCESS CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
```

The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName
The name of the group to be granted access.

ReqdAction
The action you are allowing the group to take:

- On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +dsp. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
- On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP. The authorization *ALLADM is equivalent to all these individual authorizations.
- On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.6. Granting limited administrative access to some namelists

Grant partial administrative access to some namelists on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some namelists for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -n ObjectProfile -t namelist -g GroupName ReqdAction
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*NMLIST) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQADMIN QMgrName.NAMELIST.ObjectProfile UACC(NONE)
PERMIT QMgrName.NAMELIST.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```

These commands grant access to the specified namelist. To determine which MQSC commands the user can perform on the namelist, issue the following commands for each MQSC command:

```
RDEFINE MQCMDS QMgrName.ReqdAction.NAMELIST UACC(NONE)
PERMIT QMgrName.ReqdAction.NAMELIST CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
```

To permit the user to use the DISPLAY NAMELIST command, issue the following commands:

```
RDEFINE MQCMDS QMgrName.DISPLAY.NAMELIST UACC(NONE)
PERMIT QMgrName.DISPLAY.NAMELIST CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
```

The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName

The name of the group to be granted access.

ReqdAction

The action you are allowing the group to take:

- On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +ctrl, +ctrlx, +dsp. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
- On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL, *CTRLX. The authorization *ALLADM is equivalent to all these individual authorizations.
- On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.1.7. Granting limited administrative access to some services

Grant partial administrative access to some services on a queue manager, to each group of users with a business need for it.

About this task

To grant limited administrative access to some services for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
```
setmqaut -m QMgrName -n ObjectProfile -t service -g GroupName ReqdAction
```
- For i5/OS®, issue the following command:
```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*SVC) USER(GroupName) AUT(ReqdAction) MQMNAME('QMgrName')
```
- For z/OS®, issue the following commands:
```
RDEFINE MQADMIN QMgrName.SERVICE.ObjectProfile UACC(NONE)
PERMIT QMgrName.SERVICE.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```
  These commands grant access to the specified service. To determine which MQSC commands the user can perform on the service, issue the following commands for each MQSC command:
```
RDEFINE MQCMDS QMgrName.ReqdAction.SERVICE UACC(NONE)
PERMIT QMgrName.ReqdAction.SERVICE CLASS(MQCMDS) ID(GroupName) ACCESS(ALTER)
```
  To permit the user to use the DISPLAY SERVICE command, issue the following commands:
```
RDEFINE MQCMDS QMgrName.DISPLAY.SERVICE UACC(NONE)
PERMIT QMgrName.DISPLAY.SERVICE CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

  ReqdAction

  The action you are allowing the group to take:

  - On UNIX and Windows systems, any combination of the following authorizations: +chg, +clr, +crt, +dlt, +ctrl, +ctrlx, +dsp. The authorization +alladm is equivalent to +chg +clr +dlt +dsp.
  - On i5/OS, any combination of the following authorizations: *ADMCHG, *ADMCLR, *ADMCRT, *ADMDLT, *ADMDSP, *CTRL, *CTRLX. The authorization *ALLADM is equivalent to all these individual authorizations.
  - On z/OS, one of the values ALTER, CLEAR, DEFINE, DELETE, or MOVE.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2. Granting full administrative access on a subset of queue manager resources

You need to give certain users full administrative access to some, but not all, queue manager resources. Use these tables to determine the actions you need to take.

Table 1.

The users need to administer objects of this type	Perform this action
Queues	Grant full administrative access to the required queues, as described in Granting full administrative access to some queues
Topics	Grant full administrative access to the required topics, as described in Granting full administrative access to some topics
Channels	Grant full administrative access to the required channels, as described in Granting full administrative access to some channels
The queue manager	Grant full administrative access to the queue manager, as described in Granting full administrative access to a queue manager
Processes	Grant full administrative access to the required processes, as described in Granting full administrative access to some processes
Namelists	Grant full administrative access to the required namelists, as described in Granting full administrative access to some namelists
Services	Grant full administrative access to the required services, as described in Granting full administrative access to some services

Parent topic: Security tasks for WebSphere MQ

2.2.1. Granting full administrative access to some queues

Grant full administrative access to some queues on a queue manager, to each group of users with a business need for it.

About this task

To grant full administrative access to some queues, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +alladm
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*ALLADM) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.QUEUE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.2. Granting full administrative access to some topics

Grant full administrative access to some topics on a queue manager, to each group of users with a business need for it.

About this task

To grant full administrative access to some topics for some actions, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t topic -g GroupName +alladm
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*TOPIC) USER(GroupName) AUT(ALLADM) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.TOPIC.ObjectProfile UACC(NONE)
  ```

```
PERMIT QMgrName.TOPIC.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```

The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName
The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.3. Granting full administrative access to some channels

Grant full administrative access to some channels on a queue manager, to each group of users with a business need for it.

About this task

To grant full administrative access to some channels, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
```
setmqaut -m QMgrName -n ObjectProfile -t channel -g GroupName +alladm
```
- For i5/OS®, issue the following command:
```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*CHL) USER(GroupName) AUT(ALLADM) MQMNAME('QMgrName')
```
- For z/OS®, issue the following commands:
```
RDEFINE MQADMIN QMgrName.CHANNEL.ObjectProfile UACC(NONE)
PERMIT QMgrName.CHANNEL.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```

The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName
The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.4. Granting full administrative access to a queue manager

Grant full administrative access to a queue manager, to each group of users with a business need for it.

About this task

To grant full administrative access to the queue manager, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
```
setmqaut -m QMgrName -t qmgr -g GroupName +alladm
```
- For i5/OS®, issue the following command:
```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*MQM) USER(GroupName) AUT(*ALLADM) MQMNAME('QMgrName')
```
- For z/OS®, issue the following commands:
```
RDEFINE MQADMIN QMgrName.QMGR UACC(NONE)
PERMIT QMgrName.QMGR CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```

The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName
The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.5. Granting full administrative access to some processes

Grant full administrative access to some processes on a queue manager, to each group of users with a business need for it.

About this task
To grant full administrative access to some processes, use the appropriate commands for your operating system.

Procedure
- For UNIX and Windows systems, issue the following command:
```
setmqaut -m QMgrName -n ObjectProfile -t process -g GroupName +alladm
```
- For i5/OS®, issue the following command:
```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*PRC) USER(GroupName) AUT(*ALLADM) MQMNAME('QMgrName')
```
- For z/OS®, issue the following commands:
```
RDEFINE MQADMIN QMgrName.CHANNEL.ObjectProfile UACC(NONE)
PERMIT QMgrName.PROCESS.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```
The variable names have the following meanings:

QMgrName
The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
The name of the object or generic profile for which to change authorizations.

GroupName
The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.6. Granting full administrative access to some namelists

Grant full administrative access to some namelists on a queue manager, to each group of users with a business need for it.

About this task
To grant full administrative access to some namelists, use the appropriate commands for your operating system.

Procedure
- For UNIX and Windows systems, issue the following command:
```
setmqaut -m QMgrName -n ObjectProfile -t namelist -g GroupName +alladm
```
- For i5/OS®, issue the following command:
```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*NMLIST) USER(GroupName) AUT(*ALLADM) MQMNAME('QMgrName')
```
- For z/OS®, issue the following commands:
```
RDEFINE MQADMIN QMgrName.NAMELIST.ObjectProfile UACC(NONE)
PERMIT QMgrName.NAMELIST.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
```
The variable names have the following meanings:

QMgrName

The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile

The name of the object or generic profile for which to change authorizations.

GroupName

The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.2.7. Granting full administrative access to some services

Grant full administrative access to some services on a queue manager, to each group of users with a business need for it.

About this task

To grant full administrative access to some services, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t service -g GroupName +alladm
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*SVC) USER(GroupName) AUT(*ALLADM) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQADMIN QMgrName.SERVICE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.SERVICE.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(ALTER)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.3. Granting read-only access to all resources on a queue manager

Grant read-only access to all the resources on a queue manager, to each user or group of users with a business need for it.

About this task

Use the Add Role Based Authorities wizard or the appropriate commands for your operating system.

Procedure

- Using the wizard:
  1. In the WebSphere® MQ Explorer Navigator pane, right-click the queue manager and click **Object Authorities** > **Add Role Based Authorities**
     The Add Role Based Authorities wizard opens.
- For UNIX and Windows systems, issue the following commands:
  ```
  setmqaut -m QMgrName -n ** -t queue -g GroupName +browse +dsp
  setmqaut -m QMgrName -n SYSTEM.ADMIN.COMMAND.QUEUE -t queue -g GroupName +dsp +inq +put
  setmqaut -m QMgrName -n SYSTEM.MQEXPLORER.REPLY.MODEL -t queue -g GroupName +dsp +inq +get
  setmqaut -m QMgrName -n ** -t topic -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t channel -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t clntconn -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t authinfo -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t listener -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t namelist -g GroupName +dsp
  setmqaut -m QMgrName -n ** -t process -g GroupName +dsp
  ```

```
setmqaut -m QMgrName -n ** -t service -g GroupName +dsp
setmqaut -m QMgrName -t qmgr -g GroupName +dsp +inq +connect
```

The specific authorities to SYSTEM.ADMIN.COMMAND.QUEUE and SYSTEM.MQEXPLORER.REPLY.MODEL are necessary only if you want to use the MQ Explorer.

- For i5/OS®, issue the following commands:

```
GRTMQMAUT OBJ(*ALL) OBJTYPE(*Q) USER('GroupName') AUT(*ADMDSP *BROWSE) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*TOPIC) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*CHL) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*CLTCN) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*AUTHINFO) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*LSR) USER('GroupName') AUT(*ADMDSP)MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*NMLIST) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*PRC) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ(*ALL) OBJTYPE(*SVC) USER('GroupName') AUT(*ADMDSP) MQMNAME('QMgrName')
GRTMQMAUT OBJ('object-name') OBJTYPE(*MQM)  USER('GroupName') AUT(*ADMDSP *CONNECT *INQ) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQQUEUE QMgrName.** UACC(NONE)
PERMIT QMgrName.** CLASS(MQQUEUE) ID(GroupName) ACCESS(READ)
RDEFINE MQTOPIC QMgrName.** UACC(NONE)
PERMIT QMgrName.** CLASS(MQTOPIC) ID(GroupName) ACCESS(READ)
RDEFINE MQPROC QMgrName.** UACC(NONE)
PERMIT QMgrName.** CLASS(MQPROC) ID(GroupName) ACCESS(READ)
RDEFINE MQNLIST QMgrName.** UACC(NONE)
PERMIT QMgrName.** CLASS(MQNLIST) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.BATCH UACC(NONE)
PERMIT QMgrName.BATCH CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.CICS UACC(NONE)
PERMIT QMgrName.CICS CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.IMS UACC(NONE)
PERMIT QMgrName.IMS CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.CHIN UACC(NONE)
PERMIT QMgrName.CHIN CLASS(MQCONN) ID(GroupName) ACCESS(READ)
```

The variable names have the following meanings:

QMgrName

The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

GroupName

The name of the group to be granted access.

Parent topic: Security tasks for WebSphere MQ

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.4. Granting full administrative access to all resources on a queue manager

Grant full administrative access to all the resources on a queue manager, to each user or group of users with a business need for it.

About this task

Use the Add Role Based Authorities wizard or the appropriate commands for your operating system.

Procedure

- Using the wizard:
  1. In the WebSphere® MQ Explorer Navigator pane, right-click the queue manager and click **Object Authorities** > **Add Role Based Authorities** The Add Role Based Authorities wizard opens.
- For UNIX systems, issue the following commands:

```
setmqaut -m QMgrName -n ** -t queue -g GroupName +alladm +browse
setmqaut -m QMgrName -n @class -t queue -g GroupName +crt
setmqaut -m QMgrName -n SYSTEM.ADMIN.COMMAND.QUEUE -t queue -g GroupName +dsp +inq +put
setmqaut -m QMgrName -n SYSTEM.MQEXPLORER.REPLY.QUEUE -t queue -g GroupName +dsp +inq +get
setmqaut -m QMgrName -n ** -t topic -g GroupName +alladm
setmqaut -m QMgrName -n @class -t topic -g GroupName +crt
setmqaut -m QMgrName -n ** -t channel -g GroupName +alladm
setmqaut -m QMgrName -n @class -t channel -g GroupName +crt
setmqaut -m QMgrName -n ** -t clntconn -g GroupName +alladm
setmqaut -m QMgrName -n @class -t clntconn -g GroupName +crt
setmqaut -m QMgrName -n ** -t authinfo -g GroupName +alladm
setmqaut -m QMgrName -n @class -t authinfo -g GroupName +crt
setmqaut -m QMgrName -n ** -t listener -g GroupName +alladm
setmqaut -m QMgrName -n @class -t listener -g GroupName +crt
setmqaut -m QMgrName -n ** -t namelist -g GroupName +alladm
setmqaut -m QMgrName -n @class -t namelist -g GroupName +crt
setmqaut -m QMgrName -n ** -t process -g GroupName +alladm
setmqaut -m QMgrName -n @class -t process -g GroupName +crt
setmqaut -m QMgrName -n ** -t service -g GroupName +alladm
setmqaut -m QMgrName -n @class -t service -g GroupName +crt
setmqaut -m QMgrName -t qmgr -g GroupName +alladm +conn
```

- For Windows systems, issue the same commands as for UNIX systems, but using the profile name @CLASS instead of @class.
- For i5/OS, issue the following command:

```
GRTMQMAUT OBJ(*ALL) OBJTYPE(*ALL) USER('GroupName') AUT(*ALLADM) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQADMIN QMgrName.*.** UACC(NONE)
PERMIT QMgrName.*.** CLASS(MQADMIN) ID(GroupName) ACCESS(ALTER)
```

The variable names have the following meanings:

QMgrName

The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

GroupName

The name of the group to be granted access.

Parent topic: Security tasks for WebSphere MQ

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.5. Removing connectivity to the queue manager

If you do not want user applications to connect to your queue manager, remove their authority to connect to it.

About this task

Revoke the authority of all users to connect to the queue manager by using the appropriate command for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -t qmgr -g GroupName -connect
```

- For i5/OS®, issue the following command:

```
RVKMQMAUT OBJ('QMgrName') OBJTYPE(*MQM) USER(*ALL) AUT(*CONNECT)
```

- For z/OS®, issue the following commands:

```
RDEFINE MQCONN QMgrName.BATCH UACC(NONE)
RDEFINE MQCONN QMgrName.CHIN UACC(NONE)
RDEFINE MQCONN QMgrName.CICS UACC(NONE)
RDEFINE MQCONN QMgrName.IMS UACC(NONE)
```

Do not issue any PERMIT commands.
The variable names have the following meanings:

QMgrName

The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

GroupName

The name of the group to be denied access.

Parent topic: Security tasks for WebSphere MQ

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
RVKMQMAUT
Security controls and options (z/OS)

2.6. Allowing user applications to connect to your queue manager

You want to allow user application to connect to your queue manager. Use the tables in this topic to determine what actions to take.

First, determine whether client applications will connect to your queue manager.

If none of the applications that will connect to your queue manager are client applications, disable remote access as described in Disabling remote access to the queue manager.

If one or more of the applications that will connect to your queue manager are client applications, secure remote connectivity as described in Securing remote connectivity to the queue manager.

In both cases, set up connection security as described in Setting up connection security

If you want to control access to resources for each user connecting to the queue manager, see the following table. If the statement in the first column is true, take the action listed in the second column.

Statement	Take this action
You have applications that make use of queues	See Controlling user access to queues
You have applications that make use of topics	See Controlling user access to topics.
You have applications that inquire on the queue manager object	See Granting a user authority to inquire on a queue manager.
You have applications that use process objects	See Granting a user authority to access processes
You have applications that make use of namelists	See Granting a user authority to access namelists

Parent topic: Security tasks for WebSphere MQ

2.6.1. Securing remote connectivity to the queue manager

You can secure remote connectivity to the queue manager using SSL, a security exit, or both.

About this task

Secure remote connectivity to the queue manager in one of the following ways:

Procedure

- Using SSL:
  1. Set SSLPEER on the server-connection channel to a specific value or narrow range of values.
  2. Set MCAUSER on the client-connection channel to a value matching SSLPEER.
- Using SSL with a security exit:
  1. Set MCAUSER on the client-connection channel to a user identifier with no privileges.
  2. Write a security exit to assign an MCAUSER value depending on the value of SSLPEER it receives.
- Using only a security exit:
  1. Write a security exit to authorize connections based on any property you choose, for example, the originating IP address.

Related concepts
WebSphere MQ rules for SSLPEER values

Related information
Access control in security exits
ALTER CHANNEL
Link level security using a security exit
Writing a security exit

2.6.2. Disabling remote access to the queue manager

If you do not want client applications to connect to your queue manager, disable remote access to it.

About this task

Prevent client applications connecting to the queue manager in one of the following ways:

Procedure

- Delete all server-connection channels using the MQSC command **DELETE CHANNEL**.
- Set the message channel agent user identifier (MCAUSER) of the channel to a user ID with no access rights, using the MQSC command **ALTER CHANNEL**.

Related information
ALTER CHANNEL
DELETE CHANNEL

2.6.3. Setting up connection security

Grant the authority to connect to the queue manager to each user or group of users with a business need to do so.

About this task

To set up connection security, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -t qmgr -g GroupName +connect
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('QMgrName') OBJTYPE(*MQM)  USER('GroupName') AUT(*CONNECT)
```

- For z/OS®, issue the following commands:

```
RDEFINE MQCONN QMgrName.BATCH UACC(NONE)
PERMIT QMgrName.BATCH CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.CICS UACC(NONE)
PERMIT QMgrName.CICS CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.IMS UACC(NONE)
PERMIT QMgrName.IMS CLASS(MQCONN) ID(GroupName) ACCESS(READ)
RDEFINE MQCONN QMgrName.CHIN UACC(NONE)
PERMIT QMgrName.CHIN CLASS(MQCONN) ID(GroupName) ACCESS(READ)
```

  These commands give authority to connect for batch, CICS®, IMS™ and the channel initiator (CHIN). If you do not use a particular type of connection, omit the relevant commands. The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4. Controlling user access to queues

You want to control application access to queues. Use this topic to determine what actions to take.

For each true statement in the first column, take the action indicated in the second column.

Statement	Action
The application gets messages from a queue	See Granting a user authority to get from queues
The application sets context	See Granting a user authority to set context
The application passes context	See Granting a user authority to pass context
The application puts messages on a clustered queue	See Authorizing putting messages on remote cluster queues
The application puts messages on a local queue	See Granting a user authority to put to a local queue
The application puts messages on a model queue	See Granting a user authority to put to a model queue
The application puts messages on a remote queue	See Granting a user authority to put to a remote queue

2.6.4.1. Granting a user authority to get from queues

Grant the authority to get messages from a queue or set of queues, to each group of users with a business need for it.

About this task

To grant the authority to get messages from some queues, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +get
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*GET) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(READ)
```

  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4.2. Granting a user authority to set context

Grant the authority to set context on a message that is being put, to each group of users with a business need for it.

About this task

To grant the authority to set context on some queues, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue one of the following commands:
  - To set identity context only:
    ```
    setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +setid
    ```
  - To set all context:
    ```
    setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +setall
    ```
- For i5/OS®, issue one of the following commands:
  - To set identity context only:
    ```
    GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*SETID) MQMNAME('QMgrName')
    ```
  - To set all context:
    ```
    GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*SETALL) MQMNAME('QMgrName')
    ```
- For z/OS®, issue one of the following sets of commands:
  - To set identity context only:
    ```
    RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
    PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
    ```
  - To set all context:
    ```
    RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
    PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(CONTROL)
    ```

  The variable names have the following meanings:

  QMgrName
  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile
  The name of the object or generic profile for which to change authorizations.

  GroupName
  The name of the group to be granted access.

Related concepts
Profiles for context security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4.3. Granting a user authority to pass context

Grant the authority to pass context from a retrieved message to one that is being put, to each group of users with a business need for it.

About this task

To grant the authority to pass context on some queues, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue one of the following commands:
  - To pass identity context only:
    ```
    setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +passid
    ```
  - To pass all context:
    ```
    setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +passall
    ```

- For i5/OS®, issue one of the following commands:
  - To pass identity context only:
    ```
    GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*PASSID) MQMNAME('QMgrName')
    ```
  - To pass all context:
    ```
    GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*PASSALL) MQMNAME('QMgrName')
    ```
- For z/OS®, issue the following commands to pass identity context or all context:
  ```
  RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for context security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4.4. Granting a user authority to put to a local queue

Grant the authority to put messages to a local queue or set of queues, to each group of users with a business need for it.

About this task

To grant the authority to put messages to some local queues, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +put
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
  ```
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4.5. Granting a user authority to put to a model queue

Grant the authority to put messages to a model queue or set of model queues, to each group of users with a business need for it.

About this task

Model queues are used to create dynamic queues. You must therefore grant authority to both the model and dynamic queues. To grant these authorities, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following commands:

```
setmqaut -m QMgrName -n ModelQueueName -t queue -g GroupName +put
setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +put
```

- For i5/OS®, issue the following commands:

```
GRTMQMAUT OBJ('ModelQueueName') OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQQUEUE QMgrName.ModelQueueName UACC(NONE)
PERMIT QMgrName.QUEUE.ModelQueueName CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
RDEFINE MQQUEUE QMgrName.ObjectProfile UACC(NONE)
PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
```

The variable names have the following meanings:

QMgrName
: The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ModelQueueName
: The name of the model queue on which dynamic queues are based.

ObjectProfile
: The name of the dynamic queue or generic profile for which to change authorizations.

GroupName
: The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.4.6. Granting a user authority to put to a remote queue

Grant the authority to put messages to a remote queue or set of queues, to each group of users with a business need for it.

About this task

To put a message on a remote queue, you can either put it on a local definition of a remote queue, or a fully qualified remote queue. If you are using a local definition of a remote queue, you need authority to put to the local object. If you are using a fully qualified remote queue, you need authority to put to the transmission queue on the local queue manager. Grant these authorities using the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following commands:

```
setmqaut -m QMgrName -n ObjectProfile -t queue -g GroupName +put
```

- For i5/OS®, issue the following commands:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
```

The variable names have the following meanings:

QMgrName
: The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

ObjectProfile
: The name of the queue or generic profile for which to change authorizations. This is the local name of the remote queue or the name of the transmission queue, as appropriate.

GroupName
: The name of the group to be granted access.

Related concepts
Profiles for queue security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)
Opening remote queues

2.6.5. Controlling user access to topics

You need to control the access of applications to topics. Use this topic to determine what actions to take.

For each true statement in the first column, take the action indicated in the second column.

Table 1.

Statement	Action
The application publishes messages to a topic	See Granting a user authority to publish to a topic
The application subscribes to a topic	See Granting a user authority to subscribe to topics

2.6.5.1. Granting a user authority to publish to a topic

Grant the authority to publish messages to a topic or set of topics, to each group of users with a business need for it.

About this task

To grant the authority to publish messages to some topics, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -n ObjectProfile -t topic -g GroupName +pub
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*TOPIC) USER(GroupName) AUT(*PUB) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQTOPIC QMgrName.PUBLISH.ObjectProfile UACC(NONE)
PERMIT QMgrName.PUBLISH.ObjectProfile CLASS(MQTOPIC) ID(GroupName) ACCESS(UPDATE)
```

  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.5.2. Granting a user authority to subscribe to topics

Grant the authority to subscribe to a topic or set of topics, to each group of users with a business need for it.

About this task

To grant the authority to subscribe to some topics, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:

```
setmqaut -m QMgrName -n ObjectProfile -t topic -g GroupName +sub
```

- For i5/OS®, issue the following command:

```
GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*TOPIC) USER(GroupName) AUT(*SUB) MQMNAME('QMgrName')
```

- For z/OS®, issue the following commands:

```
RDEFINE MQTOPIC QMgrName.SUBSCRIBE.ObjectProfile UACC(NONE)
PERMIT QMgrName.SUBSCRIBE.ObjectProfile CLASS(MQTOPIC) ID(GroupName) ACCESS(UPDATE)
```

  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the object or generic profile for which to change authorizations.

  GroupName

  The name of the group to be granted access.

Related concepts

Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.6. Granting a user authority to inquire on a queue manager

Grant the authority to inquire on a queue manager, to each group of users with a business need for it.

About this task

To grant the authority to inquire on a queue manager, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t qmgr -g GroupName +inq
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*MQM) USER(GroupName) AUT(*INQ) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.QMGR.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QMGR.ObjectProfile CLASS(MQADMIN) ID(GroupName ) ACCESS(READ)
  ```
  These commands grant access to the specified queue manager. To permit the user to use the MQINQ command, issue the following commands:
  ```
  RDEFINE MQCMDS QMgrName.MQINQ.QMGR UACC(NONE)
  PERMIT QMgrName.MQINQ.QMGR CLASS(MQCMDS) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName
  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile
  The name of the object or generic profile for which to change authorizations.

  GroupName
  The name of the group to be granted access.

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)

2.6.7. Granting a user authority to access processes

Grant the authority to access a process or set of processes, to each group of users with a business need for it.

About this task

To grant the authority to access some processes, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t process -g GroupName +all
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*PRC) USER(GroupName) AUT(*ALL) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQPROC QMgrName.ObjectProfile UACC(NONE)
  PERMIT QMgrName.ObjectProfile CLASS(MQTOPIC) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName
  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile
  The name of the object or generic profile for which to change authorizations.

  GroupName
  The name of the group to be granted access.

2.6.8. Granting a user authority to access namelists

Grant the authority to access a namelist or set of namelists, to each group of users with a business need for it.

About this task

To grant the authority to access some namelists, use the appropriate commands for your operating system.

Procedure

- For UNIX and Windows systems, issue the following command:
  ```
  setmqaut -m QMgrName -n ObjectProfile -t namelist -g GroupName +all
  ```
- For i5/OS®, issue the following command:
  ```
  GRTMQMAUT OBJ('ObjectProfile') OBJTYPE(*NMLIST) USER(GroupName) AUT(*ALL) MQMNAME('QMgrName')
  ```
- For z/OS®, issue the following commands:
  ```
  RDEFINE MQNLIST QMgrName.ObjectProfile UACC(NONE)
  PERMIT QMgrName.ObjectProfile CLASS(MQNLIST) ID(GroupName) ACCESS(READ)
  ```
  The variable names have the following meanings:

  QMgrName
  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile
  The name of the object or generic profile for which to change authorizations.

  GroupName
  The name of the group to be granted access.

3. Authority to administer WebSphere MQ

WebSphere® MQ administrators need authority to perform various functions. This authority is obtained in different ways on different platforms.

WebSphere MQ administrators need authority to:

- Issue commands to administer WebSphere MQ
- Use the WebSphere MQ Explorer
- Use the operations and control panels on z/OS®
- Use the WebSphere MQ utility program, CSQUTIL, on z/OS
- Access the queue manager data sets on z/OS

See the topic appropriate to your operating system.
Parent topic: Security

Related concepts
Authority to administer WebSphere MQ on z/OS
Authority to administer WebSphere MQ on UNIX and Windows systems

Related reference
WebSphere MQ authorities on i5/OS

4. Authority to work with WebSphere MQ objects

When applications access objects, the user IDs associated with the applications need appropriate authority.

Applications can access the following WebSphere® MQ objects by issuing MQI calls:

- Queue managers
- Queues
- Processes
- Namelists
- Topics

Applications can also use PCF commands to access these WebSphere MQ objects, and to access channels and authentication information objects as well. These objects are protected by WebSphere MQ and the user IDs associated with the applications need authority to access them.

Applications, in this context, include those written by users and vendors, and those supplied with WebSphere MQ for z/OS®. The applications supplied with WebSphere MQ for z/OS include:

- The operations and control panels
- The WebSphere MQ utility program, CSQUTIL

- The dead letter queue handler utility, CSQUDLQH

Applications that use the Application Messaging Interface (AMI), WebSphere MQ classes for Java, or WebSphere MQ classes for JMS still use the MQI indirectly.

MCAs also issue MQI calls and the user IDs associated with the MCAs need authority to access these WebSphere MQ objects. For more information about these user IDs and the authorities they require, see Channel security.

On z/OS, applications can also use MQSC commands to access these WebSphere MQ objects but command security and command resource security provide the authority checks in these circumstances. For more information, see Command security and command resource security and MQSC commands and the system command input queue.

On i5/OS®, a user that issues a CL command in Group 2 might require authority to access a WebSphere MQ object associated with the command. For more information, see When authority checks are performed.

When authority checks are performed
Authority checks are performed when an application attempts to access a queue manager, queue, process, or namelist.

Alternate user authority
When an application opens an object or subscribes to a topic, the application can supply a user ID on the MQOPEN, MQPUT1, or MQSUB call. It can ask the queue manager to use this user ID for authority checks instead of the one associated with the application.

Message context
Message context information allows the application that retrieves a message to find out about the originator of the message. The information is held in fields in the message descriptor and the fields are divided into three logical parts

Authority to work with WebSphere MQ objects on i5/OS, UNIX systems, and Windows systems
The authorization service component provided with WebSphere MQ is called the *Object Authority Manager (OAM)*. It provides access control when an application issues an MQI call to access a queue manager, queue, process, topic, or namelist.

Authority to work with WebSphere MQ objects on z/OS
On z/OS, there are seven categories of authority check associated with calls to the MQI. You must define certain RACF® profiles and give appropriate access to these profiles. Use the *RESLEVEL* profile to control how many users IDs are checked.

Parent topic: Security

4.1. When authority checks are performed

Authority checks are performed when an application attempts to access a queue manager, queue, process, or namelist.

On i5/OS®, authority checks might also be performed when a user issues a CL command in Group 2 that accesses any of these WebSphere® MQ objects. The checks are performed in the following circumstances:

When an application connects to a queue manager using an MQCONN or MQCONNX call

The queue manager asks the operating system for the user ID associated with the application. The queue manager then checks that the user ID is authorized to connect to it and retains the user ID for future checks.

When an application opens a WebSphere MQ object using an MQOPEN or MQPUT1 call

All authority checks are performed when an object is opened, not when it is accessed later. For example, authority checks are performed when an application opens a queue, but not when the application puts messages on the queue or gets messages from the queue.

When an application opens an object, it specifies the types of operation it needs to perform on the object. For example, an application might open a queue to browse the messages on it, get messages from it, but not to put messages on it. For each type of operation the application specifies, the queue manager checks that the user ID associated with the application has the authority to perform that operation.

When an application opens a queue, the authority checks are performed against the object named in the *ObjectName* field of the object descriptor used on the MQOPEN or MQPUT1 call. If the object is an alias queue or a remote queue definition, the authority checks are performed against the object itself, not the queue to which the alias queue or the remote queue definition resolves.

If an application references a remote queue explicitly by setting the *ObjectName* and *ObjectQMgrName* fields in the object descriptor to the names of the remote queue and the remote queue manager, the authority checks are performed against the transmission queue with the same name as the remote queue manager. If an application references a cluster queue explicitly by setting the *ObjectName* field in the object descriptor to the name of the cluster queue, the authority checks are performed against the cluster transmission queue, SYSTEM.CLUSTER.TRANSMIT.QUEUE.

The user ID that the queue manager uses for the authority checks is the user ID obtained from the operating system when the application connects to the queue manager.

When an application subscribes to a topic using an MQSUB call

When an application subscribes to a topic, it specifies the type of operation that it needs to perform. It is either creating a new subscription, altering an existing subscription, or resuming an existing subscription without changing it. For each type of operation, the queue manager checks that the user ID that is associated with the application has the authority to perform the operation.

When an application subscribes to a topic, the authority checks are performed against the topic objects that are found in the topic tree at, or above, the point in the topic tree at which the application subscribed. The authority checks might involve checks on more than one topic object.

The user ID that the queue manager uses for the authority checks is the user ID obtained from the operating system when the application connects to the queue manager.

The queue manager performs authority checks on subscriber queues but not on managed queues.

When an application deletes a permanent dynamic queue using an MQCLOSE call

If the object handle specified on the MQCLOSE call is not the one returned by the MQOPEN call that created the permanent dynamic queue, the queue manager checks that the user ID associated with the application that issued the MQCLOSE call is authorized to delete the queue.

When an application closes a subscription to remove it but the application did not create it, the appropriate authority is required to remove it.

When a PCF command that operates on a WebSphere MQ object is processed by the command server

This includes the case where a PCF command operates on an authentication information object.

The user ID that is used for the authority checks is the one found in the *UserIdentifier* field in the message descriptor of the PCF command. This user ID must have the required authorities on the queue manager where the command is processed. The equivalent MQSC command encapsulated within an Escape PCF command is treated in the same way. For more information about the *UserIdentifier* field and how it is set, see Message context.

On i5/OS, when a user issues a CL command in Group 2 that operates on a WebSphere MQ object

This includes the case where a CL command in Group 2 operates on an authentication information object.

Unless the user is a member of the QMQMADM group or has *ALLOBJ authority, checks are performed to determine whether the user has the authority to operate on a WebSphere MQ object associated with the command. The authority required depends on the type of operation that the command performs on the object. For example, the command CHGMQMQ, Change MQM Queue, requires the authority to change the attributes of the queue specified by the command. In contrast, the command DSPMQMQ, Display MQM Queue, requires the authority to display the attributes of the queue specified by the command.

Many commands operate on more than one object. For example, to issue the command DLTMQMQ, Delete MQM Queue, the following authorities are required:

- The authority to connect to the queue manager specified by the command
- The authority to delete the queue specified by the command

Some commands operate on no object at all. In this case, the user requires only i5/OS authority to issue one of these commands. STRMQMLSR, Start MQM Listener, is an example of such a command.

Parent topic: Authority to work with WebSphere MQ objects

4.2. Alternate user authority

When an application opens an object or subscribes to a topic, the application can supply a user ID on the MQOPEN, MQPUT1, or MQSUB call. It can ask the queue manager to use this user ID for authority checks instead of the one associated with the application.

The application succeeds in opening the object only if both the following conditions are met:

- The user ID associated with the application has the authority to supply a different user ID for authority checks. The application is said to have *alternate user authority*.
- The user ID supplied by the application has the authority to open the object for the types of operation requested, or to subscribe to the topic.

Parent topic: Authority to work with WebSphere MQ objects

4.3. Message context

Message context information allows the application that retrieves a message to find out about the originator of the message. The information is held in fields in the message descriptor and the fields are divided into three logical parts

These parts are as follows:

identity context

These fields contain information about the user of the application that put the message on the queue.

origin context

These fields contain information about the application itself and when the message was put on the queue.

user context

These fields contain message properties that applications can use to select messages that the queue manager should deliver.

When an application puts a message on a queue, the application can ask the queue manager to generate the context information in the message. This is the default action. Alternatively, it can specify that the context fields are to contain no information. The user ID associated with an application requires no special authority to do either of these.

An application can set the identity context fields in a message, allowing the queue manager to generate the origin context, or it can set all the context fields. An application can also pass the identity context fields from a message it has retrieved to a message it is putting on a queue, or it can pass all the context fields. However, the user ID associated with an application requires authority to set or pass context information. An application specifies that it intends to set or pass context information when it opens the queue on which it is about to put messages, and its authority is checked at this time.

Here is a brief description of each of the context fields:

Identity context

UserIdentifier

The user ID associated with the application that put the message. If the queue manager sets this field, it is set to the user ID obtained from the operating system when the application connects to the queue manager.

AccountingToken

Information that can be used to charge for the work done as a result of the message.

ApplIdentityData

If the user ID associated with an application has authority to set the identity context fields, or to set all the context fields, the application can set this field to any value related to identity. If the queue manager sets this field, it is set to blank.

Origin context

PutApplType

The type of the application that put the message; a CICS® transaction, for example.

PutApplName

The name of the application that put the message.

PutDate

The date when the message was put.

PutTime

The time when the message was put.

ApplOriginData

If the user ID associated with an application has authority to set all the context fields, the application can set this field to any value related to origin. If the queue manager sets this field, it is set to blank.

User context

The following values are supported for **MQINQMP** or **MQSETMP**:

MQPD_USER _CONTEXT

The property is associated with the user context.

No special authorization is required to be able to set a property associated with the user context using the MQSETMP call.

On a V7.0 or subsequent queue manager, a property associated with the user context is saved as described for MQOO_SAVE_ALL_CONTEXT. An MQPUT with MQOO_PASS_ALL_CONTEXT specified causes the property to be copied from the saved context into the new message.

MQPD_NO_CONTEXT

The property is not associated with a message context.

An unrecognized value is rejected with MQRC_PD_ERROR. The initial value of this field is **MQPD_NO_CONTEXT**.

For a detailed description of each of the context fields, see MQMD – Message descriptor. For more information about how to use message context, see Message context.

Parent topic: Authority to work with WebSphere MQ objects

This build: January 26, 2011 11:20:52

Notices | Trademarks | Downloads | Library | Support | Feedback

4.4. Authority to work with WebSphere MQ objects on i5/OS, UNIX systems, and Windows systems

The authorization service component provided with WebSphere® MQ is called the *Object Authority Manager (OAM)*. It provides access control when an application issues an MQI call to access a queue manager, queue, process, topic, or namelist.

On i5/OS®, UNIX systems, and Windows systems, the *authorization service* provides the access control when an application issues an MQI call to access a WebSphere MQ object that is a queue manager, queue, process, topic, or namelist. This includes checks for alternate user authority and the authority to set or pass context information.

As with other versions of Windows, the OAM gives members of the Administrators group the authority to access all MQ objects even when UAC is enabled on Windows Vista and Windows Server 2008.

Additionally, on Windows systems, the SYSTEM account has full access to WebSphere MQ resources.

The authorization service also provides authority checks when a PCF command operates on one of these WebSphere MQ objects or an authentication information object. The equivalent MQSC command encapsulated within an Escape PCF command is treated in the same way.

On i5/OS, unless the user is a member of the QMQMADM group or has *ALLOBJ authority, the authorization service also provides authority checks when a user issues a CL command in Group 2 that operates on any of these WebSphere MQ objects or an authentication information object.

The authorization service is an *installable service*, which means that it is implemented by one or more *installable service components*. Each component is invoked using a documented interface. This enables users and vendors to provide components to augment or replace those provided by the WebSphere MQ products.

The authorization service component provided with WebSphere MQ is called the *Object Authority Manager (OAM)*. The OAM is automatically enabled for each queue manager you create.

The OAM maintains an access control list (ACL) for each WebSphere MQ object it is controlling access to. On UNIX systems, only group IDs can appear in an ACL. This means that all members of a group have the same authorities. On i5/OS and on Windows systems, both user IDs and group IDs can appear in an ACL. This means that authorities can be granted to individual users as well as to groups.

A 12 character limitation applies to both the group and the user ID. UNIX platforms generally restrict the length of a user ID to 12 characters. AIX® and Linux have 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".

The OAM can authenticate a user and change appropriate identity context fields. You enable this by specifying a connection security parameters structure (MQCSP) on an MQCONNX call. The structure is passed to the OAM Authenticate User function (MQZ_AUTHENTICATE_USER), which sets appropriate identity context fields. In the case of an MQCONNX connection from a WebSphere MQ client, the information in the MQCSP is flowed to the queue manager to which the client is connecting over the client-connection and server-connection channel. If security exits are defined on that channel, the MQCSP is passed into each security exit and can be altered by the exit. Security exits can also create the MQCSP. For more details of the use of security exits in this context, see “Channel-exit programs”, in the WebSphere MQ Intercommunication manual.

On UNIX and Windows systems, the control command **setmqaut** grants and revokes authorities and is used to maintain the ACLs. For example, the command:

```
setmqaut -m JUPITER -t queue -n MOON.EUROPA -g VOYAGER +browse +get
```

allows the members of the group VOYAGER to browse messages on the queue MOON.EUROPA that is owned by the queue manager JUPITER. It allows the members to get messages from the queue as well. To revoke these authorities later, enter the following command:

```
setmqaut -m JUPITER -t queue -n MOON.EUROPA -g VOYAGER -browse -get
```

The command:

```
    setmqaut -m JUPITER -t queue -n MOON.* -g VOYAGER +put
```

allows the members of the group VOYAGER to put messages on any queue whose name commences with the characters `MOON.` . MOON.* is the name of a generic profile. A *generic profile* allows you to grant authorities for a set of objects using a single **setmqaut** command. Objects whose names match the profile name do not have to exist when the **setmqaut** command is issued. Using generic profiles, therefore, allows you to grant authorities for objects that you might create in the future. For more information about the **setmqaut** command, see the WebSphere MQ System Administration Guide.

The control command **dspmqaut** is available to display the current authorities that a user or group has for a specified object. The control command **dmpmqaut** is also available to display the current authorities associated with generic profiles. For more information about the **dspmqaut** and **dmpmqaut** commands, see the WebSphere MQ System Administration Guide.

On i5/OS, an administrator uses the CL command GRTMQMAUT to grant authorities and the CL command RVKMQMAUT to revoke authorities. Generic profiles can be used as well. For example, the CL command:

```
    GRTMQMAUT MQMNAME(JUPITER) OBJTYPE(*Q) OBJ('MOON.*') USER(VOYAGER) AUT(*PUT)
```

provides the same function as the previous example of a **setmqaut** command; it allows the members of the group VOYAGER to put messages on any queue whose name commences with the characters `MOON.` .

The CL command DSPMQMAUT displays the current authorities that user or group has for a specified object. The CL commands WRKMQMAUT and WRKMQMAUTD are also available to work with the current authorities associated with objects and generic profiles.

If you do not want any authority checks, for example, in a test environment, you can disable the OAM.

For more information about the authority to work with WebSphere MQ objects, see:

- WebSphere MQ for i5/OS System Administration Guide
- WebSphere MQ System Administration Guide, for UNIX and Windows systems

Using PCF to access OAM commands
On i5/OS, UNIX, and Windows systems, you can use PCF commands to access OAM administration commands.

Parent topic: Authority to work with WebSphere MQ objects

4.4.1. Using PCF to access OAM commands

On i5/OS®, UNIX, and Windows systems, you can use PCF commands to access OAM administration commands.

The PCF commands and their equivalent OAM commands are as follows:

Table 1. PCF commands and their equivalent OAM commands

PCF command	OAM command
Inquire Authority Records	dmpmqaut
Inquire Entity Authority	dspmqaut
Set Authority Record	setmqaut
Delete Authority Record	setmqaut with -remove option

For more information about using these commands, see the WebSphere MQ Programmable Command Formats and Administration Interface book.

Parent topic: Authority to work with WebSphere MQ objects on i5/OS, UNIX systems, and Windows systems

4.5. Authority to work with WebSphere MQ objects on z/OS

On z/OS®, there are seven categories of authority check associated with calls to the MQI. You must define certain RACF® profiles and give appropriate access to these profiles. Use the *RESLEVEL* profile to control how many users IDs are checked.

The seven categories of authority check associated with calls to the MQI:

Connection security
: The authority checks that are performed when an application connects to a queue manager

Queue security
: The authority checks that are performed when an application opens a queue or deletes a permanent dynamic queue

Process security
: The authority checks that are performed when an application opens a process object

Namelist security
: The authority checks that are performed when an application opens a namelist object

Alternate user security
: The authority checks that are performed when an application requests alternate user authority when opening an object

Context security
: The authority checks that are performed when an application opens a queue and specifies that it intends to set or pass the context information in the messages it puts on the queue

Topic security
: The authority checks that are performed when an application opens a topic

Each category of authority check is implemented in the same way that command security and command resource security are implemented. You must define certain RACF profiles and give the necessary groups and user IDs access to these profiles at the required levels. For queue security, the level of access determines the types of operation the application can perform on a queue. For context security, the level of access determines whether the application can:

- Pass all the context fields
- Pass all the context fields and set the identity context fields
- Pass and set all the context fields

Each category of authority check can be turned on or off by defining switch profiles.

All the categories, except connection security, are known collectively as *API-resource security*.

By default, when an API-resource security check is performed as a result of an MQI call from an application using a batch connection, only one user ID is checked. When a check is performed as a result of an MQI call from a CICS® or IMS™ application, or from the channel initiator, two user IDs are checked.

By defining a *RESLEVEL profile*, however, you can control whether zero, one, or two users IDs are checked. The number of user IDs that are checked is determined by the user ID associated with the type of connection when an application connects to the queue manager and the access level that user ID has to the RESLEVEL profile. The user ID associated with each type of connection is:

- The user ID of the connecting task for batch connections
- The CICS address space user ID for CICS connections
- The IMS region address space user ID for IMS connections
- The channel initiator address space user ID for channel initiator connections

For more information about the authority to work with WebSphere® MQ objects on z/OS, see the WebSphere MQ for z/OS System Setup Guide.

Parent topic: Authority to work with WebSphere MQ objects

5. Channel security

User IDs associated with message channel agents need authority to access WebSphere® MQ resources. Different user IDs can be checked in different situations.

The user IDs associated with message channel agents (MCAs) need authority to access various WebSphere MQ resources.

An MCA must be able to connect to a queue manager and open the dead letter queue. If it is a sending MCA, it must be able to open the transmission queue for the channel. If it is a receiving MCA, it must be able to open destination queues and set context information in the messages it puts on those queues.

For channels to which the PUTAUT parameter applies, if it is set to CTX (or ALTMCA on z/OS®) in the channel definition at the receiving end of a channel, the user ID in the *UserIdentifier* field in the message descriptor of each incoming message needs authority to open the destination queue for the message. In addition, the user ID associated with the receiving MCA needs alternate user authority to open the destination queue using the authority of a different user ID. Using PUTAUT (CTX) is recommended to maintain best control of who can deliver messages to queues from a remote system.

For a SVRCONN channel on distributed platforms, its MCAUserIdentifier is used to check authority for all MQI calls it issues on behalf of its client program. The MCAUserIdentifier is subject to change at runtime as per the rules described at MCAUserIdentifier.

The user ID that is used for authority checks depends on whether the MCA is connecting to a queue manager or accessing queue manager resources after it has connected to a queue manager:

The user ID for connecting to a queue manager

On i5/OS®, UNIX systems, and Windows systems, the user ID whose authority is checked when an MCA connects to a queue manager is the one under which the MCA is running. This is known as the *default user ID* of the MCA. The default user ID might be derived in various ways. Here are some examples:

- If a caller MCA is started by a channel initiator, the MCA runs under the same user ID as that of the channel initiator. This user ID might be derived in various ways. For example, if the channel initiator is started by using the WebSphere MQ Explorer, it runs under the MUSER_MQADMIN user ID. This user ID is created when you install WebSphere MQ for Windows and is a member of the mqm group.
- If a responder MCA is started by a WebSphere MQ listener, the MCA runs under the same user ID as that of the listener.
- If the communications protocol for the channel is TCP/IP and a responder MCA is started by the inet daemon, the MCA runs under the user ID obtained from the entry in the inetd.conf file that was used to start the MCA.
- If the communications protocol for the channel is SNA LU 6.2, a responder MCA might run under the user ID contained in the inbound attach request, or under the user ID specified in the transaction program (TP) definition for the MCA.

After an MCA has connected to a queue manager, it accesses certain queue manager resources as part of its initialization processing. The default user ID of the MCA is also used for the authority checks when it opens these resources. To enable the MCA to access these resources, you must ensure that the default user ID is a member of the QMQMADM group on i5/OS, the mqm group on UNIX and Windows systems, or the Administrators group on Windows systems.

On z/OS, every task in the channel initiator address space that needs to connect to the queue manager does so when the channel initiator address space is started. This includes the dispatcher tasks that run as MCAs. The channel initiator address space user ID is used to check the authority of a task to connect to the queue manager.

The user ID for subsequent authority checks

The user ID whose authority is checked when the MCA accesses queue manager resources subsequently might be different from the one that was checked when the MCA connected to the queue manager. In addition, on z/OS, zero, one, or two user IDs might be checked, depending on the access level of the channel initiator address space user ID to the RESLEVEL profile. Here are some examples of other user IDs that might be used:

- The value of the MCAUSER parameter in the channel definition
- For a channel to which the PUTAUT parameter applies, if PUTAUT is set to CTX (or, on z/OS only, ALTMCA), the user ID in the *UserIdentifier* field in the message descriptor of each incoming message
- For a SVRCONN channel, the user ID that is received from a client system when a WebSphere MQ client application issues an MQCONN call

The user ID used is displayed on the channel status.

On z/OS, the channel initiator address space user ID needs authority to open certain system queues, such as SYSTEM.CHANNEL.INITQ, independently of the MCAs that are running in the address space.

For more information about channel security, see:

- WebSphere MQ for i5/OS System Administration Guide
- WebSphere MQ System Administration Guide, for UNIX and Windows systems
- WebSphere MQ for z/OS System Setup Guide
- WebSphere MQ Clients, for MQI channels

Parent topic: Security

6. WebSphere MQ support for SSL and TLS

WebSphere® MQ supports both the Secure Sockets Layer (SSL) protocol and the Transport Layer Security (TLS) protocol.

For more information about the SSL and TLS protocols, refer to the related information.

WebSphere MQ provides the following support for SSL Version 3.0 and TLS 1.0:

i5/OS®
SSL and TLS support is integral to the i5/OS operating system.

Java and JMS clients
These clients use the JVM to provide SSL and TLS support.

Windows and UNIX systems
For all the UNIX and Windows systems, the SSL and TLS support is installed with WebSphere MQ.

z/OS®
SSL and TLS support is integral to the z/OS operating system. The SSL and TLS support on z/OS is known as *System SSL*.

For information about any prerequisites for WebSphere MQ SSL and TLS support, see WebSphere MQ requirements.

The following topics describe the provisions in WebSphere MQ that enable you to use and control the SSL support:

Channel attributes
Three attributes of the channel object determine how the channel uses TLS or SSL.

Channel status attributes
When you display the status of a channel, six attributes contain information related to TLS or SSL.

Queue manager attributes
Six queue manager attributes are relevant to WebSphere MQ TLS and SSL support.

The authentication information object (AUTHINFO)
WebSphere MQ SSL support includes a queue manager object called an authentication information object (AUTHINFO). This object holds information that allows WebSphere MQ to determine whether a certificate has been revoked.

The SSL key repository
A fully authenticated SSL connection requires a key repository (which can be known by different names on different platforms) at each end of the connection. The key repository contains digital certificates.

Federal Information Processing Standards (FIPS)
When cryptography is required on an SSL channel on Windows or UNIX, WebSphere MQ uses a cryptography package called IBM® Crypto for C (ICC). On all the Windows and UNIX platforms supported by WebSphere MQ Version 7.0, the ICC software has passed the Federal Information Processing Standards (FIPS) Cryptomodule Validation Program of the US National Institute of Standards and Technology, at level 140-2.

SSL and TLS on the WebSphere MQ client
WebSphere MQ supports SSL and TLS on clients. You can tailor the use of SSL or TLS in various ways.

Working with WebSphere MQ internet pass-thru (IPT)
Special considerations apply if you are using SSL with internet pass-thru.

Setting up communications for SSL or TLS on i5/OS
Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

Setting up communications for SSL or TLS on UNIX systems or Windows
Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

Setting up communications for SSL or TLS on z/OS
Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

Working with SSL or TLS on i5/OS
To use the WebSphere MQ TLS and SSL support for your i5/OS installation you must set up your communications to use cryptographic protocols.

Working with SSL or TLS on UNIX and Windows systems
This information describes how you set up and work with the Secure Sockets Layer (SSL) on UNIX and Windows systems.

Working with SSL or TLS on z/OS
This information describes how you set up and work with the Secure Sockets Layer (SSL) on z/OS.

Working with revoked certificates
Digital certificates can be revoked by Certification Authorities. You can check the revocation status of certificates using OCSP, or CRLs on LDAP servers, depending on platform.

Working with CipherSpecs
This collection of topics provides information about using CipherSpecs in WebSphere MQ.

WebSphere MQ rules for SSLPEER values
The SSLPEER attribute is used to check the Distinguished Name (DN) of the certificate from the peer queue manager or client at the other end of a WebSphere® MQ channel. WebSphere MQ uses certain rules when comparing these values

Understanding authentication failures
There are a number common reasons for authentication failures during the SSL handshake.

Parent topic: Security

Related concepts
Cryptographic security protocols: TLS and SSL

6.1. Alternative SSL and TLS support for Windows, UNIX, and Linux systems

By default, SSL and TLS support is provided by GSKit Version 7. You can choose to use GSKit Version 8, which has certain advantages.

The subcomponent that provides support for SSL and TLS on Windows, UNIX, and Linux systems is called GSKit. If you select SSL and TLS support when you install WebSphere® MQ V7.0.1, GSKit Version 7 is installed and run by default. Versions of WebSphere MQ V7.0.1 from Fix Pack 7.0.1.4 and later also contain an alternative, separate copy of GSKit, at Version 8. You can install and run this version instead of GSKit Version 7.

Why use GSKit Version 8?
By using GSKit v8, you can use SHA-2 hashing algorithms, secret key reset in all Java environments, and stricter FIPS certification. You can also receive error messages in languages other than English.

Both GSKit Version 8 and GSKit Version 7 on the same system
You can use both GSKit v8 and v7 in one WebSphere MQ system. Only one version can be used on each queue manager and client.

How to set up for GSKit Version 8
GSKit Version 8 packages are unpacked as part of the WebSphere MQ product media, and can then be separately installed. You must also configure your queue manager or client system to use GSKit Version 8 rather than GSKit Version 7.

Parent topic: WebSphere MQ support for SSL and TLS

6.1.1. Why use GSKit Version 8?

By using GSKit v8, you can use SHA-2 hashing algorithms, secret key reset in all Java environments, and stricter FIPS certification. You can also receive error messages in languages other than English.

Consider using GSKit Version 8 in the following circumstances:

To use SHA-2 algorithms

The following additional CipherSpecs are supported with GSKit Version 8:

- TLS_RSA_WITH_NULL_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256

As the names imply, these CipherSpecs all provide SHA-2 hashing. No GSKit Version 7 CipherSpecs support SHA-2.

To use secret key reset from a Java or JMS client

In a Java application, you can request that the SSL or TLS secret key is reset during a run by setting the sslResetCount field in the MQEnvironment class, or by setting environment property MQC.SSL_RESET_COUNT_PROPERTY in a Hashtable object. In a JMS application, you can request that the SSL or TLS secret key is reset during a run by setting the Reset count property in the Connection Factory. This property can be reset from an application by calling setSSLResetCount, or by an administrator by setting the Reset count (SSLRESETCOUNT) on the SSL page of the WebSphere MQ Explorer Connection Factory properties dialog.

However, in general, GSKit Version 7 is unable to perform SSL and TLS secret key resets when communicating with a WebSphere® MQ Java or JMS client application. There is an incompatibility between the Java JSSE2 secret key reset algorithm and the GSKit Version 7 SSL and TLS secret key reset algorithm.

The following WebSphere MQ Java and JMS client environments are unable to perform secret key resets when communicating with a WebSphere MQ server running GSKit Version 7 on a Windows, UNIX, or Linux platform:

- an HP or Sun V1.4.2 JDK
- any V1.4.2 JDK when using FIPS mode
- any V5.0 or later JDK

Therefore, only WebSphere MQ Java or JMS clients running on certain older Java environments can successfully reset the secret key when communicating with GSKit Version 7.

GSKit Version 8 supports secret key reset with all Java environments.

To enforce stricter FIPS 140-2 certification rules

You can specify in various ways that you want only cryptographic algorithms certified to FIPS 140-2 to be used. For example you can set SSLFIPS in MQSC, the MQSSLFIPS environment variable, or SSLFipsRequired in PCF.

With GSKit Version 8, the only CipherSpecs which can be used when FIPS processing is mandated are:

At TLS 1.0:

- TLS_RSA_WITH_3DES_EDE_CBC_SHA

At TLS 1.2:

- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256

To obtain improved and translated massages from gskcapicmd

Error reporting in gsk8capicmd is clearer than in gsk7capicmd. The gsk8capicmd messages are improved and are supplied in a range of national languages.

Parent topic: Alternative SSL and TLS support for Windows, UNIX, and Linux systems

6.1.2. Both GSKit Version 8 and GSKit Version 7 on the same system

You can use both GSKit v8 and v7 in one WebSphere® MQ system. Only one version can be used on each queue manager and client.

GSKit Version 8 is not a replacement for GSKit Version 7, it is an additional version. It is fully supported to install both GSKit Version 7 and GSKit Version 8 on the same system: they can be upgraded independently, without affecting each other.

Only one GSKit version can be used by each queue manager or client process at any one time. The GSKit version is configured by queue manager, and can also be configured by client. Different queue managers and clients on the same system can therefore use either GSKit Version 7 or GSKit Version 8 depending on their individual configuration.

Parent topic: Alternative SSL and TLS support for Windows, UNIX, and Linux systems

6.1.3. How to set up for GSKit Version 8

GSKit Version 8 packages are unpacked as part of the WebSphere® MQ product media, and can then be separately installed. You must also configure your queue manager or client system to use GSKit Version 8 rather than GSKit Version 7.

Installing GSKit Version 8

The GSKit Version 8 runtime packages can be found in the directory into which you expand your WebSphere MQ FixPack. The installation instructions depend on the installation platform and are documented in the quick beginnings guide for your platform.

Configuring to use GSKit V8

To make a WebSphere MQ queue manager use GSKit Version 8, set the “Alternative GSKit” value by using the WebSphere MQ Explorer. Alternatively, for WebSphere MQ on UNIX and Linux systems, you can set AltGSKit=YES in the SSL stanza of the configuration file qm.ini. On all systems, these changes take effect when the queue manager is started. Alternatively, you can make the changes effective immediately by issuing the MQSC command REFRESH SECURITY TYPE(SSL).

You can also use GSKit Version 8 on a WebSphere MQ client by setting AltGSKit=YES in the SSL stanza of the client configuration file (typically called mqclient.ini). Changes to the client configuration file take effect when a client process is started, or when all current SSL and TLS connections are ended and a new SSL or TLS connection is then started.

To reset SSL or TLS secret keys from client applications that use WebSphere MQ classes for Java, or for JMS, you must install and enable GSKit Version 8 on the remote queue manager.

Figure 1. Example of qm.ini with GSKit Version 8 enabled

```
ExitPath:
    ExitsDefaultPath=/var/mqm/exits/
    ExitsDefaultPath64=/var/mqm/exits64/

Log:
    LogPrimaryFiles=3
    LogSecondaryFiles=2
    LogFilePages=1024
    LogType=CIRCULAR
    LogBufferPages=0
    LogPath=/var/mqm/log/QMGR
    LogWriteIntegrity=TripleWrite

Service:
    Name=AuthorizationService
    EntryPoints=13

ServiceComponent:
    Service=AuthorizationService
    Name=MQSeries.UNIX.auth.service
    Module=/opt/mqm/lib/amqzfu
    ComponentDataSize=0
SSL:
    AltGSKit=YES
```

Parent topic: Alternative SSL and TLS support for Windows, UNIX, and Linux systems

Related information
Installing and uninstalling GSKit Version 8 on AIX
Installing and uninstalling GSKit Version 8 on HP-UX
Installing and uninstalling GSKit Version 8 on Linux
Installing and uninstalling GSKit Version 8 on Solaris
Installing and uninstalling GSKit Version 8 on Windows

6.2. Channel attributes

Three attributes of the channel object determine how the channel uses TLS or SSL.

WebSphere® MQ SSL or TLS support includes the following parameters on the DEFINE CHANNEL MQSC command:

SSLCIPH
The CipherSpec for the channel to use. For more information about the CipherSpecs that WebSphere MQ supports, refer to the related information.

The SSLCIPH parameter is mandatory if you want your channel to use SSL.

SSLPEER
The Distinguished Name pattern that WebSphere MQ uses to decide the entities from which messages are accepted. The SSLPEER pattern filters the Distinguished Names of the entities. For more information, refer to the related information.

SSLCAUTH
Whether the SSL or TLS server requires the corresponding client to send its digital certificate for authentication. For more information about mandatory client authentication, refer to the related information.

For more information about setting these parameters with the DEFINE CHANNEL MQSC command, see DEFINE CHANNEL.

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Working with CipherSpecs
Distinguished Names
WebSphere MQ rules for SSLPEER values
How SSL provides authentication, confidentiality, and integrity

6.3. Channel status attributes

When you display the status of a channel, six attributes contain information related to TLS or SSL.

WebSphere® MQ SSL or TLS support includes the following parameters on the DISPLAY CHSTATUS MQSC command:

SSLPEER
The Distinguished Name (DN) of the remote certificate.

SSLCERTI
Represents the full Distinguished Name (DN) of the issuer of the remote certificate. The "issuer" is the Certification Authority (CA) that issued the certificate.

SSLCERTU
Represents the Local UserId associated with the remote certificate. Supported on z/OS® only.

SSLRKEYS
Displays the number of SSL or TLS key resets successfully performed for this channel instance. The count of SSL or TLS key resets is reset when the channel instance is ended.

SSLKEYDA
Displays the date when the last SSL or TLS secret key reset was successfully issued for this channel instance. The date of the last SSL or TLS secret key reset is reset when the channel instance is ended.

SSLKEYTI
Displays the time when the last SSL or TLS secret key reset was successfully issued for this channel instance. The time of the last SSL or TLS secret key reset is reset when the channel instance is ended.

For more information about displaying these parameters with the DISPLAY CHSTATUS MQSC command, see DISPLAY CHSTATUS.

Parent topic: WebSphere MQ support for SSL and TLS

sy10940_

6.4. Queue manager attributes

Six queue manager attributes are relevant to WebSphere MQ TLS and SSL support.

WebSphere® MQ SSL or TLS support includes the following parameters on the ALTER QMGR MQSC command:

SSLKEYR

Sets a queue manager attribute, *SSLKeyRepository*, which holds the name of the SSL or TLS key repository.

SSLCRLNL

Sets a queue manager attribute, *SSLCRLNamelist*, which holds the name of a namelist of authentication information objects.

SSLCRYP

Sets a queue manager attribute, *SSLCryptoHardware*, which holds the name of the parameter string required to configure the cryptographic hardware present on the system. This parameter applies only to Windows and UNIX queue managers.

SSLTASKS

Sets a queue manager attribute, *SSLTasks*, which holds the number of server subtasks to use for processing SSL or TLS calls. If you use SSL or TLS channels you must have at least two of these tasks. This parameter applies only to z/OS® queue managers.

SSLRKEYC

Sets a numeric queue manager attribute called *SSLKeyResetCount*, the total 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 message channel agent.

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. These modules might, or might not, be FIPS-certified to a particular level. This depends on the hardware product in use. For more information about FIPS, see Federal Information Processing Standards (FIPS).This parameter applies only to Windows and UNIX queue managers.

For more information about setting these parameters with the ALTER QMGR MQSC command, see ALTER QMGR, which also describes when changes to the SSL queue manager attributes become effective.

On i5/OS®, you can also set the `SSLKEYR` and `SSLCRLNL` parameters with the CHGMQM command.

Parent topic: WebSphere MQ support for SSL and TLS

sy10950_

6.5. The authentication information object (AUTHINFO)

WebSphere® MQ SSL support includes a queue manager object called an authentication information object (AUTHINFO). This object holds information that allows WebSphere MQ to determine whether a certificate has been revoked.

An authentication information object with an AuthInfoType value of OCSP obtains the revocation status of a certificate from an OCSP responder. This is supported only in WebSphere MQ on UNIX systems and Windows, where it is the preferred method.

An authentication information object with an AuthInfoType value of CRLLDAP obtains Certificate Revocation List (CRL) information from an LDAP server.

If you change queue manager authentication objects of type CRLLDAP or OCSP, this can update the client channel definition table (CCDT) as described at Using a client channel definition table.

For more information about revoked certificates and working with authentication information objects, see Working with revoked certificates.

Parent topic: WebSphere MQ support for SSL and TLS

Related information
DEFINE AUTHINFO
ALTER AUTHINFO
DISPLAY AUTHINFO
Change, Copy, and Create Authentication Information Object
Inquire Authentication Information Object

sy10960_

6.6. The SSL key repository

A fully authenticated SSL connection requires a key repository (which can be known by different names on different platforms) at each end of the connection. The key repository contains digital certificates.

This information uses the general term *key repository* to describe the store for digital certificates and their associated private keys. The specific store names used on the platforms that support SSL are:

i5/OS®	certificate store
Windows and UNIX	key database file
z/OS®	key ring

For more information, refer to Digital certificates and Secure Sockets Layer (SSL) concepts.

A fully authenticated SSL connection requires a key repository at each end of the connection. The key repository contains:

- A number of CA certificates from various Certification Authorities that allow the queue manager or client to verify certificates that it receives from its partner at the remote end of the connection. Individual certificates might be in a certificate chain.

- One or more personal certificates received from a Certification Authority. You associate a separate personal certificate with *each* queue manager or WebSphere® MQ client. Personal certificates are essential on an SSL client if mutual authentication is required. If mutual authentication is not required, personal certificates are not needed on the SSL client.

The location of the key repository depends on the platform you are using:

i5/OS

On i5/OS, the key repository is a certificate store. The default system certificate store is located at `/QIBM/UserData/ICSS/Cert/Server/Default` in the integrated file system (IFS). On i5/OS, WebSphere MQ stores the password for the certificate store in a *password stash file*. For example, the stash file for queue manager `QM1` is `/QIBM/UserData/mqm/qmgrs/QM1/ssl/Stash.sth`.

Alternatively, you can specify that the i5/OS system certificate store is to be used instead. To do this you change the value of the queue manager's SSLKEYR attribute to *SYSTEM. This value indicates that the queue manager must use the system certificate store, and the queue manager is registered for use as an application with Digital Certificate Manager (DCM).

On i5/OS the certificate store also contains the private key for the queue manager.

For more information, see Locating the key repository for a queue manager on i5/OS.

Windows and UNIX

On Windows and UNIX systems the key repository is a key database file. The name of the key database file must have a file extension of `.kdb`. For example, on UNIX, the default key database file for queue manager `QM1` is `/var/mqm/qmgrs/QM1/ssl/key.kdb`. If WebSphere MQ is installed in the default location, the equivalent path on Windows is `C:\Program Files\IBM\WebSphere MQ\Qmgrs\QM1\ssl\key.kdb`.

On Windows and UNIX systems, each key database file has an associated password stash file. This file holds encrypted passwords that allow programs to access the key database. The password stash file must be in the same directory and have the same file stem as the key database, and must end with the suffix `.sth`, for example `/var/mqm/qmgrs/QM1/ssl/key.sth`

Note: On Windows and UNIX systems, PKCS #11 cryptographic hardware cards can contain the certificates and keys that are otherwise held in a key database file. When certificates and keys are held on PKCS #11 cards, WebSphere MQ still requires access to both a key database file and a password stash file.

On Windows and UNIX systems, the key database also contains the private key for the personal certificate associated with the queue manager or WebSphere MQ client.

z/OS

Certificates are held in a key ring in RACF®. Refer to Setting up a key repository on z/OS for more information about creating a key ring in RACF.

Other external security managers (ESMs) also use key rings for storing certificates.

On z/OS, private keys are managed by RACF.

Protecting WebSphere MQ client key repositories
The key repository for a WebSphere MQ client is a file on the client machine. Ensure that only the intended user can access the key repository file. This prevents an intruder or other unauthorized user copying the key repository file to another system, and then setting up an identical user ID on that system to impersonate the intended user.

Refreshing a key repository
Use the command REFRESH SECURITY TYPE(SSL) or the equivalent PCF command to refresh the copy of the key repository held in memory.

Resetting SSL secret keys
A secret key is generated during SSL handshake. The secret key can be periodically renegotiated.

Parent topic: WebSphere MQ support for SSL and TLS

6.6.1. Protecting WebSphere MQ client key repositories

The key repository for a WebSphere® MQ client is a file on the client machine. Ensure that only the intended user can access the key repository file. This prevents an intruder or other unauthorized user copying the key repository file to another system, and then setting up an identical user ID on that system to impersonate the intended user.

Parent topic: The SSL key repository

6.6.2. Refreshing a key repository

Use the command REFRESH SECURITY TYPE(SSL) or the equivalent PCF command to refresh the copy of the key repository held in memory.

You can refresh the copy of the key repository held in memory, without restarting the channel process, by using the MQSC command REFRESH SECURITY TYPE(SSL). This enables you to use an up-to-date version of the SSL key repository when you have added a new certificate, without having to stop the channel process.

On platforms other than z/OS®, the REFRESH SECURITY TYPE(SSL) command updates all SSL channels whether a refresh is required or not. On z/OS, if no refresh is required, REFRESH SECURITY TYPE(SSL) completes successfully and the channels are unaffected.

For more information about the REFRESH SECURITY TYPE(SSL) command, see REFRESH SECURITY.

You can also refresh the key repository using the PCF command Refresh Security (MQCMD_REFRESH_SECURITY). The SecurityType (MQSECTYPE_SSL) parameter refreshes the copy of the key repository held in memory, allowing updates to become effective once the command has completed successfully.

For more information about this command, see Refresh Security.

Parent topic: The SSL key repository

6.6.3. Resetting SSL secret keys

A secret key is generated during SSL handshake. The secret key can be periodically renegotiated.

During an SSL handshake a *secret key* is generated to encrypt data between the SSL client and SSL server. The secret key is used in a mathematical formula that is applied to the data to transform plaintext into unreadable ciphertext, and ciphertext into plaintext.

The secret key is generated from the random text sent as part of the handshake and is used to encrypt plaintext into ciphertext. The secret key is also used in the MAC (Message Authentication Code) algorithm, which is used to determine whether a message has been altered. See Message digests for more information.

If the secret key is discovered, the plaintext of a message could be deciphered from the ciphertext, or the message digest could be calculated, allowing messages to be altered without detection. Even for a complex algorithm, the plaintext can eventually be discovered by applying every possible mathematical transformation to the ciphertext. To minimize the amount of data that can be deciphered or altered if the secret key is broken, the secret key can be renegotiated periodically.

Once the secret key has been renegotiated, the previous secret key can no longer be used to decrypt data encrypted with the new secret key. The commands ALTER QMGR SSLRKEYC and DISPLAY QMGR SSLRKEYC are used to set the values used during key renegotiation. On i5/OS® and Java, you can use the CHGMQM SSLRSTCNT and DSPMQM commands.

Parent topic: The SSL key repository

Related information
ALTER QMGR
DISPLAY QMGR
Change Message Queue Manager (CHGMQM)
Display Message Queue Manager (DSPMQM)

6.7. Federal Information Processing Standards (FIPS)

When cryptography is required on an SSL channel on Windows or UNIX, WebSphere® MQ uses a cryptography package called IBM® Crypto for C (ICC). On all the Windows and UNIX platforms supported by WebSphere MQ Version 7.0, the ICC software has passed the Federal Information Processing Standards (FIPS) Cryptomodule Validation Program of the US National Institute of Standards and Technology, at level 140-2.

The FIPS 140-2 compliance of a WebSphere MQ SSL connection on UNIX systems and Windows is as follows:

- For all WebSphere MQ message channels (except SVRCONN and CLNTCONN channel types), the connection is FIPS-compliant if both the following conditions are met:
  - The installed GSkit ICC version has been certified FIPS 140-2 compliant on the installed operating system version and hardware architecture.
  - The queue manager's SSLFIPS attribute has been set to YES.
- For all WebSphere MQ client applications (except WebSphere MQ classes for Java and WebSphere MQ classes for JMS applications in client mode), the connection uses GSkit and is FIPS-compliant if both the following conditions are met:
  - The installed GSkit ICC version has been certified FIPS 140-2 compliant on the installed operating system version and hardware architecture.
  - SSLFIPS mode has been enabled on the client as described in the related topic.
- For WebSphere MQ classes for Java and WebSphere MQ classes for JMS applications using client mode, the connection uses the JRE's SSL implementation and is FIPS-compliant if both the following conditions are met:
  - The Java Runtime Environment used to run the application is FIPS-compliant on the installed operating system version and hardware architecture.
  - SSLFIPS mode has been enabled on the client as described in the related topic.

All supported AIX®, Linux, HP-UX, Solaris, and Windows platforms are FIPS 140-2 certified except as noted in the readme file included with each fix pack or refresh pack.

For SSL connections using GSkit, the component which is FIPS 140-2 certified is named *ICC*. It is the version of this component which determines GSkit FIPS compliance on any given platform. To determine the ICC version currently installed, run the gsk7ver command. Here is an example extract of the gsk7ver output relating to ICC:

```
   ICC
   ============
   @(#)CompanyName:      IBM Corporation
   @(#)LegalTrademarks:  IBM
   @(#)FileDescription:  IBM Crypto for C-language
   @(#)FileVersion:      1.4.5.0
   @(#)LegalCopyright:   Licensed Materials - Property of IBM
   @(#)                  ICC
   @(#)                  (C) Copyright IBM Corp. 2002-2005
   @(#)                  All Rights Reserved. US Government Users
   @(#)                  Restricted Rights - Use, duplication or
 disclosure
   @(#)                  restricted by GSA ADP Schedule Contract
 with IBM Corp.
   @(#)ProductName:      icc_1.4 (GoldCoast Build)
   @(#)ProductVersion:   1.4.5.0
   @(#)ProductInfo:      07/03/12.23:55:21.07/03/13.15:00:28
   @(#)CMVCInfo:         icc_1.4/icc1.4.0_051026
```

The NIST certification statement for GSkit ICC 1.4.5 (included in GSkit 7.0.4.11, applicable to WebSphere MQ Version 6.0.2.2 and later releases) can be found at the following address: http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2007.htm#775

Parent topic: WebSphere MQ support for SSL and TLS

Related information
Specifying that only FIPS-certified cryptography will be used
Enabling SSL in WebSphere MQ classes for Java
Using SSL with WebSphere MQ classes for JMS

6.8. SSL and TLS on the WebSphere MQ client

WebSphere® MQ supports SSL and TLS on clients. You can tailor the use of SSL or TLS in various ways.

WebSphere MQ provides SSL and TLS support for WebSphere MQ clients on Windows and UNIX systems. If you are using WebSphere MQ classes for Java or WebSphere MQ classes for JMS, see WebSphere MQ Using Java. The rest of this section does not apply to the Java or JMS environments.

You can specify the key repository for a WebSphere MQ client either with the MQSSLKEYR value in your WebSphere MQ client configuration file, or when your application makes an MQCONNX call. You have three options for specifying that a channel uses SSL:

- Using a channel definition table
- Using the SSL configuration options structure, MQSCO, on an MQCONNX call
- Using the Active Directory (on Windows systems)

You cannot use the MQSERVER environment variable to specify that a channel uses SSL.

You can continue to run your existing WebSphere MQ client applications without SSL, as long as SSL is not specified at the other end of the channel.

If changes are made on a client machine to the contents of the SSL Key Repository, the location of the SSL Key Repository, the Authentication Information, or the Cryptographic hardware parameters, you need to end all the SSL connections in order to reflect these changes in the client-connection channels that the application is using to connect to the queue manager. Once all the connections have ended, restart the SSL channels. All the new SSL settings are used. These settings are analogous to those refreshed by the REFRESH SECURITY TYPE(SSL) command on queue manager systems.

When your WebSphere MQ client runs on a Windows or UNIX system with cryptographic hardware, you configure that hardware with the MQSSLCRYP environment variable. This variable is equivalent to the SSLCRYP parameter on the ALTER QMGR MQSC command. Refer to Queue manager attributes for a description of the SSLCRYP parameter. If you use the GSK_PCS11 version of the SSLCRYP parameter, the PKCS #11 token label must be specified entirely in lower-case.

SSL secret key reset and FIPS are supported on WebSphere MQ clients. For more information, see Resetting SSL secret keys and Federal Information Processing Standards (FIPS).

See WebSphere MQ Clients for more information about the SSL support for WebSphere MQ clients.

Parent topic: WebSphere MQ support for SSL and TLS

6.9. Working with WebSphere MQ internet pass-thru (IPT)

Special considerations apply if you are using SSL with internet pass-thru.

For detailed information about IPT, refer to the WebSphere® MQ internet pass-thru SupportPac MS81.

When your WebSphere MQ system communicates with IPT, unless you are using SSLProxyMode in IPT, ensure that the CipherSpec used by WebSphere MQ matches the CipherSuite used by IPT:

- When IPT is acting as the SSL server and WebSphere MQ is connecting as the SSL client, the CipherSpec used by WebSphere MQ must correspond to a CipherSuite that is enabled in the relevant IPT key ring.
- When IPT is acting as the SSL client and is connecting to a WebSphere MQ SSL server, the IPT CipherSuite must match the CipherSpec defined on the receiving WebSphere MQ channel.

When you migrate from IPT to the integrated WebSphere MQ SSL support, you transfer the digital certificates from IPT Using iKeyman.

For more information about importing certificates, refer to the relevant section for your platform in Working with WebSphere MQ TLS and SSL support.

Parent topic: WebSphere MQ support for SSL and TLS

6.10. Setting up communications for SSL or TLS on i5/OS

Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

To set up your SSL installation you must define your channels to use SSL. You must also create and manage your digital certificates. On some operating systems, you can perform the tests with self–signed certificates. However, on i5/OS®, you must use personal certificates signed by a local CA.

For full information about creating and managing certificates, see Working with SSL or TLS on i5/OS.

This collection of topics introduces some of the tasks involved in setting up SSL communications, and provides step-by-step guidance on completing those tasks

You might also want to test SSL client authentication, which is an optional part of the SSL protocol. During the SSL handshake, the SSL client always obtains and validates a digital certificate from the SSL server. With the WebSphere MQ implementation, the SSL server always requests a certificate from the SSL client.

On i5/OS, the SSL client sends a certificate only if it has one labeled in the correct WebSphere MQ format, which is `ibmwebspheremq` followed by the name of your queue manager changed to lower case. For example, for `QM1`, `ibmwebspheremqqm1`.

WebSphere MQ uses the `ibmwebspheremq` prefix on a label to avoid confusion with certificates for other products. Ensure that you specify the entire certificate label in lower case.

The SSL server always validates the client certificate if one is sent. If the SSL client does not send a certificate, authentication fails only if the end of the channel acting as the SSL server is defined with either the SSLCAUTH parameter set to REQUIRED or an SSLPEER parameter value set. For more information, see Connecting a queue manager anonymously on i5/OS.

Using CA-signed certificates for mutual authentication on i5/OS
Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

Connecting a queue manager anonymously on i5/OS
Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

Parent topic: WebSphere MQ support for SSL and TLS

6.10.1. Using CA-signed certificates for mutual authentication on i5/OS

Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

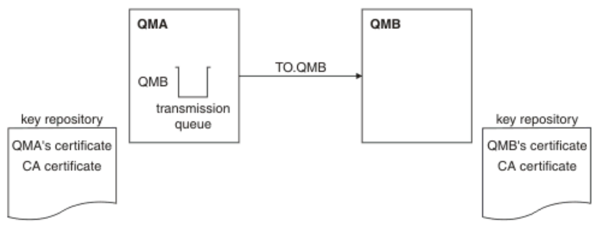
About this task

Scenario:

- You have two queue managers called QMA and QMB, which need to communicate securely. You require mutual authentication to be carried out between QMA and QMB.
- ›In the future you are planning to use this network in a production environment, and therefore you have decided to use CA-signed certificates from the beginning.‹

The resulting configuration looks like this:

Figure 1. Configuration resulting from this task



In Figure 1, the key repository for QMA contains QMA's certificate and the CA certificate. The key repository for QMB contains QMB's certificate and the CA certificate.

Procedure

1. Prepare the key repository on each queue manager, as described in Setting up a key repository on i5/OS.
2. Request a CA-signed certificate for each queue manager, as described in Requesting a server certificate on i5/OS.
3. Add the CA-signed certificate to the key repository, as described in Adding server certificates to a key repository on i5/OS.
4. Define a sender channel and associated transmission queue on queue manager QMA, as described in Defining a sender channel and transmission queue on QM1.
5. Define a receiver channel on queue manager QMB, as described in Defining a receiver channel on QM2.
6. Start the channel, as described in Starting the sender channel.

Results

Key repositories and channels are created as illustrated in Figure 1

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is like that shown in the following examples.

From queue manager QMA, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER SSLCERTI
```

```
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the queue manager QMB, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMA,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(RECEIVE)
   XMITQ( )
```

In each case, the value of SSLPEER must match that of the Distinguished Name (DN) in the partner certificate that was created in Step 2. The issuer name matches the subject DN of the CA certificate that signed the personal certificate added in Step 4.

Parent topic: Setting up communications for SSL or TLS on i5/OS

This build: January 26, 2011 11:21:44

Notices | Trademarks | Downloads | Library | Support | Feedback

6.10.1.1. Setting up a key repository on i5/OS

Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

An SSL connection requires a *key repository* at each end of the connection. Each queue manager must have access to a key repository. If you want to access the key repository using a file name and password (that is, not using the *SYSTEM option) ensure:

- the QMQM user profile has execute authority for the directory containing the key repository
- the QMQM user profile has read authority for the file containing the key repository

See The SSL key repository for more information.

On i5/OS®, digital certificates are stored in a certificate store that is managed with DCM. These digital certificates have labels, which associate a certificate with a queue manager. SSL uses the certificates for authentication purposes.

The queue manager certificate store name comprises a path and stem name. The default path is `/QIBM/UserData/ICSS/Cert/Server/` and the default stem name is `Default`. On i5/OS, the default certificate store, `/QIBM/UserData/ICSS/Cert/Server/Default.kdb`, is also known as `*SYSTEM`. Optionally, you can choose your own path and stem name.

If you choose your own path or filename, set the permissions to the file to tightly control access to it.

Changing the key repository location for a queue manager on i5/OS tells you about specifying the certificate store name. You can specify the certificate store name either before or after creating the certificate store.

Note: The operations you can perform with DCM might be limited by the authority of your user profile. For example, you require *ALLOBJ and *SECADM authorities to create a CA certificate.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Next topic: Requesting a server certificate on i5/OS

This build: January 26, 2011 11:21:06

Notices | Trademarks | Downloads | Library | Support | Feedback

6.10.1.2. Requesting a server certificate on i5/OS

Follow this procedure to create a certificate signed by your local CA, or to apply for a server certificate signed by a commercial CA.

About this task

Perform the following steps in a Web browser:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the certificate store you want to use and click **Continue**.
4. Optional: If you selected ***SYSTEM** in step 3, enter the system store password and click **Continue**.
5. Optional: If you selected **Other System Certificate Store** in step 3, in the **Certificate store path and filename** field, type the IFS path and file name you set when you created your certificate store and type a password in the **Certificate Store Password** field. Then click **Continue**
6. In the navigation panel, click **Create Certificate**.
7. In the task frame, select the **Server or client certificate** radio button and click **Continue**. The Select a Certificate Authority (CA) page is displayed in the task frame.
8. If you have a local CA on your machine you choose either the local CA or a commercial CA to sign the certificate. Select the radio button for the CA you want and click **Continue**. The Create a Certificate page is displayed in the task frame.
9. In the **Certificate label** field, type `ibmwebspheremq` followed by the name of your queue manager folded to lowercase. For example, for queue manager QM1, type `ibmwebspheremqqm1`

10. Type appropriate values in the **Common Name** and **Organization** fields, and select a country. For the remaining optional fields, type the values you require.

Results

If you selected a commercial CA to sign your certificate, DCM creates a certificate request in PEM (Privacy-Enhanced Mail) format. Forward the request to your chosen CA.

If you selected the local CA to sign your certificate, DCM informs you that the certificate has been created in the certificate store and can be used.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Setting up a key repository on i5/OS
Next topic: Requesting a server certificate on i5/OS for IBM Key Manager

Related tasks
Creating a certificate store on i5/OS

6.10.1.3. Requesting a server certificate on i5/OS® for IBM Key Manager

Follow this procedure to create a certificate signed by your local certificate authority (CA), or to apply for a server certificate signed by a commercial CA for import into the IBM® Key Management (iKeyman) utility.

About this task

A user certificate must be used when the Digital Server Manager (DCM) serves as the certificate manager for WebSphere® MQ on multiple platforms. For personal certificates distributed to other platforms and for import into the iKeyman utility, perform the following steps in a Web browser:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation pane, click **Create Certificate**. The Create Certificate page is displayed in the task frame.
3. On the Create Certificate panel, select the **User certificate** radio button and click **Continue**. The Create User Certificate page is displayed.
4. On the Create User Certificate panel, complete the required fields under Certificate Information for **Organization name**, **State** or **province**, **Country** or **region**. Optionally, put values in the **Organization unit** and **Locality** or **city** fields. Click **Continue**. The **Common name** is automatically set to the user ID with which you are logged on to the iSeries® system.
5. On the next Create User Certificate panel, click **Install certificate** and click **Continue**. A message is displayed stating, `Your personal certificate has been installed. You should keep a backup copy of this certificate.`
6. Click **OK**.
7. Depending on the internet browser you used to access DCM, do the following steps:
   a. For Internet Explorer choose: **Tools>Internet Options>Content tab>Certificates button>Personal tab>**. Select the certificate and click **Export**.
   b. For Mozilla Firefox choose: **Tools>Options>Advanced>Encryption tab>View Certificates button>Your Certificates tab>**. Select the certificate and click **Backup**. Select the path and filename and click **OK**.
8. Transfer the exported certificate to the remote system using FTP in binary format.
9. Add the exported certificate from step 7 to the iKeyman utility in the key database.
   a. If the certificate was saved using Internet Explorer, use the instructions described in Importing from a Microsoft .pfx.
   b. If the certificate was saved using Mozilla Firefox, use the instructions described in Importing a personal certificate into a key repository.

   During the import, ensure that the label name of the personal certificate and the signer certificate are changed to reflect the naming convention which WebSphere MQ is expecting. For example, the personal certificate looks like *userid's CN id*. It must be changed to *ibmwebspheremqXXXXXX* where *XXXXXX* is the name of your queue manager in lowercase letters. The command line **gsk7cmd** and **gsk7icapcmd** do not work on the distributed platforms. The iKeyman utility must be used for importing the certificates as documented.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Requesting a server certificate on i5/OS
Next topic: Adding server certificates to a key repository on i5/OS

6.10.1.4. Adding server certificates to a key repository on i5/OS

Follow this procedure to add a requested certificate to the key repository.

About this task

After the CA sends you a new server certificate, you add it to the certificate store from which you generated the request. If the CA sends the certificate as part of an e-mail message, copy the certificate into a separate file.

Note:

- You do not need to perform this procedure if the server certificate is signed by your local CA.
- Before you import a server certificate in PKCS #12 format into DCM, you must first import the corresponding CA certificate.

Use the following procedure to receive a server certificate into the queue manager certificate store:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the **Manage Certificates** task category in the navigation panel, click **Import Certificate**. The Import Certificate page displays in the task frame.
3. Select the radio button for your certificate type and click **Continue**. Either the Import Server or Client Certificate page or the Import Certificate Authority (CA) Certificate page displays in the task frame.

4. In the **Import File** field, type the file name of the certificate you want to import and click **Continue**. DCM automatically determines the format of the file.
5. If the certificate is a **Server or client** certificate, type the password in the task frame and click **Continue**. DCM informs you that the certificate has been imported.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Requesting a server certificate on i5/OS for IBM Key Manager
Next topic: Defining a sender channel and transmission queue on QM1

6.10.1.5. Defining a sender channel and transmission queue on QM1

Use the **DEFINE CHANNEL** and **DEFINE QLOCAL** commands to set up the required objects.

Procedure

On QM1, issue commands like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QM1.MACH.COM) XMITQ(QM2) SSLCIPH(RC4_MD5_US) DESCR('Sender channel u

DEFINE QLOCAL(QM2) USAGE(XMITQ)
```

This example uses CipherSpec RC4_MD5. The CipherSpecs at each end of the channel must be the same.

Only the SSLCIPH parameter is mandatory if you want your channel to use SSL. See Working with CipherSpecs for information about the permitted values for the SSLCIPH parameter.

Results

A sender channel, QM1.TO.QM2, and a transmission queue, QM2, are created.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Adding server certificates to a key repository on i5/OS
Next topic: Defining a receiver channel on QM2

6.10.1.6. Defining a receiver channel on QM2

Use the **DEFINE CHANNEL** command to set up the required object.

Procedure

On QM2, issue a command like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(RC4_MD5_US)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using SSL from QM1 to QM2')
```

The channel must have the same name as the sender channel you defined in Defining a sender channel and transmission queue on QM1, and use the same CipherSpec.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Defining a sender channel and transmission queue on QM1
Next topic: Starting the sender channel

6.10.1.7. Starting the sender channel

If necessary, start a listener program and refresh security. Then start the channel using the **START CHANNEL** command.

Procedure

1. Optional: If you have not already done so, start a listener program on QM2. The listener program listens for incoming network requests and starts the receiver channel when it is needed. For information about how to start a listener, see Starting a channel listener
2. Optional: If any SSL channels have run previously, issue the command REFRESH SECURITY TYPE(SSL). This ensures that all the changes made to the key repository are available.
3. On QM1, start the channel, using the command `START CHANNEL(QM1.TO.QM2)`.

Results

The sender channel is started.

Parent topic: Using CA-signed certificates for mutual authentication on i5/OS
Previous topic: Defining a receiver channel on QM2

6.10.2. Connecting a queue manager anonymously on i5/OS®

Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

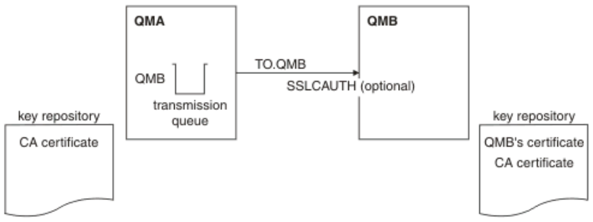
About this task

Scenario:

- Your two queue managers (QMA and QMB) have been set up as in Using CA-signed certificates for mutual authentication on UNIX systems or Windows.
- You want to change QMA so that it connects anonymously to QMB.

The resulting configuration looks like this:

Figure 1. Queue managers allowing anonymous connection



Procedure

1. Remove QMA's personal certificate from its key repository, as described in Removing certificates in i5/OS. The certificate is labeled `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`.
2. Optional: On QMA, if any SSL channels have run previously, refresh the SSL environment, as described in Refreshing the SSL environment.
3. Allow anonymous connections on the receiver, as described in Allowing anonymous connections on a receiver channel.

Results

Key repositories and channels are changed as illustrated in Figure 1

What to do next

If the sender channel was running and you issued the REFRESH SECURITY TYPE(SSL) command (in step 2), the channel restarts automatically. If the sender channel was not running, start it.

At the server end of the channel, the presence of the peer name parameter value on the channel status display indicates that a client certificate has flowed.

Verify that the task has been completed successfully by issuing some DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples:

From the QMA queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the QMB queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)                            SSLCERTI( )
   SSLPEER( )                              STATUS(RUNNING)
   SUBSTATE(RECEIVE)                       XMITQ( )
```

On QMB, the SSLPEER field is empty, showing that QMA did not send a certificate. On QMA, the value of SSLPEER matches that of the DN in QMB's personal certificate.

Parent topic: Setting up communications for SSL or TLS on i5/OS

6.10.2.1. Removing certificates in i5/OS

Use this procedure to remove personal certificates.

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the **Other System Certificate Store** check box and click **Continue**. The Certificate Store and Password page is displayed.

4. In the **Certificate store path and filename** field, type the IFS path and file name you set when you created the certificate store.
5. Type a password in the **Certificate Store Password** field. Click **Continue**. The Current Certificate Store page is displayed in the task frame.
6. In the **Manage Certificates** task category in the navigation panel, click **Delete Certificate**. The Confirm Delete Certificate page is displayed in the task frame.
7. Select the certificate you want to delete. Click **Delete**.
8. Click **Yes** to confirm that you want to delete the certificate. Otherwise, click **No**. DCM informs you if it has deleted the certificate.

Parent topic: Connecting a queue manager anonymously on i5/OS
Next topic: Refreshing the SSL environment

6.10.2.2. Refreshing the SSL environment

Refresh the SSL environment on queue manager QMA using the **REFRESH SECURITY** command.

Procedure

On QMA, enter the following command:

```
REFRESH SECURITY TYPE(SSL)
```

This ensures that all the changes made to the key repository are available.

Parent topic: Connecting a queue manager anonymously on i5/OS
Previous topic: Removing certificates in i5/OS
Next topic: Allowing anonymous connections on a receiver channel

6.10.2.3. Allowing anonymous connections on a receiver channel

Use the **ALTER CHANNEL** command to make SSL client authentication optional.

Procedure

On QMB, enter the following command:

```
ALTER CHANNEL(TO.QMB) CHLTYPE(RCVR) SSLCAUTH(OPTIONAL)
```

Parent topic: Connecting a queue manager anonymously on i5/OS
Previous topic: Refreshing the SSL environment

6.11. Setting up communications for SSL or TLS on UNIX systems or Windows

Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

To set up your SSL installation you must define your channels to use SSL. You must also create and manage your digital certificates. On UNIX systems and Windows, you can perform the tests with self–signed certificates.

Self-signed certificates cannot be revoked, which could allow an attacker to spoof an identity after a private key has been compromised. CAs can revoke a compromised certificate, which prevents its further use. CA-signed certificates are therefore safer to use in a production environment, though self-signed certificates are more convenient for a test system.

For full information about creating and managing certificates, see Working with SSL or TLS on UNIX and Windows systems.

This collection of topics introduces some of the tasks involved in setting up SSL communications, and provides step-by-step guidance on completing those tasks.

You might also want to test SSL client authentication, which is an optional part of the SSL protocol. During the SSL handshake, the SSL client always obtains and validates a digital certificate from the SSL server. With the WebSphere® MQ implementation, the SSL server always requests a certificate from the SSL client.

On UNIX systems and Windows, the SSL client sends a certificate only if it has one labeled in the correct WebSphere MQ format:

- For a queue manager, the format is `ibmwebspheremq` followed by the name of your queue manager changed to lower case. For example, for `QM1`, `ibmwebspheremqqm1`
- For a WebSphere MQ client, `ibmwebspheremq` followed by your logon user ID changed to lower case, for example `ibmwebspheremqmyuserid`.

WebSphere MQ uses the `ibmwebspheremq` prefix on a label to avoid confusion with certificates for other products. Ensure that you specify the entire certificate label in lower case.

The SSL server always validates the client certificate if one is sent. If the SSL client does not send a certificate, authentication fails only if the end of the channel acting as the SSL server is defined with either the SSLCAUTH parameter set to REQUIRED or an SSLPEER parameter value set. For more information, see Connecting a queue manager anonymously on UNIX systems or Windows.

Using self-signed certificates for mutual authentication on UNIX systems or Windows
Follow these sample instructions to implement mutual authentication between two queue managers, using self-signed SSL certificates.

Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

Connecting a queue manager anonymously on UNIX systems or Windows
Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Cryptographic security protocols: TLS and SSL

6.11.1. Using self-signed certificates for mutual authentication on UNIX systems or Windows

Follow these sample instructions to implement mutual authentication between two queue managers, using self-signed SSL certificates.

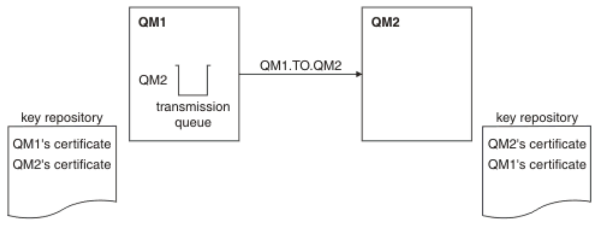
About this task

Scenario:

- You have two queue managers, QM1 and QM2, which need to communicate securely. You require mutual authentication to be carried out between QM1 and QM2.
- You have decided to test your secure communication using self-signed certificates.

The resulting configuration looks like this:

Figure 1. Configuration resulting from this task



In Figure 1, the key repository for QM1 contains QM1's certificate and the public certificate from QM2. The key repository for QM2 contains QM2's certificate and the public certificate from QM1.

Procedure

1. Prepare the key repository on each queue manager, as described in Setting up a key repository on UNIX and Windows systems.
2. Create a self-signed certificate for each queue manager, as described in Creating a self-signed personal certificate on UNIX or Windows systems.
3. Extract a copy of each certificate, as described in Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows.
   In order to authenticate a partner's certificate when using self-signed certificates, you must send a copy to the partner system. In order to send it, you must first extract it.
4. Optional: If QM1 and QM2 are running on different systems, transfer the public part of the QM1 certificate to the QM2 system and vice versa, as described in Exchanging self-signed certificates
5. Add the partner's certificate to the key repository as a signer certificate, as described in Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows.
6. On QM1, define a sender channel and transmission queue, as described in Defining a sender channel and transmission queue on QM1.
7. On QM2, define a receiver channel, as described in Defining a receiver channel on QM2.
8. Start the channel, as described in Starting the sender channel.

Results

Key repositories and channels are created as illustrated in Figure 1

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples.

From queue manager QM1, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
     4 : dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(QM1.TO.QM2)                     CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QM2)
   SSLCERTI(CN=QM2,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   SSLPEER(CN=QM2,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QM2)
```

From queue manager QM2, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
     5 : dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(QM2.TO.QM1)                      CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                      CURRENT
   RQMNAME(QM1)
   SSLCERTI(CN=QM1,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK
   SSLPEER(CN=QM1,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   STATUS(RUNNING)                          SUBSTATE(RECEIVE)
   XMITQ( )
```

In each case, the value of SSLPEER must match that of the DN in the partner certificate that was created in Step 2. The issuer's name matches the peer name because this is a self-signed certificate.

SSLPEER is optional. If it is specified, its value must be set so that the DN in the partner certificate (created in step 2) is allowed. For more information about the use of SSLPEER, see WebSphere MQ rules for SSLPEER values.

Parent topic: Setting up communications for SSL or TLS on UNIX systems or Windows

6.11.1.1. Setting up a key repository on UNIX and Windows systems

Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

An SSL connection requires a *key repository* at each end of the connection. Each WebSphere® MQ queue manager and WebSphere MQ client must have access to a key repository. See The SSL key repository for more information.

On UNIX and Windows systems, digital certificates are stored in a key database file that is managed with iKeyman, iKeycmd, or GSKCapiCmd. These digital certificates have labels. A specific label associates a personal certificate with a queue manager or WebSphere MQ client. SSL uses that certificate for authentication purposes. On UNIX and Windows systems, WebSphere MQ uses the `ibmwebspheremq` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager or WebSphere MQ client user logon ID, changed to lower case. Ensure that you specify the entire certificate label in lowercase.

The key database file name comprises a path and stem name:

- On UNIX, the default path for a queue manager (set when you create the queue manager) is `/var/mqm/qmgrs/<queue_manager_name>/ssl`.
  On Windows, the default path is *`install_directory`*`\Qmgrs\<queue_manager_name>\ssl`, where *install_directory* is the directory in which WebSphere MQ is installed. For example, `C:\Program Files\IBM\WebSphere MQ\Qmgrs\<queue_manager_name>\ssl` .
  The default stem name is `key`. Optionally, you can choose your own path and stem name, but the extension must be `.kdb`.
  If you choose your own path or filename, set the permissions to the file to tightly control access to it.
- For a WebSphere MQ client, there is no default path or stem name. Tightly control access to this file. The extension must be `.kdb`.

Note that key repositories must not be created on a file system that does not support file level locks, for example NFS version 2 on Linux.

Changing the key repository location for a queue manager on UNIX or Windows systems tells you about checking and specifying the key database file name. You can specify the key database file name either before or after creating the key database file.

The user ID from which you run iKeyman or iKeycmd must have write permission for the directory in which the key database file is created or updated. For a queue manager using the default SSL directory, the user ID from which you run iKeyman or iKeycmd must be a member of the mqm group. For a WebSphere MQ client, if you run iKeyman or iKeycmd from a user ID different from that under which the client runs, you must alter the file permissions to enable the WebSphere MQ client to access the key database file at run time. For more information, refer to Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using iKeyman

Use the following procedure to create a new key database file for either a queue manager or a WebSphere MQ client:

1. Start the iKeyman GUI (using the **gsk7ikm** command on UNIX, or the **strmqikm** command on Windows).
2. From the **Key Database File** menu, click New. The New window is displayed.
3. Click Key database type and select **CMS** (Certificate Management System).
4. In the **File Name** field, type a file name. This field already contains the text `key.kdb`. If your stem name is `key`, leave this field unchanged. If you have specified a different stem name, replace `key` with your stem name but you must not change the `.kdb`.
5. In the **Location** field, type the path, for example:
   - For a queue manager: `/var/mqm/qmgrs/QM1/ssl` (on UNIX) or `C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\ssl` (on Windows)
     The path must match the value of the SSLKeyRepository attribute of the queue manager.
   - For a WebSphere MQ client: `/var/mqm/ssl` (on UNIX) or `C:\mqm\ssl` (on Windows)
6. Click **Open**. The Password Prompt window displays.
7. Type a password in the **Password** field, and type it again in the **Confirm Password** field.
8. Select the **Stash the password to a file** check box.
   Note: If you do not stash the password, attempts to start SSL channels fail because they cannot obtain the password required to access the key database file.
9. Click **OK**. A window is displayed, confirming that the password is in file `key.sth` (unless you specified a different stem name).
10. Click **OK**. The Signer Certificates window is displayed, containing a list of the CA certificates that are provided with iKeyman and pre-installed in the key database.
11. Remove these CA certificates by selecting them and clicking **Delete**.
12. Set the access permissions, as described in Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using the command line

Use the following commands to create a new CMS key database file using iKeycmd or GSKCapiCmd:

- On UNIX:

  ```
  gsk7cmd -keydb -create -db filename -pw password -type cms -expire days
  ```

```
        -stash
```

- On Windows:

```
runmqckm -keydb -create -db filename -pw password -type cms -expire days
     -stash
```

- Using GSKCapiCmd:

```
gsk7capicmd -keydb -create -db filename -pw password -type cms -expire days
        -stash -fips -strong
```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database, and must have a file extension of `.kdb`.
`-pw` *password*	is the password for the CMS key database.
`-type` *cms*	is the type of database (for WebSphere MQ, this must be `cms`).
`-expire` *days*	is the expiration time in days of the database password. There is no default time for a database password: use the `-expire` option to set a database password expiration time explicitly.
`-stash`	tells iKeycmd or GSKCapiCmd to stash the key database password to a file.
`-fips`	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.
`-strong`	checks that the password entered satisfies the minimum requirements for password 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.

A number of CA certificates are pre-installed in the key database. Remove these using the methods described in Deleting a certificate from a key repository on UNIX systems or Windows

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Next topic: Creating a self-signed personal certificate on UNIX or Windows systems

6.11.1.2. Creating a self-signed personal certificate on UNIX or Windows systems

You can create a self-signed certificate using iKeyman, iKeycmd, or GSKCapiCmd.

Using iKeyman

When you create a key database, no personal certificates are provided. However, you need a personal certificate before you can run an SSL channel. A self-signed personal certificate can be used to run SSL channels for the purposes of testing SSL communications. These certificates can be created on either a WebSphere® MQ queue manager or WebSphere MQ client system.

Use the following procedure to obtain a self-signed certificate for your queue manager or WebSphere MQ client:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window displays.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file in which you want to save the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. From the **Create** menu, click **New Self-Signed Certificate**. The Create New Self-Signed Certificate window displays.
9. In the **Key Label** field, type:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or,
   - For a WebSphere MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
10. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining optional fields, either accept the default values, or type or select new values. Note that you can supply only one name in the **Organizational Unit** field. For more information about these fields, refer to Distinguished Names.
11. Click **OK**. The **Personal Certificates** list shows the label of the self-signed personal certificate you created.

Using the command line

Use the following commands to create a self-signed personal certificate using iKeycmd or GSKCapiCmd:

- On UNIX:

```
gsk7cmd -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
```

- On Windows:

```
runmqckm -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
```

- Using GSKCapiCmd:

```
gsk7capicmd -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
        -fips -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the key label attached to the certificate.
`-dn` *distinguished_name*	is the X.500 distinguished name enclosed in double quotes. Note that only the CN attribute is required. You can supply multiple OU attributes.
`-size` *key_size*	is the key size. For iKeycmd, the value can be 512 or 1024. For GSKCapiCmd, the value can be 512, 1024, or 2048.
`-x509version` *version*	is the version of X.509 certificate to create. The value can be 1, 2, or 3. The default is 3.
`-expire` *days*	is the expiration time in days of the certificate. The default is 365 days for a certificate.
`-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.
`-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 self-signed certificate. The value can be md5, sha1, sha224, sha256, sha384, or sha512. The default is sha1.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Setting up a key repository on UNIX and Windows systems
Next topic: Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows

6.11.1.3. Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows

Follow this procedure to extract the public part of a self-signed certificate.

Using iKeyman

Perform the following steps on the machine from which you want to extract the public part of a self-signed certificate:

1. Start the iKeyman GUI using either the gsk7ikm command (on UNIX) or the strmqikm command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file from which you want to extract the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates** and select the certificate.
9. Click **Extract certificate**. The Extract a Certificate to a File window opens.
10. Select the **Data type** of the certificate, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where you want to store the certificate, or click **Browse** to select the name and location.
12. Click **OK**. The certificate is written to the file you specified. Note that when you extract (rather than export) a certificate, only the public part of the certificate is included, so a password is not required.

Using the command line

Use the following commands to extract the public part of a self-signed certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
```
gsk7cmd -cert -extract -db filename -pw password -label label -target filename
        -format ascii
```
- On Windows:
```
runmqckm -cert -extract -db filename -pw password -label label -target filename
         -format ascii
```
- Using GSKCapiCmd:
```
gsk7capicmd -cert -extract -db filename -pw password -label label
            -target filename -format ascii -fips
```

where:

`-db` *filename*	is the fully qualified path name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the certificate.
`-target` *filename*	is the name of the destination file.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Creating a self-signed personal certificate on UNIX or Windows systems
Next topic: Exchanging self-signed certificates

6.11.1.4. Exchanging self-signed certificates

Exchange the certificates you previously extracted. If you use FTP, use the correct format.

Procedure

Transfer the CA part of the QM1 certificate to the QM2 system and vice versa, for example, by FTP.

If you transfer the certificates using FTP, you must do so in the correct format.

Transfer the following certificate types in *binary* format:

- DER encoded binary X.509
- PKCS #7 (CA certificates)
- PKCS #12 (personal certificates)

Transfer the following certificate types in ASCII format:

- PEM (privacy-enhanced mail)
- Base64 encoded X.509

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows
Next topic: Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows

6.11.1.5. Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows

Follow this procedure to add a CA certificate or the public part of a self-signed certificate to the key repository.

If the certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in strictly descending order starting from the root, followed by the CA certificate immediately below it in the chain, and so on.

Where the following instructions refer to a CA certificate, they also apply to the public part of a self-signed certificate.

Using iKeyman

Perform the following steps on the machine on which you want to add the CA certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Signer Certificates**.
9. Click **Add**. The Add CA's Certificate from a File window opens.
10. Select the **Data type** of the certificate you transferred, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where the certificate is stored, or click **Browse** to select the name and location.
12. Click **OK**. The Enter a Label window opens.
13. In the Enter a Label window, type the name of the certificate.
14. Click **OK**. The certificate is added to the key database.

Using the command line

Use the following commands to add a CA certificate using iKeycmd or GSKCapiCmd:

- On UNIX systems, issue the following command:
  ```
  gsk7cmd -cert -add -db filename -pw password -label label -file filename
          -format ascii
  ```
- On Windows, issue the following command:
  ```
  runmqckm -cert -add -db filename -pw password -label label -file filename
          -format ascii
  ```
- On UNIX systems or Windows, issue the following command:
  ```
  gsk7capicmd -cert -add -db filename -pw password -label label -file filename
          -format ascii -fips
  ```

where:

`-db filename`	is the fully qualified path name of the CMS key database.

`-pw` *`password`*	is the password for the CMS key database.
`-label` *`label`*	is the label attached to the certificate.
`-file` *`filename`*	is the name of the file containing the certificate.
`-format` *`ascii`*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Exchanging self-signed certificates
Next topic: Defining a sender channel and transmission queue on QM1

6.11.1.6. Defining a sender channel and transmission queue on QM1

Use the **DEFINE CHANNEL** and **DEFINE QLOCAL** commands to set up the required objects.

Procedure

On QM1, issue commands like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QM1.MACH.COM) XMITQ(QM2) SSLCIPH(RC4_MD5_US) DESCR('Sender channel u

DEFINE QLOCAL(QM2) USAGE(XMITQ)
```

This example uses CipherSpec RC4_MD5. The CipherSpecs at each end of the channel must be the same.

Only the SSLCIPH parameter is mandatory if you want your channel to use SSL. See Working with CipherSpecs for information about the permitted values for the SSLCIPH parameter.

Results

A sender channel, QM1.TO.QM2, and a transmission queue, QM2, are created.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows
Next topic: Defining a receiver channel on QM2

6.11.1.7. Defining a receiver channel on QM2

Use the **DEFINE CHANNEL** command to set up the required object.

Procedure

On QM2, issue a command like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(RC4_MD5_US)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using SSL from QM1 to QM2')
```

The channel must have the same name as the sender channel you defined in Defining a sender channel and transmission queue on QM1, and use the same CipherSpec.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Defining a sender channel and transmission queue on QM1
Next topic: Starting the sender channel

6.11.1.8. Starting the sender channel

If necessary, start a listener program and refresh security. Then start the channel using the **START CHANNEL** command.

Procedure

1. Optional: If you have not already done so, start a listener program on QM2. The listener program listens for incoming network requests and starts the receiver channel when it is needed. For information about how to start a listener, see Starting a channel listener
2. Optional: If any SSL channels have run previously, issue the command REFRESH SECURITY TYPE(SSL). This ensures that all the changes made to the key repository are available.
3. On QM1, start the channel, using the command `START CHANNEL(QM1.TO.QM2)`.

Results

The sender channel is started.

Parent topic: Using self-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Defining a receiver channel on QM2

Related concepts

Manipulating authentication information objects

Related information
SSL on WebSphere MQ clients

6.11.2. Using CA-signed certificates for mutual authentication on UNIX systems or Windows

Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

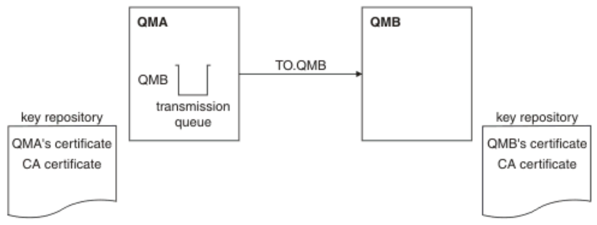
About this task

Scenario:

- You have two queue managers called QMA and QMB, which need to communicate securely. You require mutual authentication to be carried out between QMA and QMB.
- In the future you are planning to use this network in a production environment, and therefore you have decided to use CA-signed certificates from the beginning.

The resulting configuration looks like this:

Figure 1. Configuration resulting from this task



In Figure 1, the key repository for QMA contains QMA's certificate and the CA certificate. The key repository for QMB contains QMB's certificate and the CA certificate.

Procedure

1. Prepare the key repository on each queue manager, as described in Setting up a key repository on UNIX and Windows systems.
2. Request a CA-signed certificate for each queue manager, as described in Requesting a personal certificate on UNIX or Windows systems..
3. Add the Certificate Authority certificate to the key repository, as described in Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows.
4. Add the CA-signed certificate to the key repository, as described in Receiving personal certificates into a key repository on UNIX systems and Windows.
5. Define a sender channel and associated transmission queue on queue manager QMA, as described in Defining a sender channel and transmission queue on QMA.
6. Define a receiver channel on queue manager QMB, as described in Defining a receiver channel on QMB.
7. Start the channel, as described in Starting the sender channel.

Results

Key repositories and channels are created as illustrated in Figure 1

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is like that shown in the following examples.

From queue manager QMA, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the queue manager QMB, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMA,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(RECEIVE)
   XMITQ( )
```

In each case, the value of SSLPEER must match that of the Distinguished Name (DN) in the partner certificate that was created in Step 2. The issuer name matches the subject DN of the CA certificate that signed the personal certificate added in Step 4.

Extensions to the task of using CA-signed certificates
The use of CA-signed certificates makes it easier to add extra queue managers to your network, because it reduces the administration of certificates in your network.

Parent topic: Setting up communications for SSL or TLS on UNIX systems or Windows

6.11.2.1. Setting up a key repository on UNIX and Windows systems

Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

An SSL connection requires a *key repository* at each end of the connection. Each WebSphere® MQ queue manager and WebSphere MQ client must have access to a key repository. See The SSL key repository for more information.

On UNIX and Windows systems, digital certificates are stored in a key database file that is managed with iKeyman, iKeycmd, or GSKCapiCmd. These digital certificates have labels. A specific label associates a personal certificate with a queue manager or WebSphere MQ client. SSL uses that certificate for authentication purposes. On UNIX and Windows systems, WebSphere MQ uses the `ibmwebspheremq` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager or WebSphere MQ client user logon ID, changed to lower case. Ensure that you specify the entire certificate label in lowercase.

The key database file name comprises a path and stem name:

- On UNIX, the default path for a queue manager (set when you create the queue manager) is `/var/mqm/qmgrs/<queue_manager_name>/ssl`.
  On Windows, the default path is `*install_directory*\Qmgrs\<queue_manager_name>\ssl`, where *install_directory* is the directory in which WebSphere MQ is installed. For example, `C:\Program Files\IBM\WebSphere MQ\Qmgrs\<queue_manager_name>\ssl` .
  The default stem name is `key`. Optionally, you can choose your own path and stem name, but the extension must be `.kdb`.
  If you choose your own path or filename, set the permissions to the file to tightly control access to it.
- For a WebSphere MQ client, there is no default path or stem name. Tightly control access to this file. The extension must be `.kdb`.

Note that key repositories must not be created on a file system that does not support file level locks, for example NFS version 2 on Linux.

Changing the key repository location for a queue manager on UNIX or Windows systems tells you about checking and specifying the key database file name. You can specify the key database file name either before or after creating the key database file.

The user ID from which you run iKeyman or iKeycmd must have write permission for the directory in which the key database file is created or updated. For a queue manager using the default SSL directory, the user ID from which you run iKeyman or iKeycmd must be a member of the mqm group. For a WebSphere MQ client, if you run iKeyman or iKeycmd from a user ID different from that under which the client runs, you must alter the file permissions to enable the WebSphere MQ client to access the key database file at run time. For more information, refer to Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using iKeyman

Use the following procedure to create a new key database file for either a queue manager or a WebSphere MQ client:

1. Start the iKeyman GUI (using the **gsk7ikm** command on UNIX, or the **strmqikm** command on Windows).
2. From the **Key Database File** menu, click New. The New window is displayed.
3. Click Key database type and select **CMS** (Certificate Management System).
4. In the **File Name** field, type a file name. This field already contains the text `key.kdb`. If your stem name is `key`, leave this field unchanged. If you have specified a different stem name, replace `key` with your stem name but you must not change the `.kdb`.
5. In the **Location** field, type the path, for example:
   - For a queue manager: `/var/mqm/qmgrs/QM1/ssl` (on UNIX) or `C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\ssl` (on Windows)
     The path must match the value of the SSLKeyRepository attribute of the queue manager.
   - For a WebSphere MQ client: `/var/mqm/ssl` (on UNIX) or `C:\mqm\ssl` (on Windows)
6. Click **Open**. The Password Prompt window displays.
7. Type a password in the **Password** field, and type it again in the **Confirm Password** field.
8. Select the **Stash the password to a file** check box.
   Note: If you do not stash the password, attempts to start SSL channels fail because they cannot obtain the password required to access the key database file.
9. Click **OK**. A window is displayed, confirming that the password is in file `key.sth` (unless you specified a different stem name).
10. Click **OK**. The Signer Certificates window is displayed, containing a list of the CA certificates that are provided with iKeyman and pre-installed in the key database.
11. Remove these CA certificates by selecting them and clicking **Delete**.
12. Set the access permissions, as described in Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using the command line

Use the following commands to create a new CMS key database file using iKeycmd or GSKCapiCmd:

- On UNIX:
```
gsk7cmd -keydb -create -db filename -pw password -type cms -expire days
    -stash
```
- On Windows:
```
runmqckm -keydb -create -db filename -pw password -type cms -expire days
     -stash
```
- Using GSKCapiCmd:
```
gsk7capicmd -keydb -create -db filename -pw password -type cms -expire days
```

```
                -stash -fips -strong
```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database, and must have a file extension of `.kdb`.
`-pw` *password*	is the password for the CMS key database.
`-type` *cms*	is the type of database (for WebSphere MQ, this must be `cms`).
`-expire` *days*	is the expiration time in days of the database password. There is no default time for a database password: use the `-expire` option to set a database password expiration time explicitly.
`-stash`	tells iKeycmd or GSKCapiCmd to stash the key database password to a file.
`-fips`	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.
`-strong`	checks that the password entered satisfies the minimum requirements for password 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.

A number of CA certificates are pre-installed in the key database. Remove these using the methods described in Deleting a certificate from a key repository on UNIX systems or Windows

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Next topic: Requesting a personal certificate on UNIX or Windows systems.

This build: January 26, 2011 11:21:11

Notices | Trademarks | Downloads | Library | Support | Feedback

6.11.2.2. Requesting a personal certificate on UNIX or Windows systems.

You can request a personal certificate using iKeyman, iKeycmd or GSKCapiCmd.

Using iKeyman

To apply for a personal certificate, use the iKeyman tool as follows:

1. Start the iKeyman GUI using either the **gsk7ikm** command (UNIX) or the **strmqikm** command (Windows).
2. From the **Key Database File** menu, click **Open**. The Open window displays.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file from which you want to generate the request, for example `key.kdb`.
6. Click **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. From the **Create** menu, click **New Certificate Request**. The Create New Key and Certificate Request window displays.
9. In the **Key Label** field, type:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager changed to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or
   - For a WebSphere® MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
10. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining optional fields, either accept the default values, or type or select new values. Note that you can supply only one name in the **Organizational Unit** field. For more information about these fields, refer to Distinguished Names.
11. In the **Enter the name of a file in which to store the certificate request** field, either accept the default `certreq.arm`, or type a new value with a full path.
12. Click **OK**. A confirmation window displays.
13. Click **OK**. The **Personal Certificate Requests** list shows the label of the new personal certificate request you created. The certificate request is stored in the file you chose in step 11.
14. Request the new personal certificate either by sending the file to a Certification Authority (CA), or by copying the file into the request form on the Web site for the CA.

Using the command line

Use the following commands to request a personal certificate using iKeycmd or GSKCapiCmd:

- On UNIX, issue the following command:

```
gsk7cmd -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename
```

- On Windows, issue the following command:

```
runmqckm -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename
```

- On either UNIX or Windows, issue the following command:

```
gsk7capicmd -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename -fips
        -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

`-db` *filename*	The fully qualified file name of a CMS key database.
`-pw` *password*	The password for the CMS key database.
`-label` *label*	The key label attached to the certificate.
`-dn` *distinguished_name*	The X.500 distinguished name enclosed in double quotes. Note that only the CN attribute is required. You can supply multiple OU attributes.
`-size` *key_size*	The key size. If you are using iKeycmd, the value can be 512 or 1024. If you are using GSKCapiCmd, the value can be 512, 1024, or 2048.
`-file` *filename*	The file name for the 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.
`-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 request. The value can be `md5`, `sha1`, `sha224`, `sha256`, `sha384`, or `sha512`. The default is `sha1`.

If you are using cryptographic hardware, refer to Requesting a personal certificate for your PKCS #11 hardware.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Setting up a key repository on UNIX and Windows systems
Next topic: Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows

6.11.2.3. Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows

Follow this procedure to add a CA certificate or the public part of a self-signed certificate to the key repository.

If the certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in strictly descending order starting from the root, followed by the CA certificate immediately below it in the chain, and so on.

Where the following instructions refer to a CA certificate, they also apply to the public part of a self-signed certificate.

Using iKeyman

Perform the following steps on the machine on which you want to add the CA certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Signer Certificates**.
9. Click **Add**. The Add CA's Certificate from a File window opens.
10. Select the **Data type** of the certificate you transferred, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where the certificate is stored, or click **Browse** to select the name and location.
12. Click **OK**. The Enter a Label window opens.
13. In the Enter a Label window, type the name of the certificate.
14. Click **OK**. The certificate is added to the key database.

Using the command line

Use the following commands to add a CA certificate using iKeycmd or GSKCapiCmd:

- On UNIX systems, issue the following command:
```
gsk7cmd -cert -add -db filename -pw password -label label -file filename
        -format ascii
```
- On Windows, issue the following command:
```
runmqckm -cert -add -db filename -pw password -label label -file filename
        -format ascii
```
- On UNIX systems or Windows, issue the following command:
```
gsk7capicmd -cert -add -db filename -pw password -label label -file filename
        -format ascii -fips
```

where:

`-db` *filename*	is the fully qualified path name of the CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the certificate.
`-file` *filename*	is the name of the file containing the certificate.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.

`-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.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Requesting a personal certificate on UNIX or Windows systems.
Next topic: Receiving personal certificates into a key repository on UNIX systems and Windows

6.11.2.4. Receiving personal certificates into a key repository on UNIX systems and Windows

Use this procedure to receive a personal certificate into the key database file.

After the CA sends you a new personal certificate, you add it to the key database file from which you generated the new certificate request . If the CA sends the certificate as part of an e-mail message, copy the certificate into a separate file.

Using iKeyman

Ensure that the certificate file to be imported has write permission for the current user, and then use the following procedure for either a queue manager or a WebSphere® MQ client to receive a personal certificate into the key database file:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**, and then click **OK**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field. Select the **Personal Certificates** view.
8. Click **Receive**. The Receive Certificate from a File window opens.
9. Select the **Data type** of the new personal certificate, for example **Base64–encoded ASCII data** for a file with the `.arm` extension.
10. Type the certificate file name and location for the new personal certificate, or click **Browse** to select the name and location.
11. Click **OK**. If you already have a personal certificate in your key database, a window opens, asking if you want to set the key you are adding as the default key in the database.
12. Click **Yes** or **No**. The Enter a Label window opens.
13. Click **OK**. The **Personal Certificates** field shows the label of the new personal certificate you added.

Using the command line

Use the following commands to add a personal certificate to a key database file using iKeycmd or GSKCapiCmd:

- On UNIX systems, issue the following command:
```
gsk7cmd -cert -receive -file filename -db filename -pw password
        -format ascii
```
- On Windows, issue the following command:
```
runmqckm -cert -receive -file filename -db filename -pw password
        -format ascii
```
- On UNIX systems or Windows, issue the following command:
```
gsk7capicmd -cert -receive -file filename -db filename -pw password -fips
```

where:

`-file` *`filename`*	is the fully qualified file name of the file containing the personal certificate.
`-db` *`filename`*	is the fully qualified file name of a CMS key database.
`-pw` *`password`*	is the password for the CMS key database.
`-format` *`ascii`*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

If you are using cryptographic hardware, refer to Importing a personal certificate to your PKCS #11 hardware.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows
Next topic: Defining a sender channel and transmission queue on QMA

6.11.2.5. Defining a sender channel and transmission queue on QMA

Use the **DEFINE CHANNEL** and **DEFINE QLOCAL** commands to set up the required objects.

Procedure

On QMA, issue commands like the following example:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QMB.MACH.COM) XMITQ(QMB)
SSLCIPH(RC2_MD5_EXPORT) DESCR('Sender channel using SSL from QMA to QMB')

DEFINE QLOCAL(QMB) USAGE(XMITQ)
```

Results

A sender channel, TO.QMB, and a transmission queue, QMB, are created.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Receiving personal certificates into a key repository on UNIX systems and Windows
Next topic: Defining a receiver channel on QMB

6.11.2.6. Defining a receiver channel on QMB

Use the **DEFINE CHANNEL** command to set up the required object.

Procedure

On QMB, issue a command like the following example:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(RC2_MD5_EXPORT)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using SSL to QMB')
```

Results

A receiver channel, TO.QMB, is created.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Defining a sender channel and transmission queue on QMA
Next topic: Starting the sender channel

6.11.2.7. Starting the sender channel

If necessary, start a listener program and refresh security. Then start the channel using the **START CHANNEL** command.

Procedure

1. Optional: If you have not already done so, start a listener program on QMB. The listener program listens for incoming network requests and starts the receiver channel when it is needed. For information about how to start a listener, see Starting a channel listener.
2. Optional: If any SSL channels have run previously, issue the command `REFRESH SECURITY TYPE(SSL)`. This ensures that all the changes made to the key repository are available.
3. Start the channel on QMA, using the command `START CHANNEL(TO.QMB)`.

Results

The sender channel is started.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows
Previous topic: Defining a receiver channel on QMB

6.11.2.8. Extensions to the task of using CA-signed certificates

The use of CA-signed certificates makes it easier to add extra queue managers to your network, because it reduces the administration of certificates in your network.

Table 1 compares the number of certificates that need to be installed in each queue manager's key repository to be able to communicate with all the other queue managers, when using self-signed certificates and when using CA-signed certificates.

The administration of certificates includes the copying of these certificates from system to system as well as adding them to key repositories. Table 1 shows that, as your network grows, the number of certificates that must be copied into each queue manager's key repository increases when you use self-signed certificates. When you use CA-signed certificates however, the number of certificates remains the same, making the administration much simpler.

Table 1. Total number of certificates in each queue manager's key repository, both CA certificates and personal certificates, when using each scheme.

Number of queue managers in network	Using self-signed certificates	Using CA-signed certificates
2	2	2
3	3	2
4	4	2
5	5	2

You can extend this task by adding a third queue manager called QMC. QMC's key repository will contain its own certificate. The CA-signed certificate and appropriate channels can be defined to communicate with QMB, for example on QMC issue:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QMB.MACH.COM) XMITQ(QMB)
SSLCIPH(RC2_MD5_EXPORT) DESCR('Sender channel using SSL from QMC to QMB')
```

The same CipherSpec must be used on the sender channels at QMA and QMC, if generic receiver definitions are used at queue manager QMB, because the CipherSpec must match on both ends of each channel.

Parent topic: Using CA-signed certificates for mutual authentication on UNIX systems or Windows

6.11.3. Connecting a queue manager anonymously on UNIX systems or Windows

Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

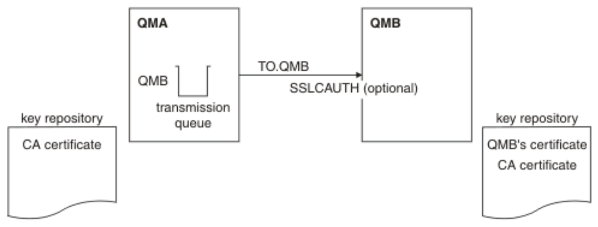
About this task

Scenario:

- Your two queue managers (QMA and QMB) have been set up as in Using CA-signed certificates for mutual authentication on UNIX systems or Windows.
- You want to change QMA so that it connects anonymously to QMB.

The resulting configuration looks like this:

Figure 1. Queue managers allowing anonymous connection



Procedure

1. Remove QMA's personal certificate from its key repository, as described in Deleting a certificate from a key repository on UNIX systems or Windows. The certificate is labeled as follows:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`.
   - For a WebSphere® MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
2. Optional: On QMA, if any SSL channels have run previously, refresh the SSL environment, as described in Refreshing the SSL environment.
3. Allow anonymous connections on the receiver, as described in Allowing anonymous connections on a receiver channel.

Results

Key repositories and channels are changed as illustrated in Figure 1

What to do next

If the sender channel was running and you issued the REFRESH SECURITY TYPE(SSL) command (in step 2), the channel restarts automatically. If the sender channel was not running, start it.

At the server end of the channel, the presence of the peer name parameter value on the channel status display indicates that a client certificate has flowed.

Verify that the task has been completed successfully by issuing some DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples:

From the QMA queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the QMB queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output will be similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)                            SSLCERTI( )
```

```
        SSLPEER( )                                STATUS(RUNNING)
        SUBSTATE(RECEIVE)                         XMITQ( )
```

On QMB, the SSLPEER field is empty, showing that QMA did not send a certificate. On QMA, the value of SSLPEER matches that of the DN in QMB's personal certificate.

Extensions to the task of connecting a queue manager anonymously
You can set up failure scenarios to familiarize yourself with what happens when the required certificates are not present.

Parent topic: Setting up communications for SSL or TLS on UNIX systems or Windows

6.11.3.1. Deleting a certificate from a key repository on UNIX systems or Windows

Use this procedure to remove personal or CA certificates.

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. ›From the drop down list, select **Personal Certificates** or **Signer Certificates**‹
9. ›Select the certificate you want to delete.‹
10. If you do not already have a copy of the certificate and you want to save it, click **Export/Import** and export it (see Exporting a personal certificate from a key repository).
11. With the certificate selected, click **Delete**. The Confirm window opens.
12. Click **Yes**. The **Personal Certificates** field no longer shows the label of the certificate you deleted.

Use the following commands to delete a certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
  ```
  gsk7cmd -cert -delete -db filename -pw password -label label
  ```
- On Windows:
  ```
  runmqckm -cert -delete -db filename -pw password -label label
  ```
- Using GSKCapiCmd:
  ```
  gsk7capicmd -cert -delete -db filename -pw password -label label -fips
  ```

where:

`-db filename`	is the fully qualified file name of a CMS key database.
`-pw password`	is the password for the CMS key database.
`-label label`	is the label attached to the personal certificate.
`-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.

Parent topic: Connecting a queue manager anonymously on UNIX systems or Windows
Next topic: Refreshing the SSL environment

6.11.3.2. Refreshing the SSL environment

Refresh the SSL environment on queue manager QMA using the **REFRESH SECURITY** command.

Procedure

On QMA, enter the following command:

```
REFRESH SECURITY TYPE(SSL)
```

This ensures that all the changes made to the key repository are available.
Parent topic: Connecting a queue manager anonymously on UNIX systems or Windows
Previous topic: Deleting a certificate from a key repository on UNIX systems or Windows
Next topic: Allowing anonymous connections on a receiver channel

6.11.3.3. Allowing anonymous connections on a receiver channel

Use the **ALTER CHANNEL** command to make SSL client authentication optional.

Procedure

On QMB, enter the following command:

```
ALTER CHANNEL(TO.QMB) CHLTYPE(RCVR) SSLCAUTH(OPTIONAL)
```

Parent topic: Connecting a queue manager anonymously on UNIX systems or Windows
Previous topic: Refreshing the SSL environment

6.11.3.4. Extensions to the task of connecting a queue manager anonymously

You can set up failure scenarios to familiarize yourself with what happens when the required certificates are not present.

In order to see the failure messages that are displayed when anonymous senders try to connect and the system is not set up to accept them, issue the following command on QMB:

```
ALTER CHANNEL(TO.QMB) CHLTYPE(RCVR) SSLCAUTH(REQUIRED)
```

and restart the sender on QMA.

Other failure scenarios could be tested by removing other certificates from the key repositories, and issuing

```
REFRESH SECURITY TYPE(SSL)
```

when the changes have been made. See Understanding authentication failures for information about the types of failures that might occur during an SSL handshake.
Parent topic: Connecting a queue manager anonymously on UNIX systems or Windows

6.12. Setting up communications for SSL or TLS on z/OS

Secure communications that use the SSL or TLS cryptographic security protocols involve setting up the communication channels and managing the digital certificates that you will use for authentication.

To set up your SSL installation you must define your channels to use SSL. You must also create and manage your digital certificates. On z/OS®, you can perform the tests with self–signed certificates, or with personal certificates signed by a local certificate authority (CA).

Self-signed certificates cannot be revoked, which could allow an attacker to spoof an identity after a private key has been compromised. CAs can revoke a compromised certificate, which prevents its further use. CA-signed certificates are therefore safer to use in a production environment, though self-signed certificates are more convenient for a test system.

For full information about creating and managing certificates, see Working with SSL or TLS on z/OS.

This collection of topics introduces some of the tasks involved in setting up SSL communications, and provides step-by-step guidance on completing those tasks.

You might also want to test SSL client authentication, which is an optional part of the SSL protocol. During the SSL handshake, the SSL client always obtains and validates a digital certificate from the SSL server. With the WebSphere® MQ implementation, the SSL server always requests a certificate from the SSL client.

On z/OS, the SSL client sends a certificate only if it has one of the following certificates:

- For a shared channel only, a certificate with a label in the format `ibmWebSphereMQ` followed by the name of your queue-sharing group, for example `ibmWebSphereMQQSG1`
- A certificate with a label in the format`ibmWebSphereMQ` followed by the name of your queue manager, for example `ibmWebSphereMQQM1`
- A default certificate (which might be the ibmWebSphereMQ certificate).

If the channel is shared, the channel first tries to find a certificate for the queue-sharing group. If it does not find a certificate for a queue-sharing group, it tries to find a certificate for the queue manager.

On z/OS, WebSphere MQ uses the `ibmWebSphereMQ` prefix on a label to avoid confusion with certificates for other products.

The SSL server always validates the client certificate if one is sent. If the SSL client does not send a certificate, authentication fails only if the end of the channel acting as the SSL server is defined with either the SSLCAUTH parameter set to REQUIRED or an SSLPEER parameter value set. For more information, see Connecting a queue manager anonymously on z/OS.

Using self-signed certificates for mutual authentication on z/OS
Follow these sample instructions to implement mutual authentication between two queue managers, using self-signed SSL certificates.

Using CA-signed certificates for mutual authentication on z/OS
Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

Connecting a queue manager anonymously on z/OS
Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

Parent topic: WebSphere MQ support for SSL and TLS

sy13090_

6.12.1. Using self-signed certificates for mutual authentication on z/OS

Follow these sample instructions to implement mutual authentication between two queue managers, using self-signed SSL certificates.

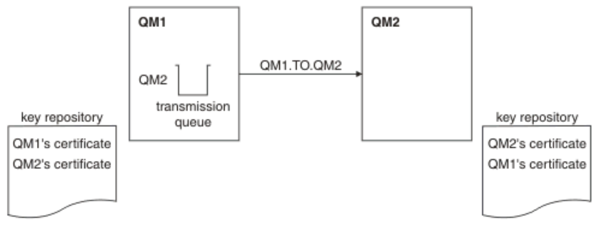
About this task

Scenario:

- You have two queue managers, QM1 and QM2, which need to communicate securely. You require mutual authentication to be carried out between QM1 and QM2.
- You have decided to test your secure communication using self-signed certificates.

The resulting configuration looks like this:

Figure 1. Configuration resulting from this task



In Figure 1, the key repository for QM1 contains QM1's certificate and the CA certificate from QM2. The key repository for QM2 contains QM2's certificate and the CA certificate from QM1.

Procedure

1. Prepare the key repository on each queue manager, as described in Setting up a key repository on z/OS.
2. Create a self-signed certificate for each queue manager, as described in Creating a self-signed personal certificate on z/OS.
3. On both QM1 and QM2, add the certificate created in step 2 to the key repository that was set up in step 1, as described in Adding personal certificates to a key repository on z/OS
4. Extract a copy of each certificate, as described in Exporting a personal certificate from a key repository on z/OS.
   In order to authenticate a partner's certificate when using self-signed certificates, you must send a copy to the partner system. In order to send it, you must first extract it.
5. Optional: If QM1 and QM2 are running on different systems, transfer the CA part of the QM1 certificate to the QM2 system and vice versa, as described in Exchanging self-signed certificates
6. Add the partner's certificate to the key repository by connecting the certificate to the key ring as described in Adding personal certificates to a key repository on z/OS.
7. On QM1, define a sender channel and transmission queue, as described in Defining a sender channel and transmission queue on QM1.
8. On QM2, define a receiver channel, as described in Defining a receiver channel on QM2.
9. Start the channel, as described in Starting the sender channel.

Results

Key repositories and channels are created as illustrated in Figure 1

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples.

From queue manager QM1, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
     4 : dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(QM1.TO.QM2)                      CHLTYPE(SDR)
   CONNAME(9.20.25.40)                      CURRENT
   RQMNAME(QM2)
   SSLCERTI(CN=QM2,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   SSLPEER(CN=QM2,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   STATUS(RUNNING)                          SUBSTATE(MQGET)
   XMITQ(QM2)
```

From queue manager QM2, enter the following command:

```
DISPLAY CHS(QM1.TO.QM2) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
     5 : dis chs(QM1.TO.QM2) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(QM2.TO.QM1)                      CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                      CURRENT
   RQMNAME(QM1)
   SSLCERTI(CN=QM1,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK
   SSLPEER(CN=QM1,OU="WebSphere MQ Development",O=IBM,ST=Hampshire,C=UK)
   STATUS(RUNNING)                          SUBSTATE(RECEIVE)
   XMITQ( )
```

In each case, the value of SSLPEER must match that of the DN in the partner certificate that was created in Step 2. The issuer's name matches the peer name because this is a self-signed certificate.

SSLPEER is optional. If it is specified, its value must be set so that the DN in the partner certificate (created in step 2) is allowed. For more information about the use of SSLPEER, see WebSphere MQ rules for SSLPEER values.

Parent topic: Setting up communications for SSL or TLS on z/OS

6.12.1.1. Setting up a key repository on z/OS

Set up a key repository at both ends of the connection. Associate each key repository with its queue manager.

An SSL connection requires a *key repository* at each end of the connection. Each queue manager must have access to a key repository. Use the SSLKEYR parameter on the ALTER QMGR command to associate a key repository with a queue manager. See The SSL key repository for more information.

On z/OS®, digital certificates are stored in a *key ring* that is managed by your External Security Manager (ESM) . These digital certificates have labels, which associate the certificate with a queue manager. SSL uses these certificates for authentication purposes. All the examples that follow use RACF® commands. Equivalent commands exist for other ESM programs.

On z/OS, WebSphere® MQ uses the `ibmWebSphereMQ` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager.

The key repository name for a queue manager is the name of a key ring in your RACF database. You can specify the key ring name either before or after creating the key ring.

Use the following procedure to create a new key ring for a queue manager:

1. Ensure that you have the appropriate authority to issue the RACDCERT command (see the *SecureWay™ Security Server RACF Command Language Reference* for more details).
2. Issue the following command:
   ```
   RACDCERT ID(userid1) ADDRING(ring-name)
   ```
   where:
   - *userid1* is the user ID of the channel initiator address space, or the user ID that is going to own the key ring (if the key ring is shared).
   - *ring-name* is the name you want to give to your key ring. The length of this name can be up to 237 characters. This name is case-sensitive. Specify *ring-name* in uppercase characters to avoid problems.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Next topic: Creating a self-signed personal certificate on z/OS

6.12.1.2. Creating a self-signed personal certificate on z/OS®

Use this procedure to create a self-signed personal certificate.

1. Generate a certificate and a public and private key pair using the following command:
   ```
   RACDCERT ID(userid2) GENCERT
   SUBJECTSDN(CN('common-name')
              T('title')
              OU('organizational-unit')
              O('organization')
              L('locality')
              SP('state-or-province')
              C('country'))
   WITHLABEL('label-name')
   ```
2. Connect the certificate to your key ring using the following command:
   ```
   RACDCERT ID(userid1)
   CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
   ```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *label-name* must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Note that *userid1* and *userid2* can be the same ID.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Setting up a key repository on z/OS
Next topic: Adding personal certificates to a key repository on z/OS

6.12.1.3. Adding personal certificates to a key repository on z/OS

Use this procedure to add or import a personal certificate to a key ring.

After the Certification Authority sends you a new personal certificate, add it to the key ring using the following procedure:

1. Add the certificate to the RACF® database using the following command:

```
        RACDCERT ID(userid2) ADD(input-data-set-name) WITHLABEL('label-name')
```

2. Connect the certificate to your key ring using the following command:

```
        RACDCERT ID(userid1)
        CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *input-data-set-name* is the name of the data set containing the CA signed certificate. The data set must be cataloged and must not be a PDS or a member of a PDS. The record format (RECFM) expected by RACDCERT is VB. RACDCERT dynamically allocates and opens the data set, and reads the certificate from it as binary data.
- *label-name* is the label name that was used when you created the original request. It must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Creating a self-signed personal certificate on z/OS
Next topic: Exporting a personal certificate from a key repository on z/OS

6.12.1.4. Exporting a personal certificate from a key repository on z/OS

Export the certificate using the RACDCERT command.

On the system from which you want to export the certificate, use the following command:

```
    RACDCERT ID(userid2) EXPORT(LABEL('label-name'))
    DSN(output-data-set-name) FORMAT(CERTB64)
```

where:

- *userid2* is the user ID under which the certificate was added to the key ring.
- *label-name* is the label of the certificate you want to extract.
- *output-data-set-name* is the data set into which the certificate is placed.
- CERTB64 is a DER encoded X.509 certificate that is in Base64 format. You can choose an alternative format, for example:

  CERTDER
    DER encoded X.509 certificate in binary format

  PKCS12B64
    PKCS #12 certificate in Base64 format

  PKCS12DER
    PKCS #12 certificate in binary format
    Note that **PKCS12DER** is supported only on OS/390® V2.10 and z/OS® V1.1 and subsequent releases.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Adding personal certificates to a key repository on z/OS
Next topic: Exchanging self-signed certificates

6.12.1.5. Exchanging self-signed certificates

Exchange the certificates you previously extracted. If you use FTP, use the correct format.

Procedure

Transfer the CA part of the QM1 certificate to the QM2 system and vice versa, for example, by FTP.

If you transfer the certificates using FTP, you must do so in the correct format.

Transfer the following certificate types in *binary* format:

- DER encoded binary X.509
- PKCS #7 (CA certificates)
- PKCS #12 (personal certificates)

Transfer the following certificate types in ASCII format:

- PEM (privacy-enhanced mail)
- Base64 encoded X.509

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Exporting a personal certificate from a key repository on z/OS
Next topic: Adding personal certificates to a key repository on z/OS

6.12.1.6. Adding personal certificates to a key repository on z/OS

Use this procedure to add or import a personal certificate to a key ring.

After the Certification Authority sends you a new personal certificate, add it to the key ring using the following procedure:

1. Add the certificate to the RACF® database using the following command:
   ```
   RACDCERT ID(userid2) ADD(input-data-set-name) WITHLABEL('label-name')
   ```
2. Connect the certificate to your key ring using the following command:
   ```
   RACDCERT ID(userid1)
   CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
   ```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *input-data-set-name* is the name of the data set containing the CA signed certificate. The data set must be cataloged and must not be a PDS or a member of a PDS. The record format (RECFM) expected by RACDCERT is VB. RACDCERT dynamically allocates and opens the data set, and reads the certificate from it as binary data.
- *label-name* is the label name that was used when you created the original request. It must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Exchanging self-signed certificates
Next topic: Defining a sender channel and transmission queue on QM1

6.12.1.7. Defining a sender channel and transmission queue on QM1

Use the **DEFINE CHANNEL** and **DEFINE QLOCAL** commands to set up the required objects.

Procedure

On QM1, issue commands like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QM1.MACH.COM) XMITQ(QM2) SSLCIPH(RC4_MD5_US) DESCR('Sender channel u

DEFINE QLOCAL(QM2) USAGE(XMITQ)
```

This example uses CipherSpec RC4_MD5. The CipherSpecs at each end of the channel must be the same.

Only the SSLCIPH parameter is mandatory if you want your channel to use SSL. See Working with CipherSpecs for information about the permitted values for the SSLCIPH parameter.

Results

A sender channel, QM1.TO.QM2, and a transmission queue, QM2, are created.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Adding personal certificates to a key repository on z/OS
Next topic: Defining a receiver channel on QM2

6.12.1.8. Defining a receiver channel on QM2

Use the **DEFINE CHANNEL** command to set up the required object.

Procedure

On QM2, issue a command like the following example:

```
DEFINE CHANNEL(QM1.TO.QM2) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(RC4_MD5_US)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using SSL from QM1 to QM2')
```

The channel must have the same name as the sender channel you defined in Defining a sender channel and transmission queue on QM1, and use the same CipherSpec.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Defining a sender channel and transmission queue on QM1
Next topic: Starting the sender channel

6.12.1.9. Starting the sender channel

If necessary, start the channel initiator, start a listener program, and refresh security. Then start the channel using the **START CHANNEL** command.

Procedure

1. Optional: if you have not already done so, start the channel initiator.
2. Optional: If you have not already done so, start a listener program on QM2. The listener program listens for incoming network requests and starts the receiver channel when it is needed. For information about how to start a listener, see Starting a channel listener
3. Optional: If the channel initiator was already running or any SSL channels have run previously, issue the command REFRESH SECURITY TYPE(SSL).

This ensures that all the changes made to the key repository are available.

4. On QM1, start the channel, using the command `START CHANNEL(QM1.TO.QM2)`.

Results

The sender channel is started.

Parent topic: Using self-signed certificates for mutual authentication on z/OS
Previous topic: Defining a receiver channel on QM2

6.12.2. Using CA-signed certificates for mutual authentication on z/OS®

Follow these sample instructions to implement mutual authentication between two queue managers, using CA-signed SSL certificates.

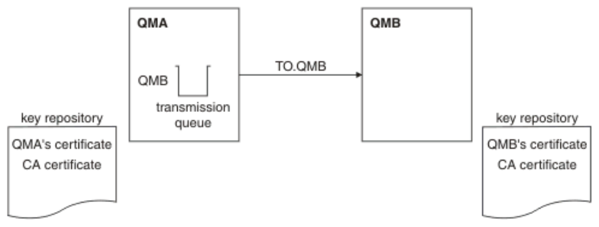
About this task

Scenario:

- You have two queue managers called QMA and QMB, which need to communicate securely. You require mutual authentication to be carried out between QMA and QMB.
- In the future you are planning to use this network in a production environment, and therefore you have decided to use CA-signed certificates from the beginning.

The resulting configuration looks like this:

Figure 1. Configuration resulting this task



In Figure 1, the key repository for QMA contains QMA's certificate and the CA certificate. The key repository for QMB contains QMB's certificate and the CA certificate.

Procedure

1. Prepare the key repository on each queue manager, as described in Setting up a key repository on z/OS.
2. Request a CA-signed certificate for each queue manager, as described in Requesting a personal certificate on z/OS.
3. Add the Certificate Authority certificate to the key repository, as described in Making CA certificates available to a queue manager on z/OS.
4. Add the CA-signed certificate to the key repository, as described in Adding personal certificates to a key repository on z/OS.
5. Define a sender channel and associated transmission queue on queue manager QMA, as described in Defining a sender channel and transmission queue on QMA.
6. Define a receiver channel on queue manager QMB, as described in Defining a receiver channel on QMB.
7. Start the channel, as described in Starting the sender channel on z/OS.

Results

Key repositories and channels are created as illustrated in Figure 1

What to do next

Check that the task has been completed successfully by using DISPLAY commands. If the task was successful, the resulting output is like that shown in the following examples.

From queue manager QMA, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the queue manager QMB, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is like the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
```

```
SSLPEER("CN=QMA,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
STATUS(RUNNING)                          SUBSTATE(RECEIVE)
XMITQ( )
```

In each case, the value of SSLPEER must match that of the Distinguished Name (DN) in the partner certificate that was created in Step 2. The issuer name matches the subject DN of the CA certificate that signed the personal certificate added in Step 4.

Extensions to the task of using CA-signed certificates
The use of CA-signed certificates makes it easier to add extra queue managers to your network, because it reduces the administration of certificates in your network.

Parent topic: Setting up communications for SSL or TLS on z/OS

6.12.2.1. Setting up a key repository on z/OS

Set up a key repository at both ends of the connection. Associate each key repository with its queue manager.

An SSL connection requires a *key repository* at each end of the connection. Each queue manager must have access to a key repository. Use the SSLKEYR parameter on the ALTER QMGR command to associate a key repository with a queue manager. See The SSL key repository for more information.

On z/OS®, digital certificates are stored in a *key ring* that is managed by your External Security Manager (ESM) . These digital certificates have labels, which associate the certificate with a queue manager. SSL uses these certificates for authentication purposes. All the examples that follow use RACF® commands. Equivalent commands exist for other ESM programs.

On z/OS, WebSphere® MQ uses the `ibmWebSphereMQ` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager.

The key repository name for a queue manager is the name of a key ring in your RACF database. You can specify the key ring name either before or after creating the key ring.

Use the following procedure to create a new key ring for a queue manager:

1. Ensure that you have the appropriate authority to issue the RACDCERT command (see the *SecureWay™ Security Server RACF Command Language Reference* for more details).
2. Issue the following command:
   ```
   RACDCERT ID(userid1) ADDRING(ring-name)
   ```
   where:
   - *userid1* is the user ID of the channel initiator address space, or the user ID that is going to own the key ring (if the key ring is shared).
   - *ring-name* is the name you want to give to your key ring. The length of this name can be up to 237 characters. This name is case-sensitive. Specify *ring-name* in uppercase characters to avoid problems.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Next topic: Requesting a personal certificate on z/OS

6.12.2.2. Requesting a personal certificate on z/OS®

Apply for a personal certificate using RACF®.

To apply for a personal certificate, use RACF as follows:

1. Create a self-signed personal certificate, as in Creating a self-signed personal certificate on z/OS. This certificate provides the request with the attribute values for the Distinguished Name.
2. Create a PKCS #10 Base64-encoded certificate request written to a data set, using the following command:
   ```
   RACDCERT ID(userid2) GENREQ(LABEL('label-name')) DSN(output-data-set-name)
   ```
   where *label-name* is the label used when creating the self-signed certificate, and *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
3. Send the data set to a Certification Authority (CA) to request a new personal certificate.
4. When the signed certificate is returned to you by the Certification Authority, add the certificate back into the RACF database, using the original label, as described in Adding personal certificates to a key repository on z/OS.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Setting up a key repository on z/OS
Next topic: Making CA certificates available to a queue manager on z/OS

6.12.2.3. Making CA certificates available to a queue manager on z/OS

After you have created your key ring, connect any relevant CA certificates to it.

For example, to connect a CA certificate for `My CA` to your key ring, use the following command:

```
RACDCERT ID(userid1)
CONNECT(CERTAUTH LABEL('My CA') RING(ring-name) USAGE(CERTAUTH))
```

where *userid1* is either the channel initiator user ID or the owner of a shared key ring.

For more information about CA certificates, refer to Digital certificates.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Requesting a personal certificate on z/OS
Next topic: Adding personal certificates to a key repository on z/OS

6.12.2.4. Adding personal certificates to a key repository on z/OS

Use this procedure to add or import a personal certificate to a key ring.

After the Certification Authority sends you a new personal certificate, add it to the key ring using the following procedure:

1. Add the certificate to the RACF® database using the following command:
   ```
   RACDCERT ID(userid2) ADD(input-data-set-name) WITHLABEL('label-name')
   ```
2. Connect the certificate to your key ring using the following command:
   ```
   RACDCERT ID(userid1)
   CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
   ```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *input-data-set-name* is the name of the data set containing the CA signed certificate. The data set must be cataloged and must not be a PDS or a member of a PDS. The record format (RECFM) expected by RACDCERT is VB. RACDCERT dynamically allocates and opens the data set, and reads the certificate from it as binary data.
- *label-name* is the label name that was used when you created the original request. It must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Making CA certificates available to a queue manager on z/OS
Next topic: Defining a sender channel and transmission queue on QMA

6.12.2.5. Defining a sender channel and transmission queue on QMA

Use the **DEFINE CHANNEL** and **DEFINE QLOCAL** commands to set up the required objects.

Procedure

On QMA, issue commands like the following example:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QMB.MACH.COM) XMITQ(QMB)
SSLCIPH(RC2_MD5_EXPORT) DESCR('Sender channel using SSL from QMA to QMB')

DEFINE QLOCAL(QMB) USAGE(XMITQ)
```

Results

A sender channel, TO.QMB, and a transmission queue, QMB, are created.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Adding personal certificates to a key repository on z/OS
Next topic: Defining a receiver channel on QMB

6.12.2.6. Defining a receiver channel on QMB

Use the **DEFINE CHANNEL** command to set up the required object.

Procedure

On QMB, issue a command like the following example:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(RCVR) TRPTYPE(TCP) SSLCIPH(RC2_MD5_EXPORT)
SSLCAUTH(REQUIRED) DESCR('Receiver channel using SSL to QMB')
```

Results

A receiver channel, TO.QMB, is created.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Defining a sender channel and transmission queue on QMA
Next topic: Starting the sender channel on z/OS

6.12.2.7. Starting the sender channel on z/OS®

If necessary, start the channel initiator, start a listener program, and refresh security. Then start the channel using the **START CHANNEL** command.

Procedure

1. Optional: If you have not already done so, start the channel initiator.
2. Optional: If you have not already done so, start a listener program on QMB. The listener program listens for incoming network requests and starts the receiver channel when it is needed. For information about how to start a listener, see Starting a channel listener.
3. Optional: If the channel initiator was already running or if any SSL channels have run previously, issue the command `REFRESH SECURITY TYPE(SSL)`. This ensures that all the changes made to the key repository are available.
4. Start the channel on QMA, using the command `START CHANNEL(TO.QMB)`.

Results

The sender channel is started.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS
Previous topic: Defining a receiver channel on QMB

6.12.2.8. Extensions to the task of using CA-signed certificates

The use of CA-signed certificates makes it easier to add extra queue managers to your network, because it reduces the administration of certificates in your network.

Table 1 compares the number of certificates that need to be installed in each queue manager's key repository to be able to communicate with all the other queue managers, when using self-signed certificates and when using CA-signed certificates.

The administration of certificates includes the copying of these certificates from system to system as well as adding them to key repositories. Table 1 shows that, as your network grows, the number of certificates that must be copied into each queue manager's key repository increases when you use self-signed certificates. When you use CA-signed certificates however, the number of certificates remains the same, making the administration much simpler.

Table 1. Total number of certificates in each queue manager's key repository, both CA certificates and personal certificates, when using each scheme.

Number of queue managers in network	Using self-signed certificates	Using CA-signed certificates
2	2	2
3	3	2
4	4	2
5	5	2

You can extend this task by adding a third queue manager called QMC. QMC's key repository will contain its own certificate. The CA-signed certificate and appropriate channels can be defined to communicate with QMB, for example on QMC issue:

```
DEFINE CHANNEL(TO.QMB) CHLTYPE(SDR) TRPTYPE(TCP) CONNAME(QMB.MACH.COM) XMITQ(QMB)
SSLCIPH(RC2_MD5_EXPORT) DESCR('Sender channel using SSL from QMC to QMB')
```

The same CipherSpec must be used on the sender channels at QMA and QMC, if generic receiver definitions are used at queue manager QMB, because the CipherSpec must match on both ends of each channel.

Parent topic: Using CA-signed certificates for mutual authentication on z/OS

6.12.3. Connecting a queue manager anonymously on z/OS

Follow these sample instructions to modify a system with mutual authentication to allow a queue manager to connect anonymously to another.

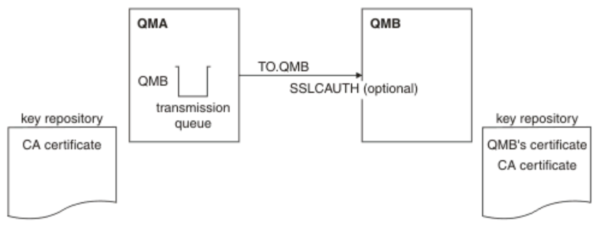
About this task

Scenario:

- Your two queue managers (QMA and QMB) have been set up as described in Using CA-signed certificates for mutual authentication on z/OS.
- You want to change QMA so that it connects anonymously to QMB.

The resulting configuration looks like this:

Figure 1. Queue managers allowing anonymous connection



Procedure

1. Remove the personal certificate for QMA and the default certificate from the QMA key repository, as described in Deleting a personal certificate from a key repository on z/OS. The certificates are labeled as follows:
   - `ibmWebSphereMQ` followed by the name of your queue manager or queue-sharing group. For example, for `QM1`, `ibmWebSphereMQQM1`.
   - The default certificate (which might be the `ibmWebSphereMQ` certificate).
2. Optional: On QMA, if the channel initiator was already running, or if any SSL channels have run previously, refresh the SSL environment, as described in Refreshing the SSL environment.
3. Allow anonymous connections on the receiver, as described in Allowing anonymous connections on a receiver channel.

Results

Key repositories and channels are changed as illustrated in Figure 1

What to do next

If the sender channel was running and you issued the REFRESH SECURITY TYPE(SSL) command (in step 2), the channel restarts automatically. If the sender channel was not running, start it.

At the server end of the channel, the presence of the peer name parameter value on the channel status display indicates that a client certificate has flowed.

Verify that the task has been completed successfully by issuing some DISPLAY commands. If the task was successful, the resulting output is similar to that shown in the following examples:

From the QMA queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     4 : dis chs(TO.QMB) SSLPEER
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(SDR)
   CONNAME(9.20.25.40)                     CURRENT
   RQMNAME(QMB)
   SSLCERTI("CN=WebSphere MQ CA,OU=WebSphere MQ Devt,O=IBM,ST=Hampshire,C=UK")
   SSLPEER("CN=QMB,OU=WebSphere MQ Development,O=IBM,ST=Hampshire,C=UK")
   STATUS(RUNNING)                         SUBSTATE(MQGET)
   XMITQ(QMB)
```

From the QMB queue manager, enter the following command:

```
DISPLAY CHS(TO.QMB) SSLPEER SSLCERTI
```

The resulting output is similar to the following example:

```
dis chs(TO.QMB) SSLPEER SSLCERTI
     5 : dis chs(TO.QMB) SSLPEER SSLCERTI
AMQ8417: Display Channel Status details.
   CHANNEL(TO.QMB)                         CHLTYPE(RCVR)
   CONNAME(9.20.35.92)                     CURRENT
   RQMNAME(QMA)                            SSLCERTI( )
   SSLPEER( )                              STATUS(RUNNING)
   SUBSTATE(RECEIVE)                       XMITQ( )
```

On QMB, the SSLPEER field is empty, showing that QMA did not send a certificate. On QMA, the value of SSLPEER matches that of the DN in the personal certificate of QMB.

Extensions to the task of connecting a queue manager anonymously
You can set up failure scenarios to familiarize yourself with what happens when the required certificates are not present.

Parent topic: Setting up communications for SSL or TLS on z/OS

6.12.3.1. Deleting a personal certificate from a key repository on z/OS

Delete a personal certificate using the RACDCERT command.

Before deleting a personal certificate, you might want to save a copy of it. To copy your personal certificate to a data set before deleting it, follow the procedure in Exporting a personal certificate from a key repository on z/OS. Then use the following command to delete your personal certificate:

```
RACDCERT ID(userid2) DELETE(LABEL('label-name'))
```

where:

- *userid2* is the user ID under which the certificate was added to the key ring.
- *label-name* is the name of the certificate you want to delete.

Parent topic: Connecting a queue manager anonymously on z/OS
Next topic: Refreshing the SSL environment

6.12.3.2. Refreshing the SSL environment

Refresh the SSL environment on queue manager QMA using the **REFRESH SECURITY** command.

Procedure

On QMA, enter the following command:

```
REFRESH SECURITY TYPE(SSL)
```

This ensures that all the changes made to the key repository are available.
Parent topic: Connecting a queue manager anonymously on z/OS
Previous topic: Deleting a personal certificate from a key repository on z/OS
Next topic: Allowing anonymous connections on a receiver channel

6.12.3.3. Allowing anonymous connections on a receiver channel

Use the **ALTER CHANNEL** command to make SSL client authentication optional.

Procedure

On QMB, enter the following command:

```
ALTER CHANNEL(TO.QMB) CHLTYPE(RCVR) SSLCAUTH(OPTIONAL)
```

Parent topic: Connecting a queue manager anonymously on z/OS
Previous topic: Refreshing the SSL environment

6.12.3.4. Extensions to the task of connecting a queue manager anonymously

You can set up failure scenarios to familiarize yourself with what happens when the required certificates are not present.

In order to see the failure messages that are displayed when anonymous senders try to connect and the system is not set up to accept them, issue the following command on QMB:

```
ALTER CHANNEL(TO.QMB) CHLTYPE(RCVR) SSLCAUTH(REQUIRED)
```

and restart the sender on QMA.

Other failure scenarios could be tested by removing other certificates from the key repositories, and issuing

```
REFRESH SECURITY TYPE(SSL)
```

when the changes have been made. See Understanding authentication failures for information about the types of failures that might occur during an SSL handshake.
Parent topic: Connecting a queue manager anonymously on z/OS

6.13. Working with SSL or TLS on i5/OS

To use the WebSphere® MQ TLS and SSL support for your i5/OS® installation you must set up your communications to use cryptographic protocols.

This collection of topics describes how you set up and work with the Secure Sockets Layer (SSL) on i5/OS.

For i5/OS, the SSL support is integral to the operating system. Ensure that you have installed the prerequisites listed in Checking hardware and software requirements.

On i5/OS, you manage keys and digital certificates with the Digital Certificate Manager (DCM) tool.

Digital Certificate Manager (DCM)
Use the DCM to manage digital certificates on i5/OS.

Assigning a certificate to a queue manager on i5/OS
Use DCM to assign a certificate to a queue manager.

Setting up a key repository on i5/OS
Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

Locating the key repository for a queue manager on i5/OS
Use this procedure to obtain the location of your queue manager's certificate store.

Changing the key repository location for a queue manager on i5/OS

Change the location of your queue manager's certificate store using either CHGMQM or ALTER QMGR.

Creating a certificate authority and certificate for testing on i5/OS
Use this procedure to create a local CA certificate to sign certificate requests, and to create and install the CA certificate.

Requesting a server certificate on i5/OS
Follow this procedure to create a certificate signed by your local CA, or to apply for a server certificate signed by a commercial CA.

Requesting a server certificate on i5/OS for IBM Key Manager
Follow this procedure to create a certificate signed by your local certificate authority (CA), or to apply for a server certificate signed by a commercial CA for import into the IBM® Key Management (iKeyman) utility.

Adding server certificates to a key repository on i5/OS
Follow this procedure to add a requested certificate to the key repository.

Exporting a certificate from a key repository on i5/OS
Follow this procedure to export a certificate.

Importing a certificate into a key repository on i5/OS
Follow this procedure to import a certificate.

Removing certificates in i5/OS
Use this procedure to remove personal certificates.

Using the *SYSTEM certificate store for one-way authentication on i5/OS
Follow these instructions to set up one-way authentication.

When changes to certificates or the certificate store become effective on i5/OS
When you change the certificates in a certificate store, or the location of the certificate store, the changes take effect depending on the type of channel and how the channel is running.

Configuring cryptographic hardware on i5/OS
Use this procedure to configure the 4758 PCI Cryptographic Coprocessor on i5/OS

Mapping DNs to user IDs
WebSphere MQ on i5/OS does not support the i5/OS function that is equivalent to the z/OS® CNFs. If you want to implement a function that maps Distinguished Names to user IDs, consider using a channel security exit.

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Cryptographic security protocols: TLS and SSL

6.13.1. Digital Certificate Manager (DCM)

Use the DCM to manage digital certificates on i5/OS.

The Digital Certificate Manager (DCM) enables you to manage digital certificates and to use them in secure applications on the i5/OS® server. With Digital Certificate Manager, you can request and process digital certificates from Certification Authorities (CAs) or other third-parties. You can also act as a local Certification Authority to create and manage digital certificates for your users.

DCM also supports using Certificate Revocation Lists (CRLs) to provide a stronger certificate and application validation process. You can use DCM to define the location where a specific Certificate Authority CRL resides on an LDAP server so that WebSphere® MQ can verify that a specific certificate has not been revoked.

DCM supports and can automatically detect certificates in the following formats: Base64, PKCS #7, PKCS #12 V1 and V,3 and the C3 encoded standard. When DCM detects a PKCS #12 encoded certificate, or a PKCS #7 certificate that contains encrypted data, it automatically prompts the user to enter the password that was used to encrypt the certificate. DCM does not prompt for PKCS #7 certificates that do not contain encrypted data.

DCM provides a browser-based user interface that you can use to manage digital certificates for your applications and users. The user interface is divided into two main frames: a navigation frame and a task frame.

You use the navigation frame to select the tasks to manage certificates or the applications that use them. Some individual tasks appear directly in the main navigation frame, but most tasks in the navigation frame are organized into categories. For example, Manage Certificates is a task category that contains various individual guided tasks, such as View certificate, Renew certificate, and Import certificate. If an item in the navigation frame is a category that contains more than one task, an arrow appears to the left of it. The arrow indicates that when you select the category link, an expanded list of tasks displays, enabling you to choose which task to perform.

For important information about DCM, see the following IBM® Redbooks® publications:

- *IBM i5/OS Wired Network Security: OS/400® V5R1 DCM and Cryptographic Enhancements*, SG24-6168. Specifically, see the appendixes for essential information about setting up your i5/OS system as a local CA.
- *AS/400® Internet Security: Developing a Digital Certificate Infrastructure*, SG24-5659. Specifically, see “Chapter 5. Digital Certificate Manager for AS/400”, which explains the AS/400 DCM.

Accessing DCM
Follow these instructions to access the DCM interface.

Parent topic: Working with SSL or TLS on i5/OS

sy11920_

6.13.1.1. Accessing DCM

Follow these instructions to access the DCM interface.

About this task

Perform the following steps in a web browser that supports frames.

Procedure

1. Go to either `http://machine.domain:2001` or `https://machine.domain:2010`, where *machine* is the name of your computer.
2. Type a valid user profile and password when requested to. Ensure that your user profile has *ALLOBJ and *SECADM special authorities to enable you to create new certificate stores. If you do not have the special authorities, you can only manage your personal certificates or view the object signatures for the objects for which you are authorized. If you are authorized to use an object signing application, you can also sign objects from DCM.
3. On the Internet Configurations page, click **Digital Certificate Manager**. The Digital Certificate Manager page is displayed.

Parent topic: Digital Certificate Manager (DCM)

6.13.2. Assigning a certificate to a queue manager on i5/OS

Use DCM to assign a certificate to a queue manager.

Use traditional i5/OS® digital certificate management to assign a certificate to a queue manager. This means that you can specify that a queue manager uses the system certificate store, and that the queue manager is registered for use as an application with Digital Certificate Manager. To do this you change the value of the queue manager's SSLKEYR attribute to *SYSTEM.

When the SSLKEYR parameter is changed to *SYSTEM, WebSphere® MQ registers the queue manager as a server application with a unique application label of QIBM_WEBSPHERE_MQ_QMGRNAME and a label with a description of Qmgrname (WMQ). The queue manager then appears as a server application in Digital Certificate Manager, and you can assign to this application any server or client certificate in the system store.

Because the queue manager is registered as an application, advanced features of DCM such as defining CA trust lists can be carried out.

If the SSLKEYR parameter is changed to a value other than *SYSTEM, WebSphere MQ deregisters the queue manager as an application with Digital Certificate Manager. If a queue manager is deleted, it is also deregistered from DCM. A user with sufficient *SECADM authority can also manually add or remove applications from DCM.

Parent topic: Working with SSL or TLS on i5/OS

6.13.3. Setting up a key repository on i5/OS

Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

An SSL connection requires a *key repository* at each end of the connection. Each queue manager must have access to a key repository. If you want to access the key repository using a file name and password (that is, not using the *SYSTEM option) ensure:

- the QMQM user profile has execute authority for the directory containing the key repository
- the QMQM user profile has read authority for the file containing the key repository

See The SSL key repository for more information.

On i5/OS®, digital certificates are stored in a certificate store that is managed with DCM. These digital certificates have labels, which associate a certificate with a queue manager. SSL uses the certificates for authentication purposes.

The queue manager certificate store name comprises a path and stem name. The default path is `/QIBM/UserData/ICSS/Cert/Server/` and the default stem name is `Default`. On i5/OS, the default certificate store, `/QIBM/UserData/ICSS/Cert/Server/Default.kdb`, is also known as `*SYSTEM`. Optionally, you can choose your own path and stem name.

If you choose your own path or filename, set the permissions to the file to tightly control access to it.

Changing the key repository location for a queue manager on i5/OS tells you about specifying the certificate store name. You can specify the certificate store name either before or after creating the certificate store.

Note: The operations you can perform with DCM might be limited by the authority of your user profile. For example, you require *ALLOBJ and *SECADM authorities to create a CA certificate.

Creating a certificate store on i5/OS
If you do not want to use the default certificate store, follow this procedure to create your own.

Stashing the certificate store password on i5/OS
Stash the certificate store password using CL commands.

Parent topic: Working with SSL or TLS on i5/OS

6.13.3.1. Creating a certificate store on i5/OS

If you do not want to use the default certificate store, follow this procedure to create your own.

About this task

Create a new certificate store only if you do not want to use the i5/OS® default certificate store.

To specify that the i5/OS system certificate store is to be used, change the value of the queue manager's SSLKEYR attribute to *SYSTEM. This value indicates that the queue manager uses the system certificate store, and the queue manager is registered for use as an application with Digital Certificate Manager (DCM).

Procedure

1. Access the DCM interface, as described in Accessing DCM
2. In the navigation panel, click **Create New Certificate Store**. The Create New Certificate Store page is displayed in the task frame.
3. In the task frame, select **Other System Certificate Store** and click **Continue**. The Create a Certificate in New Certificate Store page is displayed in the task frame.
4. Select **No - Do not create a certificate in the certificate store**and click **Continue**. The Certificate Store Name and Password page is displayed in the task frame.
5. In the **Certificate store path and filename** field, type an IFS path and file name, for example `/QIBM/UserData/mqm/qmgrs/qm1/key.kdb`
6. Type a password in the **Password** field and type it again in the **Confirm Password** field. Click **Continue**. A window opens, containing a list of the CA certificates that are preinstalled in the certificate store. This list includes the certificate for the local CA, if you have created one. Make a note of the password (which is case sensitive) because you need it when you stash the repository key.
7. To exit from DCM, close your browser window.

What to do next

When you have created the certificate store using DCM, ensure you stash the password, as described in Stashing the certificate store password on i5/OS

Parent topic: Setting up a key repository on i5/OS

6.13.3.2. Stashing the certificate store password on i5/OS

Stash the certificate store password using CL commands.

If you have specified that the system certificate store is to be used (by changing the value of the queue manager's SSLKEYR attribute to *SYSTEM) you do not need to follow the steps in this section.

When you have created the certificate store using DCM, use the following commands to stash the password:

```
STRMQM MQMNAME('queue manager name')

CHGMQM MQMNAME('queue manager name') SSLKEYRPWD('password')
```

The password must be entered as a literal (in single quotes) exactly as you entered it in 6 of Creating a certificate store on i5/OS (it is case sensitive).

Note: If you are not using the default system certificate store, and you do not stash the password, attempts to start SSL channels fail because they cannot obtain the password required to access the certificate store.

Parent topic: Setting up a key repository on i5/OS

6.13.4. Locating the key repository for a queue manager on i5/OS

Use this procedure to obtain the location of your queue manager's certificate store.

Procedure

1. Display your queue manager's attributes, using the following command:
   ```
   DSPMQM MQMNAME('queue manager name')
   ```
2. Examine the command output for the path and stem name of the certificate store. For example: `/QIBM/UserData/ICSS/Cert/Server/Default`, where `/QIBM/UserData/ICSS/Cert/Server` is the path and `Default` is the stem name.

Parent topic: Working with SSL or TLS on i5/OS

6.13.5. Changing the key repository location for a queue manager on i5/OS

Change the location of your queue manager's certificate store using either CHGMQM or ALTER QMGR.

Procedure

Use either the CHGMQM command or the ALTER QMGR MQSC command to set your queue manager's key repository attribute.

a. Using CHGMQM: `CHGMQM MQMNAME('qm1') SSLKEYR('/QIBM/UserData/ICSS/Cert/Server/MyKey')`

b. Using ALTER QMGR: `ALTER QMGR SSLKEYR('/QIBM/UserData/ICSS/Cert/Server/MyKey')`

In either case, the certificate store has the fully qualified file name: `/QIBM/UserData/ICSS/Cert/Server/MyKey.kdb`

What to do next

When you change the location of a queue manager's certificate store, certificates are not transferred from the old location. If the CA certificates preinstalled when you create the certificate store are insufficient, you must populate the new certificate store with certificates, as described in Importing a certificate into a key repository on i5/OS. You must also stash the password for the new location, as described in Stashing the certificate store password on i5/OS.

Parent topic: Working with SSL or TLS on i5/OS

6.13.6. Creating a certificate authority and certificate for testing on i5/OS

Use this procedure to create a local CA certificate to sign certificate requests, and to create and install the CA certificate.

Before you begin

The instructions in this topic assume that a local certificate authority (CA) does not exist. If a local CA does exist, go to Requesting a server certificate on i5/OS.

About this task

The CA certificates that are provided when you install SSL are signed by the issuing CA. On i5/OS®, you can generate a local certificate authority that can sign server certificates for testing SSL communications on your system. Follow these steps in a Web browser to create a local CA certificate:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Create a Certificate Authority**. The Create a Certificate Authority page is displayed in the task frame.
3. Type a password in the **Certificate store password** field and type it again in the **Confirm password** field.
4. Type a name in the **Certificate Authority (CA) name** field, for example `SSL Test Certificate Authority`.
5. Type appropriate values in the **Common Name** and **Organization** fields, and select a country. For the remaining optional fields, type the values you require.
6. Type a validity period for the local CA in the **Validity period** field. The default value is 1095 days.
7. Click **Continue**. The CA is created, and DCM creates a certificate store and a CA certificate for your local CA.
8. Click **Install certificate**. The download manager dialog box is displayed.
9. Type the full path name for the temporary file in which you want to store the CA certificate and click **Save**.
10. When download is complete, click **Open**. The Certificate window is displayed.
11. Click **Install certificate**. The Certificate Import wizard is displayed.
12. Click **Next**.
13. Select **Automatically select the certificate store based on the type of certificate** and click **Next**.
14. Click **Finish**. A confirmation window appears.
15. Click **OK**.
16. In the Certificate window, click **OK**.
17. Click **Continue**. The Certificate Authority Policy page is displayed in the task frame.
18. In the **Allow creation of user certificates** field, select **Yes**.
19. In the **Validity period** field, type the validity period of certificates that are issued by your local CA. The default value is 365 days.
20. Click **Continue**. The Create a Certificate in New Certificate Store page is displayed in the task frame.
21. Check that none of the applications are selected.
22. Click **Continue** to complete the setup of the local CA.

Parent topic: Working with SSL or TLS on i5/OS

6.13.7. Requesting a server certificate on i5/OS

Follow this procedure to create a certificate signed by your local CA, or to apply for a server certificate signed by a commercial CA.

About this task

Perform the following steps in a Web browser:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the certificate store you want to use and click **Continue**.
4. Optional: If you selected ***SYSTEM** in step 3, enter the system store password and click **Continue**.
5. Optional: If you selected **Other System Certificate Store** in step 3, in the **Certificate store path and filename** field, type the IFS path and file name you set when you created your certificate store and type a password in the **Certificate Store Password** field. Then click **Continue**
6. In the navigation panel, click **Create Certificate**.
7. In the task frame, select the **Server or client certificate** radio button and click **Continue**. The Select a Certificate Authority (CA) page is displayed in the task frame.

8. If you have a local CA on your machine you choose either the local CA or a commercial CA to sign the certificate. Select the radio button for the CA you want and click **Continue**. The Create a Certificate page is displayed in the task frame.
9. In the **Certificate label** field, type `ibmwebspheremq` followed by the name of your queue manager folded to lowercase. For example, for queue manager QM1, type `ibmwebspheremqqm1`
10. Type appropriate values in the **Common Name** and **Organization** fields, and select a country. For the remaining optional fields, type the values you require.

Results

If you selected a commercial CA to sign your certificate, DCM creates a certificate request in PEM (Privacy-Enhanced Mail) format. Forward the request to your chosen CA.

If you selected the local CA to sign your certificate, DCM informs you that the certificate has been created in the certificate store and can be used.

Parent topic: Working with SSL or TLS on i5/OS

Related tasks
Creating a certificate store on i5/OS

6.13.8. Requesting a server certificate on i5/OS for IBM Key Manager

Follow this procedure to create a certificate signed by your local certificate authority (CA), or to apply for a server certificate signed by a commercial CA for import into the IBM® Key Management (iKeyman) utility.

About this task

A user certificate must be used when the Digital Server Manager (DCM) serves as the certificate manager for WebSphere® MQ on multiple platforms. For personal certificates distributed to other platforms and for import into the iKeyman utility, perform the following steps in a Web browser:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation pane, click **Create Certificate**. The Create Certificate page is displayed in the task frame.
3. On the Create Certificate panel, select the **User certificate** radio button and click **Continue**. The Create User Certificate page is displayed.
4. On the Create User Certificate panel, complete the required fields under Certificate Information for **Organization name**, **State** or **province**, **Country** or **region**. Optionally, put values in the **Organization unit** and **Locality** or **city** fields. Click **Continue**. The **Common name** is automatically set to the user ID with which you are logged on to the iSeries® system.
5. On the next Create User Certificate panel, click **Install certificate** and click **Continue**. A message is displayed stating, `Your personal certificate has been installed. You should keep a backup copy of this certificate.`
6. Click **OK**.
7. Depending on the internet browser you used to access DCM, do the following steps:
   a. For Internet Explorer choose: **Tools>Internet Options>Content tab>Certificates button>Personal tab>**. Select the certificate and click **Export**.
   b. For Mozilla Firefox choose: **Tools>Options>Advanced>Encryption tab>View Certificates button>Your Certificates tab>**. Select the certificate and click **Backup**. Select the path and filename and click **OK**.
8. Transfer the exported certificate to the remote system using FTP in binary format.
9. Add the exported certificate from step 7 to the iKeyman utility in the key database.
   a. If the certificate was saved using Internet Explorer, use the instructions described in Importing from a Microsoft .pfx.
   b. If the certificate was saved using Mozilla Firefox, use the instructions described in Importing a personal certificate into a key repository.

   During the import, ensure that the label name of the personal certificate and the signer certificate are changed to reflect the naming convention which WebSphere MQ is expecting. For example, the personal certificate looks like *userid's CN id*. It must be changed to *ibmwebspheremqXXXXXX* where *XXXXXX* is the name of your queue manager in lowercase letters. The command line **gsk7cmd** and **gsk7icapcmd** do not work on the distributed platforms. The iKeyman utility must be used for importing the certificates as documented.

Parent topic: Working with SSL or TLS on i5/OS

6.13.9. Adding server certificates to a key repository on i5/OS

Follow this procedure to add a requested certificate to the key repository.

About this task

After the CA sends you a new server certificate, you add it to the certificate store from which you generated the request. If the CA sends the certificate as part of an e-mail message, copy the certificate into a separate file.

Note:

- You do not need to perform this procedure if the server certificate is signed by your local CA.
- Before you import a server certificate in PKCS #12 format into DCM, you must first import the corresponding CA certificate.

Use the following procedure to receive a server certificate into the queue manager certificate store:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the **Manage Certificates** task category in the navigation panel, click **Import Certificate**. The Import Certificate page displays in the task frame.
3. Select the radio button for your certificate type and click **Continue**. Either the Import Server or Client Certificate page or the Import Certificate

Authority (CA) Certificate page displays in the task frame.

4. In the **Import File** field, type the file name of the certificate you want to import and click **Continue**. DCM automatically determines the format of the file.
5. If the certificate is a **Server or client** certificate, type the password in the task frame and click **Continue**. DCM informs you that the certificate has been imported.

Parent topic: Working with SSL or TLS on i5/OS

6.13.10. Exporting a certificate from a key repository on i5/OS

Follow this procedure to export a certificate.

About this task

Perform the following steps on the computer from which you want to export the certificate:

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the certificate store you want to use and click **Continue**.
4. Optional: If you selected ***SYSTEM** in step 3, enter the system store password and click **Continue**.
5. Optional: If you selected **Other System Certificate Store** in step 3, in the **Certificate store path and filename** field, type the IFS path and file name you set when you created your certificate store and type a password in the **Certificate Store Password** field. Then click **Continue**
6. In the **Manage Certificates** task category in the navigation panel, click **Export Certificate**. The Export a Certificate page is displayed in the task frame.
7. Select the radio button for your certificate type and click **Continue**. Either the Export Server or Client Certificate page or the Export Certificate Authority (CA) Certificate page is displayed in the task frame.
8. Select the certificate you want to export.
9. Select the radio button to specify whether you want to export the certificate to a file or directly into another certificate store.
10. If you selected to export a server or client certificate to a file, provide the following information:
    - The path and file name of the location where you want to store the exported certificate.
    - For a personal certificate, the password that is used to encrypt the exported certificate and the target release. For CA certificates, you do not need to specify the password.
11. If you selected to export a certificate directly into another certificate store, specify the target certificate store and its password.
12. Click **Continue**.

Parent topic: Working with SSL or TLS on i5/OS

Related tasks
Creating a certificate store on i5/OS

6.13.11. Importing a certificate into a key repository on i5/OS

Follow this procedure to import a certificate.

> **Before you begin**
> Before you import a personal certificate in PKCS #12 format into DCM, you must first import the corresponding CA certificate.

About this task
Perform these steps on the machine to which you want to import the certificate.

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the certificate store you want to use and click **Continue**.
4. Optional: If you selected ***SYSTEM** in step 3, enter the system store password and click **Continue**.
5. Optional: If you selected **Other System Certificate Store** in step 3, in the **Certificate store path and filename** field, type the IFS path and file name you set when you created your certificate store and type a password in the **Certificate Store Password** field. Then click **Continue**
6. In the **Manage Certificates** task category in the navigation panel, click **Import Certificate**. The Import Certificate page is displayed in the task frame.
7. Select the radio button for your certificate type and click **Continue**. Either the Import Server or Client Certificate page or the Import Certificate Authority (CA) Certificate page is displayed in the task frame.
8. In the **Import File** field, type the file name of the certificate you want to import and click **Continue**. DCM automatically determines the format of the file.
9. If the certificate is a **Server or client** certificate, type the password in the task frame and click **Continue**. DCM informs you that the certificate has been imported.

Parent topic: Working with SSL or TLS on i5/OS

Related tasks

Creating a certificate store on i5/OS

6.13.12. Removing certificates in i5/OS

Use this procedure to remove personal certificates.

Procedure

1. Access the DCM interface, as described in Accessing DCM.
2. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
3. Select the **Other System Certificate Store** check box and click **Continue**. The Certificate Store and Password page is displayed.
4. In the **Certificate store path and filename** field, type the IFS path and file name you set when you created the certificate store.
5. Type a password in the **Certificate Store Password** field. Click **Continue**. The Current Certificate Store page is displayed in the task frame.
6. In the **Manage Certificates** task category in the navigation panel, click **Delete Certificate**. The Confirm Delete Certificate page is displayed in the task frame.
7. Select the certificate you want to delete. Click **Delete**.
8. Click **Yes** to confirm that you want to delete the certificate. Otherwise, click **No**. DCM informs you if it has deleted the certificate.

Parent topic: Working with SSL or TLS on i5/OS

6.13.13. Using the *SYSTEM certificate store for one-way authentication on i5/OS

Follow these instructions to set up one-way authentication.

Before you begin

- You have created a queue manager, channels, and transmission queues.
- You have created a server or client certificate on the server queue manager.
- You have transferred the CA certificate to the client queue manger and imported it into the key repository.
- You have started a listener on the server and client queue managers.

About this task

To use one-way authentication, using a computer running i5/OS as the SSL server, set the SSL Key Repository (SSLKEYR) parameter to *SYSTEM. This setting registers the WebSphere MQ queue manager as an application. You can then assign a certificate to the queue manager to enable one-way authentication.

You can also use private keystores to implement one-way authentication by creating a dummy certificate for the client queue manager in the key repository.

Procedure

1. Perform the following steps on the server and client queue managers:
   a. Alter the queue manager to set the SSLKEYR parameter by issuing the command `CHGMQM MQMNAME(SSL) SSLKEYR(*SYSTEM)`.
   b. Stash the password for the default key repository by issuing the command `CHGMQM MQMNAME(SSL) SSLKEYRPWD('xxxxxxx')`. The password must be in single quotation marks.
   c. Alter the channels to have the correct CipherSpec in the SSLCIPHER parameter.
   d. Refresh SSL security by issuing the command `RFRMQMAUT QMNAME(QMGRNAME) TYPE(*SSL)`.
2. Assign the certificate to the server queue manager using DCM, as follows:
   a. Access the DCM interface, as described in Accessing DCM.
   b. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
   c. Select the *SYSTEM certificate store and click **Continue**.
   d. In the left panel, expand **Manage Applications**.
   e. Select the **View Application** definition to check that the queue manager has been registered as an application. `SSL (WMQ)` is listed in the table.
   f. Select **Update Certificate Assignment**.
   g. Select **Server** and click **Continue**.
   h. Select QMGRNAME (WMQ) and click **Update certificate assignment**.
   i. Select the certificate and click **Assign New Certificate**. A window opens stating that the certificate has been assigned to the application.

Parent topic: Working with SSL or TLS on i5/OS

6.13.14. When changes to certificates or the certificate store become effective on i5/OS

When you change the certificates in a certificate store, or the location of the certificate store, the changes take effect depending on the type of channel and how the channel is running.

Changes to the certificates in the certificate store and to the key repository attribute become effective in the following situations:

- When a new outbound single channel process first runs an SSL channel.
- When a new inbound TCP/IP single channel process first receives a request to start an SSL channel.
- When the MQSC command REFRESH SECURITY TYPE(SSL) is issued to refresh the WebSphere MQ SSL environment.
- For client application processes, when the last SSL connection in the process is closed. The next SSL connection picks up the certificate changes.
- For channels that run as threads of a process pooling process (amqrmppa), when the process pooling process is started or restarted and first runs an SSL channel. If the process pooling process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).
- For channels that run as threads of the channel initiator, when the channel initiator is started or restarted and first runs an SSL channel. If the channel initiator process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).
- For channels that run as threads of a TCP/IP listener, when the listener is started or restarted and first receives a request to start an SSL channel. If the listener has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).

Parent topic: Working with SSL or TLS on i5/OS

6.13.15. Configuring cryptographic hardware on i5/OS

Use this procedure to configure the 4758 PCI Cryptographic Coprocessor on i5/OS®

Before you begin

Ensure your user profile has *ALLOBJ and *SECADM special authorities to enable you to configure the coprocessor hardware.

Procedure

1. Go to either `http://`*`machine`*`.domain:2001` or `https://`*`machine`*`.domain:2010`, where *machine* is the name of your computer. A dialog box is displayed, requesting a user name and a password.
2. Type a valid i5/OS user profile and password.
3. On the AS/400® Tasks page, click **4758 PCI Cryptographic Coprocessor**.

What to do next

For more information about configuring the 4758 PCI Cryptographic Coprocessor, refer to the *i5/OS Information Center* at IBM System i and IBM i Information Center.

Parent topic: Working with SSL or TLS on i5/OS

6.13.16. Mapping DNs to user IDs

WebSphere® MQ on i5/OS® does not support the i5/OS function that is equivalent to the z/OS® CNFs. If you want to implement a function that maps Distinguished Names to user IDs, consider using a channel security exit.

Parent topic: Working with SSL or TLS on i5/OS

Related concepts
Associating a user ID with a digital certificate on z/OS

Related information
Channel-exit programs for messaging channels

6.14. Working with SSL or TLS on UNIX and Windows systems

This information describes how you set up and work with the Secure Sockets Layer (SSL) on UNIX and Windows systems.

This collection of topics applies to the following:

- WebSphere® MQ for AIX®
- WebSphere MQ for HP-UX
- WebSphere MQ for Linux
- WebSphere MQ for Solaris
- WebSphere MQ for Windows

For these platforms, the SSL support is installed with WebSphere MQ.

Using iKeyman, iKeycmd, GSKCapiCmd, and GSK7Cmd

On UNIX and Windows systems, manage keys and digital certificates with the iKeyman GUI or from the command line using iKeycmd or GSKCapiCmd.

Setting up a key repository on UNIX and Windows systems
Set up a key repository at both ends of the connection. Use the default certificate stores or create your own.

Locating the key repository for a queue manager on UNIX or Windows systems
Use this procedure to obtain the location of your queue manager's key database file

Changing the key repository location for a queue manager on UNIX or Windows systems
You can change the location of your queue manager's key database file by various means including the MQSC command ALTER QMGR.

Locating the key repository for a WebSphere MQ client on UNIX and Windows systems.
The location of the key repository is given by the MQSSLKEYR variable, or specified in the MQCONNX call.

Specifying the key repository location for a WebSphere MQ client on UNIX or Windows systems
There is no default key repository for a WebSphere MQ client. You can specify its location in either of two ways. Ensure that the key database file can be accessed only by intended users or administrators to prevent unauthorized copying to other systems.

When changes to certificates or the certificate store become effective on UNIX or Windows systems.
When you change the certificates in a certificate store, or the location of the certificate store, the changes take effect depending on the type of channel and how the channel is running.

Creating a self-signed personal certificate on UNIX or Windows systems
You can create a self-signed certificate using iKeyman, iKeycmd, or GSKCapiCmd.

Requesting a personal certificate on UNIX or Windows systems.
You can request a personal certificate using iKeyman, iKeycmd or GSKCapiCmd.

Receiving personal certificates into a key repository on UNIX systems and Windows
Use this procedure to receive a personal certificate into the key database file.

Extracting a CA certificate from a key repository
Follow this procedure to extract a CA certificate.

Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows
Follow this procedure to extract the public part of a self-signed certificate.

Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows
Follow this procedure to add a CA certificate or the public part of a self-signed certificate to the key repository.

Exporting a personal certificate from a key repository
Follow this procedure to exporting a personal certificate.

Importing a personal certificate into a key repository on UNIX or Windows systems
Follow this procedure to import a personal certificate

Importing from a Microsoft .pfx file
Folow this procedreimport from a Microsoft .pfx file using iKeyman. You cannot use GSKCapiCmd to import a .pfx file.

Importing from a PKCS #7 file
The iKeyman and iKeycmd tools do not support PKCS #7 (.p7b) files. Use the GSK7Cmd tool to import certificates from a PKCS #7 file.

Deleting a certificate from a key repository on UNIX systems or Windows
Use this procedure to remove personal or CA certificates.

Configuring for cryptographic hardware on UNIX or Windows systems
You can configure cryptographic hardware for a queue manager or client in a number of ways

Mapping DNs to user IDs
UNIX systems do not have a function equivalent to the z/OS® CNFs. If you want to implement a function that maps Distinguished Names to user IDs, consider using a channel security exit.

Certificate validation and trust policy design on UNIX and Windows systems
WebSphere MQ validates SSL or TLS certificates according to two types of policy, basic and standard. Standard policy checking conforms to RFC 3280.

Migrating SSL security certificates in WebSphere MQ for Windows
SSL support changed after WebSphere MQ Version 5.3. If you are upgrading form Verison 5.3, you must migrate your certificates

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Cryptographic security protocols: TLS and SSL

6.14.1. Using iKeyman, iKeycmd, GSKCapiCmd, and GSK7Cmd

On UNIX and Windows systems, manage keys and digital certificates with the iKeyman GUI or from the command line using iKeycmd or GSKCapiCmd.

- For **UNIX** systems:
  - Use the **gsk7ikm** command to start the iKeyman GUI.
  - Use the **gsk7cmd** command to perform tasks with the iKeycmd command line interface.
  - Use the **gsk7capicmd** command to perform tasks with the GSKCapiCmd command line interface. The command syntax for **gsk7capicmd** is the same as the syntax for **gsk7cmd**.
    If you need to manage SSL certificates in a way that is FIPS compliant, use the **gsk7capicmd** command instead of the **gsk7cmd** or **gsk7ikm**

commands. ❮

See the WebSphere MQ System Administration Guide for a full description of the command line interfaces for the **gsk7cmd** and **gsk7capicmd** commands.
Before you run the **gsk7ikm** command to start the iKeyman GUI, ensure you are working on a machine that is able to run the X Window System and that you do the following:

- Set the DISPLAY environment variable, for example:
  ```
  export DISPLAY=mypc:0
  ```
- Ensure that your PATH environment variable contains **/usr/bin** and **/bin**. This is also required for the **gsk7cmd** and **gsk7capicmd** commands. For example:
  ```
  export PATH=$PATH:/usr/bin:/bin
  ```
- Set the JAVA_HOME environment variable:

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`

  These are also required for the **gsk7cmd** command.

- For **Windows** systems:
  - Use the **strmqikm** command to start the iKeyman GUI.
  - Use the **runmqckm** command to perform tasks with the iKeycmd command line interface.
  - Use the **gsk7capicmd** command to perform tasks with the GSKCapiCmd command line interface. 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 **runmqckm** or ❯**strmqikm**❮ 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
  ```

See the WebSphere MQ System Administration Guide for more information on the **strmqikm**, **runmqckm**, and **gsk7capicmd** commands.

To request SSL tracing on UNIX or Windows systems, see the WebSphere MQ System Administration Guide.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.2. Setting up a key repository on UNIX and Windows systems

❯Set up a key repository at both ends of the connection. Use the default certificate stores or create your own. ❮

An SSL connection requires a *key repository* at each end of the connection. Each WebSphere® MQ queue manager and WebSphere MQ client must have access to a key repository. See The SSL key repository for more information.

On UNIX and Windows systems, digital certificates are stored in a key database file that is managed with iKeyman, iKeycmd, or GSKCapiCmd. These digital certificates have labels. A specific label associates a personal certificate with a queue manager or WebSphere MQ client. SSL uses that certificate for authentication purposes. On UNIX and Windows systems, WebSphere MQ uses the `ibmwebspheremq` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager or WebSphere MQ client user logon ID, changed to lower case. Ensure that you specify the entire certificate label in lowercase.

The key database file name comprises a path and stem name:

- On UNIX, the default path for a queue manager (set when you create the queue manager) is `/var/mqm/qmgrs/<queue_manager_name>/ssl`.
  On Windows, the default path is *`install_directory`*`\Qmgrs\<queue_manager_name>\ssl`, where *install_directory* is the directory in which WebSphere MQ is installed. For example, `C:\Program Files\IBM\WebSphere MQ\Qmgrs\<queue_manager_name>\ssl` .
  The default stem name is `key`. Optionally, you can choose your own path and stem name, but the extension must be `.kdb`.
  If you choose your own path or filename, set the permissions to the file to tightly control access to it.
- For a WebSphere MQ client, there is no default path or stem name. Tightly control access to this file. The extension must be `.kdb`.

Note that key repositories must not be created on a file system that does not support file level locks, for example NFS version 2 on Linux.

Changing the key repository location for a queue manager on UNIX or Windows systems tells you about checking and specifying the key database file name. You can specify the key database file name either before or after creating the key database file.

The user ID from which you run iKeyman or iKeycmd must have write permission for the directory in which the key database file is created or updated. For a queue manager using the default SSL directory, the user ID from which you run iKeyman or iKeycmd must be a member of the mqm group. For a WebSphere MQ client, if you run iKeyman or iKeycmd from a user ID different from that under which the client runs, you must alter the file permissions to enable the WebSphere MQ client to access the key database file at run time. For more information, refer to Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using iKeyman

Use the following procedure to create a new key database file for either a queue manager or a WebSphere MQ client:

1. Start the iKeyman GUI (using the **gsk7ikm** command on UNIX, or the **strmqikm** command on Windows).
2. From the **Key Database File** menu, click New. The New window is displayed.
3. Click Key database type and select **CMS** (Certificate Management System).
4. In the **File Name** field, type a file name. This field already contains the text `key.kdb`. If your stem name is `key`, leave this field unchanged. If you have specified a different stem name, replace `key` with your stem name but you must not change the `.kdb`.
5. In the **Location** field, type the path, for example:
   - For a queue manager: `/var/mqm/qmgrs/QM1/ssl` (on UNIX) or `C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\ssl` (on Windows)

The path must match the value of the SSLKeyRepository attribute of the queue manager.
   - For a WebSphere MQ client: `/var/mqm/ssl` (on UNIX) or `C:\mqm\ssl` (on Windows)

6. Click **Open**. The Password Prompt window displays.
7. Type a password in the **Password** field, and type it again in the **Confirm Password** field.
8. Select the **Stash the password to a file** check box.
   Note: If you do not stash the password, attempts to start SSL channels fail because they cannot obtain the password required to access the key database file.
9. Click **OK**. A window is displayed, confirming that the password is in file `key.sth` (unless you specified a different stem name).
10. Click **OK**. The Signer Certificates window is displayed, containing a list of the CA certificates that are provided with iKeyman and pre-installed in the key database.
11. Remove these CA certificates by selecting them and clicking **Delete**.
12. Set the access permissions, as described in Accessing and securing your key database files on Windows or Accessing and securing your key database files on UNIX systems.

Using the command line

Use the following commands to create a new CMS key database file using iKeycmd or GSKCapiCmd:

- On UNIX:
```
gsk7cmd -keydb -create -db filename -pw password -type cms -expire days
    -stash
```
- On Windows:
```
runmqckm -keydb -create -db filename -pw password -type cms -expire days
     -stash
```
- Using GSKCapiCmd:
```
gsk7capicmd -keydb -create -db filename -pw password -type cms -expire days
        -stash -fips -strong
```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database, and must have a file extension of `.kdb`.
`-pw` *password*	is the password for the CMS key database.
`-type` *cms*	is the type of database (for WebSphere MQ, this must be `cms`).
`-expire` *days*	is the expiration time in days of the database password. There is no default time for a database password: use the `-expire` option to set a database password expiration time explicitly.
`-stash`	tells iKeycmd or GSKCapiCmd to stash the key database password to a file.
`-fips`	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.
`-strong`	checks that the password entered satisfies the minimum requirements for password 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.

A number of CA certificates are pre-installed in the key database. Remove these using the methods described in Deleting a certificate from a key repository on UNIX systems or Windows

Accessing and securing your key database files on Windows
The key database files might not have appropriate access permissions. You must set appropriate access to these files.

Accessing and securing your key database files on UNIX systems
The key database files might not have appropriate access permissions. You must set appropriate access to these files.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

Related concepts
Digital certificates

6.14.2.1. Accessing and securing your key database files on Windows

The key database files might not have appropriate access permissions. You must set appropriate access to these files.

Set access control to the files *key*`.kdb`, *key*`.sth`, *key*`.crl`, and *key*`.rdb`, where *key* is the stem name of your key database, to grant authority to a restricted set of users.

Consider granting access as follows:

full authority
BUILTIN\Administrators, NT AUTHORITY\SYSTEM, and the user who created the database files.

read authority

For a queue manager, the local mqm group only. This assumes that the MCA is running under a user ID in the mqm group.
For a client, the user ID under which the client process is running.

When you migrate your digital certificates from the certificate store on WebSphere® MQ for Windows V5.3 or V5.3.1 to a GSKit key repository, the .kdb file is created as part of the certificate transfer (using AMQTCERT), and the required access permissions must already be set for this to succeed.

Parent topic: Setting up a key repository on UNIX and Windows systems

6.14.2.2. Accessing and securing your key database files on UNIX systems

The key database files might not have appropriate access permissions. You must set appropriate access to these files.

For a queue manager, set permissions on the key database files so that queue manager and channel processes can read them when necessary, but other users cannot read or modify them. Normally, the mqm user needs read permissions. If you have created the key database file by logging in as the mqm user, then the permissions are probably sufficient; if you were not the mqm user, but another user in the mqm group, you probably need to grant read permissions to other users in the mqm group.

Similarly for a client, set permissions on the key database files so that client application processes can read them when necessary, but other users cannot read or modify them. Normally, the user under which the client process runs needs read permissions. If you have created the key database file by logging in as that user, then the permissions are probably sufficient; if you were not the client process user, but another user in that group, you probably need to grant read permissions to other users in the group.

Set the permissions on the files `key.kdb`, `key.sth`, `key.crl`, and `key.rdb`, where *key* is the stem name of your key database, to read and write for the file owner, and to read for the mqm or client user group (-rw-r-----).

Parent topic: Setting up a key repository on UNIX and Windows systems

6.14.3. Locating the key repository for a queue manager on UNIX or Windows systems

Use this procedure to obtain the location of your queue manager's key database file

Procedure

1. Display your queue manager's attributes, using either of the following MQSC commands:
   ```
   DISPLAY QMGR ALL
   DISPLAY QMGR SSLKEYR
   ```
   You can also display your queue manager's attributes using the WebSphere® MQ Explorer or PCF commands.
2. Examine the command output for the path and stem name of the key database file. For example,
   a. on UNIX systems: `/var/mqm/qmgrs/QM1/ssl/key`, where `/var/mqm/qmgrs/QM1/ssl` is the path and `key` is the stem name
   b. on Windows: `C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\ssl\key`, where `C:\Program Files\IBM\WebSphere MQ\qmgrs\QM1\ssl` is the path and `key` is the stem name

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.4. Changing the key repository location for a queue manager on UNIX or Windows systems

You can change the location of your queue manager's key database file by various means including the MQSC command ALTER QMGR.

You can change the location of your queue manager's key database file by using the MQSC command ALTER QMGR to set your queue manager's key repository attribute. For example, on UNIX:

```
   ALTER QMGR SSLKEYR('/var/mqm/qmgrs/QM1/ssl/MyKey')
```

The key database file has the fully qualified file name: `/var/mqm/qmgrs/QM1/ssl/MyKey.kdb`

On Windows:

```
   ALTER QMGR SSLKEYR('C:\Program Files\IBM\WebSphere MQ\Qmgrs\QM1\ssl\Mykey')
```

The key database file has the fully qualified file name: `C:\Program Files\IBM\WebSphere MQ\Qmgrs\QM1\ssl\Mykey.kdb`

You can also alter your queue manager's attributes using the WebSphere® MQ Explorer or PCF commands.

When you change the location of a queue manager's key database file, certificates are not transferred from the old location. If the CA certificates preinstalled when you create the key database file are insufficient, you must populate the new key database file with the extra CA certificates you need, as described in Importing a personal certificate into a key repository on UNIX or Windows systems.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.5. Locating the key repository for a WebSphere MQ client on UNIX and Windows systems.

The location of the key repository is given by the MQSSLKEYR variable, or specified in the MQCONNX call.

Examine the MQSSLKEYR environment variable to obtain the location of your WebSphere® MQ client's key database file. For example:

```
echo $MQSSLKEYR
```

Also check your application, because the key database file name can also be set in an MQCONNX call, as described in Specifying the key repository location for a WebSphere MQ client on UNIX or Windows systems. The value set in an MQCONNX call overrides the value of MQSSLKEYR.
Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.6. Specifying the key repository location for a WebSphere MQ client on UNIX or Windows systems

There is no default key repository for a WebSphere® MQ client. You can specify its location in either of two ways. Ensure that the key database file can be accessed only by intended users or administrators to prevent unauthorized copying to other systems.

You can specify the location of your WebSphere MQ client's key database file in either of two ways:

- Setting the MQSSLKEYR environment variable. For example, on UNIX:
  ```
  export MQSSLKEYR=/var/mqm/ssl/key
  ```
  The key database file has the fully-qualified file name:
  ```
  /var/mqm/ssl/key.kdb
  ```
  On Windows:
  ```
  set MQSSLKEYR=C:\Program Files\IBM\WebSphere MQ\ssl\key
  ```
  The key database file has the fully-qualified file name:
  ```
  C:\Program Files\IBM\WebSphere MQ\ssl\key.kdb
  ```
  Note: The `.kdb` extension is a mandatory part of the file name, but is not included as part of the value of the environment variable.
- Providing the path and stem name of the key database file in the *KeyRepository* field of the MQSCO structure when an application makes an MQCONNX call. For more information about using the MQSCO structure in MQCONNX, refer to the WebSphere MQ Application Programming Reference.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.7. When changes to certificates or the certificate store become effective on UNIX or Windows systems.

›When you change the certificates in a certificate store, or the location of the certificate store, the changes take effect depending on the type of channel and how the channel is running.‹

Changes to the certificates in the key database file and to the key repository attribute become effective in the following situations: ›

- When a new outbound single channel process first runs an SSL channel.
- When a new inbound TCP/IP single channel process first receives a request to start an SSL channel.
- When the MQSC command REFRESH SECURITY TYPE(SSL) is issued to refresh the Websphere MQ SSL environment.
- For client application processes, when the last SSL connection in the process is closed. The next SSL connection will pick up the certificate changes.
- For channels that run as threads of a process pooling process (amqrmppa), when the process pooling process is started or restarted and first runs an SSL channel. If the process pooling process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).
- For channels that run as threads of the channel initiator, when the channel initiator is started or restarted and first runs an SSL channel. If the channel initiator process has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).
- For channels that run as threads of a TCP/IP listener, when the listener is started or restarted and first receives a request to start an SSL channel. If the listener has already run an SSL channel, and you want the change to become effective immediately, run the MQSC command REFRESH SECURITY TYPE(SSL).

‹

You can also refresh the WebSphere® MQ SSL environment using the WebSphere MQ Explorer or PCF commands.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.8. Creating a self-signed personal certificate on UNIX or Windows systems

You can create a self-signed certificate using iKeyman, iKeycmd, or GSKCapiCmd.

Using iKeyman

When you create a key database, no personal certificates are provided. However, you need a personal certificate before you can run an SSL channel. A self-signed personal certificate can be used to run SSL channels for the purposes of testing SSL communications. These certificates can be created on either a WebSphere® MQ queue manager or WebSphere MQ client system.

Use the following procedure to obtain a self-signed certificate for your queue manager or WebSphere MQ client:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window displays.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file in which you want to save the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. From the **Create** menu, click **New Self-Signed Certificate**. The Create New Self-Signed Certificate window displays.
9. In the **Key Label** field, type:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or,
   - For a WebSphere MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
10. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining optional fields, either accept the default values, or type or select new values. Note that you can supply only one name in the **Organizational Unit** field. For more information about these fields, refer to Distinguished Names.
11. Click **OK**. The **Personal Certificates** list shows the label of the self-signed personal certificate you created.

Using the command line

Use the following commands to create a self-signed personal certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
```
gsk7cmd -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
```
- On Windows:
```
runmqckm -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
```
- Using GSKCapiCmd:
```
gsk7capicmd -cert -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -x509version version -expire days
        -fips -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the key label attached to the certificate.
`-dn` *distinguished_name*	is the X.500 distinguished name enclosed in double quotes. Note that only the CN attribute is required. You can supply multiple OU attributes.
`-size` *key_size*	is the key size. For iKeycmd, the value can be 512 or 1024. For GSKCapiCmd, the value can be 512, 1024, or 2048.
`-x509version` *version*	is the version of X.509 certificate to create. The value can be 1, 2, or 3. The default is 3.
`-expire` *days*	is the expiration time in days of the certificate. The default is 365 days for a certificate.
`-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.
`-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 self-signed certificate. The value can be md5, sha1, sha224, sha256, sha384, or sha512. The default is sha1.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.9. Requesting a personal certificate on UNIX or Windows systems.

You can request a personal certificate using iKeyman, iKeycmd or GSKCapiCmd.

Using iKeyman

To apply for a personal certificate, use the iKeyman tool as follows:

1. Start the iKeyman GUI using either the **gsk7ikm** command (UNIX) or the **strmqikm** command (Windows).
2. From the **Key Database File** menu, click **Open**. The Open window displays.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file from which you want to generate the request, for example `key.kdb`.
6. Click **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.

8. From the **Create** menu, click **New Certificate Request**. The Create New Key and Certificate Request window displays.
9. In the **Key Label** field, type:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager changed to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or
   - For a WebSphere® MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
10. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining optional fields, either accept the default values, or type or select new values. Note that you can supply only one name in the **Organizational Unit** field. For more information about these fields, refer to Distinguished Names.
11. In the **Enter the name of a file in which to store the certificate request** field, either accept the default `certreq.arm`, or type a new value with a full path.
12. Click **OK**. A confirmation window displays.
13. Click **OK**. The **Personal Certificate Requests** list shows the label of the new personal certificate request you created. The certificate request is stored in the file you chose in step 11.
14. Request the new personal certificate either by sending the file to a Certification Authority (CA), or by copying the file into the request form on the Web site for the CA.

Using the command line

Use the following commands to request a personal certificate using iKeycmd or GSKCapiCmd:

- On UNIX, issue the following command:

```
gsk7cmd -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename
```

- On Windows, issue the following command:

```
runmqckm -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename
```

- On either UNIX or Windows, issue the following command:

```
gsk7capicmd -certreq -create -db filename -pw password -label label
        -dn distinguished_name -size key_size -file filename -fips
        -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

`-db` *filename*	The fully qualified file name of a CMS key database.
`-pw` *password*	The password for the CMS key database.
`-label` *label*	The key label attached to the certificate.
`-dn` *distinguished_name*	The X.500 distinguished name enclosed in double quotes. Note that only the CN attribute is required. You can supply multiple OU attributes.
`-size` *key_size*	The key size. If you are using iKeycmd, the value can be 512 or 1024. If you are using GSKCapiCmd, the value can be 512, 1024, or 2048.
`-file` *filename*	The file name for the 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.
`-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 request. The value can be `md5`, `sha1`, `sha224`, `sha256`, `sha384`, or `sha512`. The default is `sha1`.

If you are using cryptographic hardware, refer to Requesting a personal certificate for your PKCS #11 hardware.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.10. Receiving personal certificates into a key repository on UNIX systems and Windows

Use this procedure to receive a personal certificate into the key database file.

After the CA sends you a new personal certificate, you add it to the key database file from which you generated the new certificate request . If the CA sends the certificate as part of an e-mail message, copy the certificate into a separate file.

Using iKeyman

Ensure that the certificate file to be imported has write permission for the current user, and then use the following procedure for either a queue manager or a WebSphere® MQ client to receive a personal certificate into the key database file:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**, and then click **OK**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field. Select the **Personal Certificates** view.
8. Click **Receive**. The Receive Certificate from a File window opens.

9. Select the **Data type** of the new personal certificate, for example **Base64–encoded ASCII data** for a file with the `.arm` extension.
10. Type the certificate file name and location for the new personal certificate, or click **Browse** to select the name and location.
11. Click **OK**. If you already have a personal certificate in your key database, a window opens, asking if you want to set the key you are adding as the default key in the database.
12. Click **Yes** or **No**. The Enter a Label window opens.
13. Click **OK**. The **Personal Certificates** field shows the label of the new personal certificate you added.

Using the command line

Use the following commands to add a personal certificate to a key database file using iKeycmd or GSKCapiCmd:

- On UNIX systems, issue the following command:
  ```
  gsk7cmd -cert -receive -file filename -db filename -pw password
          -format ascii
  ```
- On Windows, issue the following command:
  ```
  runmqckm -cert -receive -file filename -db filename -pw password
          -format ascii
  ```
- On UNIX systems or Windows, issue the following command:
  ```
  gsk7capicmd -cert -receive -file filename -db filename -pw password -fips
  ```

where:

`-file` *filename*	is the fully qualified file name of the file containing the personal certificate.
`-db` *filename*	is the fully qualified file name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

If you are using cryptographic hardware, refer to Importing a personal certificate to your PKCS #11 hardware.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.11. Extracting a CA certificate from a key repository

Follow this procedure to extract a CA certificate.

Perform the following steps on the machine from which you want to extract the CA certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. In the **Key database content** field, select **Signer Certificates** and select the certificate you want to extract.
9. Click **Extract**. The Extract a Certificate to a File window opens.
10. Select the **Data type** of the certificate, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where you want to store the certificate, or click **Browse** to select the name and location.
12. Click **OK**. The certificate is written to the file you specified.

Use the following commands to extract a CA certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
  ```
  gsk7cmd -cert -extract -db filename -pw password -label label -target filename
          -format ascii
  ```
- On Windows:
  ```
  runmqckm -cert -extract -db filename -pw password -label label -target filename
           -format ascii
  ```
- Using GSKCapiCmd:
  ```
  gsk7capicmd -cert -extract -db filename -pw password -label label
              -target filename -format ascii -fips
  ```

where:

`-db` *filename*	is the fully qualified path name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the certificate.
`-target` *filename*	is the name of the destination file.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.12. Extracting the public part of a self-signed certificate from a key repository on UNIX systems and Windows

Follow this procedure to extract the public part of a self-signed certificate.

Using iKeyman

Perform the following steps on the machine from which you want to extract the public part of a self-signed certificate:

1. Start the iKeyman GUI using either the gsk7ikm command (on UNIX) or the strmqikm command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file from which you want to extract the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates** and select the certificate.
9. Click **Extract certificate**. The Extract a Certificate to a File window opens.
10. Select the **Data type** of the certificate, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where you want to store the certificate, or click **Browse** to select the name and location.
12. Click **OK**. The certificate is written to the file you specified. Note that when you extract (rather than export) a certificate, only the public part of the certificate is included, so a password is not required.

Using the command line

Use the following commands to extract the public part of a self-signed certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
```
gsk7cmd -cert -extract -db filename -pw password -label label -target filename
        -format ascii
```
- On Windows:
```
runmqckm -cert -extract -db filename -pw password -label label -target filename
         -format ascii
```
- Using GSKCapiCmd:
```
gsk7capicmd -cert -extract -db filename -pw password -label label
            -target filename -format ascii -fips
```

where:

`-db` *filename*	is the fully qualified path name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the certificate.
`-target` *filename*	is the name of the destination file.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.13. Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows

Follow this procedure to add a CA certificate or the public part of a self-signed certificate to the key repository.

If the certificate that you want to add is in a certificate chain, you must also add all the certificates that are above it in the chain. You must add the certificates in strictly descending order starting from the root, followed by the CA certificate immediately below it in the chain, and so on.

Where the following instructions refer to a CA certificate, they also apply to the public part of a self-signed certificate.

Using iKeyman

Perform the following steps on the machine on which you want to add the CA certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Signer Certificates**.
9. Click **Add**. The Add CA's Certificate from a File window opens.
10. Select the **Data type** of the certificate you transferred, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
11. Type the certificate file name and location where the certificate is stored, or click **Browse** to select the name and location.
12. Click **OK**. The Enter a Label window opens.
13. In the Enter a Label window, type the name of the certificate.
14. Click **OK**. The certificate is added to the key database.

Using the command line

Use the following commands to add a CA certificate using iKeycmd or GSKCapiCmd:

- On UNIX systems, issue the following command:

```
gsk7cmd -cert -add -db filename -pw password -label label -file filename
        -format ascii
```

- On Windows, issue the following command:

```
runmqckm -cert -add -db filename -pw password -label label -file filename
        -format ascii
```

- On UNIX systems or Windows, issue the following command:

```
gsk7capicmd -cert -add -db filename -pw password -label label -file filename
        -format ascii -fips
```

where:

`-db` *filename*	is the fully qualified path name of the CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the certificate.
`-file` *filename*	is the name of the file containing the certificate.
`-format` *ascii*	is the format of the certificate. The value can be `ascii` for Base64-encoded ASCII or `binary` for Binary DER data. The default is `ascii`.
`-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.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.14. Exporting a personal certificate from a key repository

Follow this procedure to exporting a personal certificate.

Perform the following steps on the machine from which you want to export the personal certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates** and select the certificate you want to export.
9. Click **Export/Import**. The Export/Import key window opens.
10. Select **Export Key**.
11. Select the **Key file type** of the certificate you want to export, for example **PKCS12**.
12. Type the file name and location to which you want to export the certificate, or click **Browse** to select the name and location.
13. Click **OK**. The Password Prompt window opens. Note that when you export (rather than extract) a certificate, both the public and private parts of the certificate are included. This is why the exported file is protected by a password. When you extract a certificate, only the public part of the certificate is included, so a password is not required.
14. Type a password in the **Password** field, and type it again in the **Confirm Password** field.
15. Click **OK**. The certificate is exported to the file you specified.

Use the following commands to export a personal certificate using iKeycmd:

- On UNIX:

```
gsk7cmd -cert -export -db filename -pw password -label label -type cms
```

```
            -target filename -target_pw password -target_type pkcs12
```

- On Windows:

```
      runmqckm -cert -export -db filename -pw password -label label -type cms
              -target filename -target_pw password -target_type pkcs12
```

To export a personal certificate using GSKCapiCmd, use the following command:

```
   gsk7capicmd -cert -export -db filename -pw password -label label -type cms
           -target filename -target_pw password -target_type pkcs12
           -encryption strong | weak -fips
```

where:

`-db filename`	is the fully qualified path name of the CMS key database.
`-encryption`	is the strength of encryption used in certificate export command. The value can be `strong` or `weak`. The default is `strong`.
`-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.
`-pw password`	is the password for the CMS key database.
`-label label`	is the label attached to the certificate.
`-type cms`	is the type of the database.
`-target filename`	is the name of the destination file.
`-target_pw password`	is the password for encrypting the certificate.
`-target_type pkcs12`	is the type of the certificate.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.15. Importing a personal certificate into a key repository on UNIX or Windows systems

Follow this procedure to import a personal certificate

Before importing a personal certificate in PKCS #12 format into the key database file, you must first add the full valid chain of issuing CA certificates to the key database file (see Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows).

PKCS #12 files should be considered temporary and deleted after use.

Perform the following steps on the machine to which you want to import the personal certificate:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window displays.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window displays.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file displays in the **File Name** field.
8. In the **Key database content** field, select **Personal Certificates**.
9. Click **Export/Import**. The Export/Import key window is displayed.
10. Select **Import Key**.
11. Select the **Key file type** of the certificate you want to import, for example **PKCS12**.
12. Type the certificate file name and location where the certificate is stored, or click **Browse** to select the name and location.
13. Click **OK**. The Password Prompt window displays.
14. In the **Password** field, type the password used when the certificate was exported.
15. Click **OK**. The Select from Key Label List window is displayed.
16. From the list of certificate labels displayed, select the ones that you want to import. Ensure that you include any CA (signer) certificates that might be necessary to form a full chain for any personal certificates you are importing. You do not need to include any that are already in the target key database.
17. Click **OK**. The Change Labels window is displayed. This window allows the labels of certificates being imported to be changed if, for example, a certificate with the same label already exists in the target key database. Changing certificate labels has no effect on certificate chain validation. This can be used to change the personal certificate label to that required by WebSphere® MQ in order to associate the certificate with the particular queue manager or client (`ibmwebspheremqqm1` for example).
18. To change a label, select the required label from the **Select a label to change:** list. The label is copied into the **Enter a new label:** entry field. Replace the label text with that of the new label and click **Apply**.
19. The text in the **Enter a new label:** entry field is copied back into the **Select a label to change:** field, replacing the originally selected label and so relabelling the corresponding certificate.
20. When you have changed all the labels that needed to be changed, click **OK**. The Change Labels window closes, and the original IBM® Key Management window reappears with the **Personal Certificates** and **Signer Certificates** fields updated with the correctly labeled certificates.
21. The certificate is imported to the target key database.

To import a personal certificate using iKeycmd, use the following commands:

- On UNIX:

```
      gsk7cmd -cert -import -file filename -pw password -type pkcs12 -target filename
      -target_pw password -target_type cms -label label
```

- On Windows:

```
      runmqckm -cert -import -file filename -pw password -type pkcs12 -target filename
```

```
-target_pw password -target_type cms -label label
```

To import a personal certificate using GSKCapiCmd, use the following command:

```
gsk7capicmd -cert -import -file filename -pw password  -type pkcs12 -target filename
-target_pw password -target_type cms -label label -fips
```

where:

`-file` *filename*	is the fully qualified file name of the file containing the PKCS #12 certificate.
`-pw` *password*	is the password for the PKCS #12 certificate.
`-type` *pkcs12*	is the type of the file.
`-target` *filename*	is the name of the destination CMS key database.
`-target_pw` *password*	is the password for the CMS key database.
`-target_type` *cms*	is the type of the database specified by `-target`
`-label` *label*	is the label of the certificate to import from the source key database.
`-new_label` *label*	is the label that the certificate will be assigned in the target database. If you omit `-new_label` option, the default is to use the same as the `-label` option.
`-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.

iKeycmd does not provide a command to change certificate labels directly. Use the following steps to change a certificate label:

1. Export the certificate to a PKCS #12 file using the **-cert -export** command. Specify the existing certificate label for the `-label` option.
2. Remove the existing copy of the certificate from the original key database using the **-cert -delete** command.
3. Import the certificate from the PKCS #12 file using the **-cert -import** command. Specify the old label for the `-label` option and the required new label for the `-new_label` option. The certificate will be imported back into the key database with the required label.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

This build: January 26, 2011 11:21:21

Notices | Trademarks | Downloads | Library | Support | Feedback

6.14.16. Importing from a Microsoft .pfx file

Folow this procedreimport from a Microsoft .pfx file using iKeyman. You cannot use GSKCapiCmd to import a .pfx file.

A .pfx file can contain two certificates relating to the same key. One is a personal or site certificate (containing both a public and private key). The other is a CA (signer) certificate (containing only a public key). These certificates cannot coexist in the same CMS key database file, so only one of them can be imported. Also, the "friendly name" or label is attached to only the signer certificate.

The personal certificate is identified by a system generated Unique User Identifier (UUID). This section shows the import of a personal certificate from a pfx file while labeling it with the friendly name previously assigned to the CA (signer) certificate. The issuing CA (signer) certificates should already be added to the target key database. Note that PKCS#12 files should be considered temporary and deleted after use.

Follow these steps to import a personal certificate from a source pfx key database:

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows). The IBM® Key Management window is displayed.
2. From the **Key Database File** menu, click **Open**. The Open window is displayed.
3. Select a key database type of **PKCS12**.
4. **You are recommended to take a backup of the pfx database before performing this step.** Select the pfx key database that you want to import. Click **Open**. The Password Prompt window is displayed.
5. Enter the key database password and click **OK**. The IBM Key Management window is displayed. The title bar shows the name of the selected pfx key database file, indicating that the file is open and ready.
6. Select **Signer Certificates** from the list. The "friendly name" of the required certificate is displayed as a label in the Signer Certificates panel.
7. Select the label entry and click **Delete** to remove the signer certificate. The Confirm window is displayed.
8. Click **Yes**. The selected label is no longer displayed in the Signer Certificates panel.
9. Repeat steps 6, 7, and 8 for all the signer certificates.
10. From the **Key Database File** menu, click **Open**. The Open window is displayed.
11. Select the target key CMS database which the pfx file is being imported into. Click **Open**. The Password Prompt window is displayed.
12. Enter the key database password and click **OK**. The IBM Key Management window is displayed. The title bar shows the name of the selected key database file, indicating that the file is open and ready.
13. Select **Personal Certificates** from the list.
14. Click **Import** to import keys from the pfx key database. The Import Key window is displayed.
    - Click **Export/Import key**. The Export/Import key window is displayed.
    - Select **Import** from Choose Action Type
15. Select the PKCS12 file.
16. Enter the name of the pfx file as used in Step 4. Click **OK**. The Password Prompt window is displayed.
17. Specify the same password that you specified when you deleted the signer certificate. Click **OK**.
18. The Change Labels window is displayed (as there should be only a single certificate available for import). The label of the certificate should be a UUID which has a format `xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.
19. To change the label select the UUID from the **Select a label to change:** panel. The label will be replicated into the **Enter a new label:** field. Replace the label text with that of the friendly name that was deleted in Step 7 and click **Apply**. The friendly name must be in the form `ibmwebspheremq`, followed by the queue manager name or the WebSphere® MQ client user logon ID in lower case.
20. Click **OK**. The Change Labels window is now removed and the original IBM Key Management window reappears with the Personal Certificates and Signer Certificates panels updated with the correctly labeled personal certificate.
21. The pfx personal certificate is now imported to the (target) database.

It is not possible to change a certificate label using iKeycmd or GSKCapiCmd.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.17. Importing from a PKCS #7 file

The iKeyman and iKeycmd tools do not support PKCS #7 (.p7b) files. Use the GSK7Cmd tool to import certificates from a PKCS #7 file.

Use the following command to add a CA certificate from a PKCS #7 file:

```
gsk7cmd -cert -add -db filename -pw password -type cms -file filename
-label label
```

`-db` *filename*	is the fully qualified file name of the CMS key database.
`-pw` *password*	is the password for the key database.
`-type` *cms*	is the type of the key database.
`-file` *filename*	is the name of the PKCS #7 file.
`-label` *label*	is the label that the certificate is assigned in the target database. The first certificate takes the label given. All other certificates, if present, are labeled with their subject name.

Use the following command to import a personal certificate from a PKCS #7 file:

```
gsk7cmd -cert -import -db filename -pw password -type pkcs7 -target filename
-target_pw password -target_type cms -label label -new_label label
```

`-db` *filename*	is the fully qualified file name of the file containing the PKCS #7 certificate.
`-pw` *password*	is the password for the PKCS #7 certificate.
`-type` *pkcs7*	is the type of the file.
`-target` *filename*	is the name of the destination key database.
`-target_pw` *password*	is the password for the destination key database.
`-target_type` *cms*	is the type of the database specified by `-target`
`-label` *label*	is the label of the certificate that is to be imported.
`-new_label` *label*	is the label that the certificate will be assigned in the target database. If you omit the `-new_label` option, the default is to use the same as the `-label` option.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.18. Deleting a certificate from a key repository on UNIX systems or Windows

Use this procedure to remove personal or CA certificates.

1. Start the iKeyman GUI using either the **gsk7ikm** command (on UNIX) or the **strmqikm** command (on Windows).
2. From the **Key Database File** menu, click **Open**. The Open window opens.
3. Click **Key database type** and select **CMS** (Certificate Management System).
4. Click **Browse** to navigate to the directory that contains the key database files.
5. Select the key database file to which you want to add the certificate, for example `key.kdb`.
6. Click **Open**. The Password Prompt window opens.
7. Type the password you set when you created the key database and click **OK**. The name of your key database file is displayed in the **File Name** field.
8. From the drop down list, select **Personal Certificates** or **Signer Certificates**
9. Select the certificate you want to delete.
10. If you do not already have a copy of the certificate and you want to save it, click **Export/Import** and export it (see Exporting a personal certificate from a key repository).
11. With the certificate selected, click **Delete**. The Confirm window opens.
12. Click **Yes**. The **Personal Certificates** field no longer shows the label of the certificate you deleted.

Use the following commands to delete a certificate using iKeycmd or GSKCapiCmd:

- On UNIX:
  ```
  gsk7cmd -cert -delete -db filename -pw password -label label
  ```
- On Windows:
  ```
  runmqckm -cert -delete -db filename -pw password -label label
  ```
- Using GSKCapiCmd:
  ```
  gsk7capicmd -cert -delete -db filename -pw password -label label -fips
  ```

where:

`-db` *filename*	is the fully qualified file name of a CMS key database.
`-pw` *password*	is the password for the CMS key database.
`-label` *label*	is the label attached to the personal certificate.
`-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.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.19. Configuring for cryptographic hardware on UNIX or Windows systems

You can configure cryptographic hardware for a queue manager or client in a number of ways

You can configure cryptographic hardware for a queue manager on UNIX or Windows using either of the following methods:

- Use the ALTER QMGR MQSC command with the SSLCRYP parameter, as described in the WebSphere MQ Script (MQSC) Command Reference.
- Use WebSphere® MQ Explorer to configure the cryptographic hardware on your UNIX or Windows system. For more information, refer to the online help.

You can configure cryptographic hardware for a WebSphere MQ client on UNIX or Windows using either of the following methods:

- Set the MQSSLCRYP environment variable. The permitted values for MQSSLCRYP are the same as for the SSLCRYP parameter, as described in the WebSphere MQ Script (MQSC) Command Reference. If you use the GSK_PCS11 version of the SSLCRYP parameter, the PKCS #11 token label must be specified entirely in lower-case.
- Set the **CryptoHardware** field of the SSL configuration options structure, MQSCO, on an MQCONNX call. For more information, see the WebSphere MQ Application Programming Guide.

If you have configured cryptographic hardware which uses the PKCS #11 interface using any of these methods, you must store the personal certificate for use on your channels in the key database file for the cryptographic token you have configured. This is described in Managing certificates on PKCS #11 hardware.

Managing certificates on PKCS #11 hardware
This section tells you about managing digital certificates on cryptographic hardware that supports the PKCS #11 interface.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.19.1. Managing certificates on PKCS #11 hardware

This section tells you about managing digital certificates on cryptographic hardware that supports the PKCS #11 interface.

Note that you still need a key database file, even when you store all your certificates on your cryptographic hardware.

Perform the following steps to work with your cryptographic hardware:

1. On UNIX, login as the root user. On Windows, login as Administrator or a member of the MQM group.
2. Execute the gsk7ikm command to start the iKeyman GUI.
3. From the **Key Database File** menu, click **Open**. The Open window opens.
4. Click **Key database type** and select **Cryptographic token**.
5. In the **File Name** field, type the name of the module for managing your cryptographic hardware, for example `PKCS11_API.so`. 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.
6. In the **Location** field, type the path, for example `/usr/lib/pksc11` (on UNIX). On Windows, you can type the library name, for example `cryptoki`.
7. Click **OK**. The Open Cryptographic Token window opens.
8. In the **Cryptographic Token Password** field, type the password that you set when you configured the cryptographic hardware.
9. If your cryptographic hardware has the capacity to hold the signer certificates required to receive or import a personal certificate, clear both secondary key database check boxes and continue from step 17.
   If you require a secondary CMS key database to hold the signer certificates, select either the **Open existing secondary key database file** check box or the **Create new secondary key database file** check box.
10. In the **File Name** field, type a file name. This field already contains the text `key.kdb`. If your stem name is `key`, leave this field unchanged. If you have specified a different stem name, replace `key` with your stem name but you must not change the `.kdb`
11. In the **Location** field, type the path, for example:
    - For a queue manager: `/var/mqm/qmgrs/QM1/ssl`
    - For a WebSphere® MQ client: `/var/mqm/ssl`
12. Click **OK**. The Password Prompt window opens.
13. If you selected the **Open existing secondary key database file** check box in step 9, type a password in the **Password** field, and continue from step 17.
14. If you selected the **Create new secondary key database file** check box in step 9, type a password in the **Password** field, and type it again in the **Confirm Password** field.
15. Select the **Stash the password to a file** check box. Note that if you do not stash the password, attempts to start SSL channels fail because they cannot obtain the password required to access the key database file.
16. Click **OK**. A window opens, confirming that the password is in file `key.sth` (unless you specified a different stem name).
17. Click **OK**. The Key database content frame displays.

Requesting a personal certificate for your PKCS #11 hardware

Use this procedure for either a queue manager or a WebSphere MQ client to request a personal certificate for your cryptographic hardware.

Importing a personal certificate to your PKCS #11 hardware
Use this procedure for either a queue manager or a WebSphere MQ client to import a personal certificate to your cryptographic hardware.

Parent topic: Configuring for cryptographic hardware on UNIX or Windows systems

6.14.19.1.1. Requesting a personal certificate for your PKCS #11 hardware

Use this procedure for either a queue manager or a WebSphere® MQ client to request a personal certificate for your cryptographic hardware.

1. Perform the steps to work with your cryptographic hardware.
2. From the **Create** menu, click **New Certificate Request**. The Create New Key and Certificate Request window opens.
3. In the **Key Label** field, type:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or
   - For a WebSphere MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
4. Type a **Common Name** and **Organization**, and select a **Country**. For the remaining optional fields, either accept the default values, or type or select new values. Note that you can supply only one name in the **Organizational Unit** field. For more information about these fields, refer to Distinguished Names.
5. In the **Enter the name of a file in which to store the certificate request** field, either accept the default `certreq.arm`, or type a new value with a full path.
6. Click **OK**. A confirmation window opens.
7. Click **OK**. The **Personal Certificate Requests** list shows the label of the new personal certificate request you created. The certificate request is stored in the file you chose in step 5.
8. Request the new personal certificate either by sending the file to a Certification Authority (CA), or by copying the file into the request form on the Web site for the CA.

Parent topic: Managing certificates on PKCS #11 hardware

6.14.19.1.2. Importing a personal certificate to your PKCS #11 hardware

Use this procedure for either a queue manager or a WebSphere® MQ client to import a personal certificate to your cryptographic hardware.

1. Perform the steps to work with your cryptographic hardware.
2. Click **Receive**. The Receive Certificate from a File window displays.
3. Select the **Data type** of the new personal certificate, for example **Base64-encoded ASCII data** for a file with the `.arm` extension.
4. Type the certificate file name and location for the new personal certificate, or click **Browse** to select the name and location.
5. Click **OK**. If you already have a personal certificate in your key database, a window appears, asking if you want to set the key you are adding as the default key in the database.
6. Click **Yes** or **No**. The Enter a Label window displays.
7. Type a label, for example the label you used when you requested the personal certificate. Note that the label must be in the correct WebSphere MQ format:
   - For a queue manager, `ibmwebspheremq` followed by the name of your queue manager folded to lower case. For example, for `QM1`, `ibmwebspheremqqm1`, or,
   - For a WebSphere MQ client, `ibmwebspheremq` followed by your logon user ID folded to lower case, for example `ibmwebspheremqmyuserid`.
8. Click **OK**. The **Personal Certificates** list shows the label of the new personal certificate you added. This label is formed by adding the cryptographic token label before the label you supplied.

Parent topic: Managing certificates on PKCS #11 hardware

6.14.20. Mapping DNs to user IDs

UNIX systems do not have a function equivalent to the z/OS® CNFs. If you want to implement a function that maps Distinguished Names to user IDs, consider using a channel security exit.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

Related concepts
Associating a user ID with a digital certificate on z/OS

Related information
Channel-exit programs for messaging channels

6.14.21. Certificate validation and trust policy design on UNIX and Windows systems

WebSphere MQ validates SSL or TLS certificates according to two types of policy, basic and standard. Standard policy checking conforms to RFC 3280.

The information in these topics applies to the following:

- WebSphere® MQ for UNIX systems
- WebSphere MQ for Windows systems

The following terms are used in this section:

certificate policy
: Determines which fields in a certificate are understood and processed.

OCSP policy
: Determines which fields in an OCSP request or response are understood and processed.

CRL policy
: Determines which fields in a certificate revocation list are understood and processed.

path validation policy
: Determines how the certificate, OCSP, and CRL policy types interact with each other to determine if a certificate chain (a trust point "RootCA" to an end-entry "EE") is valid.

The basic and standard path validation policies are described separately because this reflects the implementation within WebSphere MQ for UNIX and Windows systems. However, the standard OCSP and CRL policies are the same as the basic policies, and the standard certificate policy is an extended version of the basic policy, so these are not described separately.

WebSphere MQ applies basic policy validation first. If basic policy validation fails, WebSphere MQ applies standard policy (RFC 3280) validation. If basic policy validation succeeds, standard policy validation is not applied. Thus, a validation failure means that both basic and standard policy validation failed, possibly for different reasons. A validation success means that either basic policy validation succeeded and standard policy validation was therefore not applied, or basic policy validation failed and standard policy validation succeeded.

Basic and standard certificate policies
The basic and standard certificate policies support the same fields: the standard policy supports additional certificate extensions.

Basic and standard OCSP policies
The basic and standard OCSP policies support the same fields.

Basic and standard CRL policies
The basic and standard CRL policies support the same fields and extensions.

Basic path validation policy
The basic path validation policy determines how the certificate, OCSP, and CRL policy types interact with each other to determine if a certificate chain is valid.

Standard path validation policy
The standard path validation policy determines how the certificate, OCSP, and CRL policy types interact with each other to determine if a certificate chain is valid. Standard policy checking conforms to RFC 3280.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.14.21.1. Basic and standard certificate policies

The basic and standard certificate policies support the same fields: the standard policy supports additional certificate extensions.

The supported fields for both the basic and standard policies are as follows:

- OuterSigAlgID[1]
- Signature[2]
- Version
- SerialNumber
- InnerSigAlgID[3]
- Issuer
- Validity
- SubjectName
- SubjectPublicKeyInfo
- IssuerUniqueID
- SubjectUniqueID

The supported extensions for the basic policy are as follows. Where an entry is marked as "not supported", WebSphere MQ does not attempt to process extensions containing a field of that specific type, but does process other types of the same extension.

- AuthorityKeyID
- AuthorityInfoAccess
- SubjectKeyID

- IssuerAltName
- SubjectAltName
- KeyUsage
- BasicConstraints
- PrivateKeyUsage
- CRLDistributionPoints
  - DistributionPoint
    - DistributionPointName (X.500 Name and LDAP Format URI only)
    - NameRelativeToCRLIssuer (not supported)
    - Reasons (ignored)
    - CRLIssuer fields (not supported)

The supported extensions for the standard policy are all those listed for the basic policy and those in the following list. Where an entry is marked as "not supported", WebSphere MQ does not attempt to process extensions containing a field of that specific type, but does process other types of the same extension.

- NameConstraints
- ExtendedKeyUsage
- CertificatePolicies
  - PolicyInformation
    - PolicyIdentifier
    - PolicyQualifiers (not supported)
- PolicyMappings
- PolicyConstraints

Parent topic: Certificate validation and trust policy design on UNIX and Windows systems

[1] This field is called *signatureAlgorithm* in RFC 3280.
[2] This field is called *signatureValue* in RFC 3280.
[3] This field is called *signature* in RFC 3280.

6.14.21.2. Basic and standard OCSP policies

The basic and standard OCSP policies support the same fields.

The supported fields for a request are as follows. Where an entry is marked as "not supported", WebSphere MQ does not attempt to process a request containing a field of that specific type, but does process other requests containing the same higher-level field.

- Signature (Optional)
- Version (Version 1 Only)
- RequesterName (Optional)
- RequestList (single request only)
  - CertID [1]
  - singleRequestExtensions (not supported)
- RequestExtensions
  - Nonce (if enabled)

The supported fields for a response are as follows:

- ResponseStatus
- Response
  - responseType (id-pkix-ocsp-basic)
  - BasicOCSPResponse
    - Signature
    - Certs
      - Extensions
      - extendedKeyUsage
        - id-kp-OCSPSigning
      - id-pkix-ocsp-nocheck
    - ResponseData
      - Version (Version 1 Only)
      - ResponderID (by name or by hash)
      - ProducedAt (ignored)
      - Responses (multiple responses supported)
        - SingleResponse
          - certID
          - certStatus
            - RevokedInfo (ignored)
          - thisUpdate (ignored)
          - nextUpdate

        - singleExtensions (ignored)
    - responseExtensions
        - Nonce (if enabled)

Parent topic: Certificate validation and trust policy design on UNIX and Windows systems
[1] This field is called reqCert in RFC 2560

6.14.21.3. Basic and standard CRL policies

The basic and standard CRL policies support the same fields and extensions.

The supported fields for these policies are as follows:

- OuterSigAlgID[1]
- Signature[2]
- Version
- InnerSigAlgID[3]
- Issuer
- ThisUpdate
- NextUpdate
- RevokedCertificate
    - UserCertificate
    - RevocationDate

There are no supported CRLEntry extensions.

The supported CRL extensions for these policies are as follows. Where an entry is marked as "not supported", WebSphere® MQ does not attempt to process extensions containing a field of that specific type, but does process other types of the same extension.

- AuthorityKeyID
- IssuerAltName
- CRLNumber
- IssuingDistributionPoint
    - DistributionPoint
    - DistributionPointName
        - FullName (X.500 Name and LDAP Format URI only)
        - NameRelativeToCRLIssuer (not supported)
    - Reasons (ignored)
    - CRLIssuer
    - OnlyContainsUserCerts (not supported)
    - OnlyContainsCACerts (not supported)
    - OnlySomeReasons (not supported)
    - IndirectCRL[4] (rejected)

Parent topic: Certificate validation and trust policy design on UNIX and Windows systems
[1] This field is called *signatureAlgorithm* in RFC 3280.
[2] This field is called *signatureValue* in RFC 3280.
[3] This field is called *signature* in RFC 3280.
[4] IndirectCRL extensions will result in CRL validation failing. IndirectCRL extensions must not be used because they cause identified certificates to not be rejected.

6.14.21.4. Basic path validation policy

The basic path validation policy determines how the certificate, OCSP, and CRL policy types interact with each other to determine if a certificate chain is valid.

The validation of a chain is performed in the following manner (but not necessarily in the following order):

1. Ensure that the name of the certificate's issuer is equal to the subject name in the previous certificate, and that there is not an empty issuer name in this certificate or the previous certificate subject name. If no previous certificate exists in the path and this is the first certificate in the chain, ensure that the issuer and subject name are identical and that the trust status is set for the certificate[1].
   Note: WebSphere® MQ for UNIX and Windows systems will fail path validation in situations where the previous certificate in a path has the same subject name as the current certificate.
2. Ensure that the signature algorithm used to actually sign the certificate matches the signature algorithm indicated within the certificate, by ensuring that the issuer signature algorithm identifier in the certificate matches the algorithm identifier in the signature data.
3. Ensure that the certificate was signed by the issuer, using the subject public key from the previous certificate in the path to verify the signature on the

certificate. If no previous certificate exists and this is the first certificate, use the subject public key of the certificate to verify the signature on it. WebSphere MQ supports DSA and RSA signature algorithms; however it does not support DSA Parameter Inheritance.

4. Ensure that the certificate is a known X509 version, unique IDs are not present for version 1 certificates, and extensions are not present for version 1 and version 2 certificates.
5. Ensure that the certificate has not expired, or not been activated yet, and that its validity period is good[2].
6. Ensure that there are no unknown critical extensions or any duplicate extensions.
7. Ensure that the certificate has not been revoked. Here, the following operations apply:
   a. If the OCSP connection is enabled and a Responder Address is configured or the Certificate has a valid AuthorityInfoAccess extension specifying a HTTP format GENERALNAME_uniformResourceID check revocation status with OCSP.
   b. If revocation status from 7.a above is undetermined the CRLDistributionPoints extension is checked for a list of X.500 distinguished name GENERALNAME_directoryname and URI GENERALNAME_uniformResourceID. Only LDAP, HTTP and FILE format URIs are supported. If the extension is not present, or use of the CRLDistributionPoints extension results in undetermined status and the extension is not Critical, the certificate's issuer's name is used to query revocation status. A CRL database (LDAP) is then queried for CRLs. If the certificate is not the last certificate, or if the last certificate has the basic constraint extension with the "isCA" flag turned on, the database is queried for ARLs and CRLs instead. If CRL checking is enabled, and no CRL database can be queried, the certificate is treated as revoked. Currently, the X500 directory name form and the LDAP/HTTP/FILE URI forms are the only supported name forms used to look up CRLs and ARLs[3].
      Note: RelativeDistinguishedNames are not supported.
   c. If revocation status from both 7.a and 7.b is undetermined, WebSphere MQ checks the *OCSPAuthentication* configuration setting to decide whether to allow the connection.[4]
8. If the issuerAltName extension is marked critical, ensure that the name forms are recognized. The following general name forms are currently recognized:
   - rfc822
   - DNS
   - directory
   - URI
   - IPAddress(v4/v6)
9. If the subjectAltName extension is marked critical, ensure that the name forms are recognized. The following general name forms are currently recognized:
   - rfc822
   - DNS
   - directory
   - URI
   - IPAddress(v4/v6)
10. If the KeyUsage extension is critical on a non-EE certificate, ensure that the keyCertSign flag is on, and ensure that if the BasicConstraints extension is present, the "isCA" flag is true.
    a. If the BasicConstraints extension is not present and the StrictBasicConstraintsValidation attribute is set to YES, the certificate is only valid as an EE certificate.
    b. If the BasicConstraints extension is present, the following checks are made:
       - If the "isCA" flag is false, ensure the certificate is the last certificate in the chain and that the pathLength field is not present.
       - If the "isCA" flag is true and the certificate is NOT the last certificate in the chain, ensure that the number of certificates until the last certificate in the chain is not greater than the pathLength field.

12. The AuthorityKeyID extension is not used for path validation, but is used when building the certificate chain.
13. The SubjectKeyID extension is not used for path validation, but is used when building the certificate chain.
14. The PrivateKeyUsagePeriod extension is ignored by the validation engine, because it cannot determine when the CA actually signed the certificate. The extension is always non-critical and therefore can be safely ignored.

An OCSP Response is also validated to ensure that the response itself is valid. Validation is performed in the following manner (but not necessarily the following order):

1. Ensure that response status is `Successful` and the response type is PKIX_AD_OCSP_basic.r
2. Ensure that response version data is present and the response is the correct version (Version 1)
3. Ensure that the response is correctly signed. The signature will be rejected if the signer does not meet at least one of the following criteria:
   - The signer matches a local configuration of OCSP signing authority[5] for the certificate.
   - The signer is using the CA key for which the public key is contained in the CA certificate, that is, the CA itself is directly signing the response.
   - The signer is a direct sub-ordinate of the CA that signed the certificate for which revocation information is being checked and is authorized by the CA by including the value of id-ad-ocspSigning in an ExtendedKeyUsage extension.
     Note: Revocation checking of the response signer certificate is not performed if the id-pkix-ocsp-nocheck extension is present.
4. Ensure that response hash algorithm, serialNumber, issuerNameHash, and issuerKeyHash match those of the request.
5. Ensure that the response has not expired, that is, that the nextUpdate time is greater than the current time.[6]
6. Ensure that the certificate has valid revocation status.

The validation of a CRL is also performed to ensure that the CRL itself is valid, and is performed in the following manner (but not necessarily the following order):

1. Ensure that the signature algorithm used to actually sign the CRL matches the signature algorithm indicated within the CRL, by ensuring that the issuer signature algorithm identifier in the CRL matches the algorithm identifier in the signature data.
2. Ensure that the CRL was signed by the issuer of certificate in question, verifying that the CRL has been signed with the key of the certificate issuer.
3. Ensure that the CRL has not expired[7], or not been activated yet, and that its validity period is good.
4. Ensure that if the version field is present, it is version 2. Otherwise the CRL is version 1 and must not have any extensions. However, WebSphere MQ for UNIX and Windows systems only verifies that no critical extensions are present for a version 1 CRL.
5. Ensure that the certificate in question is on the revokedCertificates field list and that the revocation date is not in the future.
6. Ensure that there are no duplicate extensions.
7. If unknown critical extensions, including critical entry extensions, are detected in the CRL, this causes identified certificates to be treated as revoked[8] (provided the CRL passes all other checks).
8. If the authorityKeyID extension in the CRL and the subjectKeyID in the CA certificate are present and if the keyIdentifier field is present within the

authorityKeyID of the CRL, match it with the CACertificate's subjectKeyID.

9. If the issuerAltName extension is marked critical, ensure that the name forms are recognized. The following general name forms are currently recognized:
   - rfc822
   - DNS
   - directory
   - URI
   - IPAddress(v4/v6)
10. If the issuingDistributionPoint extension is present in the CRL, process as follows:
    - If the issuingDistributionPoint specifies an InDirectCRL then fail the CRL validation.
    - If the issuingDistributionPoint indicates that a CRLDistributionPoint is present but no DistributionPointName is found, fail the CRL validation
    - If the issuingDistributionPoint indicates that a CRLDistributionPoint is present and specifies a DistributionPointName ensure that it is a GeneralName or LDAP format URI that matches the name given by the certificate's CRLDistributionPoint or the certificate's issuer's name. If the DistributionPointName is not a GeneralName then the CRL validation will fail.
      Note: RelativeDistinguishedNames are not supported and will fail CRL validation if encountered.

Parent topic: Certificate validation and trust policy design on UNIX and Windows systems

[1] Trust status is an administrative setting in the key database file. You can access and alter the trust status of a particular signer certificate in iKeyman. Select the required certificate from the signer list and click **View/Edit...**. The **Set the certificate as a trusted root** check box on the resulting panel indicates the trust status. You can also set Trust status using iKeycmd or GSKCapiCmd with the `-trust` flag on the **-cert -modify** command. For further information about this command, see Chapter 18, "Managing keys and certificates" in the WebSphere MQ System Administration Guide.
[2] There are no checks to ensure the subject's validity is within bounds of the issuer's validity. This is not required, and it has been shown that certificates from some CAs do not pass such a check.
[3] After they are retrieved from the database, ARLs are evaluated in exactly the same fashion as CRLs. Many CAs do not issue ARLs. However, WebSphere MQ will look for ARLs and CRLs if checking a CA certificate for revocation status.
[4] If *OCSPAuthentication* is set to `WARN`, WebSphere MQ logs the unknown revocation status and allows the connection to continue.
[5] This is a Certificate in the KeyStore a user has installed and that has Trust Status set. In the case of PKCS#11 Devices in Common Criteria mode of operation the Certificate MUST have the CKA_TRUSTED Attribute set.
[6] If no current OCSP responses are returned from the responder, WebSphere MQ will attempt to use out of date responses in determining the revocation status of a Certificate. WebSphere MQ attempts to use out of date Responses so that security will not be adversely reduced.
[7] If no current CRLs are found, WebSphere MQ for UNIX and Windows systems will attempt to use out of date CRLs to determine the revocation status of a Certificate. It is not clearly specified in RFC 3280 what action to take in the event of no current CRLs. WebSphere MQ for UNIX and Windows systems attempt to use out of date CRLs so that security will not be adversely reduced.
[8] ITU X.509 and RFC 3280 are in conflict in this case because the RFC mandates that CRLs with unknown critical extensions must fail validation. However, ITU X.509 requires that identified certificates must still be treated as revoked provided the CRL passes all other checks. WebSphere MQ for UNIX and Windows systems adopt the ITU X.509 guidance so that security will not be adversely reduced.

A potential scenario exists where the CA that issues a CRL might set an unknown critical extension to indicate that even though all other validation checks are successful, a certificate which is identified must not be considered revoked and thus not rejected by the application. In this scenario, following X.509, WebSphere MQ for UNIX and Windows systems will function in a fail-secure mode of operation. That is, they might reject certificates that the CA did not intend to be rejected and therefore might deny service to some valid users. A fail-insecure mode ignores a CRL because it has an unknown critical extension and therefore certificates that the CA intended to be revoked are still accepted. The administrator of the system should then query this behavior with the issuing CA.

6.14.21.5. Standard path validation policy

The standard path validation policy determines how the certificate, OCSP, and CRL policy types interact with each other to determine if a certificate chain is valid. Standard policy checking conforms to RFC 3280.

Path validation uses the following concepts:

- A certification path of length *n*, where the trust point or root certificate is certificate 1, and the EE is *n*.
- A set of initial policy identifiers (each comprising a sequence of policy element identifiers), that identifies one or more certificate policies, any one of which is acceptable for the purposes of certification path processing, or the special value "any-policy". Currently this is always set to "any-policy".
  Note: WebSphere® MQ for UNIX and Windows systems only supports policy identifiers that are created by WebSphere MQ for UNIX and Windows systems.
- Acceptable policy set: a set of certificate policy identifiers comprising the policy or policies recognized by the public key user, together with policies deemed equivalent through policy mapping. The initial value of the acceptable policy set is the special value "any-policy".
- Constrained subtrees: a set of root names defining a set of subtrees within which all subject names in subsequent certificates in the certification path can fall. The initial value is "unbounded".
- Excluded subtrees: a set of root names defining a set of subtrees within which no subject name in subsequent certificates in the certification path can fall. The initial value is "empty".
- Explicit policy: an integer which indicates if an explicit policy identifier is required. The integer indicates the first certificate in the path where this requirement is imposed. When set, this variable can be decreased, but cannot be increased. (That is, if a certificate in the path requires explicit policy identifiers, a later certificate cannot remove this requirement.) The initial value is n+1.
- Policy mapping: an integer which indicates if policy mapping is permitted. The integer indicates the last certificate on which policy mapping may be applied. When set, this variable can be decreased, but cannot be increased. (That is, if a certificate in the path specifies policy mapping is not permitted, it cannot be overridden by a later certificate.) The initial value is n+1.

The validation of a chain is performed in the following manner (but not necessarily the following order):

1. The information in the following paragraph is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the name of the certificate's issuer is equal to the subject name in the previous certificate, and that there is not an empty issuer name in this certificate or the previous certificate subject name. If no previous certificate exists in the path and this is the first certificate in the chain, ensure that the issuer and subject name are identical and that the trust status is set for the certificate[1].
   If the certificate does not have a subject name, the subjectAltName extension must be present and critical.
2. The information in the following paragraph is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the signature algorithm used to actually sign the certificate matches the signature algorithm indicated within the certificate, by ensuring that the issuer signature algorithm identifier in the certificate matches the algorithm identifier in the signature data.

If both the certificate's issuersUniqueID and the issuer's subjectUniqueID are present, ensure they match.

3. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the certificate was signed by the issuer, using the subject public key from the previous certificate in the path to verify the signature on the certificate. If no previous certificate exists and this is the first certificate, use the subject public key of the certificate to verify the signature on it.
4. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the certificate is a known X509 version, unique IDs are not present for version 1 certificates and extensions are not present for version 1 and version 2 certificates.
5. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the certificate has not expired, or not been activated yet, and that its validity period is good[2]
6. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that there are no unknown critical extensions, nor any duplicate extensions.
7. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   Ensure that the certificate has not been revoked. Here, the following operations apply:
   a. If the OCSP connection is enabled and a Responder Address is configured or the Certificate has a valid AuthorityInfoAccess extension specifying an HTTP format GENERALNAME_uniformResourceID check revocation status with OCSP.
      i. WebSphere MQ for UNIX and Windows systems allows the OCSP Request to be optionally signed for preconfigured responders but this has otherwise no impact on OCSP Response processing.
   b. If revocation status from 7a is undetermined the CRLDistributionPoints extension is checked for a list of X.500 distinguished name GENERALNAME_directoryname and URI GENERALNAME_uniformResourceID. If the extension is not present, the certificate's issuer's name is used. A CRL database (LDAP) is then queried for CRLs. If the certificate is not the last certificate, or if the last certificate has the basic constraint extension with the "isCA" flag turned on, the database is queried for ARL's and CRL's instead. If CRL checking is enabled, and no CRL database can be queried, the certificate is treated as revoked. Currently, the X500 directory name form and the LDAP/HTTP/FILE URI forms are the only supported name forms used to look up CRLs and ARLs15.
      Note: RelativeDistinguishedNames are not supported.
8. The following information is consistent with the basic path validation policy described in Basic path validation policy:
   If the subjectAltName extension is marked critical, ensure that the name forms are recognized. The following general name forms are currently recognized:
   - rfc822
   - DNS
   - directory
   - URI
   - IPAddress(v4/v6)
9. Ensure that the subject name and subjectAltName extension (critical or noncritical) is consistent with the constrained and excluded subtrees state variables..
10. If the EmailAddress OID is present in the subject name field as an IA5 string, and there is no subjectAltName extension, the EmailAddress must be consistent with the constrained and excluded subtrees state variable.
11. Ensure that policy information is consistent with the initial policy set:
    a. If the explicit policy state variable is less than or equal to the current certificate's numeric sequence value, a policy identifier in the certificate shall be in the initial policy set.
    b. If the policy mapping variable is less than or equal to the current certificate's numeric sequence value, the policy identifier cannot be mapped.
12. Ensure that policy information is consistent with the acceptable policy set:
    a. If the certificate policies extension is marked critical[3], the intersection of the policies extension and the acceptable policy set is non-null.
    b. The acceptable policy set is assigned the resulting intersection as its new value.
13. Ensure that the intersection of the acceptable policy set and the initial policy set is non-null. If the special Policy of anyPolicy is present then allow it only if it has not been inhibited by the inhibitAnyPolicy extension at this chain position.
14. If an inhibitAnyPolicy extension is present ensure that it is marked Critical and, if so, set the inhibitAnyPolicy state and chain position to the value of the integer value of the extension provided it is not greater than the current value. This is the number of certificates to allow with an anyPolicy Policy before disallowing the anyPolicy Policy.
15. The following steps are performed for all certificates except the last one:
    a. If the issuerAltName extension is marked critical, ensure that the name forms are recognized. The following general name forms are currently recognized:
       - rfc822
       - DNS
       - directory
       - URI
       - IPAddress(v4/v6)

       i. If the BasicConstraints extension is not present, the certificate is only valid as an EE certificate.
       ii. If the BasicConstraints extension is present, ensure that the "isCA" flag is true[4]. If the pathLength field is present, ensure the number of certificates until the last certificate is not greater than the pathLength field.
    c. If the KeyUsage extension is critical, ensure that the keyCertSign flag is on, and ensure that if the BasicConstraints extension is present, that the "isCA" flag is true[5].
    d. If a policy constraints extension is included in the certificate, modify the explicit policy and policy mapping state variables as follows:
       - i. If requireExplicitPolicy is present and has value *r*, the explicit policy state variable is set to the minimum of its current value and the sum of *r* and *i* (the current certificate in the sequence).
       - ii. If inhibitPolicyMapping is present and has value *q*, the policy mapping state variable is set to the minimum of its current value and the sum of *q* and *i* (the current certificate in the sequence).
    e. If the policyMappings extension is present (see 12(b)), ensure that it is not critical, and if policy mapping is allowed, these mappings are used to map between this certificate's policies and its signee's policies.
    f. If the nameConstraints extension is present, ensure that it is critical, and that the permitted and excluded subtrees adhere to the following rules before updating the chain's subtree's state in accordance with the algorithm described in RFC 3280 section 6.1.4 part (g):
       i. The minimum field is set to zero.
       ii. The maximum field is not present.
       iii. The base field name forms are recognized. The following general name forms are currently recognized:

- rfc822
- DNS
- directory
- URI
- IPAddress(v4/v6)

16. The ExtendedKeyUsage extension is not checked by WebSphere MQ.
17. The following information is consistent with the basic path validation policy described in Basic path validation policy:
The AuthorityKeyID extension is not used for path validation, but is used when building the certificate chain.
18. The following information is consistent with the basic path validation policy described in Basic path validation policy:
The SubjectKeyID extension is not used for path validation, but is used when building the certificate chain.
19. The following information is consistent with the basic path validation policy described in Basic path validation policy:
The PrivateKeyUsagePeriod extension is ignored by the validation engine, because it cannot determine when the CA actually signed the certificate. The extension is always non-critical and therefore can be safely ignored.

Parent topic: Certificate validation and trust policy design on UNIX and Windows systems

[1] Trust status is an administrative setting in the key database file. You can access and alter the trust status of a particular signer certificate in iKeyman. Select the required certificate from the signer list and click **View/Edit...**. The **Set the certificate as a trusted root** check box on the resulting panel indicates the trust status. You can also set Trust status using iKeycmd or GSKCapiCmd with the `-trust` flag on the **-cert -modify** command. For further information about this command, see Chapter 18, "Managing keys and certificates" in the WebSphere MQ System Administration Guide.
[2] There are no checks to ensure the subject's validity is within bounds of the issuer's validity. This is not required, and certificates from some CAs have been shown to not pass such a check.
[3] This is maintained as a legacy requirement from RFC2459 (6.1 (e)(1))
[4] "isCA" is always checked to ensure it is true to be as part of the chain building itself, however this specific test is still made.
[5] This check is in fact redundant because of step (b), but the check is still made.

6.14.22. Migrating SSL security certificates in WebSphere MQ for Windows

SSL support changed after WebSphere MQ Version 5.3. If you are upgrading form Verison 5.3, you must migrate your certificates

In WebSphere® MQ for Windows Version 5.3, support for WebSphere MQ SSL is provided using Microsoft SSL. In WebSphere MQ for Windows Version 7.0, support for WebSphere MQ SSL is provided using Global Security Kit (GSKit) SSL. This means that WebSphere MQ for Windows supports only SSL in which certificates are stored in a GSKit key database. Therefore, if you migrate from WebSphere MQ for Windows Version 5.3 to Version 7.0, you need to migrate your certificates from the Microsoft Certificate Store to a GSKit key repository. The amqtcert (Transfer Certificates) command helps to do this (see WebSphere MQ System Administration Guide for more information on using this command). The chain checker application, which is used to verify all the required certificates are there before migrating certificates from the WebSphere MQ for Windows V5.3 store to the GSKit store, is available in WebSphere MQ V5.3 Fix Pack 10 (CSD10) or later.

Migration of security certificates from WebSphere MQ Version 5.3 to Version 7.0 is described fully in WebSphere MQ Migration Information.

Parent topic: Working with SSL or TLS on UNIX and Windows systems

6.15. Working with SSL or TLS on z/OS

This information describes how you set up and work with the Secure Sockets Layer (SSL) on z/OS.

Each topic includes examples of performing each task using RACF®. You can perform similar tasks using the other external security managers.

On z/OS®, you must also set the number of server subtasks that each queue manager uses for processing SSL calls, as described in Setting the SSLTASKS parameter.

z/OS SSL support is integral to the operating system, and is known as *System SSL*. System SSL is part of the Cryptographic Services Base element of z/OS. The Cryptographic Services Base members are installed in the *pdsname*.SIEALNKE partitioned data set (PDS). When you install System SSL, ensure that you choose the appropriate options to provide the CipherSpecs that you require.

Setting the SSLTASKS parameter
Use the ALTER QMGR command to set the number of server subtasks for processing SSL calls

Setting up a key repository on z/OS
Set up a key repository at both ends of the connection. Associate each key repository with its queue manager.

Locating the key repository for a queue manager on z/OS
Use this procedure to obtain the location of your queue manager's key ring.

Specifying the key repository location for a queue manager
To specify the location of your queue manager's key ring, use the ALTER QMGR MQSC command to set your queue manager's key repository attribute.

Giving the channel initiator the correct access rights
The channel initiator (CHINIT) needs access to the key repository and to certain security profiles.

When changes to certificates or the key repository become effective on z/OS
Changes become effective when the channel initiator starts or the repository is refreshed.

Creating a self-signed personal certificate on z/OS

Use this procedure to create a self-signed personal certificate.

Requesting a personal certificate on z/OS
Apply for a personal certificate using RACF.

Creating a RACF signed personal certificate
RACF can function as a Certification Authority and issue its own CA certificate.

Adding personal certificates to a key repository on z/OS
Use this procedure to add or import a personal certificate to a key ring.

Exporting a personal certificate from a key repository on z/OS
Export the certificate using the RACDCERT command.

Deleting a personal certificate from a key repository on z/OS
Delete a personal certificate using the RACDCERT command.

Renaming a personal certificate in a key repository on z/OS
Rename a certificate using the RACDCERT command.

Associating a user ID with a digital certificate on z/OS
Where possible, WebSphere MQ uses a user ID associated with a RACF certificate as a channel user ID. Associate a user ID with a certificate by installing it under that user ID, or using a Certificate Name Filter.

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Cryptographic security protocols: TLS and SSL

Related information
ALTER QMGR

6.15.1. Setting the SSLTASKS parameter

Use the ALTER QMGR command to set the number of server subtasks for processing SSL calls

To use SSL channels, ensure that there are at least two server subtasks by setting the SSLTASKS parameter, using the ALTER QMGR command. For example:

```
ALTER QMGR SSLTASKS(5)
```

To avoid problems with storage allocation, do not set the SSLTASKS parameter to a value greater than 50.

Parent topic: Working with SSL or TLS on z/OS

6.15.2. Setting up a key repository on z/OS

Set up a key repository at both ends of the connection. Associate each key repository with its queue manager.

An SSL connection requires a *key repository* at each end of the connection. Each queue manager must have access to a key repository. Use the SSLKEYR parameter on the ALTER QMGR command to associate a key repository with a queue manager. See The SSL key repository for more information.

On z/OS®, digital certificates are stored in a *key ring* that is managed by your External Security Manager (ESM) . These digital certificates have labels, which associate the certificate with a queue manager. SSL uses these certificates for authentication purposes. All the examples that follow use RACF® commands. Equivalent commands exist for other ESM programs.

On z/OS, WebSphere® MQ uses the `ibmWebSphereMQ` prefix on a label to avoid confusion with certificates for other products. The prefix is followed by the name of the queue manager.

The key repository name for a queue manager is the name of a key ring in your RACF database. You can specify the key ring name either before or after creating the key ring.

Use the following procedure to create a new key ring for a queue manager:

1. Ensure that you have the appropriate authority to issue the RACDCERT command (see the *SecureWay™ Security Server RACF Command Language Reference* for more details).
2. Issue the following command:
   ```
   RACDCERT ID(userid1) ADDRING(ring-name)
   ```
   where:
   - *userid1* is the user ID of the channel initiator address space, or the user ID that is going to own the key ring (if the key ring is shared).
   - *ring-name* is the name you want to give to your key ring. The length of this name can be up to 237 characters. This name is case-sensitive. Specify *ring-name* in uppercase characters to avoid problems.

Making CA certificates available to a queue manager on z/OS
After you have created your key ring, connect any relevant CA certificates to it.

Parent topic: Working with SSL or TLS on z/OS

6.15.2.1. Making CA certificates available to a queue manager on z/OS

After you have created your key ring, connect any relevant CA certificates to it.

For example, to connect a CA certificate for `My CA` to your key ring, use the following command:

```
    RACDCERT ID(userid1)
    CONNECT(CERTAUTH LABEL('My CA') RING(ring-name) USAGE(CERTAUTH))
```

where *userid1* is either the channel initiator user ID or the owner of a shared key ring.

For more information about CA certificates, refer to Digital certificates.

Parent topic: Setting up a key repository on z/OS

6.15.3. Locating the key repository for a queue manager on z/OS

Use this procedure to obtain the location of your queue manager's key ring.

1. Display your queue manager's attributes, using either of the following MQSC commands:
   ```
   DISPLAY QMGR ALL
   DISPLAY QMGR SSLKEYR
   ```
2. Examine the command output for the location of the key ring.

Parent topic: Working with SSL or TLS on z/OS

6.15.4. Specifying the key repository location for a queue manager

To specify the location of your queue manager's key ring, use the ALTER QMGR MQSC command to set your queue manager's key repository attribute.

For example:

```
    ALTER QMGR SSLKEYR(CSQ1RING)
```

if the key ring is owned by the channel initiator address space, or:

```
    ALTER QMGR SSLKEYR(userid1/CSQ1RING)
```

if it is a shared key ring, where *userid1* is the user ID that owns the key ring.
Parent topic: Working with SSL or TLS on z/OS

6.15.5. Giving the channel initiator the correct access rights

The channel initiator (CHINIT) needs access to the key repository and to certain security profiles.

Granting the CHINIT access to read the key repository

Your CHINIT user ID needs read access to the IRR.DIGTCERT.LISTRING profile in the FACILITY class if you own the key repository, and update access otherwise. Grant access by using the PERMIT command with ACCESS(UPDATE) or ACCESS(READ) as appropriate:

```
    PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(userid) ACCESS(UPDATE)
```

where *userid* is the user ID of the channel initiator address space.

Granting the CHINIT read access to the appropriate CSF* profiles

Ensure your CHINIT user ID has read access to the appropriate CSF* profiles in the CSFSERV class by using the following command:

```
    PERMIT csf-resource CLASS(CSFSERV) ID(userid) ACCESS(READ)
```

where *csf-resource* is the name of the CSF* profile and *userid* is the user ID of the channel initiator address space.

Repeat this command for each of the following CSF* profile names:

- CSFPKD
- CSFPKE
- CSFPKI

- CSFDSG
- CSFDSV
- CSFKEYS

Parent topic: Working with SSL or TLS on z/OS

6.15.6. When changes to certificates or the key repository become effective on z/OS

Changes become effective when the channel initiator starts or the repository is refreshed.

Specifically, changes to the certificates in the key ring and to the key repository attribute become effective on either of the following occasions:

- When the channel initiator is started or restarted.
- When the REFRESH SECURITY TYPE(SSL) command is issued to refresh the contents of the SSL key repository.

Parent topic: Working with SSL or TLS on z/OS

6.15.7. Creating a self-signed personal certificate on z/OS®

Use this procedure to create a self-signed personal certificate.

1. Generate a certificate and a public and private key pair using the following command:
```
RACDCERT ID(userid2) GENCERT
SUBJECTSDN(CN('common-name')
           T('title')
           OU('organizational-unit')
           O('organization')
           L('locality')
           SP('state-or-province')
           C('country'))
WITHLABEL('label-name')
```
2. Connect the certificate to your key ring using the following command:
```
RACDCERT ID(userid1)
CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *label-name* must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Note that *userid1* and *userid2* can be the same ID.

Parent topic: Working with SSL or TLS on z/OS

6.15.8. Requesting a personal certificate on z/OS®

Apply for a personal certificate using RACF®.

To apply for a personal certificate, use RACF as follows:

1. Create a self-signed personal certificate, as in Creating a self-signed personal certificate on z/OS. This certificate provides the request with the attribute values for the Distinguished Name.
2. Create a PKCS #10 Base64-encoded certificate request written to a data set, using the following command:
```
RACDCERT ID(userid2) GENREQ(LABEL('label-name')) DSN(output-data-set-name)
```
   where *label-name* is the label used when creating the self-signed certificate, and *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
3. Send the data set to a Certification Authority (CA) to request a new personal certificate.
4. When the signed certificate is returned to you by the Certification Authority, add the certificate back into the RACF database, using the original label, as described in Adding personal certificates to a key repository on z/OS.

Parent topic: Working with SSL or TLS on z/OS

6.15.9. Creating a RACF signed personal certificate

RACF® can function as a Certification Authority and issue its own CA certificate.

This section uses the term *signer certificate* to denote a CA certificate issued by RACF.

The private key for the signer certificate must be in the RACF database before you carry out the following procedure:

1. Use the following command to generate a personal certificate signed by RACF, using the signer certificate contained in your RACF database:
   ```
   RACDCERT ID(userid2) GENCERT
   SUBJECTSDN(CN('common-name')
              T('title')
              OU('organizational-unit')
              O('organization')
              L('locality')
              SP('state-or-province')
              C('country'))
   WITHLABEL('label-name')
   SIGNWITH(CERTAUTH LABEL('signer-label'))
   ```
2. Connect the certificate to your key ring using the following command:
   ```
   RACDCERT ID(userid1)
   CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
   ```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *label-name* must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.
- *signer-label* is the label of your own signer certificate.

Note that *userid1* and *userid2* can be the same ID.

Parent topic: Working with SSL or TLS on z/OS

This build: January 26, 2011 11:21:27

Notices | Trademarks | Downloads | Library | Support | Feedback

6.15.10. Adding personal certificates to a key repository on z/OS

Use this procedure to add or import a personal certificate to a key ring.

After the Certification Authority sends you a new personal certificate, add it to the key ring using the following procedure:

1. Add the certificate to the RACF® database using the following command:
   ```
   RACDCERT ID(userid2) ADD(input-data-set-name) WITHLABEL('label-name')
   ```
2. Connect the certificate to your key ring using the following command:
   ```
   RACDCERT ID(userid1)
   CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
   ```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
- *ring-name* is the name you gave the key ring in Setting up a key repository on z/OS.
- *input-data-set-name* is the name of the data set containing the CA signed certificate. The data set must be cataloged and must not be a PDS or a member of a PDS. The record format (RECFM) expected by RACDCERT is VB. RACDCERT dynamically allocates and opens the data set, and reads the certificate from it as binary data.
- *label-name* is the label name that was used when you created the original request. It must be in the correct WebSphere® MQ format for a queue manager: `ibmWebSphereMQ` followed by the name of your queue manager, for example, `ibmWebSphereMQCSQ1`.

Parent topic: Working with SSL or TLS on z/OS

This build: January 26, 2011 11:21:27

Notices | Trademarks | Downloads | Library | Support | Feedback

6.15.11. Exporting a personal certificate from a key repository on z/OS

Export the certificate using the RACDCERT command.

On the system from which you want to export the certificate, use the following command:

```
RACDCERT ID(userid2) EXPORT(LABEL('label-name'))
DSN(output-data-set-name) FORMAT(CERTB64)
```

where:

- *userid2* is the user ID under which the certificate was added to the key ring.
- *label-name* is the label of the certificate you want to extract.
- *output-data-set-name* is the data set into which the certificate is placed.
- CERTB64 is a DER encoded X.509 certificate that is in Base64 format. You can choose an alternative format, for example:

  CERTDER

  DER encoded X.509 certificate in binary format

  PKCS12B64

PKCS #12 certificate in Base64 format

PKCS12DER

PKCS #12 certificate in binary format
Note that **PKCS12DER** is supported only on OS/390® V2.10 and z/OS® V1.1 and subsequent releases.

Parent topic: Working with SSL or TLS on z/OS

6.15.12. Deleting a personal certificate from a key repository on z/OS

Delete a personal certificate using the RACDCERT command.

Before deleting a personal certificate, you might want to save a copy of it. To copy your personal certificate to a data set before deleting it, follow the procedure in Exporting a personal certificate from a key repository on z/OS. Then use the following command to delete your personal certificate:

```
RACDCERT ID(userid2) DELETE(LABEL('label-name'))
```

where:

- *userid2* is the user ID under which the certificate was added to the key ring.
- *label-name* is the name of the certificate you want to delete.

Parent topic: Working with SSL or TLS on z/OS

6.15.13. Renaming a personal certificate in a key repository on z/OS

Rename a certificate using the RACDCERT command.

If you do not want a certificate with a specific label to be found, but do not want to delete it, you can rename it temporarily using the following command:

```
RACDCERT ID(userid2) LABEL('label-name') NEWLABEL('new-label-name')
```

where:

- *userid2* is the user ID under which the certificate was added to the key ring.
- *label-name* is the name of the certificate you want to rename.
- *new-label-name* is the new name of the certificate.

This can be useful when testing SSL client authentication.

Parent topic: Working with SSL or TLS on z/OS

6.15.14. Associating a user ID with a digital certificate on z/OS®

Where possible, WebSphere® MQ uses a user ID associated with a RACF® certificate as a channel user ID. Associate a user ID with a certificate by installing it under that user ID, or using a Certificate Name Filter.

When an entity at one end of an SSL channel receives a certificate from a remote connection, the entity asks RACF if there is a user ID associated with that certificate. The entity uses that user ID as the channel user ID. If there is no user ID associated with the certificate, the entity uses the user ID under which the channel initiator is running.

Associate a user ID with a certificate in either of the following ways:

- Install that certificate into the RACF database under the user ID with which you want to associate it, as described in Adding personal certificates to a key repository on z/OS.
- Use a Certificate Name Filter (CNF) to map the Distinguished Name of the subject or issuer of the certificate to the user ID, as described in Setting up a certificate name filter.

Setting up a certificate name filter
Use the RACDCERT command to define a certificate name filter (CNF), which maps a Distinguished Name to a user ID.

Parent topic: Working with SSL or TLS on z/OS

6.15.14.1. Setting up a certificate name filter

Use the RACDCERT command to define a certificate name filter (CNF), which maps a Distinguished Name to a user ID.

Perform the following steps to set up a CNF.

1. Enable CNF functions using the following command. You require update authority on the class DIGTNMAP to do this.
   ```
   SETROPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
   ```
2. Define the CNF. For example:
   ```
   RACDCERT ID(USER1) MAP WITHLABEL('filter1') TRUST
   SDNFILTER('O=IBM.C=UK') IDNFILTER('O=ExampleCA.L=Internet')
   ```
   where `USER1` is the user ID to be used when:
   - The DN of the subject has an Organization of `IBM` and a Country of `UK`.
   - The DN of the issuer has an Organization of `ExampleCA` and a Locality of `Internet`.
3. Refresh the CNF mappings:
   ```
   SETROPTS RACLIST(DIGTNMAP) REFRESH
   ```

Note:
1. If the actual certificate is stored in the RACF® database, the user ID under which it is installed is used in preference to the user ID associated with any CNF. If the certificate is not stored in the RACF database, the user ID associated with the most specific matching CNF is used. Matches of the subject DN are considered more specific than matches of the issuer DN.
2. Changes to CNFs do not apply until you refresh the CNF mappings.
3. A DN matches the DN filter in a CNF only if the DN filter is identical to the *least significant portion* of the DN. The least significant portion of the DN comprises the attributes that are usually listed at the right-most end of the DN, but which appear at the beginning of the certificate.
   For example, consider the SDNFILTER `'O=IBM.C=UK'`. A subject DN of `'CN=QM1.O=IBM.C=UK'` matches that filter, but a subject DN of `'CN=QM1.O=IBM.L=Hursley.C=UK'` does not match that filter.
   The least significant portion of some certificates can contain fields that do not match the DN filter. Consider excluding these certificates by specifying a DN pattern in the SSLPEER pattern on the DEFINE CHANNEL command.
4. If the most specific matching CNF is defined to RACF as NOTRUST, the entity uses the user ID under which the channel initiator is running.
5. RACF uses the `'.'` character as a separator. WebSphere® MQ uses either a comma or a semicolon.

You can define CNFs to ensure that the entity never sets the channel user ID to the default, which is the user ID under which the channel initiator is running. For each CA certificate in the key ring associated with the entity, define a CNF with an IDNFILTER that exactly matches the subject DN of that CA certificate. This ensures that all certificates that the entity might use match at least one of these CNFs. This is because all such certificates must either be connected to the key ring associated with the entity, or must be issued by a CA for which a certificate is connected to the key ring associated with the entity.

Refer to the *SecureWay™ Security Server RACF Security Administrator's Guide* for more information about the commands you use to manipulate CNFs.

Parent topic: Associating a user ID with a digital certificate on z/OS

6.16. Working with revoked certificates

Digital certificates can be revoked by Certification Authorities. You can check the revocation status of certificates using OCSP, or CRLs on LDAP servers, depending on platform.

During the SSL handshake, the communicating partners authenticate each other with digital certificates. Authentication can include a check that the certificate received can still be trusted. Certification Authorities (CAs) revoke certificates for various reasons, including:

- The owner has moved to a different organization
- The private key is no longer secret

CAs publish revoked personal certificates in a Certificate Revocation List (CRL). CA certificates that have been revoked are published in an Authority Revocation List (ARL).

On UNIX systems and Windows, WebSphere® MQ SSL support checks for revoked certificates using OCSP (Online Certificate Status Protocol) or using CRLs and ARLs on LDAP (Lightweight Directory Access Protocol) servers. OCSP is the preferred method. However, WebSphere MQ classes for Java and WebSphere MQ classes for JMS cannot use OCSP.

On z/Os and i5/OS® WebSphere MQ SSL support checks for revoked certificates using CRLs and ARLs on LDAP servers only.

For more information about Certification Authorities, see Digital certificates.

Revoked certificates and OCSP
WebSphere MQ determines which Online Certificate Status Protocol (OCSP) responder to use, and handles the response received. You might have to take steps to make the OCSP responder accessible.

Working with Certificate Revocation Lists and Authority Revocation Lists
WebSphere MQ's support for CRLs and ARLs varies by platform.

Manipulating authentication information objects
You can manipulate authentication information objects using MQSC or PCF commands, or the Websphere MQ Explorer.

Parent topic: WebSphere MQ support for SSL and TLS

6.16.1. Revoked certificates and OCSP

WebSphere® MQ determines which Online Certificate Status Protocol (OCSP) responder to use, and handles the response received. You might have to take steps to make the OCSP responder accessible.

Note: This information applies only to WebSphere MQ on UNIX systems and Windows.

To check the revocation status of a digital certificate using OCSP, WebSphere MQ determines which OCSP responder to contact in one of two ways:

- Using the AuthorityInfoAccess (AIA) certificate extension in the certificate to be checked.
- ›Using a URL specified in an authentication information object or specified by a client application.‹

A URL specified in an authentication information object or by a client application takes priority over a URL in an AIA certificate extension.

›The URL of the OCSP responder might lie behind a firewall; if so, reconfigure the firewall so the OCSP responder can be accessed or set up an OCSP proxy server. Specify the name of the proxy server by using the SSLHTTPProxyName variable in the SSL stanza. On client systems, you can also specify the name of the proxy server by using the environment variable MQSSLPROXY. For more details, see the related information.‹

›If you are not concerned whether TLS or SSL certificates are revoked, perhaps because you are running in a test environment, you can set OCSPCheckExtensions to NO in the SSL stanza. If you set this variable, any AIA certificate extension is ignored. This solution is unlikely to be acceptable in a production environment, where you probably do not want to allow access from users presenting revoked certificates.‹

The call to access the OCSP responder can result in one of the following three outcomes:

Good

The certificate is valid.

Revoked

The certificate is revoked.

Unknown

This outcome can arise for one of three reasons:

- WebSphere MQ cannot access the OCSP responder.
- The OCSP responder has sent a response, but WebSphere MQ cannot verify the digital signature of the response.
- The OCSP responder has sent a response that indicates that it has no revocation data for the certificate.

OCSP outcome `Unknown`

›If WebSphere MQ receives an OCSP outcome of `Unknown`, its behavior depends on the setting of the OCSPAuthentication attribute. For queue managers, this attribute is held in the SSL stanza of the `qm.ini` file for UNIX systems, or the Windows registry. It can be set using the WebSphere MQ Explorer. For clients, it is held in the SSL stanza of the client configuration file. ‹

›If an outcome of `Unknown` is received and OCSPAuthentication is set to REQUIRED (the default value), WebSphere MQ rejects the connection and issues an error message of type AMQ9716. If queue manager SSL event messages are enabled, an SSL event message of type MQRC_CHANNEL_SSL_ERROR with ReasonQualifier set to MQRQ_SSL_HANDSHAKE_ERROR is generated.‹

›If an outcome of `Unknown` is received and OCSPAuthentication is set to OPTIONAL, WebSphere MQ allows the SSL channel to start and no warnings or SSL event messages are generated.‹

›If an outcome of `Unknown` is received and OCSPAuthentication is set to WARN, the SSL channel starts but WebSphere MQ issues a warning message of type AMQ9717 in the error log. If queue manager SSL event messages are enabled, an SSL event message of type MQRC_CHANNEL_SSL_WARNING with ReasonQualifier set to MQRQ_SSL_UNKNOWN_REVOCATION is generated. ‹

›

Digital signing of OCSP responses

An OCSP responder can sign its responses in any of three ways. Your responder will inform you which method is used.

- The OCSP response can be digitally signed using the same CA certificate that issued the certificate that you are checking. In this case, you do not need to set up any additional certificate; the steps you have already taken to establish SSL connectivity are sufficient to verify the OCSP response.
- The OCSP response can be digitally signed using another certificate signed by the same certificate authority (CA) that issued the certificate you are checking. The signing certificate is flowed together with the OCSP response in this case. The certificate flowed from the OCSP responder must have an Extended Key Usage Extension set to `id-kp-OCSPSigning` so that it can be trusted for this purpose. Because the OCSP response is flowed with the certificate which signed it (and that certificate is signed by a CA which is already trusted for SSL connectivity), no additional certificate setup is required.
- The OCSP response can be digitally signed using another certificate which is not directly related to the certificate you are checking. In this case, the OCSP Response is signed by a certificate issued by the OCSP responder itself. You must add a copy of the OCSP responder certificate to the key database of the client or queue manager which performs the OCSP checking; see Adding a CA certificate (or the public part of a self-signed certificate) into a key repository, on UNIX systems or Windows. When a CA certificate is added, by default it is added as a trusted root, which is the required setting in this context. If this certificate is not added, WebSphere MQ cannot verify the digital signature on the OCSP response and the OCSP check results in an Unknown outcome, which might cause WebSphere MQ to close the channel, depending on the value of OCSPAuthentication.

‹

Parent topic: Working with revoked certificates

Related concepts
Manipulating authentication information objects

Related information
SSL on WebSphere MQ clients
SSL stanza of the queue manager configuration file
SSL stanza of the client configuration file
MQSSLPROXY environment variable

6.16.2. Working with Certificate Revocation Lists and Authority Revocation Lists

WebSphere® MQ's support for CRLs and ARLs varies by platform.

CRL and ARL support on each platform is as follows:

- On z/OS®, System SSL supports CRLs and ARLs stored in LDAP servers by the Tivoli® Public Key Infrastructure product.
- On other platforms, the CRL and ARL support complies with PKIX X.509 V2 CRL profile recommendations.

WebSphere MQ maintains a cache of CRLs and ARLs that have been accessed in the preceding 12 hours.

When a queue manager or WebSphere MQ client receives a certificate, it checks the CRL to confirm that the certificate is still valid. WebSphere MQ first checks in the cache, if there is a cache. If the CRL is not in the cache, WebSphere MQ interrogates the LDAP CRL server locations in the order they appear in the namelist of authentication information objects specified by the *SSLCRLNamelist* attribute, until WebSphere MQ finds an available CRL. If the namelist is not specified, or is specified with a blank value, CRLs are not checked.

For more information about LDAP, refer to the WebSphere MQ Application Programming Guide.

Setting up LDAP servers
Configure the LDAP Directory Information Tree structure to reflect the hierarchy of Distinguished Names of CAs. Do this using LDAP Data Interchange Format files.

Accessing CRLs and ARLs with a queue manager
A queue manger is associated with one or more authentication information objects, which hold the address of an LDAP CRL server. Websphere MQ on i5/OS® behaves differently from other platforms.

Accessing CRLs and ARLs with a WebSphere MQ client
You have three options for specifying the LDAP servers that hold CRLs for checking by a WebSphere MQ client.

Accessing CRLs and ARLs with WebSphere MQ classes for Java and WebSphere MQ classes for JMS
WebSphere MQ classes for Java and WebSphere MQ classes for JMS access CRLs differently from other platforms.

Parent topic: Working with revoked certificates

6.16.2.1. Setting up LDAP servers

Configure the LDAP Directory Information Tree structure to reflect the hierarchy of Distinguished Names of CAs. Do this using LDAP Data Interchange Format files.

Configure the LDAP Directory Information Tree (DIT) structure to use the hierarchy corresponding to the Distinguished Names of the CAs that issue the certificates and CRLs. You can set up the DIT structure with a file that uses the LDAP Data Interchange Format (LDIF). You can also use LDIF files to update a directory.

LDIF files are ASCII text files that contain the information required to define objects within an LDAP directory. LDIF files contain one or more entries, each of which comprises a Distinguished Name, at least one object class definition and, optionally, multiple attribute definitions.

The `certificateRevocationList;binary` attribute contains a list, in binary form, of revoked user certificates. The `authorityRevocationList;binary` attribute contains a binary list of CA certificates that have been revoked. For use with WebSphere® MQ SSL, the binary data for these attributes must conform to DER (Definite Encoding Rules) format. For more information about LDIF files, refer to the documentation provided with your LDAP server.

Figure 1 shows a sample LDIF file that you might create as input to your LDAP server to load the CRLs and ARLs issued by CA1, which is an imaginary Certification Authority with the Distinguished Name "CN=CA1, OU=Test, O=IBM, C=GB", set up by the Test organization within IBM®.

Figure 1. Sample LDIF file for a Certification Authority. This might vary from implementation to implementation.

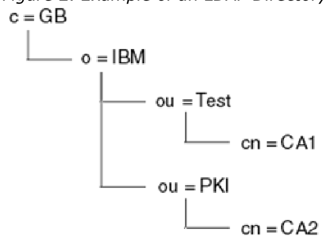
```
dn: o=IBM, c=GB
o: IBM
objectclass: top
objectclass: organization

dn: ou=Test, o=IBM, c=GB
ou: Test
objectclass: organizationalUnit

dn: cn=CA1, ou=Test, o=IBM, c=GB
cn: CA1
objectclass: cRLDistributionPoint
objectclass: certificationAuthority
authorityRevocationList;binary:: (DER format data)
certificateRevocationList;binary:: (DER format data)
caCertificate;binary:: (DER format data)
```

Figure 2 shows the DIT structure that your LDAP server creates when you load the sample LDIF file shown in Figure 1 together with a similar file for CA2, an imaginary Certification Authority set up by the PKI organization, also within IBM.

Figure 2. Example of an LDAP Directory Information Tree structure



WebSphere MQ checks both CRLs and ARLs.

Note: Ensure that the access control list for your LDAP server allows authorized users to read, search, and compare the entries that hold the CRLs and ARLs. WebSphere MQ accesses the LDAP server using the LDAPUSER and LDAPPWD properties of the AUTHINFO object.

Configuring and updating LDAP servers
Use this procedure to configure or update your LDAP server.

Parent topic: Working with Certificate Revocation Lists and Authority Revocation Lists

6.16.2.1.1. Configuring and updating LDAP servers

Use this procedure to configure or update your LDAP server.

1. Obtain the CRLs and ARLs in DER format from your Certification Authority, or Authorities.
2. Using a text editor or the tool provided with your LDAP server, create one or more LDIF files that contain the Distinguished Name of the CA and the required object class definitions. Copy the DER format data into the LDIF file as the values of either the `certificateRevocationList;binary` attribute for CRLs, the `authorityRevocationList;binary` attribute for ARLs , or both.
3. Start your LDAP server.
4. Add the entries from the LDIF file or files you created at step 2.

After you have configured your LDAP CRL server, check that it is set up correctly. First, try using a certificate that is not revoked on the channel, and check that the channel starts correctly. Then use a certificate that is revoked, and check that the channel fails to start.

Obtain updated CRLs from the Certification Authorities frequently. Consider doing this on your LDAP servers every 12 hours.

Parent topic: Setting up LDAP servers

6.16.2.2. Accessing CRLs and ARLs with a queue manager

A queue manger is associated with one or more authentication information objects, which hold the address of an LDAP CRL server. Websphere MQ on i5/OS® behaves differently from other platforms.

Note that in this section, information about Certificate Revocation Lists (CRLs) also applies to Authority Revocation Lists (ARLs).

You tell the queue manager how to access CRLs by supplying the queue manager with authentication information objects, each of which holds the address of an LDAP CRL server. The authentication information objects are held in a namelist, which is specified in the *SSLCRLNamelist* queue manager attribute.

In the following example, MQSC is used to specify the parameters:

1. Define authentication information objects using the DEFINE AUTHINFO MQSC command, with the AUTHTYPE parameter set to CRLLDAP. On i5/OS, you can also use the CRTMQMAUTI CL command.
   The value CRLLDAP for the AUTHTYPE parameter indicates that CRLs are accessed on LDAP servers. Each authentication information object with type CRLLDAP that you create holds the address of an LDAP server. When you have more than one authentication information object, the LDAP servers to which they point *must* contain identical information. This provides continuity of service if one or more LDAP servers fail.
   Additionally, on z/OS® only, all LDAP servers must be accessed using the same user ID and password. The user ID and password used are those specified in the first AUTHINFO object in the namelist.
   On all platforms, the user Id and password are sent to the LDAP server unencrypted.
2. Using the DEFINE NAMELIST MQSC command, define a namelist for the names of your authentication information objects. On z/OS, ensure that the NLTYPE namelist attribute is set to AUTHINFO.
3. Using the ALTER QMGR MQSC command, supply the namelist to the queue manager. For example:
   ```
   ALTER QMGR SSLCRLNL(sslcrlnlname)
   ```
   where `sslcrlnlname` is your namelist of authentication information objects.
   This command sets a queue manager attribute called *SSLCRLNamelist*. The queue manager's initial value for this attribute is blank.

On i5/OS, you can specify authentication information objects, but the queue manager uses neither authentication information objects nor a namelist of authentication information objects. Only WebSphere® MQ clients that use a client connection table generated by an i5/OS queue manager use the authentication information specified for that i5/OS queue manager. The *SSLCRLNamelist* queue manager attribute on i5/OS determines what authentication information such clients use. See Accessing CRLs and ARLs on i5/OS for information about telling an i5/OS queue manager how to access CRLs.

You can add up to 10 connections to alternative LDAP servers to the namelist, to ensure continuity of service if one or more LDAP servers fail. Note that the LDAP servers *must* contain identical information.

Accessing CRLs and ARLs on i5/OS
Use this procedure to access CRLs or ARLs on i5/OS.

Accessing CRLs and ARLs using WebSphere MQ Explorer
You can use WebSphere MQ Explorer to tell a queue manager how to access CRLs.

Parent topic: Working with Certificate Revocation Lists and Authority Revocation Lists

6.16.2.2.1. Accessing CRLs and ARLs on i5/OS

Use this procedure to access CRLs or ARLs on i5/OS.

Note that in this section, information about Certificate Revocation Lists (CRLs) also applies to Authority Revocation Lists (ARLs).

Follow these steps to set up a CRL location for a specific certificate on i5/OS®:

1. Access the DCM interface, as described in Accessing DCM.
2. In the **Manage CRL locations** task category in the navigation panel, click **Add CRL location**. The Manage CRL Locations page is displayed in the task frame.
3. In the **CRL Location Name** field, type a CRL location name, for example `LDAP Server #1`
4. In the **LDAP Server** field, type the LDAP server name.
5. In the **Use Secure Sockets Layer (SSL)** field, select **Yes** if you want to connect to the LDAP server using SSL. Otherwise, select **No**.
6. In the **Port Number** field, type a port number for the LDAP server, for example `389`.
7. If your LDAP server does not allow anonymous users to query the directory, type a login distinguished name for the server in the **login distinguished name** field.
8. Click **OK**. DCM informs you that it has created the CRL location.
9. In the navigation panel, click **Select a Certificate Store**. The Select a Certificate Store page is displayed in the task frame.
10. Select the **Other System Certificate Store** check box and click **Continue**. The Certificate Store and Password page is displayed.
11. In the **Certificate store path and filename** field, type the IFS path and file name you set when Creating a certificate store on i5/OS.
12. Type a password in the **Certificate Store Password** field. Click **Continue**. The Current Certificate Store page is displayed in the task frame.
13. In the **Manage Certificates** task category in the navigation panel, click **Update CRL location assignment**. The CRL Location Assignment page is displayed in the task frame.
14. Select the radio button for the CA certificate to which you want to assign the CRL location. Click **Update CRL Location Assignment**. The Update CRL Location Assignment page is displayed in the task frame.
15. Select the radio button for the CRL location which you want to assign to the certificate. Click **Update Assignment**. DCM informs you that it has updated the assignment.

Note that DCM allows you to assign a different LDAP server by Certification Authority.
Parent topic: Accessing CRLs and ARLs with a queue manager

6.16.2.2.2. Accessing CRLs and ARLs using WebSphere MQ Explorer

You can use WebSphere® MQ Explorer to tell a queue manager how to access CRLs.

Note that in this section, information about Certificate Revocation Lists (CRLs) also applies to Authority Revocation Lists (ARLs).

Use the following procedure to set up an LDAP connection to a CRL:

1. Ensure that you have started your queue manager.
2. In WebSphere MQ Explorer, expand the **Advanced** folder of your queue manager.
3. Right-click the **Authentication Information** folder and click **New -> Authentication Information**. In the property sheet that opens:
   a. On the first page **Create Authentication Information**, enter a name for the CRL(LDAP) object.
   b. On the **General** page of **Change Properties**, select the connection type. Optionally you can enter a description.
   c. Select the **CRL(LDAP)** page of **Change Properties**.
   d. Enter the LDAP server name as either the network name or the IP address.
   e. If the server requires login details, provide a user ID and if necessary a password.
   f. Click **OK**.
4. Right-click the **Namelists** folder and click **New -> Namelist**. In the property sheet that opens:
   a. Type a name for the namelist.
   b. Add the name of the CRL(LDAP) object (from step 3.a) to the list.
   c. Click **OK**.
5. Right-click the queue manager, select **Properties**, and select the **SSL** page:
   a. Select the **Check certificates received by this queue manager against Certification Revocation Lists** check box.
   b. Type the name of the namelist (from step 4.a) in the **CRL Namelist** field.

Parent topic: Accessing CRLs and ARLs with a queue manager

6.16.2.3. Accessing CRLs and ARLs with a WebSphere MQ client

You have three options for specifying the LDAP servers that hold CRLs for checking by a WebSphere® MQ client.

Note that in this section, information about Certificate Revocation Lists (CRLs) also applies to Authority Revocation Lists (ARLs).

The three ways of specifying the LDAP servers are as follows:

- Using a channel definition table
- Using the SSL configuration options structure, MQSCO, on an MQCONNX call
- Using the Active Directory (on Windows systems with Active Directory support)

For more details, refer to the related information.

You can include up to 10 connections to alternative LDAP servers to ensure continuity of service if one or more LDAP servers fail. Note that the LDAP servers *must* contain identical information.

You cannot access LDAP CRLs from a WebSphere MQ client channel running on Linux (zSeries® platform).

Parent topic: Working with Certificate Revocation Lists and Authority Revocation Lists

Related information
Specifying the location of an OCSP responder, and of LDAP servers that hold CRLs
setmqcrl

6.16.2.4. Accessing CRLs and ARLs with WebSphere MQ classes for Java and WebSphere MQ classes for JMS

WebSphere® MQ classes for Java and WebSphere MQ classes for JMS access CRLs differently from other platforms.

For information about working with CRLs and ARLs with WebSphere MQ classes for Java, see Using certificate revocation lists

For information about working with CRLs and ARLs with WebSphere MQ classes for JMS, see SSLCERTSTORES object property

Parent topic: Working with Certificate Revocation Lists and Authority Revocation Lists

6.16.3. Manipulating authentication information objects

You can manipulate authentication information objects using MQSC or PCF commands, or the Websphere MQ Explorer.

The following MQSC commands act on authentication information objects:

- DEFINE AUTHINFO
- ALTER AUTHINFO
- DELETE AUTHINFO
- DISPLAY AUTHINFO

For a complete description of these commands, see WebSphere MQ Script (MQSC) Command Reference.

The following Programmable Command Format (PCF) commands act on authentication information objects:

- Create Authentication Information
- Copy Authentication Information
- Change Authentication Information
- Delete Authentication Information
- Inquire Authentication Information
- Inquire Authentication Information Names

For a complete description of these commands, see WebSphere MQ Programmable Command Formats and Administration Interface.

On platforms where it is available, you can also use the WebSphere MQ Explorer.

Parent topic: Working with revoked certificates

6.17. Working with CipherSpecs

This collection of topics provides information about using CipherSpecs in WebSphere® MQ.

The CipherSpec identifies the combination of encryption algorithm and hash function used by an SSL or TLS connection. A CipherSpec forms part of a CipherSuite, which identifies the key exchange and authentication mechanism as well as the encryption and hash function algorithms.

WebSphere MQ supports both SSL and TLS CipherSpecs.

WebSphere MQ supports only the RSA key exchange and authentication algorithms. The size of the key used during the SSL handshake can depend on the digital certificate you use, but some of the CipherSpecs supported by WebSphere MQ include a specification of the handshake key size. Larger handshake key sizes provide stronger authentication. With smaller key sizes, the handshake is faster.

For more information, refer to CipherSuites and CipherSpecs and An overview of the SSL handshake.

Specifying CipherSpecs
Specify the CipherSpec by using the SSLCIPH parameter in either the DEFINE CHANNEL MQSC command or the ALTER CHANNEL MQSC command.

Understanding CipherSpec mismatches
Both ends of a WebSphere MQ SSL channel must use the same CipherSpec. Mismatches can be detected during the SSL handshake or during channel startup.

Parent topic: WebSphere MQ support for SSL and TLS

6.17.1. Specifying CipherSpecs

Specify the CipherSpec by using the SSLCIPH parameter in either the DEFINE CHANNEL MQSC command or the ALTER CHANNEL MQSC command.

You can choose from the CipherSpecs listed in Table 1:

Table 1. CipherSpecs that can be used with WebSphere MQ SSL and TLS support

CipherSpec name	**Protocol used**	**Hash algorithm**	**Encryption algorithm**	**Encryption bits**	**FIPS on Windows and UNIX platforms** 1
NULL_MD5	SSL 3.0	MD5	None	0	No
NULL_SHA	SSL 3.0	SHA-1	None	0	No
RC4_MD5_EXPORT	SSL 3.0	MD5	RC4	40	No
RC4_MD5_US	SSL 3.0	MD5	RC4	128	No
RC4_SHA_US	SSL 3.0	SHA-1	RC4	128	No
RC2_MD5_EXPORT	SSL 3.0	MD5	RC2	40	No
DES_SHA_EXPORT	SSL 3.0	SHA-1	DES	56	No
RC4_56_SHA_EXPORT1024 **Note:** 1. Not available for z/OS® or i5/OS® 2. Specifies a 1024–bit handshake key size	SSL 3.0	SHA-1	RC4	56	No
DES_SHA_EXPORT1024 **Note:** 1. Not available for z/OS or i5/OS 2. Specifies a 1024–bit handshake key size	SSL 3.0	SHA-1	DES	56	No
TRIPLE_DES_SHA_US	SSL 3.0	SHA-1	3DES	168	No
TLS_RSA_WITH_AES_128_CBC_SHA	TLS 1.0	SHA-1	AES	128	Yes, with GSKit v7
TLS_RSA_WITH_AES_256_CBC_SHA **Note:** This CipherSpec cannot be used to secure a connection from the WebSphere MQ Explorer to a queue manager.	TLS 1.0	SHA-1	AES	256	Yes, with GSKit v7
TLS_RSA_WITH_AES_128_CBC_SHA256 **Notes:** 1. Available only on Windows and UNIX platforms. 2. Available only with GSKit v8. See Alternative SSL and TLS support for Windows, UNIX, and Linux systems.	TLS 1.2	SHA-256	AES	128	Yes, with GSKit v8
TLS_RSA_WITH_AES_256_CBC_SHA256 **Notes:** 1. Available only on Windows and UNIX platforms. 2. Available only with GSKit v8. See Alternative SSL and TLS support for Windows, UNIX, and Linux systems.	TLS 1.2	SHA-256	AES	256	Yes, with GSKit v8
AES_SHA_US **Note:** Available on i5/OS only	SSL 3.0	SHA-1	AES	128	No
TLS_RSA_WITH_DES_CBC_SHA **Note:** Not available for z/OS	TLS 1.0	SHA-1	DES	56	No[2]
TLS_RSA_WITH_3DES_EDE_CBC_SHA **Note:** Not available for z/OS	TLS 1.0	SHA-1	3DES	168	Yes, with GSKit v7, and with GSKit v8 at TLS 1.0
FIPS_WITH_DES_CBC_SHA **Note:** Available only on Windows and UNIX platforms	SSL 3.0	SHA-1	DES	56	No[3]
FIPS_WITH_3DES_EDE_CBC_SHA **Note:** Available only on Windows and UNIX platforms	SSL 3.0	SHA-1	3DES	168	Yes, with GSKit v7
TLS_RSA_WITH_NULL_MD5 **Note:** Available on i5/OS only	TLS 1.0	MD5	None	0	No
TLS_RSA_WITH_NULL_SHA **Note:** Available on i5/OS only	TLS 1.0	SHA-1	None	0	No
TLS_RSA_WITH_NULL_SHA256 **Notes:** 1. Available only on Windows and UNIX platforms. 2. Available only with GSKit v8. See Alternative SSL and	TLS 1.2	SHA-256	None	0	No

TLS support for Windows, UNIX, and Linux systems.					
TLS_RSA_EXPORT_WITH_RC2_40_MD5 **Note:** Available on i5/OS only	TLS 1.0	MD5	RC2	40	No
TLS_RSA_EXPORT_WITH_RC4_40_MD5 **Note:** Available on i5/OS only	TLS 1.0	MD5	RC4	40	No
TLS_RSA_WITH_RC4_128_MD5 **Note:** Available on i5/OS only	TLS 1.0	MD5	RC4	128	No

Note:
1. Is the CipherSpec FIPS-certified on a FIPS-certified platform? See Federal Information Processing Standards (FIPS) for an explanation of FIPS.
   Different CipherSpecs are FIPS-certified depending on whether you are using GSKit Version 7 or GSKit Version 8. See Alternative SSL and TLS support for Windows, UNIX, and Linux systems for more information.
2. This CipherSpec was FIPS 140-2 certified before 19th May 2007.
3. This CipherSpec was FIPS 140-2 certified before 19th May 2007. The name FIPS_WITH_DES_CBC_SHA is historical and reflects the fact that this CipherSpec was previously FIPS-compliant.

Note: On i5/OS V5R3, installation of AC3 is a prerequisite for the use of SSL, but installation of AC3 is not a prerequisite on i5/OS releases later than V5R3.

When you request a personal certificate, you specify a key size for the public and private key pair. The key size that is used during the SSL handshake can depend on the size stored in the certificate and on the CipherSpec:

- On UNIX systems, Windows systems, and z/OS, when a CipherSpec name includes `_EXPORT`, the maximum handshake key size is 512 bits. If either of the certificates exchanged during the SSL handshake has a key size greater than 512 bits, a temporary 512-bit key is generated for use during the handshake.
- On UNIX and Windows systems, when a CipherSpec name includes `_EXPORT1024`, the handshake key size is 1024 bits.
- Otherwise the handshake key size is the size stored in the certificate.

Obtaining information about CipherSpecs using WebSphere MQ Explorer
You can use WebSphere MQ Explorer to display descriptions of CipherSpecs.

Alternatives for specifying CipherSpecs
For those platforms where the operating system provides the SSL support, your system might support new CipherSpecs. You can specify a new CipherSpec with the SSLCIPH parameter, but the value you supply depends on your platform.

Specifying a CipherSpec for a WebSphere MQ client
You have three options for specifying a CipherSpec for a WebSphere MQ client.

Specifying a CipherSuite with WebSphere MQ classes for Java and WebSphere MQ classes for JMS
WebSphere MQ classes for Java and WebSphere MQ classes for JMS specify CipherSuites differently from other platforms.

Parent topic: Working with CipherSpecs

6.17.1.1. Obtaining information about CipherSpecs using WebSphere MQ Explorer

You can use WebSphere® MQ Explorer to display descriptions of CipherSpecs.

Use the following procedure to obtain information about the CipherSpecs in Table 1:

1. Open **WebSphere MQ Explorer** and expand the **Queue Managers** folder.
2. Ensure that you have started your queue manager.
3. Select the queue manager you want to work with and click **Advanced –> Channels**.
4. Right–click the channel you want to work with and select **Properties**.
5. Select the **SSL** property page.
6. Select from the list the CipherSpec you want to work with. A description appears in the window below the list.

Parent topic: Specifying CipherSpecs

6.17.1.2. Alternatives for specifying CipherSpecs

For those platforms where the operating system provides the SSL support, your system might support new CipherSpecs. You can specify a new CipherSpec with the SSLCIPH parameter, but the value you supply depends on your platform.

Note: This section does not apply to UNIX or Windows systems, because the CipherSpecs are provided with the WebSphere® MQ product, so new CipherSpecs do not become available after shipment.

For those platforms where the operating system provides the SSL support, your system might support new CipherSpecs that are not included in Table 1. You can specify a new CipherSpec with the SSLCIPH parameter, but the value you supply depends on your platform. In all cases the specification *must* correspond to an SSL CipherSpec that is both valid and supported by the version of SSL your system is running.

i5/OS®
A two-character string representing a hexadecimal value.

For more information about the permitted values, refer to the appropriate information center (search on *cipher_spec*):

- For V5R3, the *iSeries® Information Center* at http://publib.boulder.ibm.com/infocenter/iseries/v5r3/index.jsp
- For V5R4, the *i5/OS Information Center* at http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp

You can use either the CHGMQMCHL or the CRTMQMCHL command to specify the value, for example:

```
CRTMQMCHL CHLNAME('channnel name') SSLCIPH('hexadecimal value')
```

You can also use the ALTER QMGR MQSC command to set the `SSLCIPH` parameter.

z/OS®

A two-character string representing a hexadecimal value. The hexadecimal codes correspond to the values defined in the SSL protocol.

For more information, refer to the description of gsk_environment_open() in the API reference chapter of *z/OS System SSL Programming*, SC24-5901, where there is a list of all the supported SSL V3.0 and TLS V1.0 cipher specifications in the form of 2-digit hexadecimal codes.

Considerations for WebSphere MQ clusters

With WebSphere MQ clusters it is safest to use the CipherSpec names in Table 1. If you use an alternative specification, be aware that the specification might not be valid on other platforms. For more information, refer to the WebSphere MQ Queue Manager Clusters book.

Parent topic: Specifying CipherSpecs

6.17.1.3. Specifying a CipherSpec for a WebSphere MQ client

You have three options for specifying a CipherSpec for a WebSphere® MQ client.

These options are as follows:

- Using a channel definition table
- Using the SSL configuration options structure, MQSCO, on an MQCONNX call
- Using the Active Directory (on Windows systems with Active Directory support)

Parent topic: Specifying CipherSpecs

Related information
Specifying that an MQI channel uses SSL

6.17.1.4. Specifying a CipherSuite with WebSphere MQ classes for Java and WebSphere MQ classes for JMS

WebSphere® MQ classes for Java and WebSphere MQ classes for JMS specify CipherSuites differently from other platforms.

For information about specifying a CipherSuite with WebSphere MQ classes for Java, refer to Secure Sockets Layer (SSL) support

For information about specifying a CipherSuite with WebSphere MQ classes for JMS, refer to Using Secure Sockets Layer (SSL) with WebSphere MQ classes for JMS

Parent topic: Specifying CipherSpecs

Related concepts
Cryptographic security protocols: TLS and SSL

6.17.2. Understanding CipherSpec mismatches

Both ends of a WebSphere® MQ SSL channel must use the same CipherSpec. Mismatches can be detected during the SSL handshake or during channel startup.

A CipherSpec identifies the combination of the encryption algorithm and hash function. Both ends of a WebSphere MQ SSL channel must use the same CipherSpec, although they can specify that CipherSpec in a different manner. Mismatches can be detected at two stages:

During the SSL handshake

The SSL handshake fails when the CipherSpec specified by the SSL client is unacceptable to the SSL support at the SSL server end of the connection. A CipherSpec failure during the SSL handshake arises when the SSL client proposes a CipherSpec that is not supported by the SSL provision on the SSL server. For example, when an SSL client running on AIX® proposes the `DES_SHA_EXPORT1024` CipherSpec to an SSL server running on i5/OS®.

During channel startup

Channel startup fails when there is a mismatch between the CipherSpec defined for the responding end of the channel and the CipherSpec defined for the calling end of channel. Channel startup also fails when only one end of the channel defines a CipherSpec.

See Specifying CipherSpecs for more information.

Note: If Global Server Certificates are used, a mismatch can be detected during channel startup even if the CipherSpecs specified on both channel definitions match.

Global Server Certificates are a special type of certificate which require that a minimum level of encryption is established on all the communications links with which they are used. If the CipherSpec requested by the WebSphere MQ channel configuration does not meet this requirement, the CipherSpec is renegotiated during the SSL handshake. This is detected as a failure during WebSphere MQ channel startup as the CipherSpec no longer matches the one specified on the channel.

In this case, change the CipherSpec at both sides of the channel to one which meets the requirements of the Global Server Certificate. To establish whether a certificate that has been issued to you is a Global Server Certificate, contact the certificate authority which issued that certificate.

SSL servers do not detect mismatches when an SSL client channel on UNIX or Windows specifies the `DES_SHA_EXPORT1024` CipherSpec, and the corresponding SSL server channel on UNIX or Windows is using the `DES_SHA_EXPORT` CipherSpec. In this case, the channel runs normally.

Parent topic: Working with CipherSpecs

6.18. WebSphere MQ rules for SSLPEER values

The SSLPEER attribute is used to check the Distinguished Name (DN) of the certificate from the peer queue manager or client at the other end of a WebSphere® MQ channel. WebSphere® MQ uses certain rules when comparing these values

When SSLPEER values are compared with DNs, the rules for specifying and matching attribute values are:

1. You can use either a comma or a semicolon as a separator.
2. Spaces before or after the separator are ignored. For example:
   ```
   CN=John Smith, O=IBM ,OU=Test , C=GB
   ```
3. The values of attribute types `CN, T, O, OU, L, ST, SP, S, C` are text strings that usually include only the following:
   - Uppercase and lowercase alphabetic characters `A` through `Z` and `a` through `z`
   - Numeric characters `0` through `9`
   - The space character
   - Characters `, . ; ' " ( ) / -`

   To avoid conversion problems between different platforms, do not use other characters in an attribute value. Note that the attribute types, for example `CN`, must be in uppercase characters.
4. Strings containing the same alphabetical characters match irrespective of case.
5. Spaces are not allowed between the attribute type and the `=` character.
6. Optionally, you can enclose attribute values in double quotes, for example `CN="John Smith"`. The quotes are discarded when matching values.
7. Spaces at either end of the string are ignored unless the string is enclosed in double quotes.
8. The comma and semicolon attribute separator characters are considered to be part of the string when enclosed in double quotes.
9. The names of attribute types, for example `CN` or `OU`, are considered to be part of the string when enclosed in double quotes.
10. Any of the attribute types `ST`, `SP`, and `S` can be used for the State or Province name.
11. Any attribute value can have an asterisk (`*`) as a pattern-matching character at the beginning, the end, or in both places. The asterisk character substitutes for any number of characters at the beginning or end of the string to be matched. This enables your SSLPEER value specification to match a range of Distinguished Names. For example, `OU=IBM*` matches every Organizational Unit beginning with IBM®, such as IBM Corporation.
    Note that the asterisk character can also be a valid character in a Distinguished Name. To obtain an exact match with an asterisk at the beginning or end of the string, the backslash escape character (`\`) must precede the asterisk: `\*`. Asterisks in the middle of the string are considered to be part of the string and do not require the backslash escape character.
12. When multiple OU attributes are specified, all must exist and be in descending hierarchical order. For an example of this, see DEFINE CHANNEL.

Parent topic: WebSphere MQ support for SSL and TLS

Related concepts
Distinguished Names

6.19. Understanding authentication failures

There are a number common reasons for authentication failures during the SSL handshake.

These include, but are not limited to, those in the following list:

A certificate has been found in a Certificate Revocation List or Authority Revocation List

You have the option to check certificates against the revocation lists published by the Certification Authorities.

A Certification Authority can revoke a certificate that is no longer trusted by publishing it in a Certificate Revocation List (CRL) or Authority Revocation List (ARL). For more information, see Working with revoked certificates.

An OCSP responder has identified a certificate as Revoked or Unknown

You have the option to check certificates using OCSP. An OCSP responder can return a response of Revoked, indicating that a certificate is no longer valid, or Unknown, indicating that it has no revocation data for that certificate. For more information, see Working with revoked certificates.

A certificate has expired or is not yet active

Each digital certificate has a date from which it is valid and a date after which it is no longer valid, so an attempt to authenticate with a certificate that is outside its lifetime fails.

A certificate is corrupted

If the information in a digital certificate is incomplete or damaged, authentication fails.

A certificate is not supported

If the certificate is in a format that is not supported, authentication fails, even if the certificate is still within its lifetime.

The SSL client does not have a certificate

The SSL server always validates the client certificate if one is sent. If the SSL client does not send a certificate, authentication fails if the end of the channel acting as the SSL server is defined:

- With the SSLCAUTH parameter set to REQUIRED or
- With an SSLPEER parameter value

There is no matching CA root certificate or the certificate chain is incomplete

Each digital certificate is issued by a Certification Authority (CA), which also provides a root certificate that contains the public key for the CA. Root certificates are signed by the issuing CA itself. If the key repository on the computer that is performing the authentication does not contain a valid root certificate for the CA that issued the incoming user certificate, authentication fails.

Authentication often involves a chain of trusted certificates. The digital signature on a user certificate is verified with the public key from the certificate for the issuing CA. If that CA certificate is a root certificate, the verification process is complete. If that CA certificate was issued by an intermediate CA, the digital signature on the intermediate CA certificate must itself be verified. This process continues along a chain of CA certificates until a root certificate is reached. In such cases, all certificates in the chain must be verified correctly. If the key repository on the computer that is performing the authentication does not contain a valid root certificate for the CA that issued the incoming root certificate, authentication fails.

However, certain SSL implementations such as GSKit, DCM, and RACF validate the certificates as long as the trust anchor (ROOT CA) is present, with some of the intermediate CA not present in the trust chain. Therefore, it is important to ensure that the server side certificate store contains the complete trust chain. Also, the technique of selectively removing signer (CA) certificates should not be used to control connectivity to the queue manager.

For more information, see How certificate chains work.

For more information about the terms used in this chapter, see:

- Secure Sockets Layer (SSL) concepts
- Digital certificates

Parent topic: WebSphere MQ support for SSL and TLS

7. Channel exit programs

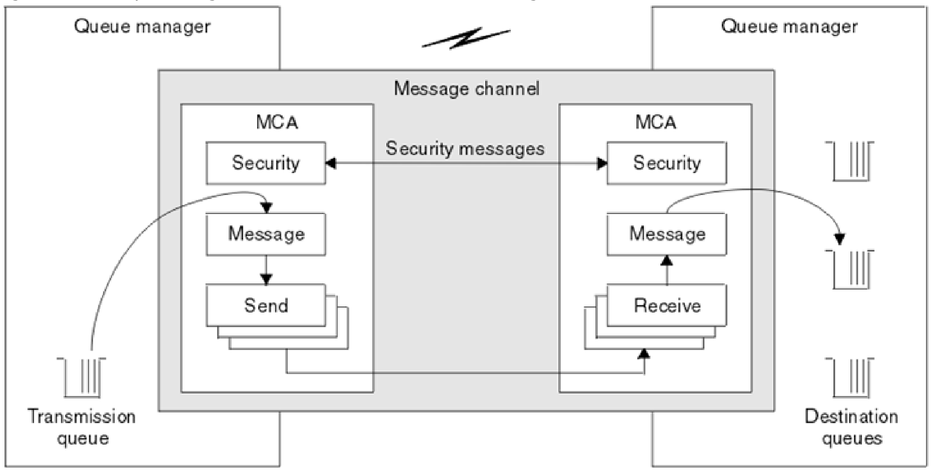
Channel exit programs are programs that are called at defined places in the processing sequence of an MCA. Users and vendors can write their own channel exit programs. Some are supplied by IBM®.

There are several types of channel exit program, but only four have a role in providing link level security:

- Security exit
- Message exit
- Send exit
- Receive exit

These four types of channel exit program are illustrated in Figure 1 and are described in the following topics.

Figure 1. Security, message, send, and receive exits on a message channel



Security exit overview
Security exits normally work in pairs. They are called before messages flow and their purpose is to allow an MCA to authenticate its partner.

Message exit
Message exits operate only on message channels and normally work in pairs. A message exit can operate on the whole message and make various changes to it.

Send and receive exits
Send and receive exits typically work in pairs. They operate on transmission segments and are best used where the structure of the data they are

processing is not relevant.

Parent topic: Security

Related information
Channel-exit programs for messaging channels

7.1. Security exit overview

Security exits normally work in pairs. They are called before messages flow and their purpose is to allow an MCA to authenticate its partner.

Security exits normally work in pairs; one at each end of a channel. They are called immediately after the initial data negotiation has completed on channel startup, but before any messages start to flow. The primary purpose of the security exit is to enable the MCA at each end of a channel to authenticate its partner. However, there is nothing to prevent a security exit from performing other function, even function that has nothing to do with security.

Security exits can communicate with each other by sending *security messages*. The format of a security message is not defined and is determined by the user. One possible outcome of the exchange of security messages is that one of the security exits might decide not to proceed any further. In that case, the channel is closed and messages do not flow. If there is a security exit at only one end of a channel, the exit is still called and can elect whether to continue or to close the channel.

Security exits can be called on both message and MQI channels. The name of a security exit is specified as a parameter in the channel definition at each end of a channel.

For more information about security exits, see Link level security using a security exit.

Parent topic: Channel exit programs

7.2. Message exit

Message exits operate only on message channels and normally work in pairs. A message exit can operate on the whole message and make various changes to it.

Message exits at the sending and receiving ends of a channel normally work in pairs. A message exit at the sending end of a channel is called after the MCA has got a message from the transmission queue. At the receiving end of a channel, a message exit is called before the MCA puts a message on its destination queue.

A message exit has access to both the transmission queue header, MQXQH, which includes the embedded message descriptor, and the application data in a message. A message exit can modify the contents of the message and change its length. A change of length might be the result of compressing, decompressing, encrypting, or decrypting the message. It might also be the result of adding data to the message, or removing data from it.

Message exits can be used for any purpose that requires access to the whole message, rather than a portion of it, and not necessarily for security.

A message exit can determine that the message it is currently processing should not proceed any further towards its destination. The MCA then puts the message on the dead letter queue. A message exit can also close the channel.

Message exits can be called only on message channels, not on MQI channels. This is because the purpose of an MQI channel is to enable the input and output parameters of MQI calls to flow between the WebSphere® MQ client application and the queue manager.

The name of a message exit is specified as a parameter in the channel definition at each end of a channel. You can also specify a list of message exits to be run in succession.

For more information about message exits, see Link level security using a message exit.

Parent topic: Channel exit programs

7.3. Send and receive exits

Send and receive exits typically work in pairs. They operate on transmission segments and are best used where the structure of the data they are processing is not relevant.

A *send exit* at one end of a channel and a *receive exit* at the other end normally work in pairs. A send exit is called just before an MCA issues a communications send to send data over a communications connection. A receive exit is called just after an MCA has regained control following a communications receive and has received data from a communications connection. If sharing conversations is in use, over an MQI channel, a different instance of a send and receive exit is called for each conversation.

The WebSphere® MQ channel protocol flows between two MCAs on a message channel contain control information as well as message data. Similarly, on an MQI channel, the flows contain control information as well as the parameters of MQI calls. Send and receive exits are called for all types of data.

Message data flows in only one direction on a message channel but, on an MQI channel, the input parameters of an MQI call flow in one direction and the output parameters flow in the other. On both message and MQI channels, control information flows in both directions. As a result, send and receive exits can be called at both ends of a channel.

The unit of data that is transmitted in a single flow between two MCAs is called a *transmission segment*. Send and receive exits have access to each transmission segment. They can modify its contents and change its length. A send exit, however, must not change the first 8 bytes of a transmission segment. These 8 bytes form part of the WebSphere MQ channel protocol header. There are also restrictions on how much a send exit can increase the length of a transmission segment. In particular, a send exit cannot increase its length beyond the maximum that was negotiated between the two MCAs at channel startup.

On a message channel, if a message is too large to be sent in a single transmission segment, the sending MCA splits the message and sends it in more than one transmission segment. As a consequence, a send exit is called for each transmission segment containing a portion of the message and, at the receiving end, a receive exit is called for each transmission segment. The receiving MCA reconstitutes the message from the transmission segments after they have been processed by the receive exit.

Similarly, on an MQI channel, the input or output parameters of an MQI call are sent in more than one transmission segment if they are too large. This might occur, for example, on an MQPUT, MQPUT1, or MQGET call if the application data is sufficiently large.

Taking these considerations into account, it is more appropriate to use send and receive exits for purposes in which they do not need to understand the structure of the data they are handling and can therefore treat each transmission segment as a binary object.

A send or a receive exit can close a channel.

The names of a send exit and a receive exit are specified as parameters in the channel definition at each end of a channel. You can also specify a list of send exits to be run in succession. Similarly, you can specify a list of receive exits.

For more information about send and receive exits, see Link level security using send and receive exits.

Parent topic: Channel exit programs

8. The SSPI channel exit program

WebSphere® MQ for Windows supplies a security exit program, which can be used on both message and MQI channels. The exit is supplied as source and object code, and provides one-way and two-way authentication.

The security exit uses the Security Support Provider Interface (SSPI), which provides the integrated security facilities of Windows platforms.

The security exit provides the following identification and authentication services:

One way authentication

This uses Windows NT LAN Manager (NTLM) authentication support. NTLM allows servers to authenticate their clients. It does not allow a client to authenticate a server, or one server to authenticate another. NTLM was designed for a network environment in which servers are assumed to be genuine. NTLM is supported on all Windows platforms that are supported by WebSphere MQ Version 7.0.

This service is typically used on an MQI channel to enable a server queue manager to authenticate a WebSphere MQ client application. A client application is identified by the user ID associated with the process that is running.

To perform the authentication, the security exit at the client end of a channel acquires an authentication token from NTLM and sends the token in a security message to its partner at the other end of the channel. The partner security exit passes the token to NTLM, which checks that the token is authentic. If the partner security exit is not satisfied with the authenticity of the token, it instructs the MCA to close the channel.

Two way, or mutual, authentication

This uses Kerberos authentication services. The Kerberos protocol does not assume that servers in a network environment are genuine. Servers can authenticate clients and other servers, and clients can authenticate servers. Kerberos is supported on all Windows platforms that are supported by WebSphere MQ Version 7.0.

This service can be used on both message and MQI channels. On a message channel, it provides mutual authentication of the two queue managers. On an MQI channel, it enables the server queue manager and the WebSphere MQ client application to authenticate each other. A queue manager is identified by its name prefixed by the string `ibmMQSeries/`. A client application is identified by the user ID associated with the process that is running.

To perform the mutual authentication, the initiating security exit acquires an authentication token from the Kerberos security server and sends the token in a security message to its partner. The partner security exit passes the token to the Kerberos server, which checks that it is authentic. The Kerberos security server generates a second token, which the partner sends in a security message to the initiating security exit. The initiating security exit then asks the Kerberos server to check that the second token is authentic. During this exchange, if either security exit is not satisfied with the authenticity of the token sent by the other, it instructs the MCA to close the channel.

The security exit is supplied in both source and object format. You can use the source code as a starting point for writing your own channel exit programs or you can use the object module as supplied. The object module has two entry points, one for one way authentication using NTLM authentication support and the other for two way authentication using Kerberos authentication services.

For more information about how the SSPI channel exit program works, and for instructions on how to implement it, see Using the SSPI security exit on Windows systems.

Parent topic: Security

9. SNA LU 6.2 security services

SNA LU 6.2 offers session level cryptography, session level authentication, and conversation level authentication.

Note: This collection of topics assumes that you have a basic understanding of Systems Network Architecture (SNA). Each of the books referenced in this section contains a brief introduction to the relevant concepts and terminology. If you require a more comprehensive technical introduction to SNA, see *Systems Network Architecture Technical Overview*, GC30-3073.

SNA LU 6.2 provides three security services:

- Session level cryptography
- Session level authentication
- Conversation level authentication

For session level cryptography and session level authentication, SNA uses the *Data Encryption Standard (DES)* algorithm. The DES algorithm is a block cipher algorithm, which uses a symmetric key for encrypting and decrypting data. Both the block and the key are 8 bytes in length.

Session level cryptography
Session level cryptography encrypts and decrypts session data using the DES algorithm. It can therefore be used to provide a link level confidentiality service on SNA LU 6.2 channels.

Session level authentication
Session level authentication is a session level security protocol that enables two LUs to authenticate each other while they are activating a session. It is also known as *LU-LU verification*.

Conversation level authentication
When a local TP attempts to allocate a conversation with a partner TP, the local LU sends an attach request to the partner LU, asking it to attach the partner TP. Under certain circumstances, the attach request can contain security information, which the partner LU can use to authenticate the local TP. This is known as *conversation level authentication*, or *end user verification*.

Parent topic: Security

9.1. Session level cryptography

Session level cryptography encrypts and decrypts session data using the DES algorithm. It can therefore be used to provide a link level confidentiality service on SNA LU 6.2 channels.

Logical units (LUs) can provide mandatory (or required) data cryptography, selective data cryptography, or no data cryptography.

On a *mandatory cryptographic session*, an LU encrypts all outbound data request units and decrypts all inbound data request units.

On a *selective cryptographic session*, an LU encrypts only the data request units specified by the sending transaction program (TP). The sending LU signals that the data is encrypted by setting an indicator in the request header. By checking this indicator, the receiving LU can tell which request units to decrypt before passing them on to the receiving TP.

In an SNA network, WebSphere® MQ MCAs are transaction programs. MCAs do not request encryption for any data that they send. Selective data cryptography is not an option therefore; only mandatory data cryptography or no data cryptography is possible on a session.

For information about how to implement mandatory data cryptography, see the books for your SNA subsystem. Refer to the same books for information about stronger forms of encryption that might be available for use on your platform, such as Triple DES 24-byte encryption on z/OS®.

For more general information about session level cryptography, see *Systems Network Architecture LU 6.2 Reference: Peer Protocols*, SC31-6808.

Parent topic: SNA LU 6.2 security services

9.2. Session level authentication

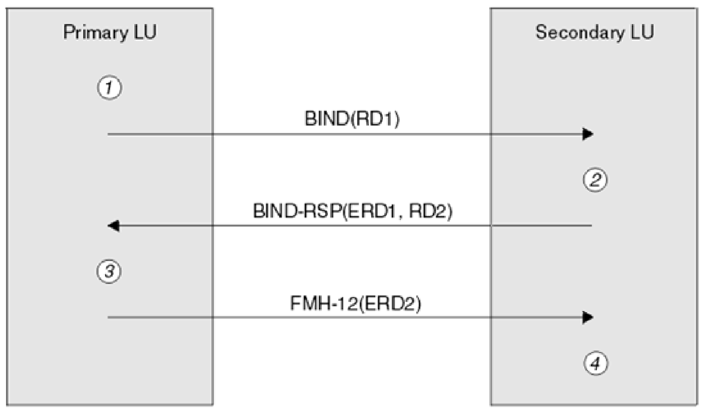
Session level authentication is a session level security protocol that enables two LUs to authenticate each other while they are activating a session. It is also known as *LU-LU verification*.

Because an LU is effectively the "gateway" into a system from the network, you might consider this level of authentication to be sufficient in certain circumstances. For example, if your queue manager needs to exchange messages with a remote queue manager that is running in a controlled and trusted environment, you might be prepared to trust the identities of the remaining components of the remote system after the LU has been authenticated.

Session level authentication is achieved by each LU verifying its partner's password. The password is called an *LU-LU password* because one password is established between each pair of LUs. The way that an LU-LU password is established is implementation dependent and outside the scope of SNA.

Figure 1 illustrates the flows for session level authentication.

Figure 1. Flows for session level authentication



Legend:

BIND	=	BIND request unit
BIND-RSP	=	BIND response unit
ERD	=	Encrypted random data
FMH-12	=	Function Management Header 12
RD	=	Random data

The protocol for session level authentication is as follows. The numbers in the procedure correspond to the numbers in Figure 1.

1. The primary LU generates a random data value (RD1) and sends it to the secondary LU in the BIND request.
2. When the secondary LU receives the BIND request with the random data, it encrypts the data using the DES algorithm with its copy of the LU-LU password as the key. The secondary LU then generates a second random data value (RD2) and sends it, with the encrypted data (ERD1), to the primary LU in the BIND response.
3. When the primary LU receives the BIND response, it computes its own version of the encrypted data from the random data it generated originally. It does this by using the DES algorithm with its copy of the LU-LU password as the key. It then compares its version with the encrypted data that it received in the BIND response. If the two values are the same, the primary LU knows that the secondary LU has the same password as it does and the secondary LU is authenticated. If the two values do not match, the primary LU terminates the session.
   The primary LU then encrypts the random data that it received in the BIND response and sends the encrypted data (ERD2) to the secondary LU in a Function Management Header 12 (FMH-12).
4. When the secondary LU receives the FMH-12, it computes its own version of the encrypted data from the random data it generated. It then compares its version with the encrypted data that it received in the FMH-12. If the two values are the same, the primary LU is authenticated. If the two values do not match, the secondary LU terminates the session.

In an enhanced version of the protocol, which provides better protection against man in the middle attacks, the secondary LU computes a DES Message Authentication Code (MAC) from RD1, RD2, and the fully qualified name of the secondary LU, using its copy of the LU-LU password as the key. The secondary LU sends the MAC to the primary LU in the BIND response instead of ERD1.

The primary LU authenticates the secondary LU by computing its own version of the MAC, which it compares with the MAC received in the BIND response. The primary LU then computes a second MAC from RD1 and RD2, and sends the MAC to the secondary LU in the FMH-12 instead of ERD2.

The secondary LU authenticates the primary LU by computing its own version of the second MAC, which it compares with the MAC received in the FMH-12.

For information about how to configure session level authentication, see the books for your SNA subsystem. For more general information about session level authentication, see *Systems Network Architecture LU 6.2 Reference: Peer Protocols*, SC31-6808.

Parent topic: SNA LU 6.2 security services

9.3. Conversation level authentication

When a local TP attempts to allocate a conversation with a partner TP, the local LU sends an attach request to the partner LU, asking it to attach the partner TP. Under certain circumstances, the attach request can contain security information, which the partner LU can use to authenticate the local TP. This is known as *conversation level authentication*, or *end user verification*.

The following topics describe how WebSphere® MQ provides support for conversation level authentication.

For more information about conversation level authentication, see *Systems Network Architecture LU 6.2 Reference: Peer Protocols*, SC31-6808. For information specific to z/OS®, see *z/OS MVS™ Planning: APPC/MVS Management*, SA22-7599.

For more information about CPI-C, see *Common Programming Interface Communications CPI-C Specification*, SC31-6180. For more information about APPC/MVS TP Conversation Callable Services, see *z/OS MVS Programming: Writing Transaction Programs for APPC/MVS*, SA22-7621.

Support for conversation level authentication in WebSphere MQ on i5/OS, UNIX systems, and Windows systems
Use this topic to gain an overview of how conversation level authentication works, on platforms other than z/OS.

Conversation level authentication and WebSphere MQ for z/OS
Use this topic to gain an overview of how conversation level authentication works, on z/OS.

Parent topic: SNA LU 6.2 security services

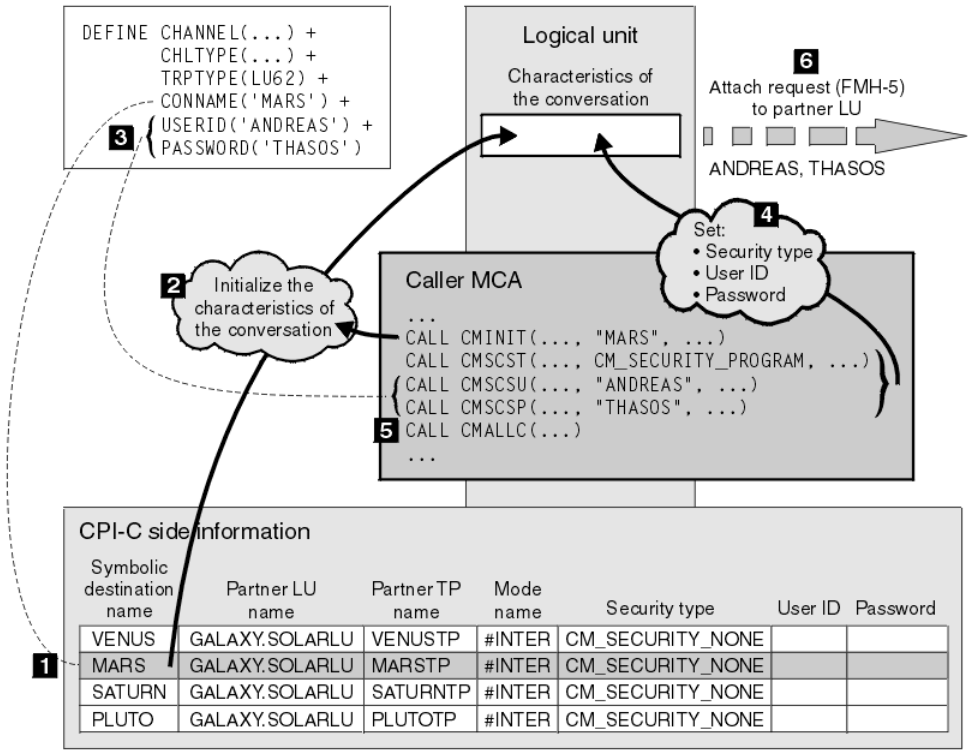
sy11150_

9.3.1. Support for conversation level authentication in WebSphere MQ on i5/OS, UNIX systems, and Windows systems

Use this topic to gain an overview of how conversation level authentication works, on platforms other than z/OS.

The support for conversation level authentication in WebSphere® MQ for i5/OS®, WebSphere MQ on UNIX systems, and WebSphere MQ for Windows is illustrated in Figure 1. The numbers in the diagram correspond to the numbers in the description that follows.

Figure 1. WebSphere MQ support for conversation level authentication



```
DEFINE CHANNEL(...) +
       CHLTYPE(...) +
       TRPTYPE(LU62) +
       CONNAME('MARS') +
       USERID('ANDREAS') +
       PASSWORD('THASOS')
```

```
...
CALL CMINIT(...., "MARS", ...)
CALL CMSCST(...., CM_SECURITY_PROGRAM, ...)
CALL CMSCSU(...., "ANDREAS", ...)
CALL CMSCSP(...., "THASOS", ...)
CALL CMALLC(...)
...
```

CPI-C side information

Symbolic destination name	Partner LU name	Partner TP name	Mode name	Security type	User ID	Password
VENUS	GALAXY.SOLARLU	VENUSTP	#INTER	CM_SECURITY_NONE		
MARS	GALAXY.SOLARLU	MARSTP	#INTER	CM_SECURITY_NONE		
SATURN	GALAXY.SOLARLU	SATURNTP	#INTER	CM_SECURITY_NONE		
PLUTO	GALAXY.SOLARLU	PLUTOTP	#INTER	CM_SECURITY_NONE		

Logical unit
Characteristics of the conversation
6
Attach request (FMH-5) to partner LU
ANDREAS, THASOS
3
4
Set:
• Security type
• User ID
• Password
2
Initialize the characteristics of the conversation
Caller MCA
5
1

On i5/OS, UNIX systems, and Windows systems, an MCA uses Common Programming Interface Communications (CPI-C) calls to communicate with a partner MCA across an SNA network. In the channel definition at the caller end of a channel, the value of the CONNAME parameter is a symbolic destination name, which identifies a CPI-C side information entry (1). This entry specifies:

- The name of the partner LU
- The name of the partner TP, which is a responder MCA
- The name of the mode to be used for the conversation

A side information entry can also specify the following security information:

- A security type.
  The commonly implemented security types are CM_SECURITY_NONE, CM_SECURITY_PROGRAM, and CM_SECURITY_SAME, but others are defined in the CPI-C specification.
- A user ID.
- A password.

A caller MCA prepares to allocate a conversation with a responder MCA by issuing the CPI-C call CMINIT, using the value of CONNAME as one of the parameters on the call. The CMINIT call identifies, for the benefit of the local LU, the side information entry that the MCA intends to use for the conversation. The local LU uses the values in this entry to initialize the characteristics of the conversation (2).

The caller MCA then checks the values of the USERID and PASSWORD parameters in the channel definition (3). If USERID is set, the caller MCA issues the following CPI-C calls (4):

- CMSCST, to set the security type for the conversation to CM_SECURITY_PROGRAM.
- CMSCSU, to set the user ID for the conversation to the value of USERID.
- CMSCSP, to set the password for the conversation to the value of PASSWORD. CMSCSP is not called unless PASSWORD is set.

The security type, user ID, and password set by these calls override any values acquired previously from the side information entry.

The caller MCA then issues the CPI-C call CMALLC to allocate the conversation (5). In response to this call, the local LU sends an attach request (Function Management Header 5, or FMH-5) to the partner LU (6).

If the partner LU will accept a user ID and a password, the values of USERID and PASSWORD are included in the attach request. If the partner LU will not accept a user ID and a password, the values are not included in the attach request. The local LU discovers whether the partner LU will accept a user ID and a password as part of an exchange of information when the LUs bind to form a session.

In a later version of the attach request, a password substitute can flow between the LUs instead of a clear password. A password substitute is a DES Message Authentication Code (MAC), or an SHA-1 message digest, formed from the password. Password substitutes can be used only if both LUs support them.

When the partner LU receives an incoming attach request containing a user ID and a password, it might use the user ID and password for the purposes of identification and authentication. By referring to access control lists, the partner LU might also determine whether the user ID has the authority to allocate a conversation and attach the responder MCA.

In addition, the responder MCA might run under the user ID included in the attach request. In this case, the user ID becomes the default user ID for the responder MCA and is used for authority checks when the MCA attempts to connect to the queue manager. It might also be used for authority checks subsequently when the MCA attempts to access the queue manager's resources.

The way in which a user ID and a password in an attach request can be used for identification, authentication, and access control is implementation dependent. For information specific to your SNA subsystem, refer to the appropriate books.

If USERID is not set, the caller MCA does not call CMSCST, CMSCSU, and CMSCSP. In this case, the security information that flows in an attach request is determined solely by what is specified in the side information entry and what the partner LU will accept.

Parent topic: Conversation level authentication

9.3.2. Conversation level authentication and WebSphere MQ for z/OS

Use this topic to gain an overview of how conversation level authentication works, on z/OS.

On WebSphere® MQ for z/OS®, MCAs do not use CPI-C. Instead, they use APPC/MVS TP Conversation Callable Services, an implementation of Advanced Program-to-Program Communication (APPC), which has some CPI-C features. When a caller MCA allocates a conversation, a security type of SAME is specified on the call. Therefore, because an APPC/MVS LU supports persistent verification only for inbound conversations, not for outbound conversations, there are two possibilities:

- If the partner LU trusts the APPC/MVS LU and will accept an already verified user ID, the APPC/MVS LU sends an attach request containing:
  - The channel initiator address space user ID
  - A security profile name, which, if RACF® is used, is the name of the current connect group of the channel initiator address space user ID
  - An already verified indicator
- If the partner LU does not trust the APPC/MVS LU and will not accept an already verified user ID, the APPC/MVS LU sends an attach request containing no security information.

On WebSphere MQ for z/OS, the USERID and PASSWORD parameters on the DEFINE CHANNEL command cannot be used for a message channel and are valid only at the client connection end of an MQI channel. Therefore, an attach request from an APPC/MVS LU never contains values specified by these parameters.

Parent topic: Conversation level authentication

10. Providing your own link level security

This collection of topics describes how you can provide your own link level security services. The main way of doing this is by writing your own channel exit programs.

Channel exit programs are introduced in Channel exit programs. The same topic also describes the channel exit program that is supplied with WebSphere® MQ for Windows (the SSPI channel exit program). This channel exit program is supplied in source format so that you can modify the source code to suit your requirements. If neither this channel exit program, nor channel exit programs available from other vendors, meets your requirements, you can design and write your own. This topic suggests ways in which channel exit programs can provide security services. For information about how to write a channel exit program, see WebSphere MQ Intercommunication.

Link level security using a security exit
Security exits normally work in pairs; one at each end of a channel. They are called immediately after the initial data negotiation has completed on channel startup.

Link level security using a message exit
A message exit can be used only on a message channel, not on an MQI channel. It has access to both the transmission queue header, MQXQH, which includes the embedded message descriptor, and the application data in a message. It can modify the contents of the message and change its length.

Link level security using send and receive exits
Send and receive exits can be used on both message and MQI channels. They are called for all types of data that flow on a channel, and for flows in both directions.

Parent topic: Security

10.1. Link level security using a security exit

Security exits normally work in pairs; one at each end of a channel. They are called immediately after the initial data negotiation has completed on channel startup.

Security exits can be used to provide identification and authentication, access control, and confidentiality.

Identification and authentication in security exits
You can use a security exit to implement one-way or mutual authentication.

Access control in security exits
You can implement access control in a security exit by use of the MCAUserIdentifier or the Object Authority Manager.

Confidentiality in security exits
Security exits can play a role in the confidentiality service by generating and distributing the symmetric key for encrypting and decrypting the data that flows on the channel. A common technique for doing this uses PKI technology.

Parent topic: Providing your own link level security

10.1.1. Identification and authentication in security exits

You can use a security exit to implement one-way or mutual authentication.

The primary purpose of a security exit is to enable the MCA at each end of a channel to authenticate its partner. At each end of a message channel, and at the server end of an MQI channel, an MCA typically acts on behalf of the queue manager to which it is connected. At the client end of an MQI channel, an MCA typically acts on behalf of the user of the WebSphere® MQ client application. In this situation, mutual authentication actually takes place between two queue managers, or between a queue manager and the user of a WebSphere MQ client application.

The supplied security exit (the SSPI channel exit) illustrates how mutual authentication can be implemented by exchanging authentication tokens that are generated, and then checked, by a trusted authentication server such as Kerberos. For more details, see The SSPI channel exit program.

Mutual authentication can also be implemented by using Public Key Infrastructure (PKI) technology. Each security exit generates some random data, signs it using the private key of the queue manager or user it is representing, and sends the signed data to its partner in a security message. The partner security exit performs the authentication by checking the digital signature using the public key of the queue manager or user. Before exchanging digital signatures, the security exits might need to agree the algorithm for generating a message digest, if more than one algorithm is available for use.

When a security exit sends the signed data to its partner, it also needs to send some means of identifying the queue manager or user it is representing. This might be a Distinguished Name, or even a digital certificate. If a digital certificate is sent, the partner security exit can validate the certificate by working through the certificate chain to the root CA certificate. This provides assurance of the ownership of the public key that is used to check the digital signature.

The partner security exit can validate a digital certificate only if it has access to a key repository that contains the remaining certificates in the certificate chain. If a digital certificate for the queue manager or user is not sent, one must be available in the key repository to which the partner security exit has access. The partner security exit cannot check the digital signature unless it can find the signer's public key.

The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) use PKI techniques like the ones just described. For more information about how the Secure Sockets Layer performs authentication, see Secure Sockets Layer (SSL) concepts.

If a trusted authentication server or PKI support is not available, other techniques can be used. A common technique, which can be implemented in security exits, uses a symmetric key algorithm.

One of the security exits, exit A, generates a random number and sends it in a security message to its partner security exit, exit B. Exit B encrypts the number using its copy of a key which is known only to the two security exits. Exit B sends the encrypted number to exit A in a security message with a second random number that exit B has generated. Exit A verifies that the first random number has been encrypted correctly, encrypts the second random number using its copy of the key, and sends the encrypted number to exit B in a security message. Exit B then verifies that the second random number has been encrypted correctly. During this exchange, if either security exit is not satisfied with the authenticity of other, it can instruct the MCA to close the channel.

An advantage of this technique is that no key or password is sent over the communications connection during the exchange. A disadvantage is that it does not provide a solution to the problem of how to distribute the shared key in a secure way. One solution to this problem is described in Confidentiality in security exits. A similar technique is used in SNA for the mutual authentication of two LUs when they bind to form a session. The technique is described in Session level authentication.

All the preceding techniques for mutual authentication can be adapted to provide one-way authentication.

Parent topic: Link level security using a security exit

10.1.2. Access control in security exits

You can implement access control in a security exit by use of the MCAUserIdentifier or the Object Authority Manager.

MCAUserIdentifier

Every instance of a channel that is current has an associated channel definition structure, MQCD. The initial values of the fields in MQCD are determined by the channel definition that is created by a WebSphere® MQ administrator. In particular, the initial value of one of the fields, *MCAUserIdentifier*, is determined by the value of the MCAUSER parameter on the DEFINE CHANNEL command, or by the equivalent to MCAUSER if the channel definition is created in another way.

The MQCD structure is passed to a channel exit program when it is called by an MCA. When a security exit is called by an MCA, the security exit can change the value of *MCAUserIdentifier*, replacing any value that was specified in the channel definition.

On i5/OS®, UNIX systems, and Windows systems, unless the value of *MCAUserIdentifier* is blank, the queue manager uses the value of *MCAUserIdentifier* as the user ID for authority checks when an MCA attempts to access the queue manager's resources after it has connected to the queue manager. If the value of *MCAUserIdentifier* is blank, the queue manager uses the default user ID of the MCA instead. This applies to RCVR, RQSTR, CLUSRCVR and SVRCONN channels. For sending MCAs, the default user ID is always used for authority checks, even if the value of *MCAUserIdentifier* is not blank.

On z/OS®, the queue manager might use the value of *MCAUserIdentifier* for authority checks, provided it is not blank. For receiving MCAs and server connection MCAs, whether the queue manager uses the value of *MCAUserIdentifier* for authority checks depends on:

- The value of the PUTAUT parameter in the channel definition
- The RACF® profile used for the checks
- The access level of the channel initiator address space user ID to the RESLEVEL profile

For sending MCAs, it depends on:

- Whether the sending MCA is a caller or a responder
- The access level of the channel initiator address space user ID to the RESLEVEL profile

The user ID that a security exit stores in *MCAUserIdentifier* can be acquired in various ways. Here are some examples:

- Provided there is no security exit at the client end of an MQI channel, a user ID associated with the WebSphere MQ client application flows from the client connection MCA to the server connection MCA when the client application issues an MQCONN call. The server connection MCA stores this user ID in the *RemoteUserIdentifier* field in the channel definition structure, MQCD. If the value of *MCAUserIdentifier* is blank at this time, the MCA stores the same user ID in *MCAUserIdentifier*. If the MCA does not store the user ID in *MCAUserIdentifier*, a security exit can do it later by setting *MCAUserIdentifier* to the value of *RemoteUserIdentifier*.
  If the user ID that flows from the client system is entering a new security domain and is not valid on the server system, the security exit can substitute the user ID for one that is valid and store the substituted user ID in *MCAUserIdentifier*.
- The user ID can be sent by the partner security exit in a security message.
  On a message channel, a security exit called by the sending MCA can send the user ID under which the sending MCA is running. A security exit called by the receiving MCA can then store the user ID in *MCAUserIdentifier*. Similarly, on an MQI channel, a security exit at the client end of the channel can send the user ID associated with the WebSphere MQ client application. A security exit at the server end of the channel can then store the user ID in *MCAUserIdentifier*. As in the previous example, if the user ID is not valid on the target system, the security exit can substitute the user ID for one that is valid and store the substituted user ID in *MCAUserIdentifier*.
  If a digital certificate is received as part of the identification and authentication service, a security exit can map the Distinguished Name in the certificate to a user ID that is valid on the target system. It can then store the user ID in *MCAUserIdentifier*.
- If SSL is used on the channel, the partner's Distinguished Name (DN) is passed to the exit in the SSLPeerNamePtr field of the MQCD, and the DN of the issuer of that certificate is passed to the exit in the SSLRemCertIssNamePtr field of the MQCXP.

For more information about the *MCAUserIdentifier* field, the channel definition structure, MQCD, and the channel exit parameter structure, MQCXP, see WebSphere MQ Intercommunication. For more information about the user ID that flows from a client system on an MQI channel, see Access control.

WebSphere MQ Object Authority Manager user authentication

On WebSphere MQ client connections, security exits can be used to modify or create the MQCSP structure used in Object Authority Manager (OAM) user authentication. This is described in Channel-exit programs for messaging channels

Parent topic: Link level security using a security exit

10.1.3. Confidentiality in security exits

Security exits can play a role in the confidentiality service by generating and distributing the symmetric key for encrypting and decrypting the data that flows on the channel. A common technique for doing this uses PKI technology.

›One security exit generates a random data value, encrypts it with the public key of the queue manager or user that the partner security exit is representing, and sends the encrypted data to its partner in a security message. The partner security exit decrypts the random data value with the private key of the queue manager or user it is representing. Each security exit can now use the random data value to derive the symmetric key independently of the other by using an algorithm known to both of them. Alternatively, they can use the random data value as the key.‹

If the first security exit has not authenticated its partner by this time, the next security message sent by the partner can contain an expected value encrypted with the symmetric key. The first security exit can now authenticate its partner by checking that the partner security exit was able to encrypt the expected value correctly.

The security exits can also use this opportunity to agree the algorithm for encrypting and decrypting the data that flows on the channel, if more than one algorithm is available for use.

Parent topic: Link level security using a security exit

10.2. Link level security using a message exit

›A message exit can be used only on a message channel, not on an MQI channel. It has access to both the transmission queue header, MQXQH, which includes the embedded message descriptor, and the application data in a message. It can modify the contents of the message and change its length.‹

A message exit can be used for any purpose that requires access to the whole message rather than a portion of it.

›Message exits can be used to provide identification and authentication, access control, confidentiality, data integrity, and non-repudiation, and for reasons other than security.‹

Identification and authentication in message exits
You can use message exits to process information to authenticate a user ID, though it might be better to implement authentication at the application level..

Access control in message exits
You might need to use a message exit to substitute one user ID with another.

Confidentiality in message exits
A message exit at the sending end of a channel can encrypt the application data in a message and another message exit at the receiving end of the channel can decrypt the data.

Data integrity in message exits
A message can be digitally signed by a message exit at the sending end of a channel. The digital signature can then be checked by a message exit at the receiving end of a channel to detect whether the message has been deliberately modified.

Non-repudiation in message exits

If incoming messages are digitally signed, a message exit at the receiving end of a channel can log sufficient evidence to enable the digital signature of a message to be checked at any time in the future. This can form the basis of a non-repudiation service with proof of origin.

Other uses of message exits
Message exits can be used for reasons other than security.

Parent topic: Providing your own link level security

10.2.1. Identification and authentication in message exits

›You can use message exits to process information to authenticate a user ID, though it might be better to implement authentication at the application level..
‹

When an application puts a message on a queue, the *UserIdentifier* field in the message descriptor contains a user ID associated with the application. However, there is no data present that can be used to authenticate the user ID. This data can be added by a message exit at the sending end of a channel and checked by a message exit at the receiving end of the channel. The authenticating data can be an encrypted password or a digital signature, for example.

This service might be more effective if it is implemented at the application level. The basic requirement is for the user of the application that receives the message to be able to identify and authenticate the user of the application that sent the message. It is therefore natural to consider implementing this service at the application level. For more discussion about this, see Identification and authentication in the API exit and API-crossing exit.

Parent topic: Link level security using a message exit

10.2.2. Access control in message exits

›You might need to use a message exit to substitute one user ID with another.‹

›Consider a client application that sends a message to a server application. The server application can extract the user ID from the *UserIdentifier* field in the message descriptor and, provided it has alternate user authority, ask the queue manager to use this user ID for authority checks when it accesses WebSphere® MQ resources on behalf of the client.‹

›If the PUTAUT parameter is set to CTX (or ALTMCA on z/OS®) in the channel definition, the user ID in the *UserIdentifier* field of each incoming message is used for authority checks when the MCA opens the destination queue.‹

In certain circumstances, when a report message is generated, it is put using the authority of the user ID in the *UserIdentifier* field of the message causing the report. In particular, confirm-on-delivery (COD) reports and expiration reports are always put with this authority.

Because of these situations, it might be necessary to substitute one user ID for another in the *UserIdentifier* field as a message enters a new security domain. This can be done by a message exit at the receiving end of the channel. Alternatively, you can ensure that the user ID in the *UserIdentifier* field of an incoming message is defined in the new security domain.

If an incoming message contains a digital certificate for the user of the application that sent the message, a message exit can validate the certificate and map the Distinguished Name in the certificate to a user ID that is valid on the receiving system. It can then set the *UserIdentifier* field in the message descriptor to this user ID.

If it is necessary for a message exit to change the value of the *UserIdentifier* field in an incoming message, it might be appropriate for the message exit to authenticate the sender of the message at the same time. For more details, see Identification and authentication in message exits.

Parent topic: Link level security using a message exit

10.2.3. Confidentiality in message exits

›A message exit at the sending end of a channel can encrypt the application data in a message and another message exit at the receiving end of the channel can decrypt the data.‹

For performance reasons, a symmetric key algorithm is normally used for this purpose. For more information about how the symmetric key can be generated and distributed, see Confidentiality in security exits.

Headers in a message, such as the transmission queue header, MQXQH, which includes the embedded message descriptor, must not be encrypted by a message exit. This is because data conversion of the message headers takes place either after a message exit is called at the sending end or before a message exit is called at the receiving end. If the headers are encrypted, data conversion fails and the channel stops.

Parent topic: Link level security using a message exit

10.2.4. Data integrity in message exits

A message can be digitally signed by a message exit at the sending end of a channel. The digital signature can then be checked by a message exit at the receiving end of a channel to detect whether the message has been deliberately modified.

Some protection can be provided by using a message digest instead of a digital signature. A message digest might be effective against casual or indiscriminate tampering, but it does not prevent the more informed individual from changing or replacing the message, and generating a completely new digest for it. This is particularly true if the algorithm that is used to generate the message digest is a well known one.

Parent topic: Link level security using a message exit

10.2.5. Non-repudiation in message exits

If incoming messages are digitally signed, a message exit at the receiving end of a channel can log sufficient evidence to enable the digital signature of a message to be checked at any time in the future. This can form the basis of a non-repudiation service with proof of origin.

Like the identification and authentication service, this service might be more effective if it is implemented at the application level. At the application level, the service can also be extended to provide proof of delivery. For more information about how this service can be implemented at the application level, see Non-repudiation in an API exit or API-crossing exit.

Parent topic: Link level security using a message exit

10.2.6. Other uses of message exits

Message exits can be used for reasons other than security.

For example, a message exit can be used for application data conversion, although a data conversion exit is normally more appropriate for this purpose. They can be used for compressing and decompressing the application data in messages if the communications subsystem cannot provide this function. Headers in a message must not be compressed by a message exit because it causes data conversion of the message headers to fail.

Message exits also play an important role in implementing reference messages. Reference messages allow a large object, such as a file, to be transferred from one system to another without needing to store the object in a WebSphere® MQ queue at either the source or destination queue manager. For more information about reference messages, see the WebSphere MQ Application Programming Guide.

Parent topic: Link level security using a message exit

10.3. Link level security using send and receive exits

Send and receive exits can be used on both message and MQI channels. They are called for all types of data that flow on a channel, and for flows in both directions.

Send and receive exits have access to each transmission segment. They can modify its contents and change its length.

On a message channel, if an MCA needs to split a message and send it in more than one transmission segment, a send exit is called for each transmission segment containing a portion of the message and, at the receiving end, a receive exit is called for each transmission segment. The same occurs on an MQI channel if the input or output parameters of an MQI call are too large to be sent in a single transmission segment.

On an MQI channel, byte 10 of a transmission segment identifies the MQI call, and indicates whether the transmission segment contains the input or output parameters of the call. Send and receive exits can examine this byte to determine whether the MQI call contains application data that might need to be protected.

When a send exit is called for the first time, to acquire and initialize any resources it needs, it can ask the MCA to reserve a specified amount of space in the buffer that holds a transmission segment. When it is called later to process a transmission segment, it can use this space to add an encrypted key or a digital signature, for example. The corresponding receive exit at the other end of the channel can remove the data added by the send exit, and use it to process the transmission segment.

Send and receive exits are best suited for purposes in which they do not need to understand the structure of the data they are handling and can therefore treat each transmission segment as a binary object.

Send and receive exits can be used to provide confidentiality and data integrity, and for uses other than security.

Confidentiality in send and receive exits
Send and receive exits can be used to encrypt and decrypt the data that flows on a channel.

Data integrity in send and receive exits
You might use send and receive exits to implement data integrity on MQI channels.

Other uses of send and receive exits
Send and receive exits can be used for reasons other than security.

Parent topic: Providing your own link level security

Related information
Identifying the API call in a channel-exit program

10.3.1. Confidentiality in send and receive exits

Send and receive exits can be used to encrypt and decrypt the data that flows on a channel.

They are more appropriate than message exits for providing this service for the following reasons:

- On a message channel, message headers can be encrypted as well as the application data in the messages.
- Send and receive exits can be used on MQI channels as well as message channels. Parameters on MQI calls might contain sensitive application data that needs to be protected while it flows on an MQI channel. You can therefore use the same send and receive exits on both kinds of channel.

Parent topic: Link level security using send and receive exits

10.3.2. Data integrity in send and receive exits

You might use send and receive exits to implement data integrity on MQI channels.

On a message channel, message exits are more appropriate for providing this service because a message exit has access to a whole message. On an MQI channel, parameters on MQI calls might contain application data that needs to be protected and only send and receive exits can provide this protection.

Parent topic: Link level security using send and receive exits

10.3.3. Other uses of send and receive exits

Send and receive exits can be used for reasons other than security.

For example, they can be used to compress and decompress the data that flows on a channel. On message channels, they are more appropriate than message exits for this purpose because message headers can be compressed as well as the application data in the messages.

Data compression on channels is supported as an integral part of WebSphere® MQ functionality. This integral support does not involve use of channel exits.

Parent topic: Link level security using send and receive exits

11. WebSphere MQ Advanced Message Security

WebSphere® MQ Advanced Message Security extends the security features available in WebSphere MQ.

If you are moving highly sensitive or valuable information, especially confidential or payment-related information such as patient records or credit card details, you must pay special attention to information security. Ensuring that information moving around the enterprise retains its integrity and is protected from unauthorized access is an ongoing challenge and responsibility. You are also likely to be required to comply with security regulations, at the risk of penalties for non-compliance.

You can develop your own security extensions to WebSphere MQ. However, such solutions require specialist skills and can be complicated and expensive to maintain. WebSphere MQ Advanced Message Security helps address these challenges when moving information around the enterprise between virtually every type of commercial IT system.

WebSphere MQ Advanced Message Security extends the security features of WebSphere MQ in the following ways:

- It provides application-level, end-to-end data protection for your point to point messaging infrastructure, using either encryption or digital signing of messages.
- It provides comprehensive security without writing complex security code or modifying or recompiling existing applications.
- It uses Public Key Infrastructure (PKI) technology to provide authentication, authorization, confidentiality, and data integrity services for messages.
- It provides administration of security policies for mainframe and distributed servers.
- It supports both WebSphere MQ servers and clients.
- It integrates with WebSphere MQ File Transfer Edition to provide an end-to-end secure messaging solution.

For more information see the WebSphere MQ Advanced Message Security webpage at http://www-01.ibm.com/software/integration/wmq/advanced-message-security/ and the WebSphere MQ Advanced Message Security Information Center at http://publib.boulder.ibm.com/infocenter/mqams/v7r0m1/index.jsp.

Parent topic: Security

12. Providing your own application level security

This collection of topics describes how you can provide your own application level security services.

To help you implement application level security, WebSphere® MQ provides two exits, the API exit and the API-crossing exit

. These exits can provide identification and authentication, access control, confidentiality, data integrity, and non-repudiation services, and other functions not related to security.

If the API exit or API-crossing exit is not supported in your system environment, you might want to consider other ways of providing your own application level security. One way is to develop a higher level API that encapsulates the MQI. Programmers then use this API, instead of the MQI, to write WebSphere MQ applications.

The most common reasons for using a higher level API are:

- To hide the more advanced features of the MQI from programmers.
- To enforce standards in the use of the MQI.
- To add function to the MQI. This additional function can be security services.

Some vendor products use this technique to provide application level security for WebSphere MQ.

If you are planning to provide security services in this way, note the following regarding data conversion:

- If a security token, such as a digital signature, has been added to the application data in a message, any code performing data conversion must be aware of the presence of this token.
- A security token might have been derived from a binary image of the application data. Therefore, any checking of the token must be done before converting the data.
- If the application data in a message has been encrypted, it must be decrypted before data conversion.

The API exit
An *API exit* is a program module that monitors or modifies the function of MQI calls. An API exit comprises multiple *API exit functions*, each with its own entry point in the module.

The API-crossing exit
An *API-crossing exit* is a program that monitors or modifies the function of MQI calls issued by CICS® applications on z/OS®.

Identification and authentication in the API exit and API-crossing exit
An application that receives a message must be able to identify and authenticate the user of the application that sent the message. This service is typically best implemented at the application level. API exits can implement the service in a number of ways.

Access control in the API exit and API-crossing exit
An API or API-crossing exit can provide access controls to supplement those provided by WebSphere MQ. In particular, the exit can provide access control at the message level. The exit can ensure that an application puts on a queue, or gets from a queue, only those messages that satisfy certain criteria.

Confidentiality in the API exit and API-crossing exit
The application data in a message can be encrypted by an API or API-crossing exit when the message is put by the sending application and decrypted by a second exit when the message is retrieved by the receiving application.

Data integrity in the API exit or API-crossing exit
A message can be digitally signed by an API or API-crossing exit when the message is put by the sending application. The digital signature can then be checked by a second exit when the message is retrieved by the receiving application. This can detect whether the message has been deliberately modified.

Non-repudiation in an API exit or API-crossing exit
An API exit or API-crossing exit can be used to form the basis of a non-repudiation service by logging information about messages received.

Parent topic: Security

12.1. The API exit

An *API exit* is a program module that monitors or modifies the function of MQI calls. An API exit comprises multiple *API exit functions*, each with its own entry point in the module.

Note: The information in this section does not apply to WebSphere® MQ for z/OS®.

There are two categories of exit function:

An exit function that is associated with an MQI call

There are two exit functions in this category for each MQI call and an additional one for an MQGET call with the MQGMO_CONVERT option. The MQCONN and MQCONNX calls share the same exit functions.

For each MQI call, one of the two exit functions is invoked before the queue manager starts to process the call and the other is invoked after the queue manager has completed processing the call. The exit function for an MQGET call with the MQGMO_CONVERT option is invoked during the MQGET call, after the message has been retrieved from the queue by the queue manager but before any data conversion takes place. This allows, for example, a message to be decrypted before data conversion.

An exit function can inspect and modify any of the parameters on an MQI call. On an MQPUT call, for example, an exit function that is invoked before the processing of the call has started can:

- Inspect and modify the contents of the application data in the message being put
- Change the length of the application data in the message
- Modify the contents of the fields in the message descriptor structure, MQMD
- Modify the contents of the fields in the put message options structure, MQPMO

An exit function that is invoked before the processing of an MQI call has started can suppress the call completely. The exit function for an MQGET call with the MQGMO_CONVERT option can suppress data conversion of the message being retrieved.

Initialization and termination exit functions

There are two exit functions in this category, the initialization exit function and the termination exit function.

The initialization exit function is invoked by the queue manager when an application connects to the queue manager. Its primary purpose is to register exit functions and their respective entry points with the queue manager and perform any initialization processing. You do not have to register all the exit functions, only those that are required for this connection. When the application disconnects from the queue manager, the registrations are removed automatically.

The initialization exit function can also be used to acquire any storage required by the exit and examine the values of any environment variables.

The termination exit function is invoked by the queue manager when an application disconnects from the queue manager. Its purpose is to release any storage used by the exit and perform any required cleanup operations.

An API exit can issue calls to the MQI but, if it does, the API exit is not invoked recursively a second time. The following exit functions, however, are not able to issue MQI calls because the correct environment is not present at the time the exit functions are invoked:

- The initialization exit function
- The exit function for an MQCONN and MQCONNX call that is invoked *before* the queue manager starts to process the call
- The exit function for the MQDISC call that is invoked *after* the queue manager has completed processing the call
- The termination exit function

An API exit can also use other APIs that might be available; for example, it can issue calls to DB2®.

An API exit can be used with a WebSphere MQ client application, but it is important to note that the exit is invoked at the *server* end of an MQI channel. See the discussion in Comparing link level security and application level security.

An API exit is written using the C programming language.

To enable an API exit, you must configure it. On i5/OS®, Windows, and UNIX systems, you do this by editing the WebSphere MQ configuration file, mqs.ini, and the queue manager configuration file, qm.ini, for each queue manager.

You configure an API exit by providing the following information:

- The descriptive name of the API exit.
- The name of the module and its location; for example, the full path name.
- The name of the entry point for the initialization exit function.
- The sequence in which the API exit is invoked relative to other API exits. You can configure more than one API exit for a queue manager.
- Optionally, any data to be passed to the API exit.

For more information about how to configure an API exit, see:

- API exits for i5/OS
- API exits for UNIX and Windows systems

For information about how to write an API exit, see Using and writing API exits.

Parent topic: Providing your own application level security

12.2. The API-crossing exit

An *API-crossing exit* is a program that monitors or modifies the function of MQI calls issued by CICS® applications on z/OS®.

Note: The information in this section applies only to CICS applications on z/OS.

The API-crossing exit program is invoked by the CICS adapter and runs in the CICS address space.

The API-crossing exit is invoked for the following MQI calls only:

MQBUFMH
MQCB
MQCB_FUNCTION
MQCLOSE
MQCRTMH
MQCTL
MQDLTMH
MQGET
MQINQ
MQOPEN
MQPUT
MQPUT1
MQSET
MQSTAT

MQSUB

MQSUBRQ

For each MQI call, it is invoked once before the processing of the call has started and once after the processing of the call has been completed.

The exit program can determine the name of an MQI call and can inspect and modify any of the parameters on the call. If it is invoked before an MQI call is processed, it can suppress the call completely.

The exit program can use any of the APIs that a CICS task-related user exit can use; for example, the IMS™, DB2®, and CICS APIs. It can also use any of the MQI calls except MQCONN, MQCONNX, and MQDISC. However, any MQI calls issued by the exit program do not invoke the exit program a second time.

You can write an API-crossing exit in any programming language supported by WebSphere® MQ for z/OS.

Before an API-crossing exit can be used, the exit program load module must be available when the CICS adapter connects to a queue manager. The load module is a CICS program that must be named CSQCAPX and reside in a library in the DFHRPL concatenation sequence. CSQCAPX must be defined in the CICS system definition file (CSD), and the program must be enabled.

An API-crossing exit can be managed using the CICS adapter control panels, CKQC. When CSQCAPX is loaded, a confirmation message is written to the adapter control panels or to the system console. The adapter control panels can also be used to enable or disable the exit program.

For more information about how to write and implement an API-crossing exit, see the ../com.ibm.mq.csqzal.doc/fg15250_.htm.

Parent topic: Providing your own application level security

12.3. Identification and authentication in the API exit and API-crossing exit

An application that receives a message must be able to identify and authenticate the user of the application that sent the message. This service is typically best implemented at the application level. API exits can implement the service in a number of ways.

At the level of an individual message, identification and authentication is a service that involves two users, the sender and the receiver of the message. The basic requirement is for the user of the application that receives the message to be able to identify and authenticate the user of the application that sent the message. Note that the requirement is for one way, not two way, authentication.

Depending on how it is implemented, the users and their applications might need to interface, or even interact, with the service. In addition, when and how the service is used might depend on where the users and their applications are located, and on the nature of the applications themselves. It is therefore natural to consider implementing the service at the application level rather than at the link level.

If you consider implementing this service at the link level, you might need to resolve issues such as the following:

- On a message channel, how do you apply the service only to those messages that require it?
- How do you enable users and their applications to interface, or interact, with the service, if this is a requirement?
- In a multi-hop situation, where a message is sent over more than one message channel on the way to its destination, where do you invoke the components of the service?

Here are some examples of how the identification and authentication service can be implemented at the application level. The term *API exit* means either an API exit or an API-crossing exit.

- When an application puts a message on a queue, an API exit can acquire an authentication token from a trusted authentication server such as Kerberos. The API exit can add this token to the application data in the message. When the message is retrieved by the receiving application, a second API exit can ask the authentication server to authenticate the sender by checking the token.
- When an application puts a message on a queue, an API exit can append the following items to the application data in the message:
  - The digital certificate of the sender
  - The digital signature of the sender

  If different algorithms for generating a message digest are available for use, the API exit can include the name of the algorithm it has used.
  When the message is retrieved by the receiving application, a second API exit can perform the following checks:
  - The API exit can validate the digital certificate by working through the certificate chain to the root CA certificate. To do this, the API exit must have access to a key repository that contains the remaining certificates in the certificate chain. This check provide assurance that the sender, identified by the Distinguished Name, is the genuine owner of the public key contained in the certificate.
  - The API exit can check the digital signature using the public key contained in the certificate. This check authenticates the sender.

  The Distinguished Name of the sender can be sent instead of the whole digital certificate. In this case, the key repository must contain the sender's certificate so that the second API exit can find the public key of the sender. Another possibility is to send all the certificates in the certificate chain.
- When an application puts a message on a queue, the *UserIdentifier* field in the message descriptor contains a user ID associated with the application. The user ID can be used to identify the sender. To enable authentication, an API exit can append some data, such as an encrypted password, to the application data in the message. When the message is retrieved by the receiving application, a second API exit can authenticate the user ID by using the data that has travelled with the message.
  This technique might be considered sufficient for messages that originate in a controlled and trusted environment, and in circumstances where a trusted authentication server or PKI support is not available.

Parent topic: Providing your own application level security

12.4. Access control in the API exit and API-crossing exit

An API or API-crossing exit can provide access controls to supplement those provided by WebSphere® MQ. In particular, the exit can provide access control at the message level. The exit can ensure that an application puts on a queue, or gets from a queue, only those messages that satisfy certain criteria.

Consider the following examples:

- A message contains information about an order. When an application attempts to put a message on a queue, an API or API-crossing exit can check that the total value of the order is less than some prescribed limit.

- Messages arrive on a destination queue from remote queue managers. When an application attempts to get a message from the queue, an API or API-crossing exit can check that the sender of the message is authorized to send a message to the queue.

Parent topic: Providing your own application level security

12.5. Confidentiality in the API exit and API-crossing exit

The application data in a message can be encrypted by an API or API-crossing exit when the message is put by the sending application and decrypted by a second exit when the message is retrieved by the receiving application.

For performance reasons, a symmetric key algorithm is normally used for this purpose. However, at the application level, where many users might be sending messages to each other, the problem is how to ensure that only the intended receiver of a message is able to decrypt the message. One solution is to use a different symmetric key for each pair of users that send messages to each other. But this solution might be difficult and time consuming to administer, particularly if the users belong to different organizations. A standard way of solving this problem is known as *digital enveloping* and uses PKI technology.

When an application puts a message on a queue, an API or API-crossing exit generates a random symmetric key and uses the key to encrypt the application data in the message. The exit encrypts the symmetric key with the public key of the intended receiver. It then replaces the application data in the message with the encrypted application data and the encrypted symmetric key. In this way, only the intended receiver can decrypt the symmetric key and therefore the application data. If an encrypted message has more than one possible intended receiver, the exit can encrypt a copy of the symmetric key for each intended receiver.

If different algorithms for encrypting and decrypting the application data are available for use, the exit can include the name of the algorithm it has used.

Parent topic: Providing your own application level security

12.6. Data integrity in the API exit or API-crossing exit

A message can be digitally signed by an API or API-crossing exit when the message is put by the sending application. The digital signature can then be checked by a second exit when the message is retrieved by the receiving application. This can detect whether the message has been deliberately modified.

As discussed in Data integrity in message exits, some protection can be provided by using a message digest instead of a digital signature.

Parent topic: Providing your own application level security

12.7. Non-repudiation in an API exit or API-crossing exit

An API exit or API-crossing exit can be used to form the basis of a non-repudiation service by logging information about messages received.

Consider an API exit that checks the digital signature of each message that is retrieved from a queue by the receiving application. If the API exit logs sufficient evidence to enable the digital signature to be checked at any time in the future, this can form the basis of a non-repudiation service with proof of origin.

The evidence that is logged might include:

- The digital certificate of the sender
- The digital signature of the sender
- The original message

The API exit can also prepare a delivery report on behalf of the receiver of the message and send it to the reply-to queue specified in the message descriptor of the message. The delivery report might include :

- The date and time of delivery of the message
- The digital certificate of the receiver
- The digital signature of the receiver
- The original message, a subset of the original message, or some means of identifying the original message

When the delivery report is retrieved from the reply-to queue, another API exit can check the digital signature to authenticate the receiver of the original message. If the API exit also logs sufficient evidence to enable the digital signature to be checked at any time in the future, this can form the basis of a non-repudiation service with proof of delivery.

Parent topic: Providing your own application level security

13. Cryptographic hardware

On UNIX and Windows systems, WebSphere® MQ provides support for a variety of cryptographic hardware using the PKCS #11 interface. On i5/OS and z/OS, the operating system provides the cryptographic hardware support.

For a list of currently supported cryptography cards, see Cryptography Card List for WebSphere MQ.

On all platforms, cryptographic hardware is used at the SSL handshaking stage and at secret key reset.

On i5/OS®, when you use DCM to create or renew certificates, you can choose to store the key directly in the coprocessor or to use the coprocessor master key to encrypt the private key and store it in a special key store file.

On z/OS®, when you use RACF® to create certificates, you can choose to store the key using ICSF (Integrated Cryptographic Service Facility) to obtain improved performance and more secure key storage.

On UNIX and Windows systems, WebSphere MQ support is also provided for SSL cryptographic hardware symmetric cipher operations. When using SSL cryptographic hardware symmetric cipher operations, data sent across an SSL or TLS connection is encrypted/decrypted by the cryptographic hardware product.

On the queue manager, this is switched on by setting the SSLCryptoHardware queue manager attribute appropriately (see WebSphere MQ Script (MQSC) Command Reference and WebSphere MQ Programmable Command Formats and Administration Interface). On the Websphere MQ client, equivalent variables are provided (see WebSphere MQ Clients). The default setting is off.

If this attribute is switched on, WebSphere MQ attempts to use symmetric cipher operations whether the cryptographic hardware product supports them for the encryption algorithm specified in the current CipherSpec or not. If the cryptographic hardware product does not provide this support, WebSphere MQ performs the encryption and decryption of data itself, and no error is reported. If the cryptographic hardware product supports symmetric cipher operations for the encryption algorithm specified in the current CipherSpec, this function is activated and the cryptographic hardware product performs the encryption and decryption of the data sent.

In a situation of low CPU usage it is often quicker to perform the encryption/decryption in software, rather than copying the data onto the card, encrypting/decrypting it, and copying it back to the SSL protocol software. Hardware symmetric cipher operations become more useful when the CPU usage is high.

On z/OS with cryptographic hardware, support is provided for symmetric cipher operations. This means that the user's data is encrypted and decrypted by the hardware if the hardware has this capability for the CipherSpec chosen, and is configured to support data encryption and decryption.

On i5/OS, cryptographic hardware is not used for encryption and decryption of the user's data, even if the hardware has the capability of performing such encryption for the encryption algorithm specified in the current CipherSpec.

Parent topic: Security

14. Setting up security on z/OS

Security considerations specific to z/OS.

Security in WebSphere MQ for z/OS is controlled using RACF or an equivalent external security manager (ESM). The instructions given here assume that you are using RACF.

Authority to administer WebSphere MQ on z/OS
This collection of topics describes various aspects of the authority you need to administer WebSphere® MQ for z/OS®.

RACF security classes
RACF® classes are used to hold the profiles required for WebSphere MQ security checking. Many of the member classes have equivalent group classes. You must activate the classes and enable them to accept generic profiles

RACF profiles
All RACF profiles used by WebSphere MQ contain a prefix, which is either the queue manager name or the queue-sharing group name. Be careful when you use the percent sign as a wildcard.

Switch profiles
To control the security checking performed by WebSphere MQ, you use *switch profiles*. A switch profile is a normal RACF profile that has a special meaning to WebSphere MQ. The access list in switch profiles is not used by WebSphere MQ.

Profiles used to control access to WebSphere MQ resources
You must define RACF profiles to control access to WebSphere MQ resources, in addition to the switch profiles that might have been defined. This collection of topics discusses the RACF profiles for the different types of WebSphere MQ resource.

The RESLEVEL security profile
You can define a special profile in the MQADMIN or MXADMIN class to control the number of user IDs checked for API-resource security. This profile is referred to as the RESLEVEL profile How this profile affects API-resource security depends on how you access WebSphere MQ.

User IDs for security checking
WebSphere MQ initiates security checks based on user IDs associated with users, terminals, applications, and other resources. This collection of topics lists which user IDs are used for each type of security check.

WebSphere MQ for z/OS security management
WebSphere MQ uses an in-storage table to hold information relating to each user and the access requests made by each user. To manage this table efficiently and to reduce the number of requests made from WebSphere MQ to the external security manager (ESM), a number of controls are available.

Security installation tasks
After installing and customizing WebSphere MQ, authorize started task procedures to RACF, authorize access to various resources, and set up RACF definitions. Optionally, configure your system for SSL.

Auditing considerations on z/OS
The normal RACF auditing controls are available for conducting a security audit of a queue manager. WebSphere MQ does not gather any security statistics of its own. The only statistics are those that can be created by auditing.

Customizing security
If you want to change the way WebSphere MQ security operates, you must do this through the SAF exit (ICHRFR00), or exits in your external security manager.

Security violation messages on z/OS
A security violation is indicated by the return code MQRC_NOT_AUTHORIZED in an application program or by a message in the job log.

What to do if access is allowed or disallowed incorrectly
In addition to the steps detailed in the *z/OS Security Server RACF Security Administrator's Guide*, use this checklist if access to a resource appears to be incorrectly controlled.

Security considerations for the channel initiator on z/OS
If you are using resource security in a distributed queuing environment, the Channel initiator address space needs appropriate access to various WebSphere MQ resources.

Security in queue manager clusters on z/OS
Security considerations for clusters are like those for queue managers and channels that are not clustered. The channel initiator needs access to some additional system queues, and some additional commands need appropriate security set.

Security considerations for using WebSphere MQ with CICS
The CICS® adapter provides information to WebSphere MQ for use in security.

Security considerations for using WebSphere MQ with IMS
Use this topic to plan your security requirements when you use WebSphere MQ with IMS™.

Security scenario: two queue managers on z/OS
In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The TCP channels can either use SSL or not. The application could be a batch application or a CICS application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

Security scenario: queue-sharing group on z/OS
In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The application is a batch application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

WebSphere MQ for z/OS security implementation checklist
This topic gives a step-by-step procedure you can use to work out and define the security implementation for each of your WebSphere MQ queue managers.

Parent topic: Security

14.1. Authority to administer WebSphere MQ on z/OS

This collection of topics describes various aspects of the authority you need to administer WebSphere® MQ for z/OS®.

Authority checks on z/OS

Command security and command resource security

MQSC commands and the system command input queue

Access to the queue manager data sets

Parent topic: Setting up security on z/OS

14.1.1. Authority checks on z/OS

WebSphere® MQ uses the System Authorization Facility (SAF) to route requests for authority checks to an external security manager (ESM) such as the z/OS® Security Server Resource Access Control Facility (RACF®). WebSphere MQ does no authority checks of its own.

This book assumes that you are using RACF as your ESM. If you are using a different ESM, you might need to interpret the information provided for RACF in a way that is relevant to your ESM.

You can specify whether you want authority checks turned on or off for each queue manager individually or for every queue manager in a queue-sharing group. This level of control is called *subsystem security*. If you turn subsystem security off for a particular queue manager, no authority checks are carried out for that queue manager.

If you turn subsystem security on for a particular queue manager, authority checks can be performed at two levels:

Queue-sharing group level security
Authority checks use RACF profiles that are shared by all queue managers in the queue-sharing group. This means that there are fewer profiles to define and maintain, making security administration easier.

Queue manager level security
Authority checks use RACF profiles specific to the queue manager.

You can use a combination of queue-sharing group and queue manager level security. For example, you can arrange for profiles specific to a queue manager to override those of the queue-sharing group to which it belongs.

Subsystem security, queue-sharing group level security, and queue manager level security are turned on or off by defining *switch profiles*. A switch profile is a normal RACF profile that has a special meaning to WebSphere MQ.

Parent topic: Authority to administer WebSphere MQ on z/OS

14.1.2. Command security and command resource security

Authority checks are carried out when a WebSphere® MQ administrator issues an MQSC command. This is called *command security*.

To implement command security, you must define certain RACF® profiles and give the necessary groups and user IDs access to these profiles at the required levels. The name of a profile for command security contains the name of an MQSC command.

Some MQSC commands perform an operation on a WebSphere MQ resource, such as the DEFINE QLOCAL command to create a local queue. When an administrator issues an MQSC command, authority checks are carried out to determine whether the requested operation can be performed on the resource specified in the command. This is called *command resource security*.

To implement command resource security, you must define certain RACF profiles and give the necessary groups and user IDs access to these profiles at the required levels. The name of a profile for command resource security contains the name of a WebSphere MQ resource and its type (QUEUE, PROCESS, NAMELIST, TOPIC, AUTHINFO, or CHANNEL).

Command security and command resource security are independent. For example, when an administrator issues the command:

```
    DEFINE QLOCAL(MOON.EUROPA)
```

the following authority checks are performed:

- Command security checks that the administrator is authorized to issue the DEFINE QLOCAL command.
- Command resource security checks that the administrator is authorized to perform an operation on the local queue called MOON.EUROPA.

Command security and command resource security can be turned on or off by defining switch profiles.

Parent topic: Authority to administer WebSphere MQ on z/OS

14.1.3. MQSC commands and the system command input queue

Command security and command resource security are also used when the command server retrieves a message containing an MQSC command from the system command input queue. The user ID that is used for the authority checks is the one found in the *UserIdentifier* field in the message descriptor of the message containing the MQSC command. This user ID must have the required authorities on the queue manager where the command is processed. For more information about the *UserIdentifier* field and how it is set, see Message context.

Messages containing MQSC commands are sent to the system command input queue in the following circumstances:

- The operations and control panels send MQSC commands to the system command input queue of the target queue manager. The MQSC commands correspond to the actions you choose on the panels. The *UserIdentifier* field in each message is set to the TSO user ID of the administrator.
- The COMMAND function of the WebSphere® MQ utility program, CSQUTIL, sends the MQSC commands in the input data set to the system command input queue of the target queue manager. The COPY and EMPTY functions send DISPLAY QUEUE and DISPLAY STGCLASS commands. The *UserIdentifier* field in each message is set to the job user ID.
- The MQSC commands in the CSQINPX data sets are sent to the system command input queue of the queue manager to which the channel initiator is connected. The *UserIdentifier* field in each message is set to the channel initiator address space user ID.
  No authority checks are performed when MQSC commands are issued from the CSQINP1 and CSQINP2 data sets. You can control who is allowed to update these data sets using RACF® data set protection.
- Within a queue-sharing group, a channel initiator might send START CHANNEL commands to the system command input queue of the queue manager to which it is connected. A command is sent when an outbound channel that uses a shared transmission queue is started by triggering. The *UserIdentifier* field in each message is set to the channel initiator address space user ID.
- An application can send MQSC commands to a system command input queue. By default, the *UserIdentifier* field in each message is set to the user ID associated with the application.
- On UNIX and Windows systems, the **runmqsc** control command can be used in indirect mode to send MQSC commands to the system command input queue of a queue manager on z/OS®. The *UserIdentifier* field in each message is set to the user ID of the administrator who issued the **runmqsc** command.

Parent topic: Authority to administer WebSphere MQ on z/OS

14.1.4. Access to the queue manager data sets

WebSphere® MQ administrators need authority to access the queue manager data sets. These data sets include:

- The data sets referred to by CSQINP1, CSQINP2, and CSQXLIB in the queue manager's started task procedure
- The queue manager's page sets, active log data sets, archive log data sets, and bootstrap data sets (BSDSs)
- The data sets referred to by CSQXLIB and CSQINPX in the channel initiator's started task procedure

You must protect the data sets so that no unauthorized user can start a queue manager or gain access to any queue manager data. To do this, use RACF® data set protection.

Parent topic: Authority to administer WebSphere MQ on z/OS

14.2. RACF security classes

RACF® classes are used to hold the profiles required for WebSphere® MQ security checking. Many of the member classes have equivalent group classes. You must activate the classes and enable them to accept generic profiles

Each RACF class holds one or more profiles used at some point in the checking sequence, as shown in Table 1.

Table 1. RACF classes used by WebSphere MQ

Member class	Group class	Contents
MQADMIN	GMQADMIN	Profiles: Used mainly for holding profiles for administration-type functions. For example: • Profiles for WebSphere MQ security switches • The RESLEVEL security profile • Profiles for alternate user security • The context security profile • Profiles for command resource security
MXADMIN	GMXADMIN	Profiles: Used mainly for holding profiles for administration-type functions. For example: • Profiles for WebSphere MQ security switches • The RESLEVEL security profile • Profiles for alternate user security • The context security profile • Profiles for command resource security This class can hold both uppercase and mixed case RACF profiles.
MQCONN		Profiles used for connection security
MQCMDS		Profiles used for command security
MQQUEUE	GMQQUEUE	Profiles used in queue resource security
MXQUEUE	GMXQUEUE	Mixed case and uppercase profiles used in queue resource security
MQPROC	GMQPROC	Profiles used in process resource security
MXPROC	GMXPROC	Mixed case and uppercase profiles used in process resource security
MQNLIST	GMQNLIST	Profiles used in namelist resource security
MXNLIST	GMXNLIST	Mixed case and uppercase profiles used in namelist resource security
MXTOPIC	GMXTOPIC	Mixed case and uppercase profiles used in topic security

Some classes have a related *group class* that enables you to put together groups of resources that have similar access requirements. For details about the difference between the member and group classes and when to use a member or group class, see the *z/OS® SecureWay™ Security Server RACF Security Administrator's Guide*.

The classes must be activated before security checks can be made. To activate all the WebSphere MQ classes, you use can use this RACF command:

```
SETROPTS CLASSACT(MQADMIN,MXADMIN,MQQUEUE,MXQUEUE,MQPROC,MXPROC,
                  MQNLIST,MXNLIST,MXTOPIC,MQCONN,MQCMDS)
```

You should also ensure that you set up the classes so that they can accept generic profiles. You also do this with the RACF command SETROPTS, for example:

```
SETROPTS GENERIC(MQADMIN,MXADMIN,MQQUEUE,MXQUEUE,MQPROC,MXPROC,
                 MQNLIST,MXNLIST,MXTOPIC,MQCONN,MQCMDS)
```

Parent topic: Setting up security on z/OS

14.3. RACF profiles

All RACF® profiles used by WebSphere® MQ contain a prefix, which is either the queue manager name or the queue-sharing group name. Be careful when you use the percent sign as a wildcard.

All RACF profiles used by WebSphere MQ contain a prefix. For queue-sharing group level security, this is the queue-sharing group name. For queue manager level security, the prefix is the queue manager name. If you are using a mixture of queue manager and queue-sharing group level security, you will use profiles with both types of prefix. (Queue-sharing group and queue manager level security are described in the WebSphere MQ for z/OS Concepts and Planning Guide.)

For example, if you want to protect a queue called QUEUE_FOR_SUBSCRIBER_LIST in queue-sharing group QSG1 at queue-sharing group level, the appropriate profile would be defined to RACF as:

```
RDEFINE MQQUEUE QSG1.QUEUE_FOR_SUBSCRIBER_LIST
```

If you want to protect a queue called QUEUE_FOR_LOST_CARD_LIST, that belongs to queue manager STCD at queue manager level, the appropriate profile would be defined to RACF as:

```
RDEFINE MQQUEUE STCD.QUEUE_FOR_LOST_CARD_LIST
```

This means that different queue managers and queue-sharing groups can share the same RACF database and yet have different security options.

Do not use generic queue manager names in profiles to avoid unanticipated user access.

WebSphere MQ allows the use of the percent sign (%) in object names. However, RACF uses the % character as a single-character wildcard. This means that when you define an object name with a % character in its name, you must consider this when you define the corresponding profile.

For example, for the queue CREDIT_CARD_%_RATE_INQUIRY, on queue manager CRDP, the profile would be defined to RACF as follows:

```
RDEFINE MQQUEUE CRDP.CREDIT_CARD_%_RATE_INQUIRY
```

This queue cannot be protected by a generic profile, such as, CRDP.**.

WebSphere MQ allows the use of mixed case characters in object names. You can protect these objects by defining:

1. Mixed case profiles in the appropriate mixed case RACF classes, or
2. Generic profiles in the appropriate uppercase RACF classes.

In order to use mixed case profiles and mixed case RACF classes you must follow the steps described in Migrating to mixed-case security.

There are some profiles, or parts of profiles, that remain uppercase only as the values are provided by WebSphere MQ. These are:

- Switch profiles.
- All high-level qualifiers (HLQ) including subsystem and Queue-Sharing Group identifiers.
- Profiles for SYSTEM objects.
- Profiles for Default objects.
- The **MQCMDS** class, so all command profiles are uppercase only.
- The **MQCONN** class, so all connection profiles are uppercase only.
- **RESLEVEL** profiles.
- The `'object'` qualification in command resource profiles; for example, `hlq.QUEUE.queuename`. The resource name only is mixed case.
- Dynamic queue profiles `hlq.CSQOREXX.*` , `hlq.CSQUTIL.*`, and `CSQXCMD.*`.
- The `'CONTEXT'` part of the `hlq.CONTEXT.resourcename`.

For example, if you have a queue called `PAYROLL.Dept1` on Queue Manager `QM01` and you are using:

- Mixed case classes; you can define a profile in the WebSphere MQ RACF class `MXQUEUE`

  ```
  RDEFINE MXQUEUE MQ01.PAYROLL.Dept1
  ```

- Uppercase classes; you can define a profile in the WebSphere MQ RACF class `MQQUEUE`

  ```
  RDEFINE MQQUEUE MQ01.PAYROLL.*
  ```

The first example, using mixed case classes, gives you more granular control over granting authority to access the resource.
Parent topic: Setting up security on z/OS

14.4. Switch profiles

To control the security checking performed by WebSphere® MQ, you use *switch profiles*. A switch profile is a normal RACF® profile that has a special meaning to WebSphere MQ. The access list in switch profiles is not used by WebSphere MQ.

WebSphere MQ maintains an internal switch for each switch type shown in tables Switch profiles for subsystem level security, Switch profiles for queue-sharing group or queue manager level security,and Switch profiles for resource checking. Switch profiles can be maintained at queue-sharing group level, or at queue manager level, or at a combination of both. Using a single set of queue-sharing group security switch profiles, you can control security on all the queue managers within a queue-sharing group.

When a security switch is set on, the security checks associated with the switch are performed. When a security switch is set off, the security checks associated with the switch are bypassed. The default is that all security switches are set on.

Switches and classes
When you start a queue manager or refresh security, WebSphere MQ sets switches according to the state of various RACF classes.

How switches work
To set off a security switch, define a NO.* switch profile for it. You can override a NO.* profile set at the queue-sharing group level by defining a YES.* profile for a queue manager.

Profiles to control subsystem security
WebSphere MQ checks whether subsystem security checks are required for the subsystem, for the queue manager, and for the queue-sharing group.

Profiles to control queue-sharing group or queue manager level security
If subsystem security checking is required, WebSphere MQ checks whether security checking is required at queue-sharing group or queue manager level.

Resource level checks
A number of switch profiles are used to control access to resources. Some stop checking being performed on either a queue manager or a queue-sharing group. These can be overridden by profiles that enable checking for specific queue managers.

An example of defining switches
Different WebSphere MQ subsystems have different security requirements, which can be implemented using different switch profiles.

Parent topic: Setting up security on z/OS

14.4.1. Switches and classes

When you start a queue manager or refresh security, WebSphere MQ sets switches according to the state of various RACF classes.

When a queue manager is started (or when the MQADMIN or MXADMIN class is refreshed by the WebSphere® MQ REFRESH SECURITY command), WebSphere MQ first checks the status of RACF® and the appropriate class:

- The MQADMIN class if you are using uppercase profiles
- The MXADMIN class if you are using mixed case profile.

It sets the subsystem security switch off if any of these conditions is true:

- RACF is inactive or not installed.
- The MQADMIN or MXADMIN class is not defined (these classes are always defined for RACF because they are included in the class descriptor table (CDT)).
- The MQADMIN or MXADMIN class has not been activated.

If both RACF and the MQADMIN or MXADMIN class are active, WebSphere MQ checks the MQADMIN or MXADMIN class to see whether any of the switch profiles have been defined. It first checks the profiles described in Profiles to control subsystem security. If subsystem security is not required, WebSphere MQ sets the internal subsystem security switch off, and performs no further checks.

The profiles determine whether the corresponding WebSphere MQ switch is set on or off.

- If the switch is off, that type of security is deactivated.
- If any WebSphere MQ switch is set on, WebSphere MQ checks the status of the RACF class associated with the type of security corresponding to the WebSphere MQ switch. If the class is not installed or not active, the WebSphere MQ switch is set off. For example, process security checks are not carried out if the MQPROC or MXPROC class has not been activated. The class not being active is equivalent to defining NO.PROCESS.CHECKS profile for every queue manager and queue-sharing group that uses this RACF database.

Parent topic: Switch profiles

14.4.2. How switches work

To set off a security switch, define a NO.* switch profile for it. You can override a NO.* profile set at the queue-sharing group level by defining a YES.* profile for a queue manager.

To set off a security switch, you need to define a NO.* switch profile for it. The existence of a NO.* profile means that security checks are **not** performed for that type of resource, unless you choose to override a queue-sharing group level setting on a particular queue manager. This is described in Overriding queue-sharing group level settings.

If your queue manager is not a member of a queue-sharing group, you do not need to define any queue-sharing group level profiles or any override profiles. However, you must remember to define these profiles if the queue manager joins a queue-sharing group at a later date.

Each NO.* switch profile that WebSphere® MQ detects turns off the checking for that type of resource. Switch profiles are activated during startup of the queue manager. If you change the switch profiles while any affected queue managers are running, you can get WebSphere MQ to recognize the changes by issuing the WebSphere MQ REFRESH SECURITY command.

The switch profiles must always be defined in the MQADMIN or MXADMIN class. Do not define them in the GMQADMIN or GMXADMIN class. Tables Table 1 and Table 1 show the valid switch profiles and the security type they control.

Overriding queue-sharing group level settings

You can override queue-sharing group level security settings for a particular queue manager that is a member of that group. If you want to perform queue manager checks on an individual queue manager that are not performed on other queue managers in the group, use the (qmgr-name.YES.*) switch profiles.

Conversely, if you do not want to perform a certain check on one particular queue manager within a queue-sharing group, define a (qmgr-name.NO.*) profile for that particular resource type on the queue manager, and do not define a profile for the queue-sharing group. (WebSphere MQ only checks for a queue-sharing group level profile if it does not find a queue manager level profile.)

Parent topic: Switch profiles

14.4.3. Profiles to control subsystem security

WebSphere® MQ checks whether subsystem security checks are required for the subsystem, for the queue manager, and for the queue-sharing group.

The first security check made by WebSphere MQ is used to determine whether security checks are required for the whole WebSphere MQ subsystem. If you specify that you do not want subsystem security, no further checks are made.

The following switch profiles are checked to determine whether subsystem security is required. Figure 1 shows the order in which they are checked.

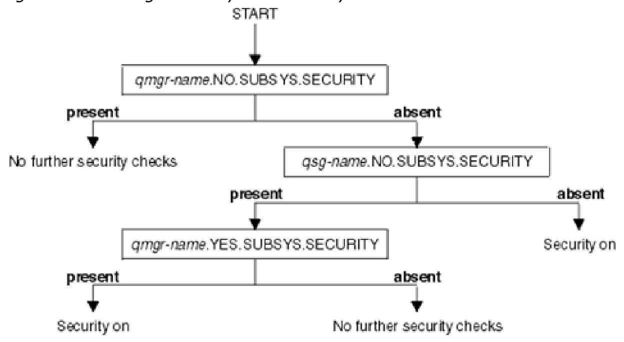
Table 1. Switch profiles for subsystem level security

Switch profile name	Type of resource or checking that is controlled
qmgr-name.NO.SUBSYS.SECURITY	Subsystem security for this queue manager
qsg-name.NO.SUBSYS.SECURITY	Subsystem security for this queue-sharing group

qmgr-name.YES.SUBSYS.SECURITY	Subsystem security override for this queue manager

If your queue manager is not a member of a queue-sharing group, WebSphere MQ checks for the qmgr-name.NO.SUBSYS.SECURITY switch profile only.

Figure 1. Checking for subsystem security


START
qmgr-name.NO.SUBSYS.SECURITY
present
absent
No further security checks
qsg-name.NO.SUBSYS.SECURITY
present
absent
qmgr-name.YES.SUBSYS.SECURITY
Security on
present
absent
Security on
No further security checks

Parent topic: Switch profiles

14.4.4. Profiles to control queue-sharing group or queue manager level security

If subsystem security checking is required, WebSphere MQ checks whether security checking is required at queue-sharing group or queue manager level.

When WebSphere® MQ has determined that security checking is required, it then determines whether checking is required at queue-sharing group or queue manager level, or both. These checks are not performed if your queue manager is not a member of a queue sharing group.

The following switch profiles are checked to determine the level required. Figure 1 and Figure 2 show the order in which they are checked.

Table 1. Switch profiles for queue-sharing group or queue manager level security

Switch profile name	Type of resource or checking that is controlled
qmgr-name.NO.QMGR.CHECKS	No queue manager level checks for this queue manager
qsg-name.NO.QMGR.CHECKS	No queue manager level checks for this queue-sharing group
qmgr-name.YES.QMGR.CHECKS	Queue manager level checks override for this queue manager
qmgr-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue manager
qsg-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue-sharing group
qmgr-name.YES.QSG.CHECKS	Queue-sharing group level checks override for this queue manager

If subsystem security is active, you cannot switch off both queue-sharing group and queue manager level security. If you try to do so, WebSphere MQ sets security checking on at both levels.

Figure 1. Checking for queue manager level security

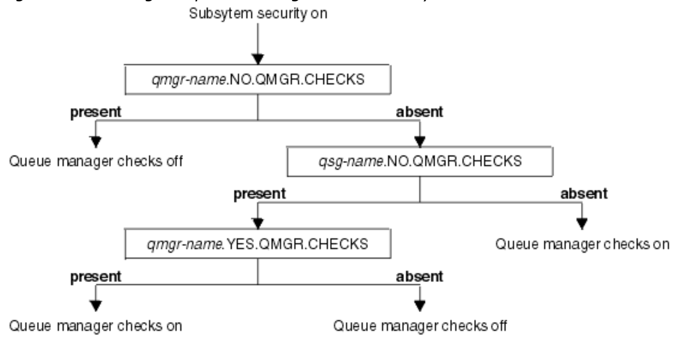
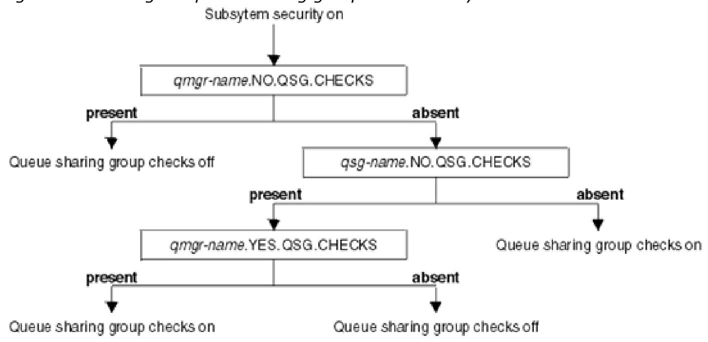


Figure 2. Checking for queue-sharing group level security



Valid combinations of security switches
Only certain combinations of switches are valid. If you use a combination of switch settings that is not valid, message CSQH026I is issued and security checking is set on at both queue-sharing group and queue manager level.

Parent topic: Switch profiles

14.4.4.1. Valid combinations of security switches

Only certain combinations of switches are valid. If you use a combination of switch settings that is not valid, message CSQH026I is issued and security checking is set on at both queue-sharing group and queue manager level.

Table 1, Table 2, Table 3, and Table 4 show the sets of combinations of switch settings that are valid for each type of security level.

Table 1. Valid security switch combinations for queue manager level security

qmgr-name.NO.QSG.CHECKS
qsg-name.NO.QSG.CHECKS
qmgr-name.NO.QSG.CHECKS qsg-name.NO.QMGR.CHECKS qmgr-name.YES.QMGR.CHECKS
qsg-name.NO.QSG.CHECKS qsg-name.NO.QMGR.CHECKS qmgr-name.YES.QMGR.CHECKS

Table 2. Valid security switch combinations for queue-sharing group level security

qmgr-name.NO.QMGR.CHECKS
qsg-name.NO.QMGR.CHECKS
qmgr-name.NO.QMGR.CHECKS qsg-name.NO.QSG.CHECKS qmgr-name.YES.QSG.CHECKS
qsg-name.NO.QMGR.CHECKS qsg-name.NO.QSG.CHECKS qmgr-name.YES.QSG.CHECKS

Table 3. Valid security switch combinations for queue manager and queue-sharing group level security

qsg-name.NO.QMGR.CHECKS qmgr-name.YES.QMGR.CHECKS No QSG.* profiles defined
No QMGR.* profiles defined qsg-name.NO.QSG.CHECKS qmgr-name.YES.QSG.CHECKS
qsg-name.NO.QMGR.CHECKS qmgr-name.YES.QMGR.CHECKS qsg-name.NO.QSG.CHECKS qmgr-name.YES.QSG.CHECKS
No profiles for either switch defined

*Table 4. Other valid security switch combinations that switch both levels of checking **on**.*

qmgr-name.NO.QMGR.CHECKS qmgr-name.NO.QSG.CHECKS
qsg-name.NO.QMGR.CHECKS qsg-name.NO.QSG.CHECKS
qmgr-name.NO.QMGR.CHECKS qsg-name.NO.QSG.CHECKS
qsg-name.NO.QMGR.CHECKS qmgr-name.NO.QSG.CHECKS

Parent topic: Profiles to control queue-sharing group or queue manager level security

14.4.5. Resource level checks

A number of switch profiles are used to control access to resources. Some stop checking being performed on either a queue manager or a queue-sharing group. These can be overridden by profiles that enable checking for specific queue managers.

Table 1 shows the switch profiles used to control access to WebSphere® MQ resources.

If your queue manager is part of a queue sharing group and you have both queue manager and queue-sharing group security active, you can use a YES.* switch profile to override queue-sharing group level profiles and specifically turn on security for a particular queue manager.

Some profiles apply to both queue managers and queue-sharing groups. These are prefixed by the string *hlq* and you should substitute the name of your queue-sharing group or queue manager, as applicable. Profile names shown prefixed by *qmgr-name* are queue-manager override profiles; you should substitute the name of your queue manager.

Table 1. Switch profiles for resource checking

Type of resource	Switch profile name	Override profile for a particular queue manager

checking that is controlled		
Connection security	hlq.NO.CONNECT.CHECKS	qmgr-name.YES.CONNECT.CHECKS
Queue security	hlq.NO.QUEUE.CHECKS	qmgr-name.YES.QUEUE.CHECKS
Process security	hlq.NO.PROCESS.CHECKS	qmgr-name.YES.PROCESS.CHECKS
Namelist security	hlq.NO.NLIST.CHECKS	qmgr-name.YES.NLIST.CHECKS
Context security	hlq.NO.CONTEXT.CHECKS	qmgr-name.YES.CONTEXT.CHECKS
Alternate user security	hlq.NO.ALTERNATE.USER.CHECKS	qmgr-name.YES.ALTERNATE.USER.CHECKS
Command security	hlq.NO.CMD.CHECKS	qmgr-name.YES.CMD.CHECKS
Command resource security	hlq.NO.CMD.RESC.CHECKS	qmgr-name.YES.CMD.RESC.CHECKS
Topic security	hlq.NO.TOPIC.CHECKS	qmgr-name.YES.TOPIC.CHECKS
Note: Generic switch profiles such as hlq.NO.** are ignored by WebSphere MQ		

For example, if you want to perform process security checks on queue manager QM01, which is a member of queue-sharing group QSG3 but you do not want to perform process security checks on any of the other queue managers in the group, define the following switch profiles:

```
QSG3.NO.PROCESS.CHECKS
QM01.YES.PROCESS.CHECKS
```

If you want to have queue security checks performed on all the queue managers in the queue-sharing group, except QM02, define the following switch profile:

```
QM02.NO.QUEUE.CHECKS
```

(There is no need to define a profile for the queue sharing group because the checks are automatically enabled if there is no profile defined.)

Parent topic: Switch profiles

14.4.6. An example of defining switches

Different WebSphere® MQ subsystems have different security requirements, which can be implemented using different switch profiles.

Four WebSphere MQ subsystems have been defined:

- MQP1 (a production system)
- MQP2 (a production system)
- MQD1 (a development system)
- MQT1 (a test system)

All four queue managers are members of queue-sharing group QS01. All WebSphere MQ RACF® classes have been defined and activated.

These subsystems have different security requirements:

- The production systems require full WebSphere MQ security checking to be active at queue-sharing group level on both systems.
  This is done by specifying the following profile:
  ```
  RDEFINE MQADMIN QS01.NO.QMGR.CHECKS
  ```
  This sets queue-sharing group level checking for all the queue managers in the queue-sharing group. You do not need to define any other switch profiles for the production queue managers because you want to check everything for these systems.
- Test queue manager MQT1 also requires full security checking. However, because you might want to change this later, security can be defined at queue-manager level so that you can change the security settings for this queue manager without affecting the other members of the queue-sharing group.
  This is done by defining the NO.QSG.CHECKS profile for MQT1 as follows:
  ```
  RDEFINE MQADMIN MQT1.NO.QSG.CHECKS
  ```
- Development queue manager MQD1 has different security requirements from the rest of the queue-sharing group. It requires only connection and queue security to be active.
  This is done by defining a `MQD1.YES.QMGR.CHECKS` profile for this queue manager, and then defining the following profiles to switch off security checking for the resources that do not need to be checked:
  ```
  RDEFINE MQADMIN MQD1.NO.CMD.CHECKS
  RDEFINE MQADMIN MQD1.NO.CMD.RESC.CHECKS
  RDEFINE MQADMIN MQD1.NO.PROCESS.CHECKS
  RDEFINE MQADMIN MQD1.NO.NLIST.CHECKS
  RDEFINE MQADMIN MQD1.NO.CONTEXT.CHECKS
  RDEFINE MQADMIN MQD1.NO.ALTERNATE.USER.CHECKS
  ```

When the queue manager is active, you can display the current security settings by issuing the DISPLAY SECURITY MQSC command.

You can also change the switch settings when the queue manager is running by defining or deleting the appropriate switch profile in the MQADMIN class. To make the changes to the switch settings active, you must issue the REFRESH SECURITY command for the MQADMIN class.

See Refreshing queue manager security on z/OS for more details about using the DISPLAY SECURITY and REFRESH SECURITY commands.

Parent topic: Switch profiles

14.5. Profiles used to control access to WebSphere MQ resources

You must define RACF® profiles to control access to WebSphere® MQ resources, in addition to the switch profiles that might have been defined. This collection of topics discusses the RACF profiles for the different types of WebSphere MQ resource.

If you do not have a resource profile defined for a particular security check, and a user issues a request that would involve making that check, WebSphere MQ denies access. You do not need to define profiles for security types relating to any security switches that you have deactivated.

Profiles for connection security
If connection security is active, you must define profiles in the MQCONN class and permit the necessary groups or user IDs access to those profiles, so that they can connect to WebSphere MQ.

Profiles for queue security
If queue security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to these profiles. Queue security profiles are named after the queue manager or queue-sharing group, and the queue to be opened.

Profiles for topic security
If topic security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

Profiles for processes
If process security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

Profiles for namelists
If namelist security is active, you define profiles in the appropriate classes and give the necessary groups or user IDs access to these profiles.

Profiles for alternate user security
If alternate user security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

Profiles for context security
WebSphere MQ uses profiles for controlling access to the context information specific to a particular message. The context is contained within the message descriptor (MQMD).

Profiles for command security
To enable security checking for commands, add profiles to the MQCMDS class. The profile names are based on the MQSC commands but control both MQSC and PCF commands. Profiles can apply to a queue manager or a queue-sharing group.

Profiles for command resource security
If you have not defined the command resource security switch profile, because you want security checking for resources associated with commands, you must add resource profiles for each resource to the appropriate class. The same security profiles control both MQSC and PCF commands.

Parent topic: Setting up security on z/OS

14.5.1. Profiles for connection security

If connection security is active, you must define profiles in the MQCONN class and permit the necessary groups or user IDs access to those profiles, so that they can connect to WebSphere® MQ.

To enable a connection to be made, you must grant users RACF® READ access to the appropriate profile. (If no queue manager level profile exists, and your queue manager is a member of a queue-sharing group, checks might be made against queue-sharing group level profiles, if the security is set up to do this.)

A connection profile qualified with a queue manager name controls access to a specific queue manager and users given access to this profile can connect to that queue manager. A connection profile qualified with queue-sharing group name controls access to all queue managers within the queue-sharing group for that connection type. For example, a user with access to `QS01.BATCH` can use a batch connection to any queue manager in queue-sharing group QS01 that has not got a queue manager level profile defined.

Note:
1. For information about the user IDs checked for different security requests, see User IDs for security checking.
2. Resource level security (RESLEVEL) checks are also made at connection time. For details, see The RESLEVEL security profile.

WebSphere MQ security recognizes the following different types of connection:
- Batch (and batch-type) connections, these include:
  - z/OS® batch jobs
  - TSO applications
  - USS sign-ons
  - DB2® stored procedures
- CICS® connections
- IMS™ connections from control and application processing regions
- The WebSphere MQ channel initiator

Connection security profiles for batch connections
Profiles for checking batch-type connections are composed of the queue manager or queue-sharing group name followed by the word *BATCH*. Give the user ID associated with the connecting address space READ access to the connection profile.

Connection security profiles for CICS connections
Profiles for checking CICS connections are composed of the queue manager or queue-sharing group name followed by the word *CICS*. Give the user ID associated with the CICS address space READ access to the connection profile.

Connection security profiles for IMS connections
Profiles for checking IMS connections are composed of the queue manager or queue-sharing group name followed by the word *IMS*. Give the IMS control and dependent region user IDs READ access to the connection profile.

Connection security profiles for the channel initiator
Profiles for checking connections from the channel initiator are composed of the queue manager or queue-sharing group name followed by the word *CHIN*. Give the user ID used by the channel initiator started task address space READ access to the connection profile.

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.1.1. Connection security profiles for batch connections

Profiles for checking batch-type connections are composed of the queue manager or queue-sharing group name followed by the word *BATCH*. Give the user ID associated with the connecting address space READ access to the connection profile.

Profiles for checking batch and batch-type connections take the form:

```
hlq.BATCH
```

where `hlq` can be either the `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails.

For batch or batch-type connection requests, you must permit the user ID associated with the connecting address space to access the connection profile. For example, the following RACF® command allows users in the CONNTQM1 group to connect to the queue manager TQM1; these user IDs will be permitted to use any batch or batch-type connection.

```
RDEFINE MQCONN TQM1.BATCH UACC(NONE)
PERMIT TQM1.BATCH CLASS(MQCONN) ID(CONNTQM1) ACCESS(READ)
```

Parent topic: Profiles for connection security

14.5.1.2. Connection security profiles for CICS connections

Profiles for checking CICS® connections are composed of the queue manager or queue-sharing group name followed by the word *CICS*. Give the user ID associated with the CICS address space READ access to the connection profile.

Profiles for checking connections from CICS take the form:

```
hlq.CICS
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by CICS, you need only permit the CICS address space user ID access to the connection profile.

For example, the following RACF® commands allow the CICS address space user ID KCBCICS to connect to the queue manager TQM1:

```
RDEFINE MQCONN TQM1.CICS UACC(NONE)
PERMIT TQM1.CICS CLASS(MQCONN) ID(KCBCICS) ACCESS(READ)
```

Parent topic: Profiles for connection security

14.5.1.3. Connection security profiles for IMS connections

Profiles for checking IMS™ connections are composed of the queue manager or queue-sharing group name followed by the word *IMS*. Give the IMS control and dependent region user IDs READ access to the connection profile.

Profiles for checking connections from IMS take the form:

```
hlq.IMS
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by IMS, permit access to the connection profile for the IMS control and dependent region user IDs.

For example, the following RACF® commands allow:

- The IMS region user ID, IMSREG, to connect to the queue manager TQM1.
- Users in group BMPGRP to submit BMP jobs.

```
RDEFINE MQCONN TQM1.IMS UACC(NONE)
PERMIT TQM1.IMS CLASS(MQCONN) ID(IMSREG,BMPGRP) ACCESS(READ)
```

Parent topic: Profiles for connection security

14.5.1.4. Connection security profiles for the channel initiator

Profiles for checking connections from the channel initiator are composed of the queue manager or queue-sharing group name followed by the word *CHIN*. Give the user ID used by the channel initiator started task address space READ access to the connection profile.

Profiles for checking connections from the channel initiator take the form:

```
hlq.CHIN
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name). If you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name. If it does not find one, it looks for a profile prefixed by the queue-sharing group name. If it fails to find either profile, the connection request fails

For connection requests by the channel initiator, define access to the connection profile for the user ID used by the channel initiator started task address space.

For example, the following RACF® commands allow the channel initiator address space running with user ID DQCTRL to connect to the queue manager TQM1:

```
RDEFINE MQCONN TQM1.CHIN UACC(NONE)
PERMIT TQM1.CHIN CLASS(MQCONN) ID(DQCTRL) ACCESS(READ)
```

Parent topic: Profiles for connection security

14.5.2. Profiles for queue security

If queue security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to these profiles. Queue security profiles are named after the queue manager or queue-sharing group, and the queue to be opened.

If queue security is active, you must:

- Define profiles in the **MQQUEUE** or **GMQQUEUE** classes if using uppercase profiles.
- Define profiles in the **MXQUEUE** or **GMXQUEUE** classes if using mixed case profiles.
- Permit the necessary groups or user IDs access to these profiles, so that they can issue WebSphere® MQ API requests that use queues.

Profiles for queue security take the form:

```
hlq.queuename
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `queuename` is the name of the queue being opened, as specified in the object descriptor on the **MQOPEN** or **MQPUT1** call.

A profile prefixed by the queue manager name controls access to a single queue on that queue manager. A profile prefixed by the queue-sharing group name controls access to access to one or more queues with that queue name on all queue managers within the queue-sharing group, or access to a shared queue by any queue manager within the group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that queue on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

If you are using shared queues, you are recommended to use queue-sharing group level security.

For details of how queue security operates when the queue name is that of an alias or a model queue, see Considerations for alias queues and Considerations for model queues.

The RACF® access required to open a queue depends on the **MQOPEN** or **MQPUT1** options specified. If more than one of the MQOO_* and MQPMO_* options is coded, the queue security check is performed for the highest RACF authority required.

Table 1. Access levels for queue security using the MQOPEN or MQPUT1 calls

MQOPEN or MQPUT1 option	RACF access level required to access hlq.queuename
MQOO_BROWSE	READ
MQOO_INQUIRE	READ
MQOO_BIND_*	UPDATE
MQOO_INPUT_*	UPDATE
MQOO_OUTPUT or MQPUT1	UPDATE
MQOO_PASS_ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT	UPDATE
MQOO_PASS_IDENTITY_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT	UPDATE
MQOO_SAVE_ALL_CONTEXT	UPDATE
MQOO_SET_IDENTITY_CONTEXT MQPMO_SET_IDENTITY_CONTEXT	UPDATE
MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT	UPDATE
MQOO_SET	ALTER

For example, on WebSphere MQ queue manager QM77, all user IDs in the RACF group PAYGRP are to be given access to get messages from or put messages to all queues with names beginning with 'PAY.'. You can do this using these RACF commands:

```
RDEFINE MQQUEUE QM77.PAY.** UACC(NONE)
PERMIT QM77.PAY.** CLASS(MQQUEUE) ID(PAYGRP) ACCESS(UPDATE)
```

Also, all user IDs in the PAYGRP group must have access to put messages on queues that do not follow the PAY naming convention. For example:

```
REQUEST_QUEUE_FOR_PAYROLL
SALARY.INCREASE.SERVER
REPLIES.FROM.SALARY.MODEL
```

You can do this by defining profiles for these queues in the GMQQUEUE class and giving access to that class as follows:

```
RDEFINE GMQQUEUE PAYROLL.EXTRAS UACC(NONE)
        ADDMEM(QM77.REQUEST_QUEUE_FOR_PAYROLL,
               QM77.SALARY.INCREASE.SERVER,
               QM77.REPLIES.FROM.SALARY.MODEL)
PERMIT PAYROLL.EXTRAS CLASS(GMQQUEUE) ID(PAYGRP) ACCESS(UPDATE)
```

Note:

1. If the RACF access level that an application has to a queue security profile is changed, the changes only take effect for any new object handles obtained (that is, new **MQOPEN**s) for that queue. Those handles already in existence at the time of the change retain their existing access to the queue. If an application is required to use its changed access level to the queue rather than its existing access level, it must close and reopen the queue for each object handle that requires the change.
2. In the example, the queue manager name `QM77` could also be the name of a queue-sharing group.

Other types of security checks might also occur at the time the queue is opened depending on the open options specified and the types of security that are active. See also Profiles for context security and Profiles for alternate user security. For a summary table showing the open options and the security authorization needed when queue, context, and alternate user security are all active, see Table 1.

If you are using publish/subscribe you must consider the following. When an MQSUB request is processed a security check is performed to ensure that the user ID making the request has the required access to put messages to the target WebSphere MQ queue as well as the required access to subscribe to the WebSphere MQ topic.

Table 2. Access levels for queue security using the MQSUB call

MQSUB option	RACF access level required to access hlq.queuename
MQSO_ALTER, MQSO_CREATE, and MQSO_RESUME	UPDATE

Note:

1. The `hlq.queuename` is the destination queue for publications. When this is a managed queue, you need access to the appropriate model queue to be used for the managed queue and the dynamic queue that are created.
2. You can use a technique like this for the destination queue you provide on an MQSUB API call if you want to distinguish between the users making the subscriptions, and the users retrieving the publications from the destination queue.

Considerations for alias queues
When you issue an **MQOPEN** or **MQPUT1** call for an alias queue, WebSphere MQ makes a resource check against the queue name specified in the object descriptor (MQOD) on the call. It does not check if the user is allowed access to the target queue name.

Using alias queues to distinguish between MQGET and MQPUT requests
The range of MQI calls available in one access level can cause a problem if you want to restrict access to a queue to allow only the **MQPUT** call or only the **MQGET** call. A queue can be protected by defining two aliases that resolve to that queue: one that enables applications to get messages from the queue, and one that enable applications to put messages on the queue.

Considerations for model queues
To open a model queue, you must be able to open both the model queue itself and the dynamic queue to which it resolves. Define generic RACF profiles for dynamic queues, including dynamic queues used by WebSphere MQ utilities.

Close options on permanent dynamic queues
If an application opens a permanent dynamic queue that was created by another application and then attempts to delete that queue with an **MQCLOSE** option, some extra security checks are applied when the attempt is made.

Security and remote queues
When a message is put on a remote queue, the queue security that is implemented by the local queue manager depends on how the remote queue is specified when it is opened.

Dead-letter queue security
Special considerations apply to the dead-letter queue, because many users must be able to put messages on it, but access to retrieve messages must be tightly restricted. You can achieve this by applying different RACF authorities to the dead-letter queue and an alias queue.

System queue security
You must set up RACF access to allow certain user IDs access to particular system queues.

API-resource security access quick reference
A summary of the **MQOPEN**, **MQPUT1**, **MQSUB**, and **MQCLOSE** options and the access required by the different resource security types.

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.2.1. Considerations for alias queues

When you issue an **MQOPEN** or **MQPUT1** call for an alias queue, WebSphere® MQ makes a resource check against the queue name specified in the object descriptor (MQOD) on the call. It does not check if the user is allowed access to the target queue name.

For example, an alias queue called PAYROLL.REQUEST resolves to a target queue of PAY.REQUEST. If queue security is active, you need only be authorized to access the queue PAYROLL.REQUEST. No check is made to see if you are authorized to access the queue PAY.REQUEST.

Parent topic: Profiles for queue security

14.5.2.2. Using alias queues to distinguish between MQGET and MQPUT requests

The range of MQI calls available in one access level can cause a problem if you want to restrict access to a queue to allow only the **MQPUT** call or only the **MQGET** call. A queue can be protected by defining two aliases that resolve to that queue: one that enables applications to get messages from the queue, and one that enable applications to put messages on the queue.

The following text gives you an example of how you can define your queues to WebSphere® MQ:

```
DEFINE QLOCAL(MUST_USE_ALIAS_TO_ACCESS) GET(ENABLED)
       PUT(ENABLED)

DEFINE QALIAS(USE_THIS_ONE_FOR_GETS) GET(ENABLED)
       PUT(DISABLED) TARGQ(MUST_USE_ALIAS_TO_ACCESS)

DEFINE QALIAS(USE_THIS_ONE_FOR_PUTS) GET(DISABLED)
       PUT(ENABLED) TARGQ(MUST_USE_ALIAS_TO_ACCESS)
```

You must also make the following RACF® definitions:

```
RDEFINE MQQUEUE hlq.MUST_USE_ALIAS_TO_ACCESS UACC(NONE)
RDEFINE MQQUEUE hlq.USE_THIS_ONE_FOR_GETS UACC(NONE)
RDEFINE MQQUEUE hlq.USE_THIS_ONE_FOR_PUTS UACC(NONE)
```

Then you ensure that no users have access to the queue hlq.MUST_USE_ALIAS_TO_ACCESS, and give the appropriate users or groups access to the alias. You can do this using the following RACF commands:

```
PERMIT hlq.USE_THIS_ONE_FOR_GETS CLASS(MQQUEUE)
       ID(GETUSER,GETGRP) ACCESS(UPDATE)
PERMIT hlq.USE_THIS_ONE_FOR_PUTS CLASS(MQQUEUE)
       ID(PUTUSER,PUTGRP) ACCESS(UPDATE)
```

This means user ID GETUSER and user IDs in the group GETGRP are only allowed to get messages on MUST_USE_ALIAS_TO_ACCESS through the alias queue USE_THIS_ONE_FOR_GETS; and user ID PUTUSER and user IDs in the group PUTGRP are only allowed to put messages through the alias queue USE_THIS_ONE_FOR_PUTS.

Note:

1. If you want to use a technique like this, you must inform your application developers, so that they can design their programs appropriately.
2. You can use a technique like this for the destination queue you provide on and MQSUB API request if you want to distinguish between the users making the subscriptions and the users 'getting' the publications from the destination queue.

Parent topic: Profiles for queue security

14.5.2.3. Considerations for model queues

To open a model queue, you must be able to open both the model queue itself and the dynamic queue to which it resolves. Define generic RACF profiles for dynamic queues, including dynamic queues used by WebSphere® MQ utilities.

When you open a model queue, WebSphere MQ security makes two queue security checks:

1. Are you authorized to access the model queue?
2. Are you authorized to access the dynamic queue to which the model queue resolves?

If the dynamic queue name contains a trailing asterisk (*) character, this * is replaced by a character string generated by WebSphere MQ, to create a dynamic queue with a unique name. However, because the whole name, including this generated string, is used for checking authority, you should define generic profiles for these queues.

For example, an **MQOPEN** call uses a model queue name of CREDIT.CHECK.REPLY.MODEL and a dynamic queue name of CREDIT.REPLY.* on queue manager (or queue-sharing group) MQSP.

To do this, you must issue the following RACF® commands to define the necessary queue profiles:

```
RDEFINE MQQUEUE MQSP.CREDIT.CHECK.REPLY.MODEL
RDEFINE MQQUEUE MQSP.CREDIT.REPLY.**
```

You must also issue the corresponding RACF PERMIT commands to allow the user access to these profiles.

A typical dynamic queue name created by an **MQOPEN** is something like CREDIT.REPLY.A346EF00367849A0. The precise value of the last qualifier is unpredictable; this is why you should use generic profiles for such queue names.

A number of WebSphere MQ utilities put messages on dynamic queues. You should define profiles for the following dynamic queue names, and provide RACF UPDATE access to the relevant user IDs (see User IDs for security checking for the correct user IDs):

```
SYSTEM.CSQUTIL.*   (used by CSQUTIL)
SYSTEM.CSQOREXX.*  (used by the operations and control panels)
SYSTEM.CSQXCMD.*   (used by the channel initiator when processing CSQINPX)
CSQ4SAMP.*         (used by the WebSphere MQ supplied samples)
```

You might also consider defining a profile to control use of the dynamic queue name used by default in the application programming copy members. The WebSphere MQ-supplied copybooks contain a default *DynamicQName*, which is CSQ.*. This enables an appropriate RACF profile to be established.

Note: Do not allow application programmers to specify a single * for the dynamic queue name. If you do, you must define an hlq.** profile in the MQQUEUE class, and you would have to give it wide-ranging access. This means that this profile could also be used for other non-dynamic queues that do not have a more specific RACF profile. Your users could, therefore, gain access to queues you do not want them to access.

Parent topic: Profiles for queue security

14.5.2.4. Close options on permanent dynamic queues

If an application opens a permanent dynamic queue that was created by another application and then attempts to delete that queue with an **MQCLOSE** option, some extra security checks are applied when the attempt is made.

Table 1. Access levels for close options on permanent dynamic queues

MQCLOSE option	RACF® access level required to hlq.queuename
MQCO_DELETE	ALTER
MQCO_DELETE_PURGE	ALTER

Parent topic: Profiles for queue security

14.5.2.5. Security and remote queues

When a message is put on a remote queue, the queue security that is implemented by the local queue manager depends on how the remote queue is specified when it is opened.

The following rules are applied:

1. If the remote queue has been defined on the local queue manager through the WebSphere® MQ DEFINE QREMOTE command, the queue that is checked is the name of the remote queue. For example, if a remote queue is defined on queue manager MQS1 as follows:
   ```
   DEFINE QREMOTE(BANK7.CREDIT.REFERENCE)
          RNAME(CREDIT.SCORING.REQUEST)
          RQMNAME(BNK7)
          XMITQ(BANK1.TO.BANK7)
   ```
   In this case, a profile for BANK7.CREDIT.REFERENCE must be defined in the MQQUEUE class.
2. If the *ObjectQMgrName* for the request does not resolve to the local queue manager, a security check is carried out against the resolved (remote) queue manager name except in the case of a cluster queue where the check is made against the cluster queue name.
   For example, the transmission queue BANK1.TO.BANK7 is defined on queue manager MQS1. An **MQPUT1** request is then issued on MQS1 specifying *ObjectName* as BANK1.INTERBANK.TRANSFERS and an *ObjectQMgrName* of BANK1.TO.BANK7. In this case, the user performing the request must have access to BANK1.TO.BANK7.
3. If you make an **MQPUT** request to a queue and specify `ObjectQMgrName` as the name of an alias of the local queue manager, only the queue name is checked for security, not that of the queue manager.

When the message gets to the remote queue manager it might be subject to additional security processing. For more information, see the WebSphere MQ Intercommunication manual.

Parent topic: Profiles for queue security

14.5.2.6. Dead-letter queue security

Special considerations apply to the dead-letter queue, because many users must be able to put messages on it, but access to retrieve messages must be tightly restricted. You can achieve this by applying different RACF authorities to the dead-letter queue and an alias queue.

Undelivered messages can be put on a special queue called the dead-letter queue. If you have sensitive data that could possibly end up on this queue, you must consider the security implications of this because you do not want unauthorized users to retrieve this data.

Each of the following must be allowed to put messages onto the dead-letter queue:

- Application programs.
- The channel initiator address space and any MCA user IDs. (If the RESLEVEL profile is not present, or is defined so that network-received user IDs are checked, the network-received user ID also needs authority to put messages on the dead-letter queue.)
- CKTI, the WebSphere® MQ-supplied CICS® task initiator.
- CSQQTRMN, the WebSphere MQ-supplied IMS™ trigger monitor.

The only application that can retrieve messages from the dead-letter queue should be a 'special' application that processes these messages. However, a problem arises if you give applications RACF® UPDATE authority to the dead-letter queue for **MQPUT**s because they can then automatically retrieve messages from the queue using **MQGET** calls. You cannot disable the dead-letter queue for get operations because, if you do, not even the 'special' applications could retrieve the messages.

One solution to this problem is set up a two-level access to the dead-letter queue. CKTI, message channel agent transactions or the channel initiator address space, and 'special' applications have direct access; other applications can only access the dead-letter queue through an alias queue. This alias is defined to allow applications to put messages on the dead-letter queue, but not to get messages from it.

This is how it might work:

1. Define the real dead-letter queue with attributes PUT(ENABLED) and GET(ENABLED), as shown in the sample thlqual.SCSQPROC(CSQ4INYG).
2. Give RACF UPDATE authority for the dead-letter queue to the following user IDs:
   - User IDs that the CKTI and the MCAs or channel initiator address space run under.
   - The user IDs associated with the 'special' dead-letter queue processing application.
3. Define an alias queue that resolves to the real dead-letter queue, but give the alias queue these attributes: PUT(ENABLED) and GET(DISABLED). Give the alias queue a name with the same stem as the dead-letter queue name but append the characters ".PUT" to this stem. For example, if the dead-letter queue name is hlq.DEAD.QUEUE, the alias queue name would be hlq.DEAD.QUEUE.PUT.
4. To put a message on the dead-letter queue, an application uses the alias queue. This is what your application must do:
   - Retrieve the name of the real dead-letter queue. To do this, it opens the queue manager object using **MQOPEN** and then issues an **MQINQ** to get the dead-letter queue name.
   - Build the name of the alias queue by appending the characters '.PUT' to this name, in this case, hlq.DEAD.QUEUE.PUT.

- Open the alias queue, hlq.DEAD.QUEUE.PUT.
- Put the message on the real dead-letter queue by issuing an **MQPUT** against the alias queue.

5. Give the user ID associated with the application RACF UPDATE authority to the alias, but no access (authority NONE) to the real dead-letter queue. This means that:
   - The application can put messages onto the dead-letter queue using the alias queue.
   - The application cannot get messages from the dead-letter queue using the alias queue because the alias queue is disabled for get operations.

   The application cannot get any messages from the real dead-letter queue either because it does have the correct RACF authority.

Table 1 summarizes the RACF authority required for the various participants in this solution.

Table 1. RACF authority to the dead-letter queue and its alias

Associated user IDs	Real dead-letter queue (hlq.DEAD.QUEUE)	Alias dead-letter queue (hlq.DEAD.QUEUE.PUT)
MCA or channel initiator address space and CKTI	UPDATE	NONE
'Special' application (for dead-letter queue processing)	UPDATE	NONE
User-written application user IDs	NONE	UPDATE

If you use this method, the application cannot determine the maximum message length (MAXMSGL) of the dead-letter queue. This is because the MAXMSGL attribute cannot be retrieved from an alias queue. Therefore, your application should assume that the maximum message length is 100 MB, the maximum size WebSphere MQ for z/OS® supports. The real dead-letter queue should also be defined with a MAXMSGL attribute of 100 MB.

Note: User-written application programs do not normally use alternate user authority to put messages on the dead-letter queue. This reduces the number of user IDs that have access to the dead-letter queue.

Parent topic: Profiles for queue security

14.5.2.7. System queue security

You must set up RACF® access to allow certain user IDs access to particular system queues.

Many of the system queues are accessed by the ancillary parts of WebSphere® MQ:

- The CSQUTIL utility
- The operations and control panels
- The channel initiator address space ❯(including the Queued Pub/Sub Daemon)❮

The user IDs under which these run must be given RACF access to these queues, as shown in Table 1.

Table 1. Access required to the SYSTEM queues by WebSphere MQ

SYSTEM queue	CSQUTIL	Operations and control panels	Channel initiator for distributed queuing
SYSTEM.ADMIN.CHANNEL.EVENT	–	–	UPDATE
❯SYSTEM.BROKER.ADMIN.STREAM❮	❯–❮	❯–❮	❯ALTER❮
❯SYSTEM.BROKER.CONTROL.QUEUE❮	❯–❮	❯–❮	❯ALTER❮
❯SYSTEM.BROKER.DEFAULT.STREAM❮	❯–❮	❯–❮	❯ALTER❮
❯SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS❮	❯–❮	❯–❮	❯UPDATE❮
SYSTEM.CHANNEL.INITQ	–	–	UPDATE
SYSTEM.CHANNEL.SYNCQ	–	–	UPDATE
SYSTEM.CLUSTER.COMMAND.QUEUE	–	–	ALTER
SYSTEM.CLUSTER.REPOSITORY.QUEUE	–	–	UPDATE
SYSTEM.CLUSTER.TRANSMIT.QUEUE	–	–	ALTER
SYSTEM.COMMAND.INPUT	UPDATE	UPDATE	UPDATE
❯SYSTEM.COMMAND.REPLY.*❮	❯–❮	❯–❮	❯UPDATE❮
SYSTEM.COMMAND.REPLY.MODEL	UPDATE	UPDATE	UPDATE
SYSTEM.CSQOREXX.*	–	UPDATE	–
SYSTEM.CSQUTIL.*	UPDATE	–	–
SYSTEM.CSQXCMD.*	–	–	UPDATE
SYSTEM.HIERARCHY.STATE	–	–	UPDATE
SYSTEM.INTER.QMGR.CONTROL	–	–	UPDATE
SYSTEM.INTER.QMGR.PUBS	–	–	UPDATE
SYSTEM.INTER.QMGR.FANREQ	–	–	UPDATE
SYSTEM.QSG.CHANNEL.SYNCQ	–	–	UPDATE
SYSTEM.QSG.TRANSMIT.QUEUE	–	–	UPDATE

Parent topic: Profiles for queue security

14.5.2.8. API-resource security access quick reference

›A summary of the **MQOPEN**, **MQPUT1**, **MQSUB**, and **MQCLOSE** options and the access required by the different resource security types.‹

Table 1. MQOPEN, MQPUT1, MQSUB, and MQCLOSE options and the security authorization required. Callouts shown like this ***(1)*** *refer to the notes following this table.*

	Minimum RACF® access level required			
RACF class:	**MXTOPIC**	**MQQUEUE or MXQUEUE(1)**	**MQADMIN or MXADMIN**	**MQADMIN or MXADMIN**
RACF profile:	**(15 or 16)**	**(2)**	**(3)**	**(4)**
MQOPEN option				
MQOO_INQUIRE		READ **(5)**	No check	No check
MQOO_BROWSE		READ	No check	No check
MQOO_INPUT_*		UPDATE	No check	No check
MQOO_SAVE_ALL_CONTEXT **(6)**		UPDATE	No check	No check
MQOO_OUTPUT (USAGE=NORMAL) **(7)**		UPDATE	No check	No check
MQOO_PASS_IDENTITY_CONTEXT **(8)**		UPDATE	READ	No check
MQOO_PASS_ALL_CONTEXT **(8) (9)**		UPDATE	READ	No check
MQOO_SET_IDENTITY_CONTEXT **(8) (9)**		UPDATE	UPDATE	No check
MQOO_SET_ALL_CONTEXT **(8) (10)**		UPDATE	CONTROL	No check
MQOO_OUTPUT (USAGE (XMITQ) **(›11‹)**		UPDATE	CONTROL	No check
MQOO_OUTPUT (topic object)	UPDATE **(16)**			
MQOO_OUTPUT (alias queue to topic object)	UPDATE **(16)**	UPDATE		
MQOO_SET		ALTER	No check	No check
MQOO_ALTERNATE_USER_AUTHORITY		**(12)**	**(12)**	UPDATE
MQPUT1 option				
Put on a normal queue **(7)**		UPDATE	No check	No check
MQPMO_PASS_IDENTITY_CONTEXT		UPDATE	READ	No check
MQPMO_PASS_ALL_CONTEXT		UPDATE	READ	No check
MQPMO_SET_IDENTITY_CONTEXT		UPDATE	UPDATE	No check
MQPMO_SET_ALL_CONTEXT		UPDATE	CONTROL	No check
MQOO_OUTPUT Put on a transmission queue **(11)**		UPDATE	CONTROL	No check
MQOO_OUTPUT (topic object)	UPDATE **(16)**			
MQOO_OUTPUT (alias queue to topic object)	UPDATE **(16)**	UPDATE		
MQPMO_ALTERNATE_USER_AUTHORITY		**(13)**	**(13)**	UPDATE
MQCLOSE option				
MQCO_DELETE **(14)**		ALTER	No check	No check
MQCO_DELETE_PURGE **(14)**		ALTER	No check	No check
MQCO_REMOVE_SUB	ALTER **(15)**			
MQSUB option				
MQSO_CREATE	ALTER **(15)**	**(17)**	**(18)**	
MQSO_ALTER	ALTER **(15)**	**(17)**	**(18)**	
MQSO_RESUME	READ **(15)**	**(17)**	No check	
MQSO_ALTERNATE_USER_AUTHORITY				UPDATE
MQSO_SET_IDENTITY_CONTEXT			**(18)**	

Note:

1. This option is not restricted to queues. Use the MQNLIST or MXNLIST class for namelists, and the MQPROC or MXPROC class for processes.
2. Use RACF profile: hlq.resourcename
3. Use RACF profile: hlq.CONTEXT.queuename
4. Use RACF profile: hlq.ALTERNATE.USER.`alternateuserid`
   `alternateuserid` is the user identifier that is specified in the *`AlternateUserId`* field of the object descriptor. Note that up to 12 characters of the *`AlternateUserId`* field are used for this check, unlike other checks where only the first 8 characters of a user identifier are used.
5. No check is made when opening the queue manager for inquiries.
6. MQOO_INPUT_* must be specified as well. This is valid for a local, model or alias queue.
7. This check is done for a local or model queue that has a *`Usage`* queue attribute of MQUS_NORMAL, and also for an alias or remote queue (that is defined to the connected queue manager.) If the queue is a remote queue that is opened specifying an *`ObjectQMgrName`* (not the name of the connected queue manager) explicitly, the check is carried out against the queue with the same name as *`ObjectQMgrName`* (which must be a local queue with a *`Usage`* queue attribute of MQUS_TRANSMISSION).
8. MQOO_OUTPUT must be specified as well.
9. MQOO_PASS_IDENTITY_CONTEXT is implied as well by this option.
10. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT and MQOO_SET_IDENTITY_CONTEXT are implied as well by this option.
11. This check is done 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.
12. At least one of MQOO_INQUIRE, MQOO_BROWSE, MQOO_INPUT_*, MQOO_OUTPUT or MQOO_SET must be specified as well. The check carried out is the same as that for the other options specified.
13. The check carried out is the same as that for the other options specified.
14. This applies only for permanent dynamic queues that have been opened directly, that is, not opened through a model queue. No security is required to delete a temporary dynamic queue.
15. Use RACF profile hlq.SUBSCRIBE.topicname.
16. Use RACF profile hlq.PUBLISH.topicname.
17. If on the MQSUB request you specified a destination queue for the publications to be sent to, then a security check is carried out against that queue to ensure that you have put authority to that queue.
18. If on the MQSUB request, with MQSO_CREATE or MQSO_ALTER options specified, you want to set any of the identity context fields in the MQSD structure, you also need to specify the MQSO_SET_IDENTITY_CONTEXT option and you also need the appropriate authority to the context profile for the destination queue.

Parent topic: Profiles for queue security

14.5.3. Profiles for topic security

If topic security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

The concept of topic security within a topic tree is described in Publish/subscribe security.

If topic security is active, you must perform the following actions:

- Define profiles in the **MXTOPIC** or **GMXTOPIC** classes.
- Permit the necessary groups or user IDs access to these profiles, so that they can issue WebSphere® MQ API requests that use topics.

Profiles for topic security take the form:

```
hlq.SUBSCRIBE.topicname
hlq.PUBLISH.topicname
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `topicname` is the name of the topic administration node in the topic tree, associated either with the topic being subscribed to through an MQSUB call, or being published to through an MQOPEN call.

A profile prefixed by the queue manager name controls access to a single topic on that queue manager. A profile prefixed by the queue-sharing group name controls access to access to one or more topics with that topic name on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that topic on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

Subscribe

To subscribe to a topic, you need access to both the topic you are trying to subscribe to, and the target queue for the publications.

When you issue an MQSUB request, the following security checks take place:

- Whether you are allowed to subscribe to that topic, and also that the target queue (if specified) is opened for output
- Whether you have the appropriate level of access to that target queue.

Table 1. Access level required for topic security to subscribe

MQSUB to a topic	RACF® access required to *hlq.SUBSCRIBE.topicname* profile in MXTOPIC class
MQSO_CREATE and MQSO_ALTER	ALTER
MQSO_RESUME	READ
MQSUB - additional authority to non managed destination queues.	RACF access required to *hlq.CONTEXT.queuename* profile in MQADMIN or MXADMIN class
MQSO_CREATE, MQSO_ALTER, and MQSO_RESUME	UPDATE
	RACF access required to *hlq.queuename* profile in MQQUEUE or MXQUEUE class
MQSO_CREATE and MQSO_ALTER	UPDATE
	RACF access required to *hlq.ALTERNATE.USER.alternateuserid* profile in MQADMIN or MXADMIN class
MQSO_ALTERNATE_USER	UPDATE

Considerations for managed queues for subscriptions

A security check is carried out to see if you are allowed to subscribe to the topic. However, no security checks are carried out when the managed queue is created, or to determine if you are allowed to 'Put' to this destination queue.

You are not allowed to close delete a managed queue.

The model queues used are:`SYSTEM.DURABLE.MODEL.QUEUE` and `SYSTEM.NDURABLE.MODEL.QUEUE`.

The managed queues created from these model queues are of the form `SYSTEM.MANAGED.DURABLE.A346EF00367849A0` and `SYSTEM.MANAGED.NDURABLE.A346EF0036785EA0` where the last qualifier is unpredictable.

Do not give any user access to these queues. The queues can be protected using generic profiles of the form `SYSTEM.MANAGED.DURABLE.*` and `SYSTEM.MANAGED.NDURABLE.*` with no authorities granted.

Messages can be retrieved from these queues using the handle returned on the MQSUB request.

If you explicitly issue an MQCLOSE call for a subscription with the MQCO_REMOVE_SUB option specified, and you did not create the subscription you are closing under this handle, a security check is performed at the time of closure to ensure that you have the correct authority to perform the operation.

Table 2. Access level required to profiles for topic security for closure of a subscribe operation

MQCLOSE (of a subscription)	RACF access required to *hlq.SUBSCRIBE.topicname* profile in MXTOPIC class
MQCO_REMOVE_SUB	ALTER

Publish

To publish on a topic you need access to the topic and, if you are using alias queues, to the alias queue as well.

Table 3. Access level required to profiles for topic security for a publish operation

MQOPEN (of a topic)	RACF access required to *hlq.PUBLISH.topicname* profile in MXTOPIC class
MQOO_OUTPUT or MQPUT1	UPDATE
MQOPEN (Alias queue to topic)	RACF access required to *hlq.queuename* profile in MQQUEUE or MXQUEUE class for the alias queue
MQOO_OUTPUT or MQPUT1	UPDATE

For details of how topic security operates when an alias queue that resolves to a topic name is opened for publish, see Considerations for alias queues that resolve to topics for a publish operation.

When you consider alias queues used for destination queues for PUT or GET restrictions, see Considerations for alias queues.

If the RACF access level that an application has to a topic security profile is changed, the changes take effect only for any new object handles obtained, (that is, a new MQSUB or MQOPEN) for that topic. Those handles already in existence at the time of the change retain their existing access to the topic. Also, existing subscribers retain their access to any subscriptions that they have already made.

Considerations for alias queues that resolve to topics for a publish operation

When you issue an MQOPEN or MQPUT1 call for an alias queue that resolves to a topic, WebSphere MQ makes two resource checks:

- The first one against the alias queue name specified in the object descriptor (MQOD) on the MQOPEN or MQPUT1 call.
- The second against the topic to which the alias queue resolves

You need to be aware that this is different from the behavior you get when alias queues resolve to other queues. You need the correct access to both profiles in order for the publish action to proceed.

System topic security

Many of the system topics are accessed by the ancillary parts of WebSphere MQ, for example the channel initiator address space.

The user IDs under which this runs must be given RACF access to these queues, as shown in Table 4.

Table 4. Access required to the SYSTEM topics

SYSTEM topic	Profile	Channel Initiator for Distributed queueing
SYSTEM.BROKER.ADMIN.STREAM	PUBLISH.topicname	UPDATE
SYSTEM.BROKER.ADMIN.STREAM	SUBSCRIBE.topicname	ALTER

Parent topic: Profiles used to control access to WebSphere MQ resources

This build: January 26, 2011 11:22:35

Notices | Trademarks | Downloads | Library | Support | Feedback

14.5.4. Profiles for processes

If process security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

If process security is active, you must:

- Define profiles in the **MQPROC** or **GMQPROC** classes if using uppercase profiles.
- Define profiles in the **MXPROC** or **GMXPROC** classes if using mixed case profiles.
- Permit the necessary groups or user IDs access to these profiles, so that they can issue WebSphere® MQ API requests that use processes.

Profiles for processes take the form:

```
hlq.processname
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `processname` is the name of the process being opened.

A profile prefixed by the queue manager name controls access to a single process definition on that queue manager. A profile prefixed by the queue-sharing group name controls access to one or more process definitions with that name on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that process definition on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access required for opening a process.

Table 1. Access levels for process security

MQOPEN option	RACF® access level required to hlq.processname
MQOO_INQUIRE	READ

For example, on queue manager MQS9, the RACF group INQVPRC must be able to inquire (**MQINQ**) on all processes starting with the letter V. The RACF definitions for this would be:

```
RDEFINE MQPROC MQS9.V* UACC(NONE)
PERMIT MQS9.V* CLASS(MQPROC) ID(INQVPRC) ACCESS(READ)
```

Alternate user security might also be active, depending on the open options specified when a process definition object is opened.

Parent topic: Profiles used to control access to WebSphere MQ resources

This build: January 26, 2011 11:22:10

Notices | Trademarks | Downloads | Library | Support | Feedback

14.5.5. Profiles for namelists

If namelist security is active, you define profiles in the appropriate classes and give the necessary groups or user IDs access to these profiles.

If namelist security is active, you must:

- Define profiles in the **MQNLIST** or **GMQNLIST** classes if using uppercase profiles.
- Define profiles in the **MXNLIST** or **GMXNLIST** classes if using mixed case profiles.

- Permit the necessary groups or user IDs access to these profiles.

Profiles for namelists take the form:

```
hlq.namelistname
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `namelistname` is the name of the namelist being opened.

A profile prefixed by the queue manager name controls access to a single namelist on that queue manager. A profile prefixed by the queue-sharing group name controls access to access to one or more namelists with that name on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that namelist on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access required for opening a namelist.

Table 1. Access levels for namelist security

MQOPEN option	RACF® access level required to hlq.namelistname
MQOO_INQUIRE	READ

For example, on queue manager (or queue-sharing group) PQM3, the RACF group DEPT571 must be able to inquire (**MQINQ**) on these namelists:

- All namelists starting with "DEPT571".
- PRINTER/DESTINATIONS/DEPT571
- AGENCY/REQUEST/QUEUES
- WAREHOUSE.BROADCAST

The RACF definitions to do this are:

```
RDEFINE MQNLIST PQM3.DEPT571.** UACC(NONE)
PERMIT PQM3.DEPT571.** CLASS(MQNLIST) ID(DEPT571) ACCESS(READ)

RDEFINE GMQNLIST NLISTS.FOR.DEPT571 UACC(NONE)
        ADDMEM(PQM3.PRINTER/DESTINATIONS/DEPT571,
               PQM3.AGENCY/REQUEST/QUEUES,
               PQM3.WAREHOUSE.BROADCAST)
PERMIT NLISTS.FOR.DEPT571 CLASS(GMQNLIST) ID(DEPT571) ACCESS(READ)
```

Alternate user security might be active, depending on the options specified when a namelist object is opened.

System namelist security

Many of the system namelists are accessed by the ancillary parts of WebSphere MQ:

- The CSQUTIL utility
- The operations and control panels
- The channel initiator address space (including the Queued Publish/Subscribe Daemon)

The user IDs under which these run must be given RACF access to these namelists, as shown in Table 2.

Table 2. Access required to the SYSTEM namelists by WebSphere MQ

SYSTEM namelist	CSQUTIL	Operations and control panels	Channel initiator for distributed queuing
SYSTEM.QPUBSUB.QUEUE.NAMELIST	–	–	READ
SYSTEM.QPUBSUB.SUBPOINT.NAMELIST	–	–	READ

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.6. Profiles for alternate user security

If alternate user security is active, you must define profiles in the appropriate classes and permit the necessary groups or user IDs access to those profiles.

If alternate user security is active, you must:

- Define profiles in the MQADMIN or GMQADMIN classes if you are using uppercase profiles.
- Define profiles in the MXADMIN or GMXADMIN classes if you are using mixed case profiles.

Permit the necessary groups or user IDs access to these profiles, so that they can use the ALTERNATE_USER_AUTHORITY options when the object is opened.

Profiles for alternate user security can be specified at subsystem level or at queue-sharing group level and take the following form:

```
hlq.ALTERNATE.USER.alternateuserid
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), and `alternateuserid` is the value of the *`AlternateUserId`* field in the object descriptor.

A profile prefixed by the queue manager name controls use of an alternate user ID on that queue manager. A profile prefixed by the queue-sharing group name controls use of an alternate user ID on all queue managers within the queue-sharing group. This alternate user ID can be used on any queue manager within the queue-sharing group by a user that has the correct access. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that alternate user ID on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

The following table shows the access when specifying an alternate user option.

Table 1. Access levels for alternate user security

MQOPEN, MQSUB, or MQPUT1 option	RACF® access level required
MQOO_ALTERNATE_USER_AUTHORITY MQSO_ALTERNATE_USER_AUTHORITY MQPMO_ALTERNATE_USER_AUTHORITY	UPDATE

In addition to alternate user security checks, other security checks for queue, process, namelist, and context security can also be made. The alternate user ID, if provided, is only used for security checks on queue, process definition, or namelist resources. For alternate user and context security checks, the user ID requesting the check is used. For details about how user IDs are handled, see User IDs for security checking. For a summary table showing the open options and the security checks required when queue, context and alternate user security are all active, see Table 1.

An alternate user profile gives the requesting user ID access to resources associated with the user ID specified in the alternate user ID. For example, the payroll server running under user ID PAYSERV on queue manager QMPY processes requests from personnel user IDs, all of which start with PS. To cause the work performed by the payroll server to be carried out under the user ID of the requesting user, alternate user authority is used. The payroll server knows which user ID to specify as the alternate user ID because the requesting programs generate messages using the MQPMO_DEFAULT_CONTEXT put message option. See User IDs for security checking for more details about from where alternate user IDs are obtained.

The following example RACF definitions enable the server program to specify alternate user IDs starting with the characters PS:

```
RDEFINE MQADMIN QMPY.ALTERNATE.USER.PS* UACC(NONE)
PERMIT QMPY.ALTERNATE.USER.PS* CLASS(MQADMIN) ID(PAYSERV) ACCESS(UPDATE)
```

Note:

1. The *AlternateUserId* fields in the object descriptor and subscription descriptor are 12 bytes long. All 12 bytes are used in the profile checks, but only the first 8 bytes are used as the user ID by WebSphere MQ. If this user ID truncation is not desirable, application programs making the request must translate any alternate user ID over 8 bytes into something more appropriate.
2. If you specify MQOO_ALTERNATE_USER_AUTHORITY, MQSO_ALTERNATE_USER_AUTHORITY, or MQPMO_ALTERNATE_USER_AUTHORITY and you do not specify an *AlternateUserId* field in the object descriptor, a user ID of blanks is used. For the purposes of the alternate user security check the user ID used for the *AlternateUserId* qualifier is -BLANK-. For example `RDEF MQADMIN hlq.ALTERNATE.USER.-BLANK-`.
   If the user is allowed to access this profile, all further checks are made with a user ID of blanks. For details of blank user IDs, see Blank user IDs and UACC levels.

The administration of alternate user IDs is easier if you have a naming convention for user IDs that enables you to use generic alternate user profiles. If they do not, you could use the RACF RACVARS feature. For details about using RACVARS, see the *z/OS® SecureWay™ Security Server RACF Security Administrator's Guide*.

When a message is put to a queue that has been opened with alternate user authority and the context of the message has been generated by the queue manager, the MQMD_USER_IDENTIFIER field is set to the alternate user ID.

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.7. Profiles for context security

WebSphere MQ uses profiles for controlling access to the context information specific to a particular message. The context is contained within the message descriptor (MQMD).

Using profiles for context security

If context security is active, you must:

- Define a profile in the **MQADMIN** class if using uppercase profiles.
- Define profile in the **MXADMIN** class if using mixed case profiles.

The profile is called `hlq.CONTEXT.queuename`, where:

hlq
Can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name).

queuename
Can be either the full name of the queue you want to define the context profile for, or a generic profile.

Special considerations apply if you are migrating from a previous version; see Migrating from a previous version.

A profile prefixed by the queue manager name, and with `**` specified as the queue name, allows control for context security on all queues belonging to that queue manager. This can be overridden on an individual queue by defining a queue level profile for context on that queue.

A profile prefixed by the queue-sharing group name, and with `**` specified as the queue name, allows control for context on all queues belonging to the queue managers within the queue-sharing group. This can be overridden on an individual queue manager by defining a queue-manager level profile for context on that queue manager, by specifying a profile prefixed by the queue manager name. It can also be overridden on an individual queue by specifying a profile suffixed with the queue name.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

You must give the necessary groups or user IDs access to this profile. The following table shows the access level required, depending on the specification of the context options when the queue is opened.

Table 1. Access levels for context security

MQOPEN or MQPUT1 option	RACF® access level required to hlq.CONTEXT.queuename
MQPMO_NO_CONTEXT	No context security check
MQPMO_DEFAULT_CONTEXT	No context security check
MQOO_SAVE_ALL_CONTEXT	No context security check
MQOO_PASS_IDENTITY_CONTEXT MQPMO_PASS_IDENTITY_CONTEXT	READ
MQOO_PASS_ALL_CONTEXT MQPMO_PASS_ALL_CONTEXT	READ
MQOO_SET_IDENTITY_CONTEXT MQPMO_SET_IDENTITY_CONTEXT	UPDATE

MQOO_SET_ALL_CONTEXT MQPMO_SET_ALL_CONTEXT	CONTROL
MQOO_OUTPUT or MQPUT1 (USAGE(XMITQ))	CONTROL
MQSUB option	
MQSO_SET_IDENTITY_CONTEXT(**Note 2**)	UPDATE

Note:

1. The user IDs used for distributed queuing require CONTROL access to `hlq.CONTEXT.queuename` to put messages on the destination queue. See User IDs used by the channel initiator for information about the user IDs used.
2. If on the MQSUB request, with MQSO_CREATE or MQSO_ALTER options specified, you want to set any of the identity context fields in the MQSD structure, you need to specify the MQSO_SET_IDENTITY_CONTEXT option. You require also, the appropriate authority to the context profile for the destination queue.

If you put commands on the system-command input queue, use the default context put message option to associate the correct user ID with the command.

For example, the WebSphere MQ-supplied utility program CSQUTIL can be used to off-load and reload messages in queues. When off-loaded messages are restored to a queue, the CSQUTIL utility uses the MQOO_SET_ALL_CONTEXT option to return the messages to their original state. In addition to the queue security required by this open option, context authority is also required. For example, if this authority is required by the group BACKGRP on queue manager MQS1, this would be defined by:

```
RDEFINE MQADMIN MQS1.CONTEXT.** UACC(NONE)
PERMIT MQS1.CONTEXT.** CLASS(MQADMIN) ID(BACKGRP) ACCESS(CONTROL)
```

Depending on the options specified, and the types of security performed, other types of security checks might also occur when the queue is opened. These include queue security (see Profiles for queue security), and alternate user security (see Profiles for alternate user security). For a summary table showing the open options and the security checks required when queue, context and alternate user security are all active, see Table 1.

System queue context security

Many of the system queues are accessed by the ancillary parts of WebSphere MQ, for example the channel initiator address space.

The user IDs under which this runs must be given RACF access to these queues, as shown in Table 2.

Table 2. Access required to the SYSTEM queues for context operations

SYSTEM queue	Channel Initiator for Distributed queueing
SYSTEM.BROKER.CONTROL.QUEUE	CONTROL
SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS	CONTROL
SYSTEM.CHANNEL.SYNCQ	CONTROL
SYSTEM.CLUSTER.TRANSMIT.QUEUE	CONTROL

Parent topic: Profiles used to control access to WebSphere MQ resources

This build: January 26, 2011 11:22:11

Notices | Trademarks | Downloads | Library | Support | Feedback

14.5.8. Profiles for command security

To enable security checking for commands, add profiles to the MQCMDS class. The profile names are based on the MQSC commands but control both MQSC and PCF commands. Profiles can apply to a queue manager or a queue-sharing group.

If you want security checking for commands (so you have not defined the command security switch profile hlq.NO.CMD.CHECKS) you must add profiles to the MQCMDS class.

The same security profiles control both MQSC and PCF commands. The names of the RACF® profiles for command security checking are based on the MQSC command names themselves. These profiles take the form:

```
hlq.verb.pkw
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name), `verb` is the verb part of the command name, for example ALTER, and `pkw` is the object type, for example QLOCAL for a local queue.

Thus, the profile name for the ALTER QLOCAL command in subsystem CSQ1 is:

```
CSQ1.ALTER.QLOCAL
```

You can use generic profiles to protect sets of commands so that you need to maintain fewer profiles and, therefore, fewer access lists. Consider creating a generic profile that applies to all commands not protected by a more specific profile. Define this profile with UACC(NONE) and grant ALTER access only to the RACF groups containing administrators. You might then create a generic profile applicable to all DISPLAY commands and grant widespread access to it. Between these extremes, you might identify groups of users needing access to certain sets of commands, in which case you could create profiles for those sets and grant access to RACF groups representing those classes of user. Avoid giving users access to commands they do not require: apply the principle of least privilege, so that users only have access to the commands that are required for their jobs.

A profile prefixed by the queue manager name controls the use of the command on that queue manager. A profile prefixed by the queue-sharing group name controls the use of the command on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that command on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

By setting up command profiles at queue manager level, a user can be restricted from issuing commands on a particular queue manager. Alternatively, you can define one profile for a queue-sharing group for each command verb, and all security checks take place against that profile instead of individual queue managers.

If both subsystem security and queue-sharing group security are active and a local profile is not found, a command security check is performed to see if the user has access to a queue-sharing group profile.

If you use the CMDSCOPE attribute to route a command to other queue managers in a queue-sharing group, security is checked on each queue manager where the command is executed, but not necessarily on the queue manager where the command is entered.

Table 1 shows, for each WebSphere MQ MQSC command, the profiles required for command security checking to be carried out, and the corresponding access level for each profile in the MQCMDS class.

Table 2 shows, for each WebSphere MQ PCF command, the profiles required for command security checking to be carried out, and the corresponding access level for each profile in the MQCMDS class.

Table 1. MQSC commands, profiles, and their access levels

Command	Command profile for MQCMDS	Access level for MQCMDS	Command resource profile for MQADMIN or MXADMIN	Access level for MQADMIN or MXADMIN
ALTER AUTHINFO	hlq.ALTER.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
ALTER BUFFPOOL	hlq.ALTER.BUFFPOOL	ALTER	No check	–
ALTER CFSTRUCT	hlq.ALTER.CFSTRUCT	ALTER	No check	–
ALTER CHANNEL	hlq.ALTER.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
ALTER NAMELIST	hlq.ALTER.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
ALTER PROCESS	hlq.ALTER.PROCESS	ALTER	hlq.PROCESS.process	ALTER
ALTER PSID	hlq.ALTER.PSID	ALTER	No check	–
ALTER QALIAS	hlq.ALTER.QALIAS	ALTER	hlq.QUEUE.queue	ALTER
ALTER QLOCAL	hlq.ALTER.QLOCAL	ALTER	hlq.QUEUE.queue	ALTER
ALTER QMGR	hlq.ALTER.QMGR	ALTER	No check	–
ALTER QMODEL	hlq.ALTER.QMODEL	ALTER	hlq.QUEUE.queue	ALTER
ALTER QREMOTE	hlq.ALTER.QREMOTE	ALTER	hlq.QUEUE.queue	ALTER
ALTER SECURITY	hlq.ALTER.SECURITY	ALTER	No check	–
ALTER STGCLASS	hlq.ALTER.STGCLASS	ALTER	No check	–
ALTER SUB	hlq.ALTER.SUB	ALTER	No check	–
ALTER TOPIC	hlq.ALTER.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
ALTER TRACE	hlq.ALTER.TRACE	ALTER	No check	–
ARCHIVE LOG	hlq.ARCHIVE.LOG	CONTROL	No check	–
BACKUP CFSTRUCT	hlq.BACKUP.CFSTRUCT	CONTROL	No check	–
CLEAR QLOCAL	hlq.CLEAR.QLOCAL	ALTER	hlq.QUEUE.queue	ALTER
›CLEAR TOPICSTR 3‹	›hlq.CLEAR.TOPICSTR‹	›ALTER‹	›hlq.TOPIC.topic‹	›ALTER‹
DEFINE AUTHINFO	hlq.DEFINE.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
DEFINE BUFFPOOL	hlq.DEFINE.BUFFPOOL	ALTER	No check	–
DEFINE CFSTRUCT	hlq.DEFINE.CFSTRUCT	ALTER	No check	–
DEFINE CHANNEL	hlq.DEFINE.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
DEFINE LOG	hlq.DEFINE.LOG	ALTER	No check	–
DEFINE MAXSMSGS	hlq.DEFINE.MAXSMSGS	ALTER	No check	–
DEFINE NAMELIST	hlq.DEFINE.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
DEFINE PROCESS	hlq.DEFINE.PROCESS	ALTER	hlq.PROCESS.process	ALTER
DEFINE PSID	hlq.DEFINE.PSID	ALTER	No check	–
DEFINE QALIAS	hlq.DEFINE.QALIAS	ALTER	hlq.QUEUE.queue	ALTER
DEFINE QLOCAL	hlq.DEFINE.QLOCAL	ALTER	hlq.QUEUE.queue	ALTER
DEFINE QMODEL	hlq.DEFINE.QMODEL	ALTER	hlq.QUEUE.queue	ALTER
DEFINE QREMOTE	hlq.DEFINE.QREMOTE	ALTER	hlq.QUEUE.queue	ALTER
DEFINE STGCLASS	hlq.DEFINE.STGCLASS	ALTER	No check	–
DEFINE SUB	hlq.DEFINE.SUB	ALTER	No check	–
DEFINE TOPIC	hlq.DEFINE.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
DELETE AUTHINFO	hlq.DELETE.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
DELETE BUFFPOOL	hlq.DELETE.BUFFPOOL	ALTER	No check	–
DELETE CFSTRUCT	hlq.DELETE.CFSTRUCT	ALTER	No check	–
DELETE CHANNEL	hlq.DELETE.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
DELETE NAMELIST	hlq.DELETE.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
DELETE PROCESS	hlq.DELETE.PROCESS	ALTER	hlq.PROCESS.process	ALTER
DELETE PSID	hlq.DELETE.PSID	ALTER	No check	–
DELETE QALIAS	hlq.DELETE.QALIAS	ALTER	hlq.QUEUE.queue	ALTER
DELETE QLOCAL	hlq.DELETE.QLOCAL	ALTER	hlq.QUEUE.queue	ALTER
DELETE QMODEL	hlq.DELETE.QMODEL	ALTER	hlq.QUEUE.queue	ALTER
DELETE QREMOTE	hlq.DELETE.QREMOTE	ALTER	hlq.QUEUE.queue	ALTER
DELETE STGCLASS	hlq.DELETE.STGCLASS	ALTER	No check	–
DELETE SUB	hlq.DELETE.SUB	ALTER	No check	–
DELETE TOPIC	hlq.DELETE.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
DISPLAY ARCHIVE 1	hlq.DISPLAY.ARCHIVE	READ	No check	–
DISPLAY AUTHINFO	hlq.DISPLAY.AUTHINFO	READ	No check	–
DISPLAY CFSTATUS	hlq.DISPLAY.CFSTATUS	READ	No check	–
DISPLAY CFSTRUCT	hlq.DISPLAY.CFSTRUCT	READ	No check	–
DISPLAY CHANNEL	hlq.DISPLAY.CHANNEL	READ	No check	–
DISPLAY CHINIT	hlq.DISPLAY.CHINIT	READ	No check	–
DISPLAY CHSTATUS	hlq.DISPLAY.CHSTATUS	READ	No check	–
DISPLAY CLUSQMGR	hlq.DISPLAY.CLUSQMGR	READ	No check	–
DISPLAY CMDSERV	hlq.DISPLAY.CMDSERV	READ	No check	–
DISPLAY CONN 1	hlq.DISPLAY.CONN	READ	No check	–
DISPLAY GROUP	hlq.DISPLAY.GROUP	READ	No check	–
DISPLAY LOG 1	hlq.DISPLAY.LOG	READ	No check	–

DISPLAY MAXSMSGS	hlq.DISPLAY.MAXSMSGS	READ	No check	–
DISPLAY NAMELIST	hlq.DISPLAY.NAMELIST	READ	No check	–
DISPLAY PROCESS	hlq.DISPLAY.PROCESS	READ	No check	–
DISPLAY PUBSUB	hlq.DISPLAY.PUBSUB	READ	No check	-
DISPLAY QALIAS	hlq.DISPLAY.QALIAS	READ	No check	–
DISPLAY QCLUSTER	hlq.DISPLAY.QCLUSTER	READ	No check	–
DISPLAY QLOCAL	hlq.DISPLAY.QLOCAL	READ	No check	–
DISPLAY QMGR	hlq.DISPLAY.QMGR	READ	No check	–
DISPLAY QMODEL	hlq.DISPLAY.QMODEL	READ	No check	–
DISPLAY QREMOTE	hlq.DISPLAY.QREMOTE	READ	No check	–
DISPLAY QSTATUS	hlq.DISPLAY.QSTATUS	READ	No check	–
DISPLAY QUEUE	hlq.DISPLAY.QUEUE	READ	No check	–
DISPLAY SBSTATUS	hlq.DISPLAY.SBSTATUS	READ	No check	–
DISPLAY SUB	hlq.DISPLAY.SUB	READ	No check	–
DISPLAY SECURITY	hlq.DISPLAY.SECURITY	READ	No check	–
DISPLAY STGCLASS	hlq.DISPLAY.STGCLASS	READ	No check	–
DISPLAY SYSTEM 1	hlq.DISPLAY.SYSTEM	READ	No check	–
DISPLAY THREAD	hlq.DISPLAY.THREAD	READ	No check	–
DISPLAY TPSTATUS	hlq.DISPLAY.TPSTATUS	READ	No check	–
DISPLAY TOPIC	hlq.DISPLAY.TOPIC	READ	No check	–
DISPLAY TPSTATUS	hlq.DISPLAY.TPSTATUS	READ	No check	–
DISPLAY TRACE	hlq.DISPLAY.TRACE	READ	No check	–
DISPLAY USAGE 1	hlq.DISPLAY.USAGE	READ	No check	–
MOVE QLOCAL	hlq.MOVE.QLOCAL	ALTER	hlq.QUEUE.from-queue hlq.QUEUE.to-queue	ALTER
PING CHANNEL	hlq.PING.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
RECOVER BSDS	hlq.RECOVER.BSDS	CONTROL	No check	–
RECOVER CFSTRUCT	hlq.RECOVER.CFSTRUCT	CONTROL	No check	–
REFRESH CLUSTER	hlq.REFRESH.CLUSTER	ALTER	No check	–
REFRESH QMGR	hlq.REFRESH.QMGR	ALTER	No check	–
REFRESH SECURITY	hlq.REFRESH.SECURITY	ALTER	No check	–
RESET CHANNEL	hlq.RESET.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
RESET CLUSTER	hlq.RESET.CLUSTER	CONTROL	No check	–
RESET QMGR	hlq.RESET.QMGR	CONTROL	No check	-
RESET QSTATS	hlq.RESET.QSTATS	CONTROL	hlq.QUEUE.queue	CONTROL
RESET TPIPE	hlq.RESET.TPIPE	CONTROL	No check	–
RESOLVE CHANNEL	hlq.RESOLVE.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
RESOLVE INDOUBT	hlq.RESOLVE.INDOUBT	CONTROL	No check	–
RESUME QMGR	hlq.RESUME.QMGR	CONTROL	No check	–
RVERIFY SECURITY	hlq.RVERIFY.SECURITY	ALTER	No check	–
SET ARCHIVE	hlq.SET.ARCHIVE	CONTROL	No check	–
SET LOG	hlq.SET.LOG	CONTROL	No check	–
SET SYSTEM	hlq.SET.SYSTEM	CONTROL	No check	–
START CHANNEL	hlq.START.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
START CHINIT	hlq.START.CHINIT	CONTROL	No check	–
START CMDSERV	hlq.START.CMDSERV	CONTROL	No check	–
START LISTENER	hlq.START.LISTENER	CONTROL	No check	–
START QMGR	None 2	–	–	–
START TRACE	hlq.START.TRACE	CONTROL	No check	–
STOP CHANNEL	hlq.STOP.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
STOP CHINIT	hlq.STOP.CHINIT	CONTROL	No check	–
STOP CMDSERV	hlq.STOP.CMDSERV	CONTROL	No check	–
STOP LISTENER	hlq.STOP.LISTENER	CONTROL	No check	–
STOP QMGR	hlq.STOP.QMGR	CONTROL	No check	–
STOP TRACE	hlq.STOP.TRACE	CONTROL	No check	–
SUSPEND QMGR	hlq.SUSPEND.QMGR	CONTROL	No check	–

Notes:

1. These commands might be issued internally by the queue manager; no authority is checked in these cases.
2. WebSphere MQ does not check the authority of the user who issues the START QMGR command. However, you can use RACF facilities to control access to the START xxxxMSTR command that is issued as a result of the START QMGR command. This is done by controlling access to the MVS™.START.STC.xxxxMSTR profile in the RACF operator commands (OPERCMDS) class. For details of this procedure, see the *z/OS® Secureway Security Server RACF Security Administrator's Guide*. If you use this technique, and an unauthorized user tries to start the queue manager, it terminates with a reason code of 00F30216.
3. The **hlq.TOPIC.topic** resource refers to the Topic object derived from the TOPICSTR. For more details, see Publish/Subscribe security.

Table 2. PCF commands, profiles, and their access levels

Command	**Command profile for MQCMDS**	**Access level for MQCMDS**	**Command resource profile for MQADMIN or MXADMIN**	**Access level for MQADMIN or MXADMIN**
Backup CF Structure	hlq.BACKUP.CFSTRUCT	CONTROL	No check	–
Change Authentication Information Object	hlq.ALTER.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
Change CF Structure	hlq.ALTER.CFSTRUCT	ALTER	No check	–

Change Channel	hlq.ALTER.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
Change Namelist	hlq.ALTER.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
Change Process	hlq.ALTER.PROCESS	ALTER	hlq.PROCESS.process	ALTER
Change Queue	hlq.ALTER.QUEUE	ALTER	hlq.QUEUE.queue	ALTER
Change Queue Manager	hlq.ALTER.QMGR	ALTER	No check	–
Change Security	hlq.ALTER.SECURITY	ALTER	No check	–
Change Storage Class	hlq.ALTER.STGCLASS	ALTER	No check	–
Change Subscription	hlq.ALTER.SUB	ALTER	No check	–
Change Topic	hlq.ALTER.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
Clear Queue	hlq.CLEAR.QLOCAL	ALTER	hlq.QUEUE.queue	ALTER
Clear Topic String [1]	hlq.CLEAR.TOPICSTR	ALTER	hlq.TOPIC.topic	ALTER
Copy Authentication Information Object	hlq.DEFINE.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
Copy CF Structure	hlq.DEFINE.CFSTRUCT	ALTER	No check	–
Copy Channel	hlq.DEFINE.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
Copy Namelist	hlq.DEFINE.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
Copy Process	hlq.DEFINE.PROCESS	ALTER	hlq.PROCESS.process	ALTER
Copy Queue	hlq.DEFINE.QUEUE	ALTER	hlq.QUEUE.queue	ALTER
Copy Subscription	hlq.DEFINE.SUB	ALTER	No check	–
Copy Storage Class	hlq.DEFINE.STGCLASS	ALTER	No check	–
Copy Topic	hlq.DEFINE.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
Create Authentication Information Object	hlq.DEFINE.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
Create CF Structure	hlq.DEFINE.CFSTRUCT	ALTER	No check	–
Create Channel	hlq.DEFINE.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
Create Namelist	hlq.DEFINE.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
Create Process	hlq.DEFINE.PROCESS	ALTER	hlq.PROCESS.process	ALTER
Create Queue	hlq.DEFINE.QUEUE	ALTER	hlq.QUEUE.queue	ALTER
Create Storage Class	hlq.DEFINE.STGCLASS	ALTER	No check	–
Create Subscription	hlq.DEFINE.SUB	ALTER	No check	–
Create Topic	hlq.DEFINE.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
Delete Authentication Information Object	hlq.DELETE.AUTHINFO	ALTER	hlq.AUTHINFO.resourcename	ALTER
Delete CF Structure	hlq.DELETE.CFSTRUCT	ALTER	No check	–
Delete Channel	hlq.DELETE.CHANNEL	ALTER	hlq.CHANNEL.channel	ALTER
Delete Namelist	hlq.DELETE.NAMELIST	ALTER	hlq.NAMELIST.namelist	ALTER
Delete Process	hlq.DELETE.PROCESS	ALTER	hlq.PROCESS.process	ALTER
Delete Queue	hlq.DELETE.QUEUE	ALTER	hlq.QUEUE.queue	ALTER
Delete Storage Class	hlq.DELETE.STGCLASS	ALTER	No check	–
Delete Subscription	hlq.DELETE.SUB	ALTER	No check	–
Delete Topic	hlq.DELETE.TOPIC	ALTER	hlq.TOPIC.topic	ALTER
Inquire Archive	hlq.DISPLAY.ARCHIVE	READ	No check	–
Inquire Authentication Information Object	hlq.DISPLAY.AUTHINFO	READ	No check	–
Inquire Authentication Information Object Names	hlq.DISPLAY.AUTHINFO	READ	No check	–
Inquire CF Structure	hlq.DISPLAY.CFSTRUCT	READ	No check	–
Inquire CF Structure Names	hlq.DISPLAY.CFSTRUCT	READ	No check	–
Inquire CF Structure Status	hlq.DISPLAY.CFSTATUS	READ	No check	–
Inquire Channel	hlq.DISPLAY.CHANNEL	READ	No check	–
Inquire Channel Initiator	hlq.DISPLAY.CHINIT	READ	No check	–
Inquire Channel Names	hlq.DISPLAY.CHANNEL	READ	No check	–
Inquire Channel Status	hlq.DISPLAY.CHSTATUS	READ	No check	–
Inquire Cluster Queue Manager	hlq.DISPLAY.CLUSQMGR	READ	No check	–
Inquire Connection	hlq.DISPLAY.CONN	READ	No check	–
Inquire Group	hlq.DISPLAY.GROUP	READ	No check	–
Inquire Log	hlq.DISPLAY.LOG	READ	No check	–
Inquire Namelist	hlq.DISPLAY.NAMELIST	READ	No check	–
Inquire Namelist Names	hlq.DISPLAY.NAMELIST	READ	No check	–
Inquire Process	hlq.DISPLAY.PROCESS	READ	No check	–
Inquire Process Names	hlq.DISPLAY.PROCESS	READ	No check	–
Inquire Pub/Sub Status	hlq.DISPLAY.PUBSUB	READ	No check	-
Inquire Queue	hlq.DISPLAY.QUEUE	READ	No check	–
Inquire Queue Manager	hlq.DISPLAY.QMGR	READ	No check	–
Inquire Queue Names	hlq.DISPLAY.QUEUE	READ	No check	–
Inquire Queue Status	hlq.DISPLAY.QSTATUS	READ	No check	–
Inquire Security	hlq.DISPLAY.SECURITY	READ	No check	–
Inquire Storage Class	hlq.DISPLAY.STGCLASS	READ	No check	–
Inquire Storage Class Names	hlq.DISPLAY.STGCLASS	READ	No check	–
Inquire Subscription	hlq.INQUIRE.SUB	READ	No check	–
Inquire Subscription Status	hlq.INQUIRE.SBSTATUS	READ	No check	–
Inquire System	hlq.DISPLAY.SYSTEM	READ	No check	–
Inquire Topic	hlq.DISPLAY.TOPIC	READ	No check	–

Inquire Topic Names	hlq.DISPLAY.TOPIC	READ	No check	–
Inquire Topic Status	hlq.DISPLAY.TPSTATUS	READ	No check	–
Inquire Usage	hlq.DISPLAY.USAGE	READ	No check	–
Move Queue	hlq.MOVE.QLOCAL	ALTER	hlq.QUEUE.from-queue hlq.QUEUE.to-queue	ALTER
Ping Channel	hlq.PING.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
Recover CF Structure	hlq.RECOVER.CFSTRUCT	CONTROL	No check	–
Refresh Cluster	hlq.REFRESH.CLUSTER	ALTER	No check	–
Refresh Queue Manager	hlq.REFRESH.QMGR	ALTER	No check	–
Refresh Security	hlq.REFRESH.SECURITY	ALTER	No check	–
Reset Channel	hlq.RESET.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
Reset Cluster	hlq.RESET.CLUSTER	CONTROL	No check	–
Reset Queue Manager	hlq.RESET.QMGR	CONTROL	No check	-
Reset Queue Statistics	hlq.RESET.QSTATS	CONTROL	hlq.QUEUE.queue	CONTROL
Resolve Channel	hlq.RESOLVE.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
Resume Queue Manager	hlq.RESUME.QMGR	CONTROL	No check	–
Resume Queue Manager Cluster	hlq.RESUME.QMGR	CONTROL	No check	–
Reverify Security	hlq.RVERIFY.SECURITY	ALTER	No check	–
Set Archive	hlq.SET.ARCHIVE	CONTROL	No check	–
Set Log	hlq.SET.LOG	CONTROL	No check	–
Set System	hlq.SET.SYSTEM	CONTROL	No check	–
Start Channel	hlq.START.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
Start Channel Initiator	hlq.START.CHINIT	CONTROL	No check	–
Start Channel Listener	hlq.START.LISTENER	CONTROL	No check	–
Stop Channel	hlq.STOP.CHANNEL	CONTROL	hlq.CHANNEL.channel	CONTROL
Stop Channel Initiator	hlq.STOP.CHINIT	CONTROL	No check	–
Stop Channel Listener	hlq.STOP.LISTENER	CONTROL	No check	–
Suspend Queue Manager	hlq.SUSPEND.QMGR	CONTROL	No check	–
Suspend Queue Manager Cluster	hlq.SUSPEND.QMGR	CONTROL	No check	–

Notes:
1. The **hlq.TOPIC.topic** resource refers to the Topic object derived from the TOPICSTR. For more details, see Publish/Subscribe security.

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.9. Profiles for command resource security

If you have not defined the command resource security switch profile, because you want security checking for resources associated with commands, you must add resource profiles for each resource to the appropriate class. The same security profiles control both MQSC and PCF commands.

If you have not defined the command resource security switch profile, `hlq.NO.CMD.RESC.CHECKS`, because you want security checking for resources associated with commands, you must:

- Add a resource profile in the **MQADMIN** class, if using uppercase profiles, for each resource.
- Add a resource profile in the **MXADMIN** class, if using mixed case profiles, for each resource.

The same security profiles control both MQSC and PCF commands.

Profiles for command resource security checking take the form:

```
hlq.type.resourcename
```

where `hlq` can be either `qmgr-name` (queue manager name) or `qsg-name` (queue-sharing group name).

A profile prefixed by the queue manager name controls access to the resources associated with commands on that queue manager. A profile prefixed by the queue-sharing group name controls access to the resources associated with commands on all queue managers within the queue-sharing group. This access can be overridden on an individual queue manager by defining a queue-manager level profile for that command resource on that queue manager.

If your queue manager is a member of a queue-sharing group and you are using both queue manager and queue-sharing group level security, WebSphere® MQ checks for a profile prefixed by the queue manager name first. If it does not find one, it looks for a profile prefixed by the queue-sharing group name.

For example, the RACF® profile name for command resource security checking against the model queue CREDIT.WORTHY in subsystem CSQ1 is:

```
CSQ1.QUEUE.CREDIT.WORTHY
```

Because the profiles for all types of command resource are held in the MQADMIN class, the “type” part of the profile name is needed in the profile to distinguish between resources of different types that have the same name. The “type” part of the profile name can be CHANNEL, QUEUE, TOPIC, PROCESS, or NAMELIST. For example, a user might be authorized to define hlq.QUEUE.PAYROLL.ONE, but not authorized to define hlq.PROCESS.PAYROLL.ONE

If the resource type is a queue, and the profile is a queue-sharing group level profile, it controls access to one or more local queues within the queue sharing group, or access to a single shared queue from any queue manager in the queue-sharing group.

MQSC commands, profiles, and their access levels shows, for each WebSphere MQ MQSC command, the profiles required for command security checking to be carried out, and the corresponding access level for each profile in the MQCMDS class.

PCF commands, profiles, and their access levels shows, for each WebSphere MQ PCF command, the profiles required for command security checking to be carried out, and the corresponding access level for each profile in the MQCMDS class.

Command resource security checking for alias queues and remote queues

Alias queue and remote queues both provide indirection to another queue. Additional points apply when you consider security checking for these queues.

Parent topic: Profiles used to control access to WebSphere MQ resources

14.5.9.1. Command resource security checking for alias queues and remote queues

Alias queue and remote queues both provide indirection to another queue. Additional points apply when you consider security checking for these queues.

Alias queues

When you define an alias queue, command resource security checks are only performed against the name of the alias queue, not against the name of the target queue to which the alias resolves.

Alias queues can resolve to both local and remote queues. If you do not want to permit users access to certain local or remote queues, you must do both of the following:

1. Do not allow the users access to these local and remote queues.
2. Restrict the users from being able to define aliases for these queues. That is, prevent them from being able to issue DEFINE QALIAS and ALTER QALIAS commands.

Remote queues

When you define a remote queue, command resource security checks are performed only against the name of the remote queue. No checks are performed against the names of the queues specified in the RNAME or XMITQ attributes in the remote queue object definition.

Parent topic: Profiles for command resource security

Related information
START CHINIT

14.6. The RESLEVEL security profile

You can define a special profile in the MQADMIN or MXADMIN class to control the number of user IDs checked for API-resource security. This profile is referred to as the RESLEVEL profile How this profile affects API-resource security depends on how you access WebSphere® MQ.

When an application tries to connect to WebSphere MQ, WebSphere MQ checks the access that the user ID associated with the connection has to a profile in the MQADMIN or MXADMIN class called:

```
hlq.RESLEVEL
```

where `hlq` can be either `ssid` (subsystem ID) or `qsg` (queue-sharing group ID).

The user IDs associated with each connection type are:

- The user ID of the connecting task for batch connections
- The CICS® address space user ID for CICS connections
- The IMS™ region address space user ID for IMS connections
- The channel initiator address space user ID for channel initiator connections

Attention: RESLEVEL is a very powerful option; it can cause the bypassing of all resource security checks for a particular connection.

If you do not have a RESLEVEL profile defined, you must be careful that no other profile in the MQADMIN class matches hlq.RESLEVEL. For example, if you have a profile in MQADMIN called hlq.** and no hlq.RESLEVEL profile, beware of the consequences of the hlq.** profile because it is used for the RESLEVEL check.

Define an hlq.RESLEVEL profile and set the UACC to NONE, rather than have no RESLEVEL profile at all. Have as few users or groups in the access list as possible. For details about how to audit RESLEVEL access, see Auditing considerations on z/OS.

If you are using queue manager level security only, WebSphere MQ performs RESLEVEL checks against the `qmgr-name.RESLEVEL` profile. If you are using queue-sharing group level security only, WebSphere MQ performs RESLEVEL checks against the `qsg-name.RESLEVEL` profile. If you are using a combination of both queue manager and queue-sharing group level security, WebSphere MQ first checks for the existence of a RESLEVEL profile at queue manager level. If it does not find one, it checks for a RESLEVEL profile at queue-sharing group level.

If it cannot find a RESLEVEL profile, WebSphere MQ enables checking of both the job and task (or alternate user) ID for a CICS or an IMS connection. For a batch connection, WebSphere MQ enables checking of the job (or alternate) user ID. For the channel initiator, WebSphere MQ enables checking of the channel user ID and the MCA (or alternate) user ID.

If there is a RESLEVEL profile, the level of checking depends on the environment and access level for the profile.

Remember that if your queue manager is a member of a queue-sharing group and you do not define this profile at queue-manager level, there might be one defined at queue-sharing group level that will affect the level of checking. To activate the checking of two user IDs, you define a RESLEVEL profile (prefixed with either the queue manager name of the queue-sharing group name) with a UACC(NONE) and ensure that the relevant users do not have access granted against this profile.

When you consider the access that the channel initiator's user ID has to RESLEVEL, remember that the connection established by the channel initiator is also the connection used by the channels. A setting that causes the bypassing of all resource security checks for the channel initiator's user ID effectively bypasses security checks for all channels. If the channel initiator's user ID access to RESLEVEL is something other than NONE, then only one user ID (for an access level of READ or UPDATE) or no user IDs (for an access level of CONTROL or ALTER) is checked for access. If you grant the channel initiator's user ID an access level other than NONE to RESLEVEL, be sure that you understand the effect of this setting on the security checks done for channels.

Using the RESLEVEL profile means that normal security audit records are not taken. For example, if you put UAUDIT on a user, the access to the hlq.RESLEVEL profile in MQADMIN is not audited.

If you use the RACF WARNING option on the hlq.RESLEVEL profile, no RACF warning messages are produced for profiles in the RESLEVEL class.

Security checking for report messages such as CODs are controlled by the RESLEVEL profile associated with the originating application. For example, if a batch job's userid has CONTROL or ALTER authority to a RESLEVEL profile, then all resource checking performed by the batch job are bypassed, including the security check of report messages.

If you change the RESLEVEL profile, users must disconnect and connect again before the change takes place. (This includes stopping and restarting the channel initiator if the access that the distributed queuing address space user ID has to the RESLEVEL profile is changed.)

To switch RESLEVEL auditing off, use the RESAUDIT system parameter.

RESLEVEL and batch connections
By default, when a WebSphere MQ resource is being accessed through batch and batch-type connections, the user must be authorized to access that resource for the particular operation. You can bypass the security check by setting up an appropriate RESLEVEL definition.

RESLEVEL and system functions
The application of RESLEVEL to the operation and control panels, and to CSQUTIL.

RESLEVEL and CICS connections
By default, when an API-resource security check is made on a CICS connection, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

RESLEVEL and IMS connections
By default, when an API-resource security check is made for an IMS connection, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

RESLEVEL and the channel initiator connection
By default, when an API-resource security check is made by the channel initiator, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

RESLEVEL and intra-group queuing
By default, when an API-resource security check is made by the intra-group queuing agent, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

RESLEVEL and the user IDs checked
Example of setting a RESLEVEL profile and granting access to it.

Parent topic: Setting up security on z/OS

This build: January 26, 2011 11:22:18

Notices | Trademarks | Downloads | Library | Support | Feedback

14.6.1. RESLEVEL and batch connections

By default, when a WebSphere® MQ resource is being accessed through batch and batch-type connections, the user must be authorized to access that resource for the particular operation. You can bypass the security check by setting up an appropriate RESLEVEL definition.

Whether the user is checked or not is based on the user ID used at connect time, the same user ID used for the connection check.

For example, you can set up RESLEVEL so that when a user you trust accesses certain resources through a batch connection, no API-resource security checks are done; but when a user you do not trust tries to access the same resources, security checks are carried out as normal. You should set up RESLEVEL checking to bypass API-resource security checks only when you sufficiently trust the user and the programs run by that user.

The following table shows the checks made for batch connections.

Table 1. Checks made at different RACF access levels for batch connections

RACF® access level	Level of checking
NONE	Resource checks performed
READ	Resource checks performed
UPDATE	Resource checks performed
CONTROL	No check.
ALTER	No check.

Parent topic: The RESLEVEL security profile

This build: January 26, 2011 11:22:18

Notices | Trademarks | Downloads | Library | Support | Feedback

14.6.2. RESLEVEL and system functions

The application of RESLEVEL to the operation and control panels, and to CSQUTIL.

The operation and control panels and the CSQUTIL utility are batch-type applications that make requests to the queue manager's command server, and so they are subject to the considerations described in RESLEVEL and batch connections. You can use RESLEVEL to bypass security checking for the SYSTEM.COMMAND.INPUT and SYSTEM.COMMAND.REPLY.MODEL queues that they use, but not for the dynamic queues SYSTEM.CSQXCMD.*, SYSTEM.CSQOREXX.*, and SYSTEM.CSQUTIL.*.

The command server is an integral part of the queue manager and so does not have connection or RESLEVEL checking associated with it. To maintain security, therefore, the command server must confirm that the user ID of the requesting application has authority to open the queue being used for replies. For the operations and control panels, this is SYSTEM.CSQOREXX.*. For CSQUTIL, it is SYSTEM.CSQUTIL.*. Users must be authorized to use these queues, as described in System queue security, in addition to any RESLEVEL authorization they are given.

For other applications using the command server, it is the queue they name as their reply-to queue. Such other applications might deceive the command server into placing messages on unauthorized queues by passing (in the message context) a more trusted user ID than its own to the command server. To prevent this, use a CONTEXT profile to protect the identity context of messages placed on SYSTEM.COMMAND.INPUT.

Parent topic: The RESLEVEL security profile

14.6.3. RESLEVEL and CICS connections

By default, when an API-resource security check is made on a CICS® connection, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

By default, when an API-resource security check is made on a CICS connection, two user IDs are checked to see if access is allowed to the resource.

The first user ID checked is that of the CICS address space. This is the user ID on the job card of the CICS job, or the user ID assigned to the CICS started task by the z/OS® STARTED class or the started procedures table. (It is not the CICS DFLTUSER.)

The second user ID checked is the user ID associated with the CICS transaction.

If one of these user IDs does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED. Both the CICS address space user ID and the user ID of the person running the CICS transaction must have access to the resource at the correct level.

How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested. The possible checks are:

- Check the CICS address space user ID and the transaction user ID.
- Check the CICS address space user ID only.
- If the transaction is defined to CICS with RESSEC(NO), check the CICS address space user ID only. (The status of the CICS security is NOT checked when considering the transaction RESSEC setting. For example, if CICS has been started with SEC=NO, but the transaction has been defined with RESSEC(YES), WebSphere® MQ still checks both user IDs.)
- If the transaction is defined to CICS with RESSEC(YES), check the CICS address space user ID and the transaction user ID.
- Do not check any user IDs.

The user IDs checked depend on the user ID used at connection time, that is, the CICS address space user ID. This control enables you to bypass API-resource security checking for WebSphere MQ requests coming from one system (for example, a test system, TESTCICS,) but to implement them for another (for example, a production system, PRODCICS).

Note: If you set up your CICS address space user ID with the "trusted" attribute in the STARTED class or the RACF® started procedures table ICHRIN03, this overrides any user ID checks for the CICS address space established by the RESLEVEL profile for your queue manager (that is, the queue manager does not perform the security checks for the CICS address space). For more information, see the *CICS Transaction Server for z/OS V3.2 RACF Security Guide*.

The following table shows the checks made for CICS connections.

Table 1. Checks made at different RACF access levels for CICS connections

RACF access level	Level of checking
NONE	Check the CICS address space user ID and the task or alternate user ID.
READ	Check the CICS address space user ID.
UPDATE	Check the CICS address space user ID and, if the transaction has been defined with RESSEC=YES, also check the task or alternate user ID.
CONTROL	No check.
ALTER	No check.

Parent topic: The RESLEVEL security profile

14.6.4. RESLEVEL and IMS connections

By default, when an API-resource security check is made for an IMS™ connection, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

By default, when an API-resource security check is made for an IMS connection, two user IDs are checked to see if access is allowed to the resource.

The first user ID checked is that of the address space of the IMS region. This is taken from either the USER field from the job card or the user ID assigned to the region from the z/OS® STARTED class or the started procedures table (SPT).

The second user ID checked is associated with the work being done in the dependent region. It is determined according to the type of the dependent region as shown in Table 2.

If either the first or second IMS user ID does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED.

The setting of WebSphere® MQ RESLEVEL profiles cannot alter the user ID under which IMS transactions are scheduled from the IBM-supplied MQ-IMS trigger monitor program CSQQTRMN. This user ID is the PSBNAME of that trigger monitor, which by default is CSQQTRMN.

How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested. The possible checks are:

- Check the IMS region address space user ID and the second user ID or alternate user ID.
- Check IMS region address space user ID only.
- Do not check any user IDs.

The following table shows the checks made for IMS connections.

Table 1. Checks made at different RACF access levels for IMS connections

RACF® access level	Level of checking
NONE	Check the IMS address space user ID and the IMS second user ID or alternate user ID.
READ	Check the IMS address space user ID.
UPDATE	Check the IMS address space user ID.
CONTROL	No check.
ALTER	No check.

Parent topic: The RESLEVEL security profile

14.6.5. RESLEVEL and the channel initiator connection

By default, when an API-resource security check is made by the channel initiator, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

By default, when an API-resource security check is made by the channel initiator, two user IDs are checked to see if access is allowed to the resource.

The user IDs checked can be that specified by the MCAUSER channel attribute, that received from the network, that of the channel initiator address space, or the alternate user ID for the message descriptor. Which user IDs are checked depends on the communication protocol you are using and the setting of the PUTAUT channel attribute. See User IDs used by the channel initiator for more information.

If one of these user IDs does not have access to the resource, the request fails with a completion code of MQRC_NOT_AUTHORIZED.

How RESLEVEL can affect the checks made

Depending on how you set up your RESLEVEL profile, you can change which user IDs are checked when access to a resource is requested, and how many are checked.

The following table shows the checks made for the channel initiator's connection, and for all channels since they use this connection.

Table 1. Checks made at different RACF access levels for channel initiator connections

RACF® access level	Level of checking
NONE	Check two user IDs.
READ	Check one user ID.
UPDATE	Check one user ID.
CONTROL	No check.
ALTER	No check.
Note: See User IDs used by the channel initiator for a definition of the user IDs checked	

Parent topic: The RESLEVEL security profile

14.6.6. RESLEVEL and intra-group queuing

By default, when an API-resource security check is made by the intra-group queuing agent, two user IDs are checked. You can change which user IDs are checked by setting up a RESLEVEL profile.

By default, when an API-resource security check is made by the intra-group queuing agent, two user IDs are checked to see if access is allowed to the resource.

The user IDs checked can be the user ID determined by the IGQUSER attribute of the receiving queue manager, the user ID of the queue manager within the queue-sharing group that put the message on to the SYSTEM.QSG.TRANSMIT.QUEUE, or the alternate user ID specified in the `UserIdentifier` field of the message descriptor of the message. See User IDs used by the intra-group queuing agent for more information.

Because the intra-group queuing agent is an internal queue manager task, it does not issue an explicit connect request and runs under the user ID of the queue manager. The intra-group queuing agent starts at queue manager initialization. During the initialization of the intra-group queuing agent, WebSphere® MQ checks the access that the user ID associated with the queue manager has to a profile in the MQADMIN class called:

```
hlq.RESLEVEL
```

This check is always performed unless the hlq.NO.SUBSYS.SECURITY switch has been set.

If there is no RESLEVEL profile, WebSphere MQ enables checking for two user IDs. If there is a RESLEVEL profile, the level of checking depends on the access level granted to the user ID of the queue manager for the profile. Table 1 shows the checks made for the intra-group queuing agent.

Table 1. Checks made at different RACF access levels for the intra-group queuing agent

RACF® access level	Level of checking
NONE	Check two user IDs.
READ	Check one user ID.
UPDATE	Check one user ID.
CONTROL	No check.
ALTER	No check.

Note: See User IDs used by the intra-group queuing agent for a definition of the user IDs checked

If the permissions granted to the RESLEVEL profile for the queue manager's user ID are changed, the intra-group queuing agent must be stopped and restarted to pick up the new permissions. Because there is no way to independently stop and restart the intra-group queuing agent, the queue manager must be stopped and restarted to achieve this.

Parent topic: The RESLEVEL security profile

14.6.7. RESLEVEL and the user IDs checked

Example of setting a RESLEVEL profile and granting access to it.

Table 1 through Table 1 show how RESLEVEL affects which user IDs are checked for different MQI requests.

For example, you have a queue manager called QM66 with the following requirements:

- User WS21B is to be exempt from resource security.
- CICS® started task WXNCICS running under address space user ID CICSWXN is to perform full resource checking only for transactions defined with RESSEC(YES).

To define the appropriate RESLEVEL profile, issue the following RACF® command:

```
RDEFINE MQADMIN QM66.RESLEVEL UACC(NONE)
```

Then give the users access to this profile, using the following commands:

```
PERMIT QM66.RESLEVEL CLASS(MQADMIN) ID(WS21B) ACCESS(CONTROL)
PERMIT QM66.RESLEVEL CLASS(MQADMIN) ID(CICSWXN) ACCESS(UPDATE)
```

If you make these changes while the user IDs are connected to queue manager QM66, the users must disconnect and connect again before the change takes place.

If subsystem security is not active when a user connects but, while this user is still connected, subsystem security becomes active, full resource security checking is applied to the user. The user must reconnect to get the correct RESLEVEL processing.

Parent topic: The RESLEVEL security profile

14.7. User IDs for security checking

WebSphere® MQ initiates security checks based on user IDs associated with users, terminals, applications, and other resources. This collection of topics lists which user IDs are used for each type of security check.

User IDs for connection security
The user ID used for connection security depends on the type of connection.

User IDs for command security and command resource security
The user ID used for command security or command resource security depends on where the command is issued from.

User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)
This information shows the contents of the user IDs for normal and alternate user IDs for each type of connection. The number of checks is defined by the RESLEVEL profile. The user ID checked is that used for **MQOPEN**, **MQSUB**, or **MQPUT1** calls.

Blank user IDs and UACC levels
If a blank user ID occurs, a RACF® undefined user is signed on. Do not grant wide-ranging access to the undefined user.

Parent topic: Setting up security on z/OS

14.7.1. User IDs for connection security

The user ID used for connection security depends on the type of connection.

Connection type	User ID contents
Batch connection	The user ID of the connecting task. For example:

	• The TSO user ID • The user ID assigned to a batch job by the USER JCL parameter • The user ID assigned to a started task by the STARTED class or the started procedures table
CICS® connection	The CICS address space user ID.
IMS™ connection	The IMS region address space user ID.
Channel initiator connection	The channel initiator address space user ID.

Parent topic: User IDs for security checking

14.7.2. User IDs for command security and command resource security

The user ID used for command security or command resource security depends on where the command is issued from.

Issued from...	User ID contents
CSQINP1 or CSQINP2	No check is made.
System command input queue	The user ID found in the `UserIdentifier` of the message descriptor of the message that contains the command. If the message does not contain a `UserIdentifier`, a user ID of blanks is passed to the security manager.
Console	The user ID signed onto the console. If the console is not signed on, the default user ID set by the CMDUSER system parameter in CSQ6SYSP. To issue commands from a console, the console must have the z/OS® SYS AUTHORITY attribute.
SDSF/TSO console	TSO or job user ID.
Operations and control panels	TSO user ID. If you are going to use the operations and control panels, you must have the appropriate authority to issue the commands corresponding to the actions that you choose. In addition, you must have READ access to all the hlq.DISPLAY.*object* profiles in the MQCMDS class because the panels use the various DISPLAY commands to gather the information that they present.
MGCRE	If MGCRE is used with Utoken, the user ID in the Utoken. If MGCRE is issued without the Utoken, the TSO or job user ID is used.
CSQUTIL	Job user ID.
CSQINPX	User ID of the channel initiator address space.

Parent topic: User IDs for security checking

14.7.3. User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

This information shows the contents of the user IDs for normal and alternate user IDs for each type of connection. The number of checks is defined by the RESLEVEL profile. The user ID checked is that used for **MQOPEN**, **MQSUB**, or **MQPUT1** calls.

Note: All user ID fields are checked exactly as they are received. No conversions take place, and, for example, three user ID fields containing "Bob", "BOB", and "bob" are not equivalent.

User IDs checked for batch connections
The user ID checked for a batch connection depends on how the task is run and whether an alternate user ID. has been specified.

User IDs checked for CICS connections
The user IDs checked for CICS® connections depend on whether one or two checks are to be carried out, and whether an alternate user ID is specified.

User IDs checked for IMS connections
The user IDs checked for IMS™ connections depend on whether one or two checks are to be -performed, and whether an alternate user ID is specified. If a second user ID is checked, it depends on the type of dependent region and on which user IDs are available.

User IDs used by the channel initiator

User IDs used by the intra-group queuing agent
The user IDs that are checked when the intra-group queuing agent opens destination queues are determined by the values of the IGQAUT and IGQUSER queue manager attributes.

Parent topic: User IDs for security checking

14.7.3.1. User IDs checked for batch connections

The user ID checked for a batch connection depends on how the task is run and whether an alternate user ID. has been specified.

Table 1. User ID checking against profile name for batch connections

Alternate user ID specified on open?	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
No	–	JOB	JOB
Yes	JOB	JOB	ALT

Key:

ALT

Alternate user ID.

JOB

- The user ID of a TSO or USS sign-on.
- The user ID assigned to a batch job.
- The user ID assigned to a started task by the STARTED class or the started procedures table.
- The user ID associated with the executing DB2® stored procedure

A Batch job is performing an **MQPUT1** to a queue called Q1 with RESLEVEL set to READ and alternate user ID checking turned off.

Table 1 and Table 1 show that the job user ID is checked against profile hlq.Q1.

Parent topic: User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

14.7.3.2. User IDs checked for CICS connections

The user IDs checked for CICS® connections depend on whether one or two checks are to be carried out, and whether an alternate user ID is specified.

Table 1. User ID checking against profile name for CICS-type user IDs

Alternate user ID specified on open?	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
No, 1 check	–	ADS	ADS
No, 2 checks	–	ADS+TXN	ADS+TXN
Yes, 1 check	ADS	ADS	ADS
Yes, 2 checks	ADS+TXN	ADS+TXN	ADS+ALT

Key:

ALT

Alternate user ID

ADS

The user ID associated with the CICS batch job or, if CICS is running as a started task, through the STARTED class or the started procedures table.

TXN

The user ID associated with the CICS transaction. This is normally the user ID of the terminal user who started the transaction. It can be the CICS DFLTUSER, a PRESET security terminal, or a manually signed-on user.

Determine the user IDs checked for the following conditions:

- The RACF® access level to the RESLEVEL profile, for a CICS address space user ID, is set to NONE.
- An **MQOPEN** call is made against a queue with MQOO_OUTPUT and MQOO_PASS_IDENTITY_CONTEXT.

Answer: First, see how many CICS user IDs are checked based on the CICS address space user ID access to the RESLEVEL profile. From Table 1, two user IDs are checked if the RESLEVEL profile is set to NONE. Then, from Table 1, these checks are carried out:

- The hlq.ALTERNATE.USER.userid profile is not checked.
- The hlq.CONTEXT.queuename profile is checked with both the CICS address space user ID and the CICS transaction user ID.
- The hlq.resourcename profile is checked with both the CICS address space user ID and the CICS transaction user ID.

This means that four security checks are made for this **MQOPEN** call.

Parent topic: User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

14.7.3.3. User IDs checked for IMS connections

The user IDs checked for IMS™ connections depend on whether one or two checks are to be -performed, and whether an alternate user ID is specified. If a second user ID is checked, it depends on the type of dependent region and on which user IDs are available.

Table 1. User ID checking against profile name for IMS-type user IDs

Alternate user ID specified on open?	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
No, 1 check	–	REG	REG
No, 2 checks	–	REG+SEC	REG+SEC
Yes, 1 check	REG	REG	REG
Yes, 2 checks	REG+SEC	REG+SEC	REG+ALT

Key:

ALT
Alternate user ID.

REG
The user ID is normally set through the STARTED class or the started procedures table or, if IMS is running, from a submitted job, by the USER JCL parameter.

SEC
The second user ID is associated with the work being done in a dependent region. It is determined according to Table 2.

Table 2. How the second user ID is determined for the IMS connection

Types of dependent region	Hierarchy for determining the second user ID
• BMP message driven and successful GET UNIQUE issued. • IFP and GET UNIQUE issued. • MPP.	User ID associated with the IMS transaction if the user is signed on. LTERM name if available. PSBNAME.
• BMP message driven and successful GET UNIQUE not issued. • BMP not message driven. • IFP and GET UNIQUE not issued.	User ID associated with the IMS dependent region address space if this is not all blanks or all zeros. PSBNAME.

Parent topic: User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

14.7.3.4. User IDs used by the channel initiator

The following sections describe the user IDs used and checked for the following:

- TCP/IP receiving channels.
- LU 6.2 receiving channels.
- Client MQI requests issued over server-connection channels for both TCP/IP and LU 6.2.

You can use the PUTAUT parameter of the receiving channel definition to determine the type of security checking used. To get consistent security checking throughout your WebSphere® MQ network, you can use the ONLYMCA and ALTMCA options.

You can use the DISPLAY CHSTATUS command to determine the user identifier used by the MCA. See WebSphere MQ Script (MQSC) Command Reference.

Receiving channels using TCP/IP
The user IDs checked depend on the PUTAUT option of the channel and on whether one or two checks are to be performed.

Receiving channels using LU 6.2
The user IDs checked depend on the PUTAUT option of the channel and on whether one or two checks are to be performed.

Client MQI requests
Various user IDs can be used, depending on which user IDs and environment variables have been set. These user IDs are checked against various profiles, depending on the PUTAUT option used and whether an alternate user ID is specified.

Channel initiator example
An example of how user IDs are checked against RACF® profiles.

Parent topic: User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

Related information
DISPLAY CHSTATUS
DEFINE CHANNEL

14.7.3.4.1. Receiving channels using TCP/IP

The user IDs checked depend on the PUTAUT option of the channel and on whether one or two checks are to be performed.

Table 1. User IDs checked against profile name for TCP/IP channels

PUTAUT option specified on receiver or requester channel	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
DEF, 1 check	–	CHL	CHL
DEF, 2 checks	–	CHL + MCA	CHL + MCA
CTX, 1 check	CHL	CHL	CHL
CTX, 2 checks	CHL + MCA	CHL + MCA	CHL + ALT
ONLYMCA, 1 check	–	MCA	MCA
ONLYMCA, 2 checks	–	MCA	MCA

ALTMCA, 1 check	MCA	MCA	MCA
ALTMCA, 2 checks	MCA	MCA	MCA + ALT

Key:

MCA (MCA user ID)

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

CHL (Channel user ID)

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

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

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

ALT (Alternate user ID)

The user ID from the context information (that is, the *UserIdentifier* field) within the message descriptor of the message. This user ID is moved into the *AlternateUserID* field in the object descriptor before an **MQOPEN**or **MQPUT1** call is issued for the target destination queue.

Parent topic: User IDs used by the channel initiator

14.7.3.4.2. Receiving channels using LU 6.2

The user IDs checked depend on the PUTAUT option of the channel and on whether one or two checks are to be performed.

Table 1. User IDs checked against profile name for LU 6.2 channels

PUTAUT option specified on receiver or requester channel	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
DEF, 1 check	–	CHL	CHL
DEF, 2 checks	–	CHL + MCA	CHL + MCA
CTX, 1 check	CHL	CHL	CHL
CTX, 2 checks	CHL + MCA	CHL + MCA	CHL + ALT
ONLYMCA, 1 check	–	MCA	MCA
ONLYMCA, 2 checks	–	MCA	MCA
ALTMCA, 1 check	MCA	MCA	MCA
ALTMCA, 2 checks	MCA	MCA	MCA + ALT

Key:

MCA (MCA user ID)

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

CHL (Channel user ID)

Requester-server channels

If the channel is started from the requester, there is no opportunity to receive a network user ID (the channel user ID).

If the PUTAUT parameter is set to DEF or CTX on the requester channel, the channel user ID is that of the channel initiator address space of the requester because no user ID is received from the network.

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

Other channel types

If the PUTAUT parameter is set to DEF or CTX on the receiver or requester channel, the channel user ID is the user ID received from the communications system when the channel is initiated.

- If the sending channel is on z/OS®, the channel user ID received is the channel initiator address space user ID of the sender.
- If the sending channel is on a different platform (for example, AIX® or HP-UX), the channel user ID received is typically provided by the USERID parameter of the channel definition.

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

ALT (Alternate user ID)

The user ID from the context information (that is, the *UserIdentifier* field) within the message descriptor of the message. This user ID is moved into the *AlternateUserID* field in the object descriptor before an **MQOPEN**or **MQPUT1** call is issued for the target destination queue.

Parent topic: User IDs used by the channel initiator

14.7.3.4.3. Client MQI requests

Various user IDs can be used, depending on which user IDs and environment variables have been set. These user IDs are checked against various profiles, depending on the PUTAUT option used and whether an alternate user ID is specified.

This section describes the user IDs checked for client MQI requests issued over server-connection channels for TCP/IP and LU 6.2. The MCA user ID and channel user ID are as for the TCP/IP and LU 6.2 channels described in the previous sections.

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

See WebSphere MQ Clients for more information.

For client **MQOPEN**, **MQSUB**, and **MQPUT1** requests, use the following rules to determine the profile that is checked:

- If the request specifies alternate-user authority, a check is made against the *hlq*.ALTERNATE.USER.*userid* profile.
- If the request specifies context authority, a check is made against the *hlq*.CONTEXT.*queuename* profile.
- For all **MQOPEN**, **MQSUB**, and **MQPUT1** requests, a check is made against the *hlq.resourcename* profile.

When you have determined which profiles are checked, use the following table to determine which user IDs are checked against these profiles.

Table 1. User IDs checked against profile name for LU 6.2 and TCP/IP server-connection channels

PUTAUT option specified on server-connection channel	Alternate user ID specified on open?	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
***DEF*, 1 check**	No	–	CHL	CHL
***DEF*, 1 check**	Yes	CHL	CHL	CHL
***DEF*, 2 checks**	No	–	CHL + MCA	CHL + MCA
***DEF*, 2 checks**	Yes	CHL + MCA	CHL + MCA	CHL + ALT
***ONLYMCA*, 1 check**	No	–	MCA	MCA
***ONLYMCA*, 1 check**	Yes	MCA	MCA	MCA
***ONLYMCA*, 2 checks**	No	–	MCA	MCA
***ONLYMCA*, 2 checks**	Yes	MCA	MCA	MCA + ALT

Key:

ALT
Alternate user ID.

CHL
Channel user ID.

MCA
MCA user ID.

Parent topic: User IDs used by the channel initiator

14.7.3.4.4. Channel initiator example

An example of how user IDs are checked against RACF® profiles.

A user performs an **MQPUT1** operation to a queue on queue manager QM01 that resolves to a queue called QB on queue manager QM02. The message is sent on a TCP/IP channel called QM01.TO.QM02. RESLEVEL is set to NONE, and the open is performed with alternate user ID and context checking. The receiver channel definition has PUTAUT(CTX) and the MCA user ID is set. Which user IDs are used on the receiving channel to put the message to queue QB?

Answer: Table 1 shows that two user IDs are checked because RESLEVEL is set to NONE.

Table 1 shows that, with PUTAUT set to CTX and 2 checks, the following user IDs are checked:

- The channel initiator user ID and the MCAUSER user ID are checked against the hlq.ALTERNATE.USER.userid profile.
- The channel initiator user ID and the MCAUSER user ID are checked against the hlq.CONTEXT.queuename profile.
- The channel initiator user ID and the alternate user ID specified in the message descriptor (MQMD) are checked against the hlq.Q2 profile.

Parent topic: User IDs used by the channel initiator

14.7.3.5. User IDs used by the intra-group queuing agent

The user IDs that are checked when the intra-group queuing agent opens destination queues are determined by the values of the IGQAUT and IGQUSER queue manager attributes.

The possible user IDs are:

Intra-group queuing user ID (IGQ)

The user ID determined by the IGQUSER attribute of the receiving queue manager. If this is set to blanks, the user ID of the receiving queue manager is used. However, because the receiving queue manager has authority to access all queues defined to it, security checks are not performed for the receiving queue manager's user ID. In this case:

- If only one user ID is to be checked and the user ID is that of the receiving queue manager, no security checks take place. This can occur when IGAUT is set to ONLYIGQ or ALTIGQ.
- If two user IDs are to be checked and one of the user IDs is that of the receiving queue manager, security checks take place for the other user ID only. This can occur when IGAUT is set to DEF, CTX, or ALTIGQ.
- If two user IDs are to be checked and both user IDs are that of the receiving queue manager, no security checks take place. This can occur when IGAUT is set to ONLYIGQ.

Sending queue manager user ID (SND)

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

Alternate user ID (ALT)

The user ID specified in the *UserIdentifier* field in the message descriptor of the message.

Table 1. User IDs checked against profile name for intra-group queuing

IGQAUT option specified on receiving queue manager	hlq.ALTERNATE.USER.userid profile	hlq.CONTEXT.queuename profile	hlq.resourcename profile
DEF, 1 check	–	SND	SND
DEF, 2 checks	–	SND +IGQ	SND +IGQ
CTX, 1 check	SND	SND	SND
CTX, 2 checks	SND + IGQ	SND +IGQ	SND + ALT
ONLYIGQ, 1 check	–	IGQ	IGQ
ONLYIGQ, 2 checks	–	IGQ	IGQ
ALTIGQ, 1 check	–	IGQ	IGQ
ALTIGQ, 2 checks	IGQ	IGQ	IGQ + ALT

Key:

ALT
Alternate user ID.

IGQ
IGQ user ID.

SND
Sending queue manager user ID.

Parent topic: User IDs for resource security (MQOPEN, MQSUB, and MQPUT1)

14.7.4. Blank user IDs and UACC levels

If a blank user ID occurs, a RACF® undefined user is signed on. Do not grant wide-ranging access to the undefined user.

Blank user IDs can exist when a user is manipulating messages using context or alternate-user security, or when WebSphere® MQ is passed a blank user ID. For example, a blank user ID is used when a message is written to the system-command input queue without context.

Note: A user ID of "`*       `" (that is, an asterisk character followed by seven spaces) is treated as an undefined user ID.

WebSphere MQ passes the blank user ID to RACF and a RACF undefined user is signed on. All security checks then use the universal access (UACC) for the relevant profile. Depending on how you have set your access levels, the UACC might give the undefined user a wide-ranging access.

For example, if you issue this RACF command from TSO:

```
RDEFINE MQQUEUE Q.AVAILABLE.TO.EVERYONE UACC(UPDATE)
```

you define a profile that enables both z/OS-defined user IDs (that have not been put in the access list) and the RACF undefined user ID to put messages on, and get messages from, that queue.

To protect against blank user IDs you must plan your access levels carefully, and limit the number of people who can use context and alternate-user security. You must prevent people using the RACF undefined user ID from getting access to resources that they must not access. However, at the same time, you must allow access to people with defined user IDs. To do this, you can specify a user ID of asterisk (*) in a RACF command PERMIT, giving access to resources for all defined user IDs. Therefore all undefined user IDs (such as "`*       `") are denied access. For example, these RACF commands prevent the RACF undefined user ID from gaining access to the queue to put or get messages:

```
RDEFINE MQQUEUE Q.AVAILABLE.TO.RACF.DEFINED.USERS.ONLY UACC(NONE)
PERMIT Q.AVAILABLE.TO.RACF.DEFINED.USERS.ONLY CLASS(MQQUEUE) ACCESS(UPDATE) ID(*)
```

Parent topic: User IDs for security checking

14.8. WebSphere MQ for z/OS security management

WebSphere® MQ uses an in-storage table to hold information relating to each user and the access requests made by each user. To manage this table efficiently and to reduce the number of requests made from WebSphere MQ to the external security manager (ESM), a number of controls are available.

These controls are available through both the operations and control panels and WebSphere MQ commands.

User ID reverification
If the RACF® definition of a user who is using WebSphere MQ resources has been changed—for example, by connecting the user to a new group—you can tell the queue manager to sign this user on again the next time it tries to access a WebSphere MQ resource. You can do this by using the WebSphere MQ command RVERIFY SECURITY.

User ID timeouts
You can make WebSphere MQ sign a user off a queue manager after a period of inactivity.

Refreshing queue manager security on z/OS
WebSphere MQ for z/OS® caches RACF data to improve performance. When you change certain security classes, you must refresh this cached information. Refresh security infrequently, for performance reasons. You can also choose to refresh only SSL security information.

Displaying security status
To display the status of the security switches, and other security controls, issue the MQSC DISPLAY SECURITY command.

Parent topic: Setting up security on z/OS

14.8.1. User ID reverification

If the RACF® definition of a user who is using WebSphere® MQ resources has been changed—for example, by connecting the user to a new group—you can tell the queue manager to sign this user on again the next time it tries to access a WebSphere MQ resource. You can do this by using the WebSphere MQ command RVERIFY SECURITY.

- User HX0804 is getting and putting messages to the PAYROLL queues on queue manager PRD1. However HX0804 now requires access to some of the PENSION queues on the same queue manager (PRD1).
- The data security administrator connects user HX0804 to the RACF group that allows access to the PENSION queues.
- So that HX0804 can access the PENSION queues immediately—that is, without shutting down queue manager PRD1, or waiting for HX0804 to time out—you must use the WebSphere MQ command:

  ```
  RVERIFY SECURITY(HX0804)
  ```

Note: If you turn off user ID timeout for long periods of time (days or even weeks) while the queue manager is running, you must remember to run the RVERIFY SECURITY command for any users that have been revoked or deleted in that time.

Parent topic: WebSphere MQ for z/OS security management

14.8.2. User ID timeouts

You can make WebSphere MQ sign a user off a queue manager after a period of inactivity.

When a user accesses a WebSphere® MQ resource, the queue manager tries to sign this user on to the queue manager (if subsystem security is active). This means that the user is authenticated to the ESM. This user remains signed on to WebSphere MQ until either the queue manager is shut down, or until the user ID is *timed out* (the authentication lapses) or reverified (reauthenticated).

When a user is timed out, the user ID is *signed off* within the queue manager and any security-related information retained for this user is discarded. The signing on and off of the user within the queue manager is not apparent to the application program or to the user.

Users are eligible for timeout when they have not used any WebSphere MQ resources for a predetermined amount of time. This time period is set by the MQSC ALTER SECURITY command.

Two values can be specified in the ALTER SECURITY command:

TIMEOUT
The time period in minutes that an unused user ID and its associated resources can remain within the WebSphere MQ queue manager.

INTERVAL
The time period in minutes between checks for user IDs and their associated resources, to determine whether the *TIMEOUT* has expired.

For example, if the *TIMEOUT* value is 30 and the *INTERVAL* value is 10, every 10 minutes WebSphere MQ checks user IDs and their associated resources to determine whether any have not been used for 30 minutes. If a timed-out user ID is found, that user ID is signed off within the queue manager. If any timed-out resource information associated with non-timed-out user IDs is found, that resource information is discarded. If you do not want to time out user IDs, set the *INTERVAL* value to zero. However, if the *INTERVAL* value is zero, storage occupied by user IDs and their associated resources is not freed until you issue a **REFRESH SECURITY** or **RVERIFY SECURITY** command.

Tuning this value can be important if you have many one-off users. If you set small interval and timeout values, resources that are no longer required are freed.

Note: If you use values for *INTERVAL* or *TIMEOUT* other than the defaults, you must reenter the command at every queue manager startup. You can do this automatically by putting the **ALTER SECURITY** command in the CSQINP1 data set for that queue manager.

Parent topic: WebSphere MQ for z/OS security management

Related information
ALTER SECURITY

14.8.3. Refreshing queue manager security on z/OS

WebSphere® MQ for z/OS® caches RACF® data to improve performance. When you change certain security classes, you must refresh this cached information. Refresh security infrequently, for performance reasons. You can also choose to refresh only SSL security information.

When a queue is opened for the first time (or for the first time since a security refresh) WebSphere MQ performs a RACF check to obtain the user's access rights and places this information in the cache. The cached data includes user IDs and resources on which security checking has been performed. If the queue is opened again by the same user, the presence of the cached data means that WebSphere MQ does not have to issue RACF checks, which improves performance. The action of a security refresh is to discard any cached security information and so force WebSphere MQ to make a new check against RACF. Whenever you add, change or delete a RACF resource profile that is held in the MQADMIN, MXADMIN, MQPROC, MXPROC, MQQUEUE, MXQUEUE, MQNLIST, MXNLIST, or MXTOPIC class, you must tell the queue managers that use this class to refresh the security information that they hold. To do this, issue the following commands:

- The RACF SETROPTS RACLIST(classname) REFRESH command to refresh at the RACF level.
- The WebSphere MQ REFRESH SECURITY command to refresh the security information held by the queue manager. This command needs to be issued by each queue manager that accesses the profiles that have changed. If you have a queue-sharing group, you can use the command scope attribute to direct the command to all the queue managers in the group.

If you are using generic profiles in any of the WebSphere MQ classes, you must also issue normal RACF refresh commands if you change, add, or delete any generic profiles. For example, SETROPTS GENERIC(classname) REFRESH.

However, because WebSphere MQ uses the RACF dataspace, WebSphere MQ can use RACF profiles as soon as they become available. If a RACF resource profile is added, changed or deleted and the resource to which it applies has not yet been accessed (so no information is cached), WebSphere MQ uses the new RACF information without a security refresh being carried out.

If RACF auditing is turned on, (for example, by using the RACF RALTER AUDIT(access-attempt (audit_access_level)) command), no caching takes place, and therefore WebSphere MQ refers directly to the RACF dataspace for every check. Changes are therefore picked up immediately and REFRESH SECURITY is not necessary to access the changes. You can confirm whether RACF auditing is on by using the RACF RLIST command. For example, you could issue the command

```
RLIST MQQUEUE (qmgr.SYSTEM.COMMAND.INPUT) GEN
```

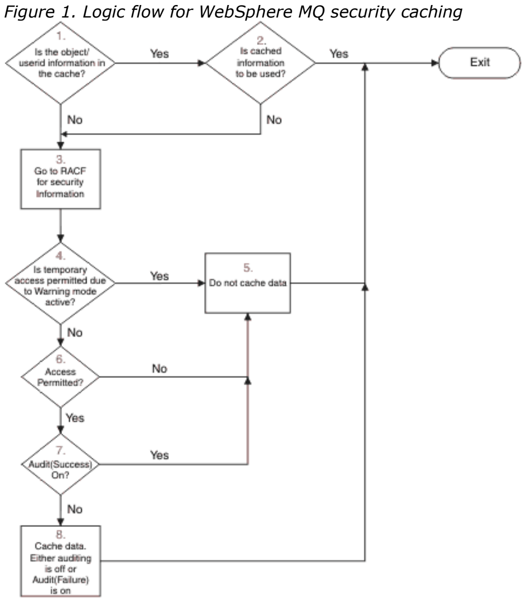
and receive the results

```
CLASS      NAME
-----      ----
MQQUEUE    QP*.SYSTEM.COMMAND.*.** (G)
    AUDITING
    --------
    FAILURES(READ)
```

This indicates that auditing is set on. For more information, refer to the *z/OS Security Server RACF Auditor's Guide* and the *z/OS Security Server RACF Command Language Reference*.

Figure 1 summarizes the situations in which security information is cached and in which cached information is used.

Figure 1. Logic flow for WebSphere MQ security caching


1. Is the object/ userid information in the cache?
Yes
2. Is cached information to be used?
Yes
Exit
No
No
3. Go to RACF for security Information
4. Is temporary access permitted due to Warning mode active?
Yes
5. Do not cache data
No
6. Access Permitted?
No
Yes
7. Audit(Success) On?
Yes
No
8. Cache data. Either auditing is off or Audit(Failure) is on

If you change your security settings by adding or deleting switch profiles in the MQADMIN or MXADMIN classes, use one of these commands to pick up these changes dynamically:

REFRESH SECURITY(*)

REFRESH SECURITY(MQADMIN)

REFRESH SECURITY(MXADMIN)

This means you can activate new security types, or deactivate them without having to restart the queue manager.

For performance reasons, these are the only classes affected by the REFRESH SECURITY command. You do *not* need to use REFRESH SECURITY if you change a profile in either the MQCONN or MQCMDS classes.

Note: A refresh of the MQADMIN or MXADMIN class is not required if you change a RESLEVEL security profile.

For performance reasons, use REFRESH SECURITY as infrequently as possible, ideally at off-peak times. You can minimize the number of security refreshes by connecting users to RACF groups that are already in the access list for WebSphere MQ profiles, rather than putting individual users in the access lists. In this way, you change the user rather than the resource profile. You can also RVERIFY SECURITY the appropriate user instead of refreshing security.

As an example of REFRESH SECURITY, suppose you define the new profiles to protect access to queues starting with INSURANCE.LIFE on queue manager PRMQ. You use these RACF commands:

```
RDEFINE MQQUEUE PRMQ.INSURANCE.LIFE.** UACC(NONE)
PERMIT PRMQ.INSURANCE.LIFE.** ID(LIFEGRP) ACCESS(UPDATE)
```

You must issue the following command to tell RACF to refresh the security information that it holds, for example:

```
SETROPTS RACLIST(MQQUEUE) REFRESH
```

Because these profiles are generic, you must tell RACF to refresh the generic profiles for MQQUEUE. For example:

```
SETROPTS GENERIC(MQQUEUE) REFRESH
```

Then you must use this command to tell queue manager PRMQ that the queue profiles have changed:

```
REFRESH SECURITY(MQQUEUE)
```

Refreshing SSL security

To refresh the cached view of the SSL Key Repository, issue the REFRESH SECURITY command with the option TYPE(SSL). This enables you to update some of your SSL settings without having to restart your channel initiator.

Parent topic: WebSphere MQ for z/OS security management

Related information
REFRESH SECURITY

14.8.4. Displaying security status

To display the status of the security switches, and other security controls, issue the MQSC DISPLAY SECURITY command.

The following figure shows typical output of the DISPLAY SECURITY ALL command.
Figure 1. Typical output from the DISPLAY SECURITY command

```
CSQH015I +CSQ1 Security timeout = 54 MINUTES
CSQH016I +CSQ1 Security interval = 12 MINUTES
CSQH030I +CSQ1 Security switches ...
CSQH034I +CSQ1 SUBSYSTEM: ON, 'SQ05.NO.SUBSYS.SECURITY' not found
CSQH032I +CSQ1 QMGR: ON, 'CSQ1.YES.QMGR.CHECKS' found
CSQH031I +CSQ1 QSG: OFF, 'SQ05.NO.QSG.CHECKS' found
CSQH031I +CSQ1 CONNECTION: OFF, 'CSQ1.NO.CONNECT.CHECKS' found
CSQH034I +CSQ1 COMMAND: ON, 'CSQ1.NO.COMMAND.CHECKS' not found
CSQH031I +CSQ1 CONTEXT: OFF, 'CSQ1.NO.CONTEXT.CHECKS' found
CSQH034I +CSQ1 ALTERNATE USER: ON, 'CSQ1.NO.ALTERNATE.USER.CHECKS' not found
CSQH034I +CSQ1 PROCESS: ON, 'CSQ1.NO.PROCESS.CHECKS' not found
CSQH034I +CSQ1 NAMELIST: ON, 'CSQ1.NO.NLIST.CHECKS' not found
CSQH034I +CSQ1 QUEUE: ON, 'CSQ1.NO.QUEUE.CHECKS' not found
CSQH034I +CSQ1 TOPIC: ON, 'CSQ1.NO.TOPIC.CHECKS' not found
CSQH031I +CSQ1 COMMAND RESOURCES: OFF, 'CSQ1.NO.CMD.RESC.CHECKS' found
CSQ9022I +CSQ1 CSQHPDTC ' DISPLAY SECURITY' NORMAL COMPLETION
```

The example shows that the queue manager that replied to the command has subsystem, command, alternate user, process, namelist, and queue security active at queue manager level but not at queue-sharing group level. Connection, command resource, and context security are not active. It also shows that user ID timeouts are active, and that every 12 minutes the queue manager checks for user IDs that have not been used in this queue manager for 54 minutes and removes them.

Note: This command shows the current security status. It does not necessarily reflect the current status of the switch profiles defined to RACF®, or the status of the RACF classes. For example, the switch profiles might have been changed since the last restart of this queue manager or REFRESH SECURITY command.

Parent topic: WebSphere MQ for z/OS security management

Related information
DISPLAY SECURITY

14.9. Security installation tasks

After installing and customizing WebSphere® MQ, authorize started task procedures to RACF®, authorize access to various resources, and set up RACF definitions. Optionally, configure your system for SSL.

When WebSphere MQ is first installed and customized, you must perform these security-related tasks:

1. Set up WebSphere MQ data set and system security by:
    - Authorizing the queue manager started-task procedure xxxxMSTR and the distributed queuing started-task procedure xxxxCHIN to run under RACF.
    - Authorizing access to queue manager data sets.
    - Authorizing access to resources for those user IDs that will use the queue manager and utility programs.
    - Authorizing access for those queue managers that will use the coupling facility list structures.
    - Authorizing access for those queue managers that will use DB2®.
2. Set up RACF definitions for WebSphere MQ security.

3. If you want to use the Secure Sockets Layer (SSL), prepare your system to use certificates and keys.

Setting up WebSphere MQ for z/OS data set security
There are many types of WebSphere MQ user. Use RACF to control their access to system data sets.

Setting up WebSphere MQ for z/OS resource security
There are many types of WebSphere MQ user. Use RACF to control their access to WebSphere MQ resources.

Configuring your system to use the Secure Sockets Layer (SSL)
Use this topic as example of how to configure WebSphere MQ for z/OS® with SSL using RACF commands.

Parent topic: Setting up security on z/OS

14.9.1. Setting up WebSphere MQ for z/OS data set security

There are many types of WebSphere® MQ user. Use RACF® to control their access to system data sets.

The possible users of WebSphere MQ data sets include the following entities:

- The queue manager itself.
- The channel initiator
- WebSphere MQ administrators, who need to create WebSphere MQ data sets, run utility programs, and so on.
- Application programmers who need to use the WebSphere MQ-supplied copybooks, include data sets, macros, and so on.
- Applications involving one or more of:
  - Batch jobs
  - TSO users
  - CICS® regions
  - IMS™ regions
- Data sets CSQOUTX and CSQSNAP
- Dynamic queues SYSTEM.CSQXCMD.*

For all these potential users, protect the WebSphere MQ data sets with RACF.

You must also control access to all your 'CSQINP' data sets.

RACF authorization of started-task procedures
Some WebSphere MQ data sets are for the exclusive use of the queue manager. If you protect your WebSphere MQ data sets using RACF, you must also authorize the queue manager started-task procedure xxxxMSTR, and the distributed queuing started-task procedure xxxxCHIN, using RACF. To do this, use the STARTED class. Alternatively, you can use the started procedures table (ICHRIN03), but then you must IPL your z/OS system before the changes take effect.

Authorizing access to data sets
The WebSphere MQ data sets should be protected so that no unauthorized user can run a queue manager instance, or gain access to any queue manager data. To do this, use normal z/OS® RACF data set protection.

Parent topic: Security installation tasks

14.9.1.1. RACF authorization of started-task procedures

Some WebSphere MQ data sets are for the exclusive use of the queue manager. If you protect your WebSphere MQ data sets using RACF®, you must also authorize the queue manager started-task procedure xxxxMSTR, and the distributed queuing started-task procedure xxxxCHIN, using RACF. To do this, use the STARTED class. Alternatively, you can use the started procedures table (ICHRIN03), but then you must IPL your z/OS system before the changes take effect.

For more information, see the *z/OS® Security Server RACF System Programmer's Guide*.

The RACF user ID identified must have the required access to the data sets in the started-task procedure. For example, if you associate a queue manager started task procedure called CSQ1MSTR with the RACF user ID QMGRCSQ1, the user ID QMGRCSQ1 must have access to the z/OS resources accessed by the CSQ1 queue manager.

Also, the content of the GROUP field in the user ID of the queue manager must be the same as the content of the GROUP field in the STARTED profile for that queue manager. If the content in each GROUP field does not match then the appropriate user ID is prevented from entering the system. This situation causes WebSphere MQ to run with an undefined user ID and consequently close due to a security violation.

The RACF user IDs associated with the queue manager and channel initiator started task procedures must not have the TRUSTED attribute set.

Parent topic: Setting up WebSphere MQ for z/OS data set security

14.9.1.2. Authorizing access to data sets

The WebSphere® MQ data sets should be protected so that no unauthorized user can run a queue manager instance, or gain access to any queue manager data. To do this, use normal z/OS® RACF® data set protection.

Table 1 summarizes the RACF access that the queue manager started task procedure must have to the different data sets.

Table 1. RACF access to data sets associated with a queue manager

RACF access	Data sets
READ	• thlqual.SCSQAUTH and thlqual.SCSQANLx (where x is the language letter for your national language). • The data sets referred to by CSQINP1, CSQINP2 and CSQXLIB in the queue manager's started task procedure.
UPDATE	• All page sets and log and BSDS data sets.
ALTER	• All archive data sets.

Table 2 summarizes the RACF access that the started task procedure for distributed queuing must have to the different data sets.

Table 2. RACF access to data sets associated with distributed queuing

RACF access	Data sets
READ	• thlqual.SCSQAUTH, thlqual.SCSQANLx (where x is the language letter for your national language), and thlqual.SCSQMVR1. • LE library data sets. • The data sets referred to by CSQXLIB and CSQINPX in the distributed queuing started task procedure.
UPDATE	• Data sets CSQOUTX and CSQSNAP • Dynamic queues SYSTEM.CSQXCMD.*

For more information, see the *z/OS Security Server RACF Security Administrator's Guide*.

Parent topic: Setting up WebSphere MQ for z/OS data set security

14.9.2. Setting up WebSphere MQ for z/OS resource security

There are many types of WebSphere® MQ user. Use RACF® to control their access to WebSphere MQ resources.

The possible users of WebSphere MQ resources, such as queues and channels include the following entities:

- The queue manager itself.
- The channel initiator
- WebSphere MQ administrators, who need to create WebSphere MQ data sets, run utility programs, and so on.
- Application programmers who need to use the WebSphere MQ-supplied copybooks, include data sets, macros, and so on.
- Applications involving one or more of:
  - Batch jobs
  - TSO users
  - CICS® regions
  - IMS™ regions
- Data sets CSQOUTX and CSQSNAP
- Dynamic queues SYSTEM.CSQXCMD.*

For all these potential users, protect the WebSphere MQ resources with RACF. In particular, note that the channel initiator needs access to various resources, as described in Security considerations for the channel initiator on z/OS, and so the user ID under which it runs must be authorized to access these resources.

If you are using a queue-sharing group, the queue manager might issue various commands internally, so the user ID it uses must be authorized to issue such commands. The commands are:

- DEFINE, ALTER, and DELETE for every object that has QSGDISP(GROUP)
- START and STOP CHANNEL for every channel used with CHLDISP(SHARED)

Parent topic: Security installation tasks

14.9.3. Configuring your system to use the Secure Sockets Layer (SSL)

Use this topic as example of how to configure WebSphere® MQ for z/OS® with SSL using RACF® commands.

If you want to use the Secure Sockets Layer (SSL) for channel security, there are a number of tasks you need to perform on your system. (For details on using RACF commands for certificates and key repositories (key rings), see Working with SSL or TLS on z/OS.)

1. Create a key ring in RACF to hold all the keys and certificates for your system, using the RACF RACDCERT command. For example:
   ```
   RACDCERT ID(CHINUSER) ADDRING(QM1RING)
   ```
   The ID should be either the channel initiator address space user ID or the user ID you want to own the key ring if it is to be a shared key ring.

2. Create a digital certificate for each queue manager, using the RACF RACDCERT command.
   The label of the certificate must be of the form `ibmWebSphereMQqmgr-name`, so in this example it is `ibmWebSphereMQQM1`.
   For example:
   ```
   RACDCERT ID(USERID) GENCERT
   SUBJECTSDN(CN('username') O('IBM') OU('departmentname') C('England'))
   WITHLABEL('ibmWebSphereMQQM1')
   ```
3. Connect the certificate in RACF to the key ring, using the RACF RACDCERT command. For example:
   ```
   RACDCERT CONNECT(ID(USERID) LABEL('ibmWebSphereMQQM1') RING(QM1RING))
   CONNECT ID(CHINUSER)
   ```
   You also need to connect any relevant signer certificates (from a Certification Authority) to the key ring. That is, all Certification Authorities for this queue manager's SSL certificate and all Certification Authorities for all SSL certificates that this queue manager communicates with. For example:
   ```
   RACDCERT ID(CHINUSER)
   CONNECT(CERTAUTH LABEL('My CA') RING(QM1RING) USAGE(CERTAUTH))
   ```
4. On each of your queue managers, use the WebSphere MQ ALTER QMGR command to specify the key repository that the queue manager needs to point to. For example, if the key ring is owned by the channel initiator address space:
   ```
   ALTER QMGR SSLKEYR(QM1RING)
   ```
   or if you are using a shared key ring:
   ```
   ALTER QMGR SSLKEYR(userid/QM1RING)
   ```
   where *userid* is the user ID that owns the shared key ring.
5. Certificate Revocation Lists (CRLs) allow the Certification Authorities to revoke certificates that can no longer be trusted. CRLs are stored in LDAP servers. To access this list on the LDAP server, you first need to create an AUTHINFO object of AUTHTYPE CRLLDAP, using the WebSphere MQ DEFINE AUTHINFO command. For example:
   ```
   DEFINE AUTHINFO(LDAP1)
   AUTHTYPE(CRLLDAP)
   CONNAME(ldap.server(389))
   LDAPUSER('')
   LDAPPWD('')
   ```
   In this example, the certificate revocation list is stored in a public area of the LDAP server, so the LDAPUSER and LDAPPWD fields are not necessary.
   Next, put your AUTHINFO object into a namelist, using the WebSphere MQ DEFINE NAMELIST command. For example:
   ```
   DEFINE NAMELIST(LDAPNL) NAMES(LDAP1)
   ```
   Finally, associate the namelist with each queue manager, using the WebSphere MQ ALTER QMGR command. For example:
   ```
   ALTER QMGR SSLCRLNL(LDAPNL)
   ```
6. Set up your queue manager to run SSL calls, using the WebSphere MQ ALTER QMGR command. This defines server subtasks that handle SSL calls only, which leaves the normal dispatchers to continue processing as normal without being affected by any SSL calls. You must have at least two of these subtasks. For example:
   ```
   ALTER QMGR SSLTASKS(8)
   ```
   This change only takes effect when the channel initiator is restarted.
7. Specify the cipher specification to be used for each channel, using the WebSphere MQ DEFINE CHANNEL or ALTER CHANNEL command. For example:
   ```
   ALTER CHANNEL(LDAPCHL)
   CHLTYPE(SDR)
   SSLCIPH(RC4_MD5_US)
   ```
   Both ends of the channel must specify the same cipher specification.

Parent topic: Security installation tasks

14.10. Auditing considerations on z/OS®

The normal RACF® auditing controls are available for conducting a security audit of a queue manager. WebSphere® MQ does not gather any security statistics of its own. The only statistics are those that can be created by auditing.

RACF auditing can be based upon:

- User IDs
- Resource classes
- Profiles

For more details, see the *z/OS Security Server RACF Auditor's Guide*.

Note: Auditing degrades performance; the more auditing you implement, the more performance is degraded. This is also a consideration for the use of the RACF WARNING option.

Auditing RESLEVEL
Use the RESAUDIT system parameter to control the production of RESLEVEL audit records. RACF GENERAL audit records are produced.

Parent topic: Setting up security on z/OS

14.10.1. Auditing RESLEVEL

Use the RESAUDIT system parameter to control the production of RESLEVEL audit records. RACF® GENERAL audit records are produced.

Produce RESLEVEL audit records by setting the RESAUDIT system parameter to YES. If the RESAUDIT parameter is set to NO, audit records are not produced. For more details about setting this parameter, see Using CSQ6SYSP.

If RESAUDIT is set to YES, no normal RACF audit records are taken when the RESLEVEL check is made to see what access an address space user ID has to the hlq.RESLEVEL profile. Instead, WebSphere® MQ requests that RACF create a GENERAL audit record (event number 27). These checks are only carried out at connect time, so the performance cost is minimal.

You can report the WebSphere MQ general audit records using the RACF report writer (RACFRW). You could use the following RACFRW commands to report the RESLEVEL access:

```
RACFRW
SELECT PROCESS
EVENT GENERAL
LIST
END
```

A sample report from RACFRW, excluding the `Date`, `Time`, and `SYSID` fields, is shown in Figure 1.
Figure 1. Sample output from RACFRW showing RESLEVEL general audit records

```
          RACF REPORT - LISTING OF PROCESS RECORDS                                     PAGE   4
                                 E
                                 V  Q
                                 E  U
   *JOB/USER *STEP/  --TERMINAL--  N  A
      NAME    GROUP     ID    LVL  T  L

    WS21B    MQMGRP IGJZM000   0   27 0  JOBID=(WS21B 05.111 09:44:57),USERDATA=()
       TRUSTED USER                      AUTH=(NONE),REASON=(NONE)
                                  SESSION=TSOLOGON,TERMINAL=IGJZM000,
                                  LOGSTR='CSQH RESLEVEL CHECK PERFORMED AGAINST PROFILE(QM66.RESLEVEL),
                                  CLASS(MQADMIN), ACCESS EQUATES TO (CONTROL)',RESULT=SUCCESS,MQADMIN
```

From checking the LOGSTR data in the output above, you can see that TSO user WS21B has CONTROL access to QM66.RESLEVEL. This means that all resource security checks are bypassed when user WS21B access QM66 resources.

For more information about using RACFRW, see the *z/OS® Security Server RACF Auditor's Guide*.

Parent topic: Auditing considerations on z/OS

14.11. Customizing security

If you want to change the way WebSphere® MQ security operates, you must do this through the SAF exit (ICHRFR00), or exits in your external security manager.

To find out more about RACF® exits, see the *z/OS® Security Server RACROUTE Macro Reference* manual.

Note: Because WebSphere MQ optimizes calls to the ESM, RACROUTE requests might not be made on, for example, every open for a particular queue by a particular user.

Parent topic: Setting up security on z/OS

14.12. Security violation messages on z/OS®

A security violation is indicated by the return code MQRC_NOT_AUTHORIZED in an application program or by a message in the job log.

A return code of MQRC_NOT_AUTHORIZED can be returned to an application program for the following reasons:

- A user is not allowed to connect to the queue manager. In this case, you get an ICH408I message in the Batch/TSO, CICS®, or IMS™ job log.
- A user sign-on to the queue manager has failed because, for example, the job user ID is not valid or appropriate, or the task user ID or alternate user ID is not valid. One or more of these user IDs might not be valid because they have been revoked or deleted. In this case, you get an ICHxxxx message and possibly an IRRxxxx message in the queue manager job log giving the reason for the sign-on failure. For example:

```
ICH408I USER(NOTDFND ) GROUP(        ) NAME(???                )
  LOGON/JOB INITIATION - USER AT TERMINAL          NOT RACF-DEFINED
IRR012I  VERIFICATION FAILED. USER PROFILE NOT FOUND
```

- An alternate user has been requested, but the job or task user ID does not have access to the alternate user ID. For this failure, you get a violation message in the job log of the relevant queue manager.
- A context option has been used or is implied by opening a transmission queue for output, but the job user ID or, where applicable, the task or alternate user ID does not have access to the context option. In this case, a violation message is put in the job log of the relevant queue manager.
- An unauthorized user has attempted to access a secured queue manager object, for example, a queue. In this case, an ICH408I message for the violation is put in the job log of the relevant queue manager. This violation might be due to the job or, when applicable, the task or alternate user ID.

Violation messages for command security and command resource security can also be found in the job log of the queue manager.

If the ICH408I violation message shows the queue manager jobname rather than a user ID, this is normally the result of a blank alternate user ID being specified. For example:

```
ICH408I JOB(MQS1MSTR) STEP(MQS1MSTR)
          MQS1.PAYROLL.REQUEST CL(MQQUEUE)
          INSUFFICIENT ACCESS AUTHORITY
          ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

You can find out who is allowed to use blank alternate user IDs by checking the access list of the MQADMIN profile hlq.ALTERNATE.USER.-BLANK-.

An ICH408I violation message can also be generated by:

- A command being sent to the system-command input queue without context. User-written programs that write to the system-command input queue should always use a context option. For more information, see Profiles for context security.

- When the job accessing the WebSphere® MQ resource does not have a user ID associated with it, or when a WebSphere MQ adapter cannot extract the user ID from the adapter environment.

Violation messages might also be issued if you are using both queue-sharing group and queue manager level security. You might get messages indicating that no profile has been found at queue manager level, but still be granted access because of a queue-sharing group level profile.

```
ICH408I JOB(MQS1MSTR) STEP(MQS1MSTR)
          MQS1.PAYROLL.REQUEST CL(MQQUEUE)
          PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
          ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE    )
```

Parent topic: Setting up security on z/OS

14.13. What to do if access is allowed or disallowed incorrectly

In addition to the steps detailed in the *z/OS® Security Server RACF® Security Administrator's Guide*, use this checklist if access to a resource appears to be incorrectly controlled.

- Are the switch profiles correctly set?
  - Is RACF active?
  - Are the WebSphere® MQ RACF classes installed and active?
    Use the RACF command, SETROPTS LIST, to check this.
  - Use the WebSphere MQ DISPLAY SECURITY command to display the current switch status from the queue manager.
  - Check the switch profiles in the MQADMIN class.
    Use the RACF commands, SEARCH and RLIST, for this.
  - Recheck the RACF switch profiles by issuing the WebSphere MQ REFRESH SECURITY(MQADMIN) command.
- Has the RACF resource profile changed? For example, has universal access on the profile changed or has the access list of the profile changed?
  - Is the profile generic?
    If it is, issue the RACF command, SETROPTS GENERIC(classname) REFRESH.
  - Have you refreshed the security on this queue manager?
    If required, issue the RACF command SETROPTS RACLIST(classname) REFRESH.
    If required, issue the WebSphere MQ REFRESH SECURITY(*) command.
- Has the RACF definition of the user changed? For example, has the user been connected to a new group or has the user access authority been revoked?
  - Have you reverified the user by issuing the WebSphere MQ RVERIFY SECURITY(userid) command?
- Are security checks being bypassed due to RESLEVEL?
  - Check the connecting user ID's access to the RESLEVEL profile. Use the RACF audit records to determine what the RESLEVEL is set to.
  - For channels, remember that the access level that the channel initiator's userid has to RESLEVEL is inherited by all channels, so an access level, such as ALTER, that causes all checks to be bypassed causes security checks to be bypassed for all channels.
  - If you are running from CICS®, check the transaction's RESSEC setting.
  - If RESLEVEL has been changed while a user is connected, they must disconnect and reconnect before the new RESLEVEL setting takes effect.
- Are you using queue-sharing groups?
  - If you are using both queue-sharing group and queue manager level security, check that you have defined all the correct profiles. If queue manager profile is not defined, a message is sent to the log stating that the profile was not found.
  - Have you used a combination of switch settings that is not valid so that full security checking has been set on?
  - Do you need to define security switches to override some of the queue-sharing group settings for your queue manager?
  - Is a queue manager level profile taking precedence over a queue-sharing group level profile?

Parent topic: Setting up security on z/OS

14.14. Security considerations for the channel initiator on z/OS

If you are using resource security in a distributed queuing environment, the Channel initiator address space needs appropriate access to various WebSphere® MQ resources.

If you are using resource security, consider the following points if you are using distributed queuing:

System queues
: The channel initiator address space needs RACF® UPDATE access to the system queues listed at System queue security, and to all the user destination queues and the dead-letter queue (but see Dead-letter queue security).

Transmission queues
: The channel initiator address space needs ALTER access to all the user transmission queues.

Context security
: The channel user ID (and the MCA user ID if one has been specified) need RACF CONTROL access to the hlq.CONTEXT.queuename profiles in the MQADMIN class. Depending on the RESLEVEL profile, the network-received user ID might also need CONTROL access to these profiles.

  All channels need CONTROL access to the MQADMIN hlq.CONTEXT. dead-letter-queue profile. All channels (whether initiating or responding) can generate reports, and consequently they need CONTROL access to the hlq.CONTEXT.reply-q profile.

SENDER, CLUSSDR, and SERVER channels need CONTROL access to the hlq.CONTEXT.xmit-queue-name profiles since messages can be put onto the transmission queue to wake up the channel to end gracefully.

Note: If the channel user ID, or a RACF group to which the channel user ID is connected, has CONTROL or ALTER access to the hlq.RESLEVEL, then there are no resource checks for the channel initiator or any of its channels.

See Profiles for context security RESLEVEL and the channel initiator connection and User IDs for security checking for more information.

CSQINPX

If you are using the CSQINPX input data set, the channel initiator also needs READ access to CSQINPX, and UPDATE access to data set CSQOUTX and dynamic queues SYSTEM.CSQXCMD.*.

Connection security

The channel initiator address space connection requests use a connection type of CHIN, for which appropriate access security must be set, see Connection security profiles for the channel initiator.

Data sets

The channel initiator address space needs appropriate access to queue manager data sets, see Authorizing access to data sets.

Commands

The distributed queuing commands (for example, DEFINE CHANNEL, START CHINIT, START LISTENER, and other channel commands) must have appropriate command security set, see Table 1.

If you are using a queue-sharing group, the channel initiator might issue various commands internally, so the user ID it uses must be authorized to issue such commands. These commands are START and STOP CHANNEL for every channel used with CHLDISP(SHARED).

Channel security

Channels, particularly receivers and server-connections, need appropriate security to be set up; see User IDs for security checking for more information.

You can also use the Secure Sockets Layer (SSL) protocol to provide security on channels. See the WebSphere MQ Security documentation for a detailed description of SSL.

See also the WebSphere MQ Clients manual for information about server-connection security.

User IDs

The user IDs described in User IDs used by the channel initiator and User IDs used by the intra-group queuing agent need the following access:

- RACF UPDATE access to the appropriate destination queues and the dead-letter queue
- RACF CONTROL access to the `hlq.CONTEXT.queuename` profile if context checking is performed at the receiver
- Appropriate access to the hlq.ALTERNATE.USER.userid profiles they might need to use.
- For clients, the appropriate RACF access to the resources to be used.

APPC security

Set appropriate APPC security if you are using the LU 6.2 transmission protocol. (Use the APPCLU RACF class for example.) For information about setting up security for APPC, see the following manuals:

- *z/OS V1R2.0 MVS™ Planning: APPC Management*
- *Multiplatform APPC Configuration Guide*, an IBM® Redbooks® publication

Outbound transmissions use the "SECURITY(SAME)" APPC option. As a result, the user ID of the channel initiator address space and its default profile (RACF GROUP) are flowed across the network to the receiver with an indicator that the user ID has already been verified (ALREADYV).

If the receiving side is also z/OS, the user ID and profile are verified by APPC and the user ID is presented to the receiver channel and used as the channel user ID.

In an environment where the queue manager is using APPC to communicate with another queue manager on the same or another z/OS system, you need to ensure that either:

- The VTAM® definition for the communicating LU specifies SETACPT(ALREADYV)
- There is a RACF APPCLU profile for the connection between LUs that specifies CONVSEC(ALREADYV)

Changing security settings

If the RACF access level that either the channel user ID or MCA user ID has to a destination queue is changed, this change takes effect only for new object handles (that is, new **MQOPEN**s) for the destination queue. The times when MCAs open and close queues is variable; if a channel is already running when such an access change is made, the MCA can continue to put messages on the destination queue using the existing security access of the user IDs rather than the updated security access. Stopping and restarting the channels to enforce the updated access level avoids this scenario.

Automatic restart

If you are using the z/OS Automatic Restart Manager (ARM) to restart the channel initiator, the user ID associated with the XCFAS address space must be authorized to issue the WebSphere MQ START CHINIT command.

Parent topic: Setting up security on z/OS

14.15. Security in queue manager clusters on z/OS®

Security considerations for clusters are like those for queue managers and channels that are not clustered. The channel initiator needs access to some additional system queues, and some additional commands need appropriate security set.

You can use the MCA user ID and security exits to authenticate cluster channels (as with conventional channels). The security exit on the cluster-receiver channel must check that the queue manager is permitted access to the server queue manager's clusters. You can start to use WebSphere® MQ cluster support without having to change your existing queue access security, however you must allow other queue managers in the cluster to write to the SYSTEM.CLUSTER.COMMAND.QUEUE if they are to join the cluster.

WebSphere MQ cluster support does not provide a mechanism to limit a member of a cluster to the client role only. As a result, you must be sure that you trust any queue managers that you allow into the cluster. If any queue manager in the cluster creates a queue with a particular name, it can receive messages for that queue, regardless of whether the application putting messages to that queue intended this or not.

To restrict the membership of a cluster, take the same action that you would take to prevent queue managers connecting to receiver channels. You can achieve this by writing a security exit program on the receiver channel or by writing an exit program to prevent unauthorized queue managers from writing to the SYSTEM.CLUSTER.COMMAND.QUEUE.

Note: It is not advisable to permit applications to open the SYSTEM.CLUSTER.TRANSMIT.QUEUE directly, just as it is not advisable to permit an application to open any other transmission queue directly.

If you are using resource security, consider the following points in addition to the considerations discussed in Security considerations for the channel initiator on z/OS:

System queues

The channel initiator needs RACF® ALTER access to the following system queues:

- SYSTEM.CLUSTER.COMMAND QUEUE
- SYSTEM.CLUSTER.TRANSMIT.QUEUE.

and UPDATE access to SYSTEM.CLUSTER.REPOSITORY.QUEUE

It also needs READ access to any namelists used for clustering.

Commands

Set appropriate command security (as described in Table 1) for the cluster support commands (REFRESH and RESET CLUSTER, SUSPEND and RESUME QMGR.

Parent topic: Setting up security on z/OS

14.16. Security considerations for using WebSphere MQ with CICS

The CICS® adapter provides information to WebSphere® MQ for use in security.

The CICS adapter provides the following information to WebSphere MQ specifically for use in WebSphere MQ security:

- Whether CICS resource-level security is active for this transaction—as specified on the RESSEC or RSLC operand of the RDO TRANSACTION definition.
- User IDs.
  For terminal tasks where a user has not signed on, the user ID is the CICS user ID associated with the terminal and is either:
  - The default CICS user ID as specified on the CICS parameter DFLTUSER SIT
  - A preset security user ID specified on the terminal definition

  For non-terminal tasks, the CICS adapter tries to get a user ID with an EXEC CICS ASSIGN command. If this is unsuccessful, the adapter tries to get the user ID using EXEC CICS INQUIRE TASK. If security is active in CICS, and the non-terminal attached transaction is defined with CMDSEC(YES), the CICS adapter passes a user ID of blanks to WebSphere MQ.

For more information about RACF® security management in the CICS environment, see the *CICS Transaction Server for z/OS® V3.2 RACF Security Guide*.

Controlling the security of CICS transactions supplied by WebSphere MQ
If you want a user to administer the CICS adapter, grant the user authorization to certain transactions.

The CICS adapter user ID
The user ID associated with the CICS adapter is that of the WebSphere MQ-supplied task initiator transaction, CKTI. This has a number of implications.

Security considerations for the CICS bridge
When you run the CICS bridge, you can specify the level of authentication you want to take place.

Parent topic: Setting up security on z/OS

14.16.1. Controlling the security of CICS transactions supplied by WebSphere MQ

If you want a user to administer the CICS® adapter, grant the user authorization to certain transactions.

The CKTI and CKAM transactions are designed to be run without a terminal; no user should have access to these transactions. These transactions are examples of what the *CICS RACF Security Guide* calls “category 1 transactions”. For information about how to set up these transactions in CICS and RACF®, see the information about category 1 transactions in the *CICS RACF Security Guide*.

If you want a user to administer the CICS adapter, you must grant the user authorization to these transactions:

CKQC	Controls the CICS adapter functions
CKBM	Controls the CICS adapter functions
CKRT	Controls the CICS adapter functions
CKCN	Connect
CKSD	Disconnect
CKRS	Statistics
CKDP	Full screen display
CKDL	Line mode display
CKSQ	CKTI START/STOP

If required, you can restrict access to specific functions of the adapter. For example, if you want to allow users to display the status of the adapter through the full screen interface, but nothing else, give them access to CKQC, CKBM, CKRT, and CKDP only.

Define these transactions to CICS with RESSEC(NO) and CMDSEC(NO). For more details, see the *CICS RACF Security Guide*.

Parent topic: Security considerations for using WebSphere MQ with CICS

14.16.2. The CICS adapter user ID

The user ID associated with the CICS® adapter is that of the WebSphere® MQ-supplied task initiator transaction, CKTI. This has a number of implications.

User ID checking for WebSphere MQ resources during PLTPI and PLTSD

If a WebSphere MQ resource is accessed during the CICS PLTPI phase, the user ID passed to WebSphere MQ is blanks. If a WebSphere MQ resource is accessed during the CICS PLTSD phase, the user ID passed to WebSphere MQ is the user ID associated with the shutdown transaction.

If CKTI is started during the CICS PLTPI phase, the user ID of the CKTI task is the CICS sysidnt. This means that a user ID with the same name as the CICS sysidnt must be defined and given access to the required WebSphere MQ resources, for example, initiation queues.

Terminal user IDs

If CKTI is started from a terminal from the CKQC transaction or a user-written program that links to CSQCSSQ, the user ID that CKTI uses is the same as the user ID of the terminal that started CKTI.

Automating starting of CKTI

To automate the starting of CKTIs under a specific user ID, you can use an automation product, for example, Tivoli® NetView® for z/OS®. You can use this to sign on a CICS console and issue the STARTCKTI command.

You can also use preset security sequential terminals, which have been defined to emulate a CRLP terminal, with the sequential terminal input containing the CKQC STARTCKTI command.

However, when the CICS adapter alert monitor reconnects CICS to WebSphere MQ, after, for example, a WebSphere MQ restart, only the CKTI specified at the initial WebSphere MQ connection is restarted. You must automate starting any extra CKTIs yourself.

Propagating the CKTI user ID to other CICS transactions

If CKTI starts other CICS transactions, for example, message channel agents (MCAs) or user-written CICS applications, the user ID of CKTI is propagated to these applications. For example, if CKTI is running under user ID *USERNAME* and a trigger event occurs that requires the sender MCA transaction, *TRNA*, to be started, the *TRNA* transaction also runs under user ID *USERNAME*. Therefore user ID *USERNAME* must have access to the required transmission queue.

Parent topic: Security considerations for using WebSphere MQ with CICS

14.16.3. Security considerations for the CICS bridge

When you run the CICS® bridge, you can specify the level of authentication you want to take place.

If requested, the bridge checks the user ID and password extracted from the WebSphere® MQ request message before running the CICS program named in the request message. The queue manager uses the external security manager (ESM) (for example RACF®) to do authentication. Therefore user IDs in the request message have to be defined to the ESM.

Note:

1. If you have not specified a user ID in the message descriptor (MQMD) or password in the CICS bridge header (MQCIH) of a message, the bridge task runs with the LOCAL level of authentication, even if you started the bridge monitor with a different authentication option.
2. The options that include password (or PassTicket) validation require an MQCIH to be provided. See MQCIH – CICS bridge header for more information about the MQCIH header.
3. PassTicket validation is performed using WebSphere MQ services, not EXEC CICS VERIFY, as the CICS service does not allow you to specify an APPLID.

The level of authentication you can use is described below:

LOCAL

This is the default. CICS programs run by the bridge task are started with the CICS DFLTUSER user ID, therefore run with the authority associated with this user ID. There is no checking of user IDs or passwords. If a CICS program is run that tries to access protected resources, it will probably fail.

IDENTIFY

When you start the monitor task with the IDENTIFY authentication option, the bridge task is started with the user ID specified in the message (MQMD). CICS programs run by the bridge run with the user ID from the MQMD. There is no password checking, the user ID is treated as trusted.

VERIFY_UOW

If MQMD.PutApplType is set to MQAT_NO_CONTEXT, the CICS DFLTUSER user ID is used, as for the LOCAL authentication option. Otherwise, the bridge monitor checks the user ID (in the MQMD) and password (in the CIH) before starting the bridge task, and CICS programs run by the bridge run with the user ID extracted from the MQMD. If the user ID or password is invalid, the request fails with return code MQCRC_SECURITY_ERROR. Subsequent messages processed by this transaction are not checked.

Bridge tasks that cannot be started because of a security failure are retried if ROUTEMEM=N (the default) is specified. However, if ROUTEMEM=Y is specified the bridge messages are put to the dead-letter queue.

VERIFY_ALL

This is the same as VERIFY_UOW except that the bridge task checks the user ID and password in **every** message. This is not applicable for 3270 transactions when using CICS earlier than CICS Transaction Server Version 2 Release 2.

A PassTicket can be used in place of a password to avoid the need to flow passwords in messages (see Security Server RACF System Programmer's Guide). When generating a PassTicket an APPLID must be specified. If you are using a single bridge monitor, the APPLID is the CICS APPLID unless a different value was specified when the bridge was started. If you are using multiple bridge monitors for a queue, you must specify the APPLID to be used via the PASSTKTA=*applid* parameter at bridge startup.

If you have not specified a user ID in a message, or you have not provided a password, the CICS program started by the CICS bridge runs with the user ID set to the user ID used to start the bridge monitor, regardless of the option requested. If you want more than one level of authentication checking performed, run a monitor task for each level you need.

When a CICS DPL request is read by the bridge monitor it starts the transaction specified in the CICS bridge header (MQCIH) or, if this is blank, transaction CKBP. The user IDs under which the bridge monitor runs must have authority to start the various transactions that might be requested. The default transaction ID for the CICS bridge monitor is CKBR but you can change this or define additional transaction IDs if you want more granular access to queues and transactions. You can use CICS surrogate security to restrict which user ID and transaction combinations a bridge monitor transaction and user ID can start.

Table 1 and Table 2 summarize the level of authority of the bridge monitor and the bridge tasks, and the use of the MQMD user ID.

Table 1. CICS bridge monitor security

Monitor started by	At a signed on terminal	Monitor authority
From a terminal or EXEC CICS LINK within a program	Yes	Signed on user ID
From a terminal or EXEC CICS LINK within a program	No	CICS default user ID
EXEC CICS START with user ID	–	User ID from START
EXEC CICS START without user ID	–	CICS default user ID
The WebSphere MQ trigger monitor CKTI	–	CICS default user ID

Table 2. CICS bridge task security

AUTH	Bridge task authority
LOCAL	CICS default user ID
IDENTIFY	MQMD UserIdentifier
VERIFY_UOW	MQMD UserIdentifier
VERIFY_ALL	MQMD UserIdentifier

The options IDENTIFY, VERIFY_UOW, and VERIFY_ALL need the user ID of the bridge monitor defined to RACF as a surrogate of all the user IDs used in request messages. This is in addition to the user ID in the message being defined to RACF. (A surrogate user is one who has the authority to start work on behalf of another user, without knowing the other user's password.)

For more information on surrogate user security, see the *CICS RACF Security Guide*.

Note: When IDENTIFY security is being used, you might see abend AICO for CKBP if you try to run with a user ID that has been revoked. The error reply will have return code MQCRC_BRIDGE_ERROR with reason MQFB_CICS_BRIDGE_FAILURE.

Authority for the CICS bridge
Components of the bridge need authority to either put to or get from the various WebSphere MQ queues.

Parent topic: Security considerations for using WebSphere MQ with CICS

14.16.3.1. Authority for the CICS bridge

Components of the bridge need authority to either put to or get from the various WebSphere® MQ queues.

In summary, the required authority is as follows:

- The monitor and all bridge tasks need authority to get messages from the bridge request queue.
- A bridge task needs authority to put messages to its own reply-to queue.
- To ensure that any error replies are received, the monitor needs authority to put messages to all reply-to queues.
- Bridge tasks need authority to put messages to the dead-letter queue.
- The monitor needs authority to put messages to the dead-letter queue, unless you want the bridge to stop if an error occurs.
- The monitor and all bridge tasks need authority to put messages to the backout requeue queue, if one is defined

See Table 1 to determine the correlation between user IDs and authority.

Parent topic: Security considerations for the CICS bridge

14.17. Security considerations for using WebSphere MQ with IMS

Use this topic to plan your security requirements when you use WebSphere® MQ with IMS™.

Using the OPERCMDS class

If you are using RACF® to protect resources in the OPERCMDS class, ensure that the userid associated with your WebSphere MQ queue manager address space has authority to issue the MODIFY command to any IMS system to which it can connect.

Security considerations for the IMS bridge

There are four aspects that you must consider when deciding your security requirements for the IMS bridge, these are:

- What security authorization is needed to connect WebSphere MQ to IMS
- How much security checking is performed on applications using the bridge to access IMS
- Which IMS resources these applications are allowed to use
- What authority is to be used for messages that are put and got by the bridge

When you define your security requirements for the IMS bridge you must consider the following:

- Messages passing across the bridge might have originated from applications on platforms that do not offer strong security features
- Messages passing across the bridge might have originated from applications that are not controlled by the same enterprise or organization

Security considerations for connecting to IMS
Grant the user ID of the WebSphere MQ queue manager address space access to the OTMA group.

Application access control for the IMS bridge
Define a RACF profile in the FACILITY class for each IMS system. Grant an appropriate level of access to the WebSphere MQ queue manager user ID.

Security checking on IMS
Messages that pass across the bridge contain security information. The security checks made depend on the setting of the IMS command /SECURE OTMA.

Security checking done by the IMS bridge
Different authorities are used depending on the action being performed.

Using RACF PassTickets in the IMS header
You can use a PassTicket in place of a password in the IMS header.

Parent topic: Setting up security on z/OS

14.17.1. Security considerations for connecting to IMS

Grant the user ID of the WebSphere® MQ queue manager address space access to the OTMA group.

The IMS™ bridge is an OTMA client. The connection to IMS operates under the user ID of the WebSphere MQ queue manager address space. This is normally defined as a member of the started task group. This user ID must be granted access to the OTMA group (unless the /SECURE OTMA setting is NONE).

To do this, define the following profile in the FACILITY class:

```
IMSXCF.xcfgname.mqxcfmname
```

Where `xcfgname` is the XCF group name and `mqxcfmname` is the XCF member name of WebSphere MQ.

You must give your WebSphere MQ queue manager user ID read access to this profile.

Note:

1. If you change the authorities in the FACILITY class, you must issue the RACF® command SETROPTS RACLIST(FACILITY) REFRESH to activate the changes.
2. If profile hlq.NO.SUBSYS.SECURITY exists in the MQADMIN class, no user ID is passed to IMS and the connection fails unless the /SECURE OTMA setting is NONE.

Parent topic: Security considerations for using WebSphere MQ with IMS

14.17.2. Application access control for the IMS bridge

Define a RACF® profile in the FACILITY class for each IMS™ system. Grant an appropriate level of access to the WebSphere® MQ queue manager user ID.

For each IMS system that the IMS bridge connects to, you can define the following RACF profile in the FACILITY class to determine how much security checking is performed for each message passed to the IMS system.

```
IMSXCF.xcfgname.imsxcfmname
```

Where `xcfgname` is the XCF group name and `imsxcfmname` is the XCF member name for IMS. (You need to define a separate profile for each IMS system.)

The access level you allow for the WebSphere MQ queue manager user ID in this profile is returned to WebSphere MQ when the IMS bridge connects to IMS, and indicates the level of security that is required on subsequent transactions. For subsequent transactions, WebSphere MQ requests the appropriate services from RACF and, where the user ID is authorized, passes the message to IMS.

OTMA does not support the IMS /SIGN command; however, WebSphere MQ allows you to set the access checking for each message to enable implementation of the necessary level of control.

The following access level information can be returned:

NONE or NO PROFILE FOUND

These values indicate that maximum security is required, that is, authentication is required for every transaction. A check is made to verify that the user ID specified in the *UserIdentifier* field of the MQMD structure, and the password or PassTicket in the *Authenticator* field of the MQIIH structure are known to RACF, and are a valid combination. A UTOKEN is created with a password or PassTicket, and passed to IMS; the UTOKEN is not cached.

Note: If profile hlq.NO.SUBSYS.SECURITY exists in the MQADMIN class, this level of security overrides whatever is defined in the profile.

READ

This value indicates that the same authentication is to be performed as for `NONE` under the following circumstances:

- The first time that a specific user ID is encountered
- When the user ID has been encountered before but the cached UTOKEN was not created with a password or PassTicket

WebSphere MQ requests a UTOKEN if required, and passes it to IMS.

Note: If a request to reverify security has been acted on, all cached information is lost and a UTOKEN is requested the first time each user ID is later encountered.

UPDATE

A check is made that the user ID in the *UserIdentifier* field of the MQMD structure is known to RACF.

A UTOKEN is built and passed to IMS; the UTOKEN is cached.

CONTROL/ALTER

These values indicate that no security UTOKENs need to be provided for any user IDs for this IMS system. (You would probably only use this option for development and test systems.)

Note:

1. This access is defined when WebSphere MQ connects to IMS, and lasts for the duration of the connection. To change the security level, the access to the security profile must be changed and then the bridge stopped and restarted (for example, by stopping and restarting OTMA).
2. If you change the authorities in the FACILITY class, you must issue the RACF command SETROPTS RACLIST(FACILITY) REFRESH to activate the changes.
3. You can use a password or a PassTicket, but you must remember that the IMS bridge does not encrypt data. For information about using PassTickets, see Using RACF PassTickets in the IMS header.
4. Some of these results might be affected by security settings in IMS, using the /SECURE OTMA command.
5. Cached UTOKEN information is held for the duration defined by the INTERVAL and TIMEOUT parameters of the WebSphere MQ ALTER SECURITY command.

Parent topic: Security considerations for using WebSphere MQ with IMS

14.17.3. Security checking on IMS

Messages that pass across the bridge contain security information. The security checks made depend on the setting of the IMS™ command /SECURE OTMA.

Each WebSphere® MQ message that passes across the bridge contains the following security information:

- A user ID contained in the *UserIdentifier* field of the MQMD structure
- The security scope contained in the *SecurityScope* field of the MQIIH structure (if the MQIIH structure is present)
- A UTOKEN (unless the WebSphere MQ sub system has CONTROL or ALTER access to the relevant `IMSXCF.xcfgname.imsxcfmname` profile)

The security checks made depend on the setting of the IMS command /SECURE OTMA, as follows:

/SECURE OTMA NONE

No security checks are made for the transaction.

/SECURE OTMA CHECK

The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking.

An ACEE (Accessor Environment Element) is built in the IMS control region.

/SECURE OTMA FULL

The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking.

An ACEE is built in the IMS dependent region as well as the IMS control region.

/SECURE OTMA PROFILE

The *UserIdentifier* field of the MQMD structure is passed to IMS for transaction or command authority checking

The *SecurityScope* field in the MQIIH structure is used to determine whether to build an ACEE in the IMS dependent region as well as the control region.

Note:

1. If you change the authorities in the TIMS or CIMS class, or the associated group classes GIMS or DIMS, you must issue the following IMS commands to activate the changes:
   - /MODIFY PREPARE RACF®
   - /MODIFY COMMIT
2. If you do not use /SECURE OTMA PROFILE, any value specified in the *SecurityScope* field of the MQIIH structure is ignored.

Parent topic: Security considerations for using WebSphere MQ with IMS

14.17.4. Security checking done by the IMS bridge

Different authorities are used depending on the action being performed.

When the bridge puts or gets a message, the following authorities are used:

Getting a message from the bridge queue
No security checks are performed.
Putting an exception, or COA report message
Uses the authority of the user ID in the *UserIdentifier* field of the MQMD structure.
Putting a reply message
Uses the authority of the user ID in the *UserIdentifier* field of the MQMD structure of the original message
Putting a message to the dead-letter queue
No security checks are performed.

Note:

1. If you change the WebSphere® MQ class profiles, you must issue the WebSphere MQ REFRESH SECURITY(*) command to activate the changes.
2. If you change the authority of a user, you must issue the MQSC RVERIFY SECURITY command to activate the change.

Parent topic: Security considerations for using WebSphere MQ with IMS

14.17.5. Using RACF PassTickets in the IMS header

You can use a PassTicket in place of a password in the IMS header.

If you want to use a PassTicket instead of a password in the IMS™ header (MQIIH), specify the application name against which the PassTicket will be validated in the PASSTKTA attribute of the STGCLASS definition of the IMS Bridge queue to which the message will be routed.

If the PASSTKTA value is left blank, you must arrange to have a PassTicket generated. The application name in this case must be of the form MVSxxxx, where xxxx is the SMFID of the z/OS® system on which the target queue manager runs.

A PassTicket is built from a user ID, the target application name, and a secret key. It is an 8-byte value containing uppercase alphabetic and numeric characters. It can be used only once, and is valid for a 20 minute period. If a PassTicket is generated by a local RACF® system, RACF only checks that the profile exists and not that the user has authority against the profile. If the PassTicket was generated on a remote system, RACF validates the access of the userid to the profile. For full information about PassTickets, see the *z/OS SecureWay™ Security Server RACF Security Administrator's Guide*.

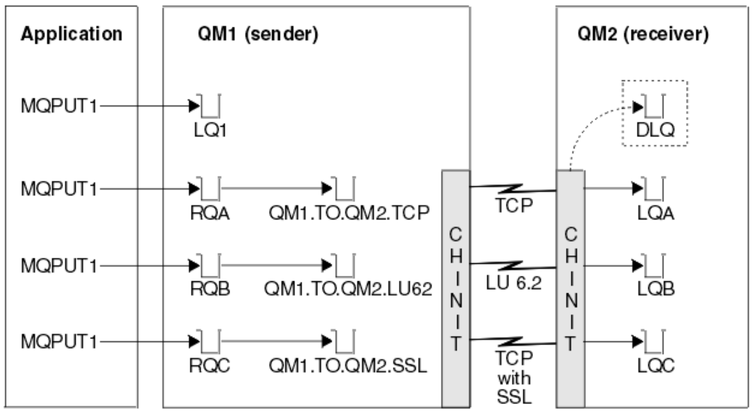
PassTickets in IMS headers are given to RACF by WebSphere® MQ, not IMS.

Parent topic: Security considerations for using WebSphere MQ with IMS

14.18. Security scenario: two queue managers on z/OS®

In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The TCP channels can either use SSL or not. The application could be a batch application or a CICS® application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

This is illustrated in Figure 1.
Figure 1. Example security scenario



The following assumptions are made about the queue managers:

- All the required WebSphere® MQ definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.
If they have not, you need the appropriate access authority to the commands needed to define these objects.
- All the RACF® profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.
If they have not, you need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You also need the appropriate authority to issue the MQSC security commands to start using the new security profiles.
- All digital certificates required have been created and connected to key rings. The digital certificate sent by QM1 as part of the SSL handshake is recognized by RACF on QM2's system, either because it is also installed in that RACF, or because a matching Certificate Name File (CNF) filter exists.

Parent topic: Setting up security on z/OS

14.18.1. Security switch settings for two-queue-manager scenario

Switch settings and RACF® profiles.

The following security switches are set for both queue managers:

- Subsystem security on
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn off process and namelist security:

```
QM1.NO.PROCESS.CHECKS
QM1.NO.NLIST.CHECKS
QM2.NO.PROCESS.CHECKS
QM2.NO.NLIST.CHECKS
```

Parent topic: Security scenario: two queue managers on z/OS

14.18.2. Queue manager QM1 in two-queue-manager scenario

Queues and channels for QM1.

The following queues are defined on queue manager QM1:

LQ1
A local queue.

RQA
A remote queue definition, with the following attributes:
- RNAME(LQA)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

RQB
A remote queue definition, with the following attributes:
- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

RQC
A remote queue definition, with the following attributes:
- RNAME(LQC)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.SSL)

QM1.TO.QM2.TCP
A transmission queue.

QM1.TO.QM2.LU62
A transmission queue.

QM1.TO.QM2.SSL
A transmission queue.

The following channels are defined on QM1:

QM1.TO.QM2.TCP
A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

QM1.TO.QM2.LU62
A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See Security considerations for the channel initiator on z/OS for information about setting up APPC security.)

QM1.TO.QM2.SSL
A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.SSL)
- CONNAME(QM2TCP)
- SSLCIPH(RC4_MD5_EXPORT)

Parent topic: Security scenario: two queue managers on z/OS

14.18.3. Queue manager QM2 in two-queue-manager scenario

Queues and channels for QM2.

The following queues have been defined on queue manager QM2:

LQA
A local queue.

LQB
A local queue.

LQC
A local queue.

DLQ
A local queue that is used as the dead-letter queue.

The following channels have been defined on QM2:

QM1.TO.QM2.TCP
A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

QM1.TO.QM2.LU62
A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See Security considerations for the channel initiator on z/OS for information about setting up APPC security.)

QM1.TO.QM2.SSL
A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCASSL)
- SSLCIPH(RC4_MD5_EXPORT)

Parent topic: Security scenario: two queue managers on z/OS

14.18.4. User IDs used in two-queue-manager scenario

Explanation of the user IDs in the scenario.

The following user IDs are used:

BATCHID
Batch application (Job or TSO ID)

MSGUSR
UserIdentifier in MQMD (context user ID)

MOVER1
QM1 channel initiator address space user ID

MOVER2
QM2 channel initiator address space user ID

MCATCP
MCAUSER specified on the TCP/IP without SSL receiver channel definition

MCALU62
MCAUSER specified on the LU 6.2 receiver channel definition

MCASSL
MCAUSER specified on the TCP/IP with SSL receiver channel definition

CICSAD1
CICS® address space ID

CICSTX1
CICS task user ID

CERTID
The user ID associated by RACF® with the flowed certificate.

Parent topic: Security scenario: two queue managers on z/OS

14.18.5. Security profiles and accesses required for the two-queue-manager scenario

Security profiles and accesses for either a batch or CICS implementation of the two-queue-manager scenario.

Table 1, Table 1, and Table 1 show the security profiles that are required to enable the scenario to work:

Table 1. Security profiles for the example scenario

Class	Profile	User ID	Access
MQCONN	QM1.CHIN	MOVER1	READ
MQADMIN	QM1.RESLEVEL	BATCHID CICSAD1 MOVER1	NONE
MQADMIN	QM1.CONTEXT.**	MOVER1	CONTROL
MQQUEUE	QM1.SYSTEM.COMMAND.INPUT	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.CHANNEL.SYNCQ	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.CHANNEL.INITQ	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.COMMAND.REPLY.MODEL	MOVER1	UPDATE
MQQUEUE	QM1.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER1	UPDATE
MQQUEUE	QM1.QM1.TO.QM2.TCP	MOVER1	ALTER
MQQUEUE	QM1.QM1.TO.QM2.LU62	MOVER1	ALTER
MQQUEUE	QM1.QM1.TO.QM2.SSL	MOVER1	ALTER
MQCONN	QM2.CHIN	MOVER2	READ
MQADMIN	QM2.RESLEVEL	MOVER2	NONE
MQADMIN	QM2.CONTEXT.**	MOVER2	CONTROL
MQQUEUE	QM2.SYSTEM.COMMAND.INPUT	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.CHANNEL.SYNCQ	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.CHANNEL.INITQ	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.COMMAND.REPLY.MODEL	MOVER2	UPDATE
MQQUEUE	QM2.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER2	UPDATE
MQQUEUE	QM2.DLQ	MOVER2	UPDATE

Security profiles required for a batch application
Additional security profiles required for a batch implementation of the two-queue-manager scenario.

Security profiles required for a CICS application
Additional security profiles required for a CICS implementation of the two-queue-manager scenario.

Parent topic: Security scenario: two queue managers on z/OS

14.18.5.1. Security profiles required for a batch application

Additional security profiles required for a batch implementation of the two-queue-manager scenario.

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:

- LQ1
- RQA
- RQB
- RQC

It uses the MQPMO_SET_ALL_CONTEXT and MQPMO_ALTERNATE_USER_AUTHORITY options. The alternate user ID found in the `UserIdentifier` field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 1. Sample security profiles for the batch application on queue manager QM1

Class	Profile	User ID	Access
MQCONN	QM1.BATCH	BATCHID	READ
MQADMIN	QM1.CONTEXT.**	BATCHID	CONTROL
MQQUEUE	QM1.LQ1	BATCHID	UPDATE
MQQUEUE	QM1.RQA	BATCHID	UPDATE
MQQUEUE	QM1.RQB	BATCHID	UPDATE
MQQUEUE	QM1.RQC	BATCHID	UPDATE

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel not using SSL):

Table 2. Sample security profiles for queue manager QM2 using TCP/IP and not SSL

Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCATCP MOVER2	UPDATE
MQADMIN	QM2.CONTEXT.**	MCATCP MOVER2	CONTROL
MQQUEUE	QM2.LQA	MOVER2 MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	MOVER2 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).
2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the `UserIdentifier` in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

Table 3. Sample security profiles for queue manager QM2 using LU 6.2

Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCALU62 MOVER1	UPDATE
MQADMIN	QM2.CONTEXT.**	MCALU62 MOVER1	CONTROL
MQQUEUE	QM2.LQB	MOVER1 MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	MOVER1 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

The following profiles are required on queue manager QM2 for messages put to queue RQC on queue manager QM1 (for the TCP/IP channel using SSL):

Table 4. Sample security profiles for queue manager QM2 using TCP/IP and SSL

Class	Profile	User ID	Access
MQADMIN	QM2.ALTERNATE.USER.MSGUSR	MCASSL CERTID	UPDATE
MQADMIN	QM2.CONTEXT.**	MCASSL CERTID	CONTROL
MQQUEUE	QM2.LQC	CERTID MSGUSR	UPDATE
MQQUEUE	QM2.DLQ	CERTID MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCASSL).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCASSL).
3. Because the certificate flowed by the channel from QM1 as part of the SSL handshake might be installed on QM2's system, or might match a certificate name filter on QM2's system, the user ID found during that matching is used as the channel user ID (CERTID).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCASSL and CERTID are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The CERTID and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

Parent topic: Security profiles and accesses required for the two-queue-manager scenario

14.18.5.2. Security profiles required for a CICS application

Additional security profiles required for a CICS implementation of the two-queue-manager scenario.

The CICS® application uses a CICS address space user ID of CICSAD1 and a CICS task user ID of CICSTX1. The security profiles required on queue manager QM1 are different from those profiles required for the batch application. The profiles required on queue manager QM2 are the same as for the batch application.

The following profiles are required on queue manager QM1:

Table 1. Sample security profiles for the CICS application on queue manager QM1

Class	Profile	User ID	Access
MQCONN	QM1.CICS	CICSAD1	READ
MQADMIN	QM1.CONTEXT.**	CICSAD1 CICSTX1	CONTROL
MQQUEUE	QM1.LQ1	CICSAD1 CICSTX1	UPDATE
MQQUEUE	QM1.RQA	CICSAD1 CICSTX1	UPDATE
MQQUEUE	QM1.RQB	CICSAD1 CICSTX1	UPDATE

Parent topic: Security profiles and accesses required for the two-queue-manager scenario

14.19. Security scenario: queue-sharing group on z/OS

In this scenario, an application uses the **MQPUT1** call to put messages to queues on queue manager QM1. Some of the messages are then forwarded to queues on QM2, using TCP and LU 6.2 channels. The application is a batch application, and the messages are put using the MQPMO_SET_ALL_CONTEXT option.

This is illustrated in Figure 1.

The following assumptions are made about the queue managers:

- All the required WebSphere MQ definitions have been predefined or have been made through the CSQINP2 data set processed at queue manager startup.
  If they have not, you need the appropriate access authority to the commands needed to define these objects.
- All the RACF® profiles required have been defined and appropriate access authorities have been granted, before the queue manager and channel initiators started.
  If they have not, you need the appropriate authority to issue the RACF commands required to define all the profiles needed and grant the appropriate access authorities to those profiles. You also need the appropriate authority to issue the MQSC security commands to start using the new security profiles.

Security switch settings for queue-sharing group scenario
Switch settings and RACF profiles.

Queue manager QM1 in queue-sharing group scenario
Queues and channels for QM1.

Queue manager QM2 in queue-sharing group scenario
Queues and channels for QM2.

User IDs used in queue-sharing group scenario
Explanation of the user IDs in the scenario.

Security profiles and accesses required for queue-sharing group scenario
Security profiles and accesses for either a batch or CICS implementation of the two-queue-manager scenario.

Parent topic: Setting up security on z/OS

14.19.1. Security switch settings for queue-sharing group scenario

Switch settings and RACF® profiles.

The following security switches are set for the queue-sharing group:

- Subsystem security on
- Queue-sharing group security on
- Queue manager security off
- Queue security on
- Alternate user security on
- Context security on
- Process security off
- Namelist security off
- Connection security on
- Command security on
- Command resource security on

The following profiles are defined in the MQADMIN class to turn process, namelist, and queue-manager level security off:

```
QSGA.NO.PROCESS.CHECKS
QSGA.NO.NLIST.CHECKS
QSGA.NO.QMGR.CHECKS
```

Parent topic: Security scenario: queue-sharing group on z/OS

14.19.2. Queue manager QM1 in queue-sharing group scenario

Queues and channels for QM1.

The following queues are defined on queue manager QM1:

LQ1
A local queue.

RQA
A remote queue definition, with the following attributes:

- RNAME(LQA)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.TCP)

RQB
A remote queue definition, with the following attributes:

- RNAME(LQB)
- RQMNAME(QM2)
- XMITQ(QM1.TO.QM2.LU62)

QM1.TO.QM2.TCP
A transmission queue.

QM1.TO.QM2.LU62
A transmission queue.

The following channels are defined on QM1:

QM1.TO.QM2.TCP
A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(TCP)
- XMITQ(QM1.TO.QM2.TCP)
- CONNAME(QM2TCP)

QM1.TO.QM2.LU62
A sender channel definition, with the following attributes:

- CHLTYPE(SDR)
- TRPTYPE(LU62)
- XMITQ(QM1.TO.QM2.LU62)
- CONNAME(QM2LU62)

(See Security considerations for the channel initiator on z/OS for information about setting up APPC security.)

Parent topic: Security scenario: queue-sharing group on z/OS zs13280_

14.19.3. Queue manager QM2 in queue-sharing group scenario

Queues and channels for QM2.

The following queues have been defined on queue manager QM2:

LQA
A local queue.

LQB
A local queue.

DLQ
A local queue that is used as the dead-letter queue.

The following channels have been defined on QM2:

QM1.TO.QM2.TCP
A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(TCP)
- PUTAUT(CTX)
- MCAUSER(MCATCP)

QM1.TO.QM2.LU62
A receiver channel definition, with the following attributes:

- CHLTYPE(RCVR)
- TRPTYPE(LU62)
- PUTAUT(CTX)
- MCAUSER(MCALU62)

(See Security considerations for the channel initiator on z/OS for information about setting up APPC security.)

Parent topic: Security scenario: queue-sharing group on z/OS

14.19.4. User IDs used in queue-sharing group scenario

Explanation of the user IDs in the scenario.

The following user IDs are used:

BATCHID
Batch application (Job or TSO ID)

MSGUSR
`UserIdentifier` in MQMD (context user ID)

MOVER1
QM1 channel initiator address space user ID

MOVER2
QM2 channel initiator address space user ID

MCATCP
MCAUSER specified on the TCP/IP receiver channel definition

MCALU62
MCAUSER specified on the LU 6.2 receiver channel definition

Parent topic: Security scenario: queue-sharing group on z/OS

14.19.5. Security profiles and accesses required for queue-sharing group scenario

Security profiles and accesses for either a batch or CICS implementation of the two-queue-manager scenario.

Table 1 through Table 3 show the security profiles that are required to enable the scenario to work:

Table 1. Security profiles for the example scenario

Class	Profile	User ID	Access
MQCONN	QSGA.CHIN	MOVER1 MOVER2	READ
MQADMIN	QSGA.RESLEVEL	BATCHID MOVER1 MOVER2	NONE
MQADMIN	QSGA.CONTEXT.**	MOVER1	CONTROL

		MOVER2	
MQQUEUE	QSGA.SYSTEM.COMMAND.INPUT	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.CHANNEL.SYNCQ	MOVER1 MOVER	UPDATE
MQQUEUE	QSGA.SYSTEM.CHANNEL.INITQ	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.COMMAND.REPLY.MODEL	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.ADMIN.CHANNEL.EVENT	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.QSG.CHANNEL.SYNCQ	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.SYSTEM.QSG.TRANSMIT.QUEUE	MOVER1 MOVER2	UPDATE
MQQUEUE	QSGA.QM1.TO.QM2.TCP	MOVER1	ALTER
MQQUEUE	QSGA.QM1.TO.QM2.LU62	MOVER1	ALTER
MQQUEUE	QSGA.DLQ	MOVER2	UPDATE

Security profiles required for a batch application
Additional security profiles required for a batch implementation of the queue-sharing group scenario.

Parent topic: Security scenario: queue-sharing group on z/OS

14.19.5.1. Security profiles required for a batch application

Additional security profiles required for a batch implementation of the queue-sharing group scenario.

The batch application runs under user ID BATCHID on QM1. It connects to queue manager QM1 and puts messages to the following queues:

- LQ1
- RQA
- RQB

It uses the MQPMO_SET_ALL_CONTEXT and MQPMO_ALTERNATE_USER_AUTHORITY options. The alternate user ID found in the `UserIdentifier` field of the message descriptor (MQMD) is MSGUSR.

The following profiles are required on queue manager QM1:

Table 1. Sample security profiles for the batch application on queue manager QM1

Class	Profile	User ID	Access
MQCONN	QSGA.BATCH	BATCHID	READ
MQADMIN	QSGA.CONTEXT.**	BATCHID	CONTROL
MQQUEUE	QSGA.LQ1	BATCHID	UPDATE
MQQUEUE	QSGA.RQA	BATCHID	UPDATE
MQQUEUE	QSGA.RQB	BATCHID	UPDATE

The following profiles are required on queue manager QM2 for messages put to queue RQA on queue manager QM1 (for the TCP/IP channel):

Table 2. Sample security profiles for queue manager QM2 using TCP/IP

Class	Profile	User ID	Access
MQADMIN	QSGA.ALTERNATE.USER.MSGUSR	MCATCP MOVER2	UPDATE
MQADMIN	QSGA.CONTEXT.**	MCATCP MOVER2	CONTROL
MQQUEUE	QSGA.LQA	MOVER2 MSGUSR	UPDATE
MQQUEUE	QSGA.DLQ	MOVER2 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCATCP).
2. The MCAUSER field of the receiver channel definition is set to MCATCP; this user ID is used in addition to the channel initiator address space user ID for the checks carried out against the alternate user ID and context profile.
3. The MOVER2 user ID and the `UserIdentifier` in the message descriptor (MQMD) are used for the resource checks against the queue.
4. The MOVER2 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.
5. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.

The following profiles are required on queue manager QM2 for messages put to queue RQB on queue manager QM1 (for the LU 6.2 channel):

Table 3. Sample security profiles for queue manager QM2 using LU 6.2

Class	Profile	User ID	Access
MQADMIN	QSGA.ALTERNATE.USER.MSGUSR	MCALU62 MOVER1	UPDATE
MQADMIN	QSGA.CONTEXT.**	MCALU62 MOVER1	CONTROL
MQQUEUE	QSGA.LQB	MOVER1 MSGUSR	UPDATE
MQQUEUE	QSGA.DLQ	MOVER1 MSGUSR	UPDATE

Notes:

1. The user ID passed in the MQMD of the message is used as the user ID for the **MQPUT1** on queue manager QM2 because the receiver channel was defined with PUTAUT(CTX) and MCAUSER(MCALU62).
2. The MCA user ID is set to the value of the MCAUSER field of the receiver channel definition (MCALU62).
3. Because LU 6.2 supports security on the communications system for the channel, the user ID received from the network is used as the channel user ID (MOVER1).
4. Two user IDs are checked on all three checks performed because RESLEVEL is set to NONE.
5. MCALU62 and MOVER1 are used for the checks performed against the alternate user ID and Context profiles, and MSGUSR and MOVER1 are used for the checks against the queue profile.
6. The MOVER1 and MSGUSR user IDs both need access to the dead-letter queue so that messages that cannot be put to the destination queue can be sent there.

Parent topic: Security profiles and accesses required for queue-sharing group scenario

14.20. WebSphere MQ for z/OS security implementation checklist

This topic gives a step-by-step procedure you can use to work out and define the security implementation for each of your WebSphere® MQ queue managers.

Refer to other sections for details, in particular Profiles used to control access to WebSphere MQ resources.

If you require security checking, follow this checklist to implement it:

1. Activate the RACF MQADMIN (uppercase profiles) and MXADMIN (mixed case profiles) classes
   - Do you want security at queue-sharing group level, queue-manager level, or a combination of both?
     Refer to Profiles to control queue-sharing group or queue manager level security.
2. Do you need connection security?
   - **Yes**: Activate the MQCONN class. Define appropriate connection profiles at either queue manager level or queue-sharing group level in the MQCONN class and permit the appropriate users or groups access to these profiles.
     Note: Only users of the **MQCONN** API request or CICS® or IMS™ address space user IDs need to have access to the corresponding connection profile.
   - **No**: Define an hlq.NO.CONNECT.CHECKS profile at either queue manager level or queue-sharing group level in the MQADMIN or MXADMIN class.
3. Do you need security checking on commands?
   - **Yes**: Activate the MQCMDS class. Define appropriate command profiles at either queue manager level or queue-sharing group level in the MQCMDS class and permit the appropriate users or groups access to these profiles.
     If you are using a queue-sharing group, you might need to include the user IDs used by the queue manager itself and the channel initiator, see Setting up WebSphere MQ for z/OS resource security.
   - **No**: Define an hlq.NO.CMD.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
4. Do you need security on the resources used in commands?
   - **Yes**: Ensure the MQADMIN or MXADMIN class is active. Define appropriate profiles for protecting resources on commands at either queue manager level or queue-sharing group level in the MQADMIN or MXADMIN class and permit the appropriate users or groups access to these profiles. Set the CMDUSER parameter in CSQ6SYSP to the default user ID to be used for command security checks.
     If you are using a queue-sharing group, you might need to include the user IDs used by the queue manager itself and the channel initiator, see Setting up WebSphere MQ for z/OS resource security.
   - **No**: Define an hlq.NO.CMD.RESC.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
5. Do you need queue security?
   - **Yes**: Activate the MQQUEUE or MXQUEUE class. Define appropriate queue profiles for the required queue manager or queue-sharing group in the MQQUEUE or MXQUEUEclass and permit the appropriate users or groups access to these profiles.
   - **No**: Define an hlq.NO.QUEUE.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
6. Do you need process security?
   - **Yes**: Activate the MQPROC or MXPROC class. Define appropriate process profiles at either queue manager or queue-sharing group level and permit the appropriate users or groups access to these profiles.
   - **No**: Define an hlq.NO.PROCESS.CHECKS profile for the appropriate queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
7. Do you need namelist security?
   - **Yes**: Activate the MQNLIST or MXNLISTclass. Define appropriate namelist profiles at either queue manager level or queue-sharing group level in the MQNLIST or MXNLIST class and permit the appropriate users or groups access to these profiles.
   - **No**: Define an hlq.NO.NLIST.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
8. Do you need topic security?
   - **Yes**: Activate the MXTOPIC class. Define appropriate topic profiles at either queue manager level or queue-sharing group level in the MXTOPIC class and permit the appropriate users or groups access to these profiles.
   - **No**: Define an hlq.NO.TOPIC.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
9. Do any users need to protect the use of the **MQOPEN** or **MQPUT1** options relating to the use of context?
   - **Yes**: Ensure the MQADMIN or MXADMIN class is active. Define hlq.CONTEXT.queuename profiles at the queue, queue manager, or queue-sharing group level in the MQADMIN or MXADMIN class and permit the appropriate users or groups access to these profiles.
   - **No**: Define an hlq.NO.CONTEXT.CHECKS profile for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
10. Do you need to protect the use of alternate user IDs?
    - **Yes**: Ensure the MQADMIN or MXADMIN class is active. Define the appropriate hlq.ALTERNATE.USER.*alternateuserid* profiles for the required queue manager or queue-sharing group and permit the required users or groups access to these profiles.
    - **No**: Define the profile hlq.NO.ALTERNATE.USER.CHECKS for the required queue manager or queue-sharing group in the MQADMIN or MXADMIN class.
11. Do you need to tailor which user IDs are to be used for resource security checks through RESLEVEL?
    - **Yes**: Ensure the MQADMIN or MXADMIN class is active. Define an hlq.RESLEVEL profile at either queue manager level or queue-sharing group level in the MQADMIN or MXADMIN class and permit the required users or groups access to the profile.

- **No**: Ensure that no generic profiles exist in the MQADMIN or MXADMIN class that could apply to hlq.RESLEVEL. Define an hlq.RESLEVEL profile for the required queue manager or queue-sharing group and ensure that no users or groups have access to it.

12. Do you need to 'time out' unused user IDs from WebSphere MQ?
    - **Yes**: Determine what timeout values you would like to use and issue the MQSC ALTER SECURITY command to change the TIMEOUT and INTERVAL parameters.
    - **No**: Issue the MQSC ALTER SECURITY command to set the INTERVAL value to zero.

    Note: Update the CSQINP1 initialization input data set used by your subsystem so that the MQSC ALTER SECURITY command is issued automatically at every queue manager start up.
13. Do you use distributed queuing?
    - **Yes**: Determine the appropriate MCAUSER attribute value for each channel, and provide suitable channel security exits.
14. Do you want to use the Secure Sockets Layer (SSL)?
    - **Yes**: Plan your SSL infrastructure. Install the System SSL feature of z/OS®. In RACF, set up your certificate name filters (CNFs), if you are using them, and your digital certificates. Set up your SSL key ring. Ensure that the SSLKEYR queue manager attribute is nonblank and points to your SSL key ring, and ensure that the value of SSLTASKS is at least 2.
    - **No**: Ensure that SSLKEYR is blank, and SSLTASKS is zero.

    For further details about SSL, see WebSphere MQ support for SSL and TLS.
15. Do you use clients?
    - **Yes**: Determine the appropriate MCAUSER attribute value for each server-connection channel, and provide suitable channel security exits if required.
16. Check your switch settings.
    WebSphere MQ issues messages at queue manager startup that display your security settings. Use these messages to determine whether your switches are set correctly. For an example of these messages, see User messages on startup.

Parent topic: Setting up security on z/OS

This build: January 26, 2011 11:22:35

Notices | Trademarks | Downloads | Library | Support | Feedback

15. 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.

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 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.

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: Security

15.1. 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

Related concepts
Authority to work with WebSphere MQ objects on UNIX systems and Windows

15.1.1. 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

15.2. 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

15.2.1. 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

15.2.2. 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

15.2.3. 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

15.2.3.1. 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

15.2.3.2. 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

15.2.4. 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

15.2.5. 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

15.3. 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

15.4. 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

15.5. 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: Setting up security on UNIX systems and Windows

15.5.1. 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

15.5.2. 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

15.5.3. 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

15.5.4. 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

15.6. 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: Setting up security on UNIX systems and Windows

15.6.1. 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

15.6.2. 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

15.6.3. 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

15.6.4. 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

15.7. 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: Setting up security on UNIX systems and Windows

15.7.1. 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

15.7.2. 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

15.7.3. 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

15.7.4. 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

15.8. 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: Setting up security on UNIX systems and Windows

15.8.1. 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

15.8.2. 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

15.8.3. 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

15.8.4. 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

15.9. 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: Setting up security on UNIX systems and Windows

15.9.1. 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

15.9.2. 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

15.9.3. 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

15.9.4. 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

15.10. 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

15.10.1. 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.

Parent topic: Using the OAM to control access to objects on UNIX systems and Windows

15.10.1.1. 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

15.10.2. 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

15.10.2.1. 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

15.10.2.2. 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

15.10.2.3. 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

15.10.3. 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

15.10.4. 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

15.10.5. 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

15.11. 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

15.11.1. 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

15.11.2. 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

15.11.3. 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

15.11.4. 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

15.12. 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

15.12.1. 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 required for:	**Queue object (1)**	**Process object**	**Queue manager 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

15.12.2. 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

15.12.3. 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

15.13. 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

15.13.1. 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

15.13.2. 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

15.13.2.1. 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

15.13.2.2. 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

15.13.3. 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

15.13.4. 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

16. Setting up security on i5/OS

Security for WebSphere® MQ for i5/OS® is implemented using the WebSphere MQ Object Authority Manager (OAM) and i5/OS object level security.

Security considerations that must be made when determining access authority to WebSphere MQ objects.

You need to consider the following points when setting up authorities to the users in your enterprise:

1. Grant and revoke authorities to the WebSphere MQ for i5/OS commands using the i5/OS **GRTOBJAUT** and **RVKOBJAUT** commands.
   In the `QMQM` library, certain noncommand (*cmd) objects are set to have ***PUBLIC** authority to ***USE**. Do not change the authorities of these objects or use an authorization list to provide authority. Any incorrect authority might compromise WebSphere MQ functionality.
2. During installation of WebSphere MQ for i5/OS, the following special user profiles are created:

   QMQM

   Is used primarily for internal product-only functions. However, it can be used to run trusted applications using MQCNO_FASTPATH_BINDINGS; see the WebSphere MQ Application Programming Guide for further information.

   QMQMADM

   Is used as a group profile for administrators of WebSphere MQ. The group profile gives access to CL commands and WebSphere MQ resources.
3. If you are sending channel commands to remote queue managers, ensure that your user profile is a member of the group QMQMADM on the target system. For a list of PCF and MQSC channel commands, see WebSphere MQ for i5/OS CL commands.
4. The group set associated with a user is cached when the group authorizations are computed by the OAM.
   Any changes made to a user's group memberships after the group set has been cached are not recognized until you restart the queue manager or execute RFRMQMAUT to refresh security.
5. Limit the number of users who have authority to work with commands that are particularly sensitive. These commands include:
   - Create Message Queue Manager (**CRTMQM**)
   - Delete Message Queue Manager (**DLTMQM**)
   - Start Message Queue Manager (**STRMQM**)
   - End Message Queue Manager (**ENDMQM**)
   - Start Command Server (**STRMQMCSVR**)
   - End Command Server (**ENDMQMCSVR**)
6. Channel definitions contain a security exit program specification. Channel creation and modification requires special considerations. Details of security exits are given in WebSphere MQ Intercommunication.
7. The channel exit and trigger monitor programs can be substituted. The security of such replacements is the responsibility of the programmer.

The Object Authority Manager on i5/OS
The Object Authority Manager (OAM) manages users' authorizations to manipulate WebSphere MQ objects, including queues and process definitions. It also provides a command interface through which you can grant or revoke access authority to an object for a specific group of users. The decision to allow access to a resource is made by the OAM, and the queue manager follows that decision. If the OAM cannot make a decision, the queue manager prevents access to that resource.

WebSphere MQ authorities on i5/OS
To access WebSphere MQ objects, you need authority to issue the command and to access the object referenced. Administrators have access to all WebSphere MQ resources.

Authorization specification tables for i5/OS

Use this information to determine what authorization is required to use particular API calls, and particular options of those calls, on queue objects, process objects, and queue manager objects.

Generic OAM profiles on i5/OS
Object authority manager (OAM) generic profiles enable you to set the authority a user has to many objects at once, rather than having to issue separate **GRTMQMAUT** commands against each individual object when it is created. Using generic profiles in the **GRTMQMAUT** command enables you to set a generic authority for all future objects created that fit that profile.

Specifying the installed authorization service on i5/OS
You can specify which authorization service component to use.

Working with and without authority profiles on i5/OS
Use this information to learn how to work with authority profiles and how to work without authority profiles.

Object Authority Manager guidelines for i5/OS
Additional hints and tips for using the Object Authority Manager (OAM)

Parent topic: Security

16.1. The Object Authority Manager on i5/OS

The Object Authority Manager (OAM) manages users' authorizations to manipulate WebSphere® MQ objects, including queues and process definitions. It also provides a command interface through which you can grant or revoke access authority to an object for a specific group of users. The decision to allow access to a resource is made by the OAM, and the queue manager follows that decision. If the OAM cannot make a decision, the queue manager prevents access to that resource.

Through the OAM you can control:

- Access to WebSphere MQ objects through the MQI. When an application program attempts to access an object, the OAM checks that the user profile making the request has the authorization for the operation requested.
  In particular, this means that queues, and the messages on queues, can be protected from unauthorized access.
- Permission to use PCF and MQSC commands.

Different groups of users can have different kinds of access authority to the same object. For example, for a specific queue, one group could 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 get and put authority to a queue, but not be allowed to alter or delete the queue.

WebSphere MQ for i5/OS® provides commands to grant, revoke, and display the authority that an application or user has to issue WebSphere MQ for i5/OS commands and perform operations on WebSphere MQ for i5/OS objects

Parent topic: Setting up security on i5/OS

16.2. WebSphere MQ authorities on i5/OS

To access WebSphere® MQ objects, you need authority to issue the command and to access the object referenced. Administrators have access to all WebSphere MQ resources.

Access to WebSphere MQ objects is controlled by authorities to:

1. Issue the WebSphere MQ command
2. Access the WebSphere MQ objects referenced by the command

All WebSphere MQ for i5/OS® CL commands are shipped with an owner of QMQM, and the administration profile (QMQMADM) has *USE rights with the *PUBLIC access set to *EXCLUDE.

Note: The QSRDUPER program is used by the WebSphere MQ for i5/OS licensed program install to duplicate Command (*CMD) objects in QSYS. In i5/OS V5R4 and later, the QSRDUPER program was changed so that the default behavior is to create a proxy command rather than a duplicate of the original command. A proxy command redirects command execution to another command and has an attribute of PRX. If a proxy command by the same name as the command being copied exists in library QSYS, private authorities to the proxy command are not granted to the command in the product library. Attempts to prompt or run the proxy command in QSYS check the authority of the target command in the product library. Any changes in authority to *CMD objects therefore need to be done in the product library (QMQM) and those in QSYS do not need to be modified. For example:

```
GRTOBJAUT OBJ(QMQM/DSPMQMQ) OBJTYPE(*CMD) USER(MQUSER) AUT(*USE)
```

Changes to the authority structure of some of the product's CL commands allows public use of these commands, if you have the required OAM authority to the WebSphere MQ objects to make these changes.

To be a WebSphere MQ administrator on i5/OS, you must be a member of the *QMQMADM group*. This group has properties like the properties of the mqm group on UNIX and Windows systems. In particular, the QMQMADM group is created when you install WebSphere MQ for i5/OS, and members of the QMQMADM group have access to all WebSphere MQ resources on the system. You also have access to all WebSphere MQ resources if you have *ALLOBJ authority.

Administrators can use CL commands to administer WebSphere MQ. One of these commands is GRTMQMAUT, which is used to grant authorities to other users. Another command, STRMQMMQSC, enables an administrator to issue MQSC commands to a local queue manager.

Access authorities for WebSphere MQ objects on i5/OS
Access authorities required for running WebSphere MQ CL commands.

Access authorizations on i5/OS

Use this information to understand the access authorization commands.

Using the access authorization commands on i5/OS
Use this information to learn about the access authorization commands, and use the command examples.

Parent topic: Setting up security on i5/OS

16.2.1. Access authorities for WebSphere MQ objects on i5/OS

Access authorities required for running WebSphere® MQ CL commands.

WebSphere MQ for i5/OS® categorizes the product's CL commands into two groups:

Group 1

Users must be in the QMQMADM user group, or have *ALLOBJ authority, to process these commands. Users having either of these authorities can process all commands in all categories without requiring any extra authority.

Note: These authorities override any OAM authority.

These commands can be grouped as follows:

- Command Server Commands
  - ENDMQMCSVR, End WebSphere MQ Command Server
  - STRMQMCSVR, Start WebSphere MQ Command Server
- Dead-Letter Queue Handler Command
  - STRMQMDLQ, Start WebSphere MQ Dead-Letter Queue Handler
- Listener Command
  - ENDMQMLSR, End WebSphere MQ listener
  - STRMQMLSR, Start non-object listener
- Media Recovery Commands
  - RCDMQMIMG, Record WebSphere MQ Object Image
  - RCRMQMOBJ, Re-create WebSphere MQ Object
  - WRKMQMTRN, Work with WebSphere MQ Transactions
- Queue Manager Commands
  - CRTMQM, Create Message Queue Manager
  - DLTMQM, Delete Message Queue Manager
  - ENDMQM, End Message Queue Manager
  - STRMQM, Start Message Queue Manager
- Security Commands
  - GRTMQMAUT, Grant WebSphere MQ Object Authority
  - RVKMQMAUT, Revoke WebSphere MQ Object Authority
- Trace Command
  - TRCMQM, Trace WebSphere MQ Job
- Transaction Commands
  - RSVMQMTRN, Resolve WebSphere MQ Transaction
- Trigger Monitor Commands
  - STRMQMTRM, Start Trigger Monitor
- WebSphere MQSC Commands
  - RUNMQSC, Run WebSphere MQSC Commands
  - STRMQMMQSC, Start WebSphere MQSC Commands

Group 2

The rest of the commands, for which two levels of authority are required:

1. i5/OS authority to run the command. A WebSphere MQ administrator sets this using the **GRTOBJAUT** command to override the *PUBLIC (*EXCLUDE) restriction for a user or group of users.
   For example:
   ```
   GRTOBJAUT OBJ(DSPMQMQ) OBJTYPE(*CMD) USER(MQUSER) AUT(*USE)
   ```
2. WebSphere MQ authority to manipulate the WebSphere MQ objects associated with the command, or commands, given the correct i5/OS authority in Step 1.
   This authority is controlled by the user having the appropriate OAM authority for the required action, set by a WebSphere MQ administrator using the **GRTMQMAUT** command
   For example:
   ```
   CHGMQMQ *connect authority to the queue manager + *admchg authority to
                  the queue
   ```

The commands can be grouped as follows:

- Channel Commands
  - CHGMQMCHL, Change WebSphere MQ Channel
  - CPYMQMCHL, Copy WebSphere MQ Channel
  - CRTMQMCHL, Create WebSphere MQ Channel

  - DLTMQMCHL, Delete WebSphere MQ Channel
  - RSVMQMCHL, Resolve WebSphere MQ Channel

- Display commands
  To process the DSP commands you must grant the user `*connect` and `*admdsp` authority to the queue manager, together with any specific option listed:
  - DSPMQM, Display Message Queue Manager
  - DSPMQMAUT, Display WebSphere MQ Object Authority
  - DSPMQMAUTI, Display WebSphere MQ Authentication Information – `*admdsp` to the authentication information object
  - DSPMQMCHL, Display WebSphere MQ Channel – `*admdsp` to the channel
  - DSPMQMCSVR, Display WebSphere MQ Command Server
  - DSPMQMNL, Display WebSphere MQ Namelist – `*admdsp` to the namelist
  - DSPMQMOBJN, Display WebSphere MQ Object Names
  - DSPMQMPRC, Display WebSphere MQ Process – `*admdsp` to the process
  - DSPMQMQ, Display WebSphere MQ Queue – `*admdsp` to the queue
  - DSPMQMTOP, Display WebSphere MQ Topic – `*admdsp` to the topic
- Work with commands
  To process the WRK commands and display the options panel you must grant the user `*connect` and `*admdsp` authority to the queue manager, together with any specific option listed:
  - WRKMQM, Work with Message Queue Managers
  - WRKMQMAUT, Work with WebSphere MQ Object Authority
  - WRKMQMAUTD, Work with WebSphere MQ Object Authority Data
  - WRKMQMAUTI, Work with WebSphere MQ Authentication Information»
    - `*admchg` for the Change WebSphere MQ Authentication Information Object command.
    - `*admcpy` for the Copy WebSphere MQ Authentication Information Object command.
    - `*admcrt` for the Create WebSphere MQ Authentication Information Object command.
    - `*admdlt` for the Delete WebSphere MQ Authentication Information Object command.
    - `*admdsp` for the Display WebSphere MQ Authentication Information Object command.

    «
  - WRKMQMCHL, Work with WebSphere MQ Channel
    »This requires the following authorities:
    - `*admchg` for the Change WebSphere MQ Channel command.
    - `*admclr` for the Clear WebSphere MQ Channel command.
    - `*admcpy` for the Copy WebSphere MQ Channel command.
    - `*admcrt` for the Create WebSphere MQ Channel command.
    - `*admdlt` for the Delete WebSphere MQ Channel command.
    - `*admdsp` for the Display WebSphere MQ Channel command.
    - »`*ctrl` for the Start WebSphere MQ Channel command.«
    - »`*ctrl` for the End WebSphere MQ Channel command.«
    - »`*ctrl` for the Ping WebSphere MQ Channel command.«
    - »`*ctrlx` for the Reset WebSphere MQ Channel command.«
    - »`*ctrlx` for the Resolve WebSphere MQ Channel command.«

    «
  - WRKMQMCHST, Work with WebSphere MQ Channel Status
    »This requires `*admdsp` authority to the channel.«
  - WRKMQMCL, Work with WebSphere MQ Clusters
  - WRKMQMCLQ, Work with WebSphere MQ Cluster Queues
  - WRKMQMCLQM, Work with WebSphere MQ Cluster Queue Manager
  - WRKMQMLSR, Work with WebSphere MQ Listener
  - WRKMQMMSG, Work with WebSphere MQ Messages
    This requires `*browse` authority to the queue
  - WRKMQMNL, Work with WebSphere MQ Namelists
    This requires the following authorities:
    - `*admchg` for the Change WebSphere MQ Namelist command.
    - `*admcpy` for the Copy WebSphere MQ Namelist command.
    - `*admcrt` for the Create WebSphere MQ Namelist command.
    - `*admdlt` for the Delete WebSphere MQ Namelist command.
    - `*admdsp` for the Display WebSphere MQ Namelist command.
  - WRKMQMPRC, Work with WebSphere MQ Processes
    This requires the following authorities:
    - `*admchg` for the Change WebSphere MQ Process command.
    - `*admcpy` for the Copy WebSphere MQ Process command.
    - `*admcrt` for the Create WebSphere MQ Process command.
    - `*admdlt` for the Delete WebSphere MQ Process command.
    - `*admdsp` for the Display WebSphere MQ Process command.
  - WRKMQMQ, Work with WebSphere MQ queues
    This requires the following authorities:
    - `*admchg` for the Change WebSphere MQ Queue command.
    - `*admclr` for the Clear WebSphere MQ Queue command.
    - `*admcpy` for the Copy WebSphere MQ Queue command.

        - *admcrt for the Create WebSphere MQ Queue command.
        - *admdlt for the Delete WebSphere MQ Queue command.
        - *admdsp for the Display WebSphere MQ Queue command.
    - WRKMQMQSTS, Work with WebSphere MQ Queue Status
    - ❯WRKMQMTOP, Work with WebSphere MQ Topics
      This requires the following authorities❯
        - *admchg for the Change WebSphere MQ Topic command.
        - *admcpy for the Copy WebSphere MQ Topic command.
        - *admcrt for the Create WebSphere MQ Topic command.
        - *admdlt for the Delete WebSphere MQ Topic command.
        - *admdsp for the Display WebSphere MQ Topic command.

      ❮
      ❮
    - ❯WRKMQMSUB, Work with WebSphere MQ Subscriptions❮
- Other Channel commands
  To process the channel commands you must grant the user the specific authorities listed:
    - ENDMQMCHL, End WebSphere MQ Channel
      This requires *connect authority to the queue manager and *allmqi authority to the transmission queue associated with the channel.
    - ❯ENDMQMLSR, End WebSphere MQ Listener
      This requires *connect authority to the queue manager and *ctrl authority to the named listener object.
      ❮
    - PNGMQMCHL, Ping WebSphere MQ Channel
      This requires *connect and *inq authority to the queue manager.
    - RSTMQMCHL, Reset WebSphere MQ Channel
      This requires *connect authority to the queue manager.
    - ❯STRMQMCHL, Start WebSphere MQ Channel
      This requires *connect authority to the queue manager and *ctrl authority to the channel object.
      ❮
    - STRMQMCHLI, Start WebSphere MQ Channel Initiator
      This requires *connect and *inq authority to the queue manager, and *allmqi authority to the initiation queue associated with the transmission queue of the channel.
    - STRMQMLSR, Start WebSphere MQ Listener
      ❯This requires *connect authority to the queue manager and *ctrl authority to the named listener object.❮
- Other commands:
  To process the following commands you must grant the user the specific authorities listed:
    - CCTMQM, Connect to Message Queue Manager
      This requires no WebSphere MQ object authority.
    - CHGMQM, Change Message Queue Manager
      This requires *connect and *admchg authority to the queue manager.
    - ❯CHGMQMAUTI, Change WebSphere MQ Authentication Information
      This requires *connect authority to the queue manager and *admchg and *admdsp authority to the authentication information object.
      ❮
    - CHGMQMNL, Change WebSphere MQ Namelist
      This requires *connect authority to the queue manager and *admchg authority to the namelist.
    - CHGMQMPRC, Change WebSphere MQ Process
      This requires *connect authority to the queue manager and *admchg authority to the process.
    - CHGMQMQ, Change WebSphere MQ Queue
      This requires *connect authority to the queue manager and *admchg authority to the queue.
    - CLRMQMQ, Clear WebSphere MQ Queue
      This requires *connect authority to the queue manager and ❯*admclr❮ authority to the queue.
    - ❯CPYMQMAUTI, Copy WebSphere MQ Authentication Information
      This requires *connect authority to the queue manager and *admdsp authority to the authentication information object and *admcrt authority to the authentication information object class.
      ❮
    - CPYMQMNL, Copy WebSphere MQ Namelist
      This requires *connect and *admcrt authority to the queue manager.
    - CPYMQMPRC, Copy WebSphere MQ Process
      This requires *connect and *admcrt authority to the queue manager.
    - CPYMQMQ, Copy WebSphere MQ Queue
      This requires *connect and *admcrt authority to the queue manager.
    - ❯CRTMQMAUTI, Create WebSphere MQ Authentication Information
      This requires *connect authority to the queue manager and *admdsp authority to the authentication information object and *admcrt authority to the authentication information object class.
      ❮
    - CRTMQMNL, Create WebSphere MQ Namelist
      This requires *connect and *admcrt authority to the queue manager and *admdsp authority to the default namelist.
    - CRTMQMPRC, Create WebSphere MQ Process
      This requires *connect and *admcrt authority to the queue manager and *admdsp authority to the default process.
    - CRTMQMQ, Create WebSphere MQ Queue
      This requires *connect and *admcrt authority to the queue manager and *admdsp authority to the default queue.
    - CVTMQMDTA, Convert WebSphere MQ Data Type Command
      This requires no WebSphere MQ object authority.
    - ❯DLTMQMAUTI, Delete WebSphere MQ Authentication Information
      This requires *connect authority to the queue manager and *ctrlx authority to the authentication information object.
      ❮
    - DLTMQMNL, Delete WebSphere MQ Namelist
      This requires *connect authority to the queue manager and *admdlt authority to the namelist.

- DLTMQMPRC, Delete WebSphere MQ Process
  This requires `*connect` authority to the queue manager and `*admdlt` authority to the process.
- DLTMQMQ, Delete WebSphere MQ Queue
  This requires `*connect` authority to the queue manager and `*admdlt` authority to the queue.
- DSCMQM, Disconnect from Message Queue Manager
  This requires no WebSphere MQ object authority.
- RFRMQMAUT, Refresh Security
  This requires `*connect` authority to the queue manager.
- RFRMQMCL, Refresh Cluster
  This requires `*connect` authority to the queue manager.
- RSMMQMCLQM, Resume Cluster Queue Manager
  This requires `*connect` authority to the queue manager.
- RSTMQMCL, Reset Cluster
  This requires `*connect` authority to the queue manager.
- SPDMQMCLQM, Suspend Cluster Queue Manager
  This requires `*connect` authority to the queue manager.

Parent topic: WebSphere MQ authorities on i5/OS

16.2.2. Access authorizations on i5/OS®

Use this information to understand the access authorization commands.

Authorizations defined by the AUT keyword on the **GRTMQMAUT** and **RVKMQMAUT** commands can be categorized as follows:

- Authorizations related to MQI calls
- Authorization-related administration commands
- Context authorizations
- General authorizations, that is, for MQI calls, for commands, or both

The following tables list the different authorities, using the AUT parameter for MQI calls, Context calls, MQSC and PCF commands, and generic operations.

Table 1. Authorizations for MQI calls

AUT	Description
***ALTUSR**	Allow another user's authority to be used for MQOPEN and MQPUT1 calls.
***BROWSE**	Retrieve a message from a queue by issuing an MQGET call with the BROWSE option.
***CONNECT**	Connect the application to the specified queue manager by issuing an MQCONN call.
***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.
***PUB**	Open a topic to publish a message using an MQPUT call.
***PUT**	Put a message on a specific queue by issuing an MQPUT call.
***RESUME**	Resume a subscription using an MQSUB call.
***SET**	Set attributes on a queue from the MQI by issuing an MQSET call. If you open a queue for multiple options, you must be authorized for each of them.
***SUB**	Create, Alter or Resume a subscription to a topic using an MQSUB call.

Table 2. Authorizations for context calls

AUT	Description
***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.

Table 3. Authorizations for MQSC and PCF calls

AUT	Description
***ADMCHG**	Change the attributes of the specified object.
***ADMCLR**	Clear the specified object (PCF Clear object command only).
***ADMCRT**	Create objects of the specified type.
***ADMDLT**	Delete the specified object.
***ADMDSP**	Display the attributes of the specified object.

Table 4. Authorizations for generic operations

AUT	Description
***ALL**	Use all operations applicable to the object.
***ALLADM**	Perform all administration operations applicable to the object.
***ALLMQI**	Use all MQI calls applicable to the object.
***CTRL**	Control startup and shutdown of channels, listeners, and services.
***CTRLX**	Reset sequence number and resolve indoubt channels.

Parent topic: WebSphere MQ authorities on i5/OS

16.2.3. Using the access authorization commands on i5/OS

Use this information to learn about the access authorization commands, and use the command examples.

Using the GRTMQMAUT command

If you have the required authorization, you can use the **GRTMQMAUT** command to grant authorization of a user profile or user group to access a particular object. The following examples illustrate how the **GRTMQMAUT** command is used:

1.
   ```
   GRTMQMAUT OBJ(RED.LOCAL.QUEUE) OBJTYPE(*LCLQ) USER(GROUPA) +
             AUT(*BROWSE *PUT) MQMNAME('saturn.queue.manager')
   ```
   In this example:
   - RED.LOCAL.QUEUE is the object name.
   - *LCLQ (local queue) is the object type.
   - GROUPA is the name of a user profile on the system whose authorizations are to change. This profile can be used as a group profile for other users.
   - *BROWSE and *PUT are the authorizations being granted to the specified queue.
     *BROWSE adds authorization to browse messages on the queue (to issue MQGET with the browse option).
     *PUT adds authorization to put (MQPUT) messages on the queue.
   - saturn.queue.manager is the queue manager name.
2. The following command grants to users JACK and JILL all applicable authorizations, to all process definitions, for the default queue manager.
   ```
   GRTMQMAUT OBJ(*ALL) OBJTYPE(*PRC) USER(JACK JILL) AUT(*ALL)
   ```
3. The following command grants user GEORGE authority to put a message on the queue ORDERS, on the queue manager TRENT.
   ```
   GRTMQMAUT OBJ(TRENT) OBJTYPE(*MQM) USER(GEORGE) AUT(*CONNECT) MQMNAME (TRENT)
   GRTMQMAUT OBJ(ORDERS) OBJTYPE(*Q) USER(GEORGE) AUT(*PUT) MQMNAME (TRENT)
   ```

Using the RVKMQMAUT command

If you have the required authorization, you can use the **RVKMQMAUT** command to remove previously granted authorization of a user profile or user group to access a particular object. The following examples illustrate how the **RVKMQMAUT** command is used:

1.
   ```
   RVKMQMAUT OBJ(RED.LOCAL.QUEUE) OBJTYPE(*LCLQ) USER(GROUPA) +
   AUT(*PUT) MQMNAME('saturn.queue.manager')
   ```
   The authority to put messages to the specified queue, that was granted in the previous example, is removed for GROUPA.
2.
   ```
   RVKMQMAUT OBJ(PAY*) OBJTYPE(*Q) USER(*PUBLIC) AUT(*GET) +
   MQMNAME(PAYROLLQM)
   ```
   Authority to get messages from any queue whose name starts with the characters PAY, owned by queue manager PAYROLLQM, is removed from all users of the system unless they, or a group to which they belong, have been separately authorized.

Using the DSPMQMAUT command

The display MQM authority (**DSPMQMAUT**) command shows, for the specified object and user, the list of authorizations that the user has for the object. The following example illustrates how the command is used:

```
DSPMQMAUT OBJ(ADMINNL) OBJTYPE(*NMLIST) USER(JOE) OUTPUT(*PRINT) +
MQMNAME(ADMINQM)
```

Using the RFRMQMAUT command

The refresh MQM security (**RFRMQMAUT**) command enables you to update the OAM's authorization group information immediately, reflecting changes made at the operating system level, without needing to stop and restart the queue manager. The following example illustrates how the command is used:

```
RFRMQMAUT MQMNAME(ADMINQM)
```

Parent topic: WebSphere MQ authorities on i5/OS

16.3. Authorization specification tables for i5/OS

Use this information to determine what authorization is required to use particular API calls, and particular options of those calls, on queue objects, process objects, and queue manager objects.

The authorization specification tables starting in Table 1 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 data:

Action to be performed
: MQI option, MQSC command, or PCF command.

Access control object
: Queue, process definition, queue manager, namelist, channel, client connection channel, listener, service, or authentication information object.

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 **GRTMQMAUT** and **RVKMQMAUT** commands for the particular entity. For example, MQZAO_BROWSE corresponds to the keyword `*BROWSE`; similarly, the keyword MQZAO_SET_ALL_CONTEXT corresponds to the keyword *SETALL, and so on. These constants are defined in the header file cmqzc.h, which is supplied with the product.

MQI authorizations

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 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 can 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 name resolution that is not a queue-manager alias, unless the queue-manager alias definition is opened directly; that is, its name appears in the `ObjectName` field of the object descriptor. Authority is always needed for the particular 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.

Note: These tables do not mention namelists, channels, client connection channels, listeners, services, or authentication information objects. 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.

Table 1. Security authorization needed for MQCONN calls

Authorization required for:	**Queue object (1)**	**Process object**	**Queue manager object**
MQCONN option	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 (**2**)	MQZAO_INQUIRE (**2**)	MQZAO_INQUIRE (**2**)
MQOO_BROWSE	MQZAO_BROWSE	Not applicable	No check
MQOO_INPUT_*	MQZAO_INPUT	Not applicable	No check
MQOO_SAVE_ ALL_CONTEXT (**3**)	MQZAO_INPUT	Not applicable	Not applicable
MQOO_OUTPUT (Normal queue) (**4**)	MQZAO_OUTPUT	Not applicable	Not applicable
MQOO_PASS_ IDENTITY_CONTEXT (**5**)	MQZAO_PASS_ IDENTITY_CONTEXT	Not applicable	No check
MQOO_PASS_ALL_ CONTEXT (**5**, **6**)	MQZAO_PASS _ALL_CONTEXT	Not applicable	No check
MQOO_SET_ IDENTITY_CONTEXT (**5**, **6**)	MQZAO_SET_ IDENTITY_CONTEXT	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (**7**)
MQOO_SET_ ALL_CONTEXT (**5**, **8**)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (**7**)
MQOO_OUTPUT (Transmission queue) (**9**)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (**7**)
MQOO_SET	MQZAO_SET	Not applicable	No check
MQOO_ALTERNATE_ USER_AUTHORITY	(**10**)	(**10**)	MQZAO_ALTERNATE_ USER_AUTHORITY (**10**, **11**)

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 (**12**)	Not applicable	No check
MQPMO_PASS_ALL _CONTEXT	MQZAO_PASS_ ALL_CONTEXT (**12**)	Not applicable	No check
MQPMO_SET_ IDENTITY_CONTEXT	MQZAO_SET_ IDENTITY_CONTEXT (**12**)	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (**7**)
MQPMO_SET_ ALL_CONTEXT	MQZAO_SET_ ALL_CONTEXT (**12**)	Not applicable	MQZAO_SET_ ALL_CONTEXT (**7**)
(Transmission queue) (**9**)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (**7**)
MQPMO_ALTERNATE_ USER_AUTHORITY	(**13**)	Not applicable	MQZAO_ALTERNATE_ USER_AUTHORITY (**11**)

Table 4. Security authorization needed for MQCLOSE calls

Authorization required for:	**Queue object (1)**	**Process object**	**Queue manager object**
MQCO_DELETE	MQZAO_DELETE (**14**)	Not applicable	Not applicable
MQCO_DELETE _PURGE	MQZAO_DELETE (**14**)	Not applicable	Not applicable

Notes for the tables:

1. If a model queue is being opened:
   - 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. Either the queue, process, namelist, or queue manager object is checked, depending on the type of object being opened.
3. MQOO_INPUT_* must also be specified. This option is valid for a local, model, or alias queue.
4. This check is performed for all output cases, except the case specified in note 9.
5. MQOO_OUTPUT must also be specified.
6. MQOO_PASS_IDENTITY_CONTEXT is also implied by this option.
7. This authority is required for both the queue manager object and the particular queue.
8. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT, and MQOO_SET_IDENTITY_CONTEXT are also implied by this option.
9. 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).

10. At least one of MQOO_INQUIRE (for any object type), or (for queues) MQOO_BROWSE, MQOO_INPUT_*, MQOO_OUTPUT, or MQOO_SET 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.
11. This authorization allows any *AlternateUserId* to be specified.
12. An MQZAO_OUTPUT check is also carried out if the queue does not have a *Usage* queue attribute of MQUS_TRANSMISSION.
13. The check carried out is as for the other options specified, using the supplied alternate-user identifier for the named queue authority, and the current application authority for the MQZAO_ALTERNATE_USER_IDENTIFIER check.
14. 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 that returned the object handle being used.

    Otherwise, there is no check.

General notes:

1. The special authorization MQZAO_ALL_MQI includes all the following authorizations that are relevant to the object type:
    - MQZAO_CONNECT
    - MQZAO_INQUIRE
    - MQZAO_SET
    - MQZAO_BROWSE
    - MQZAO_INPUT
    - MQZAO_OUTPUT
    - MQZAO_PASS_IDENTITY_CONTEXT
    - MQZAO_PASS_ALL_CONTEXT
    - MQZAO_SET_IDENTITY_CONTEXT
    - MQZAO_SET_ALL_CONTEXT
    - MQZAO_ALTERNATE_USER_AUTHORITY
2. MQZAO_DELETE (see note 14) and MQZAO_DISPLAY are classed as administration authorizations. They are not therefore included in MQZAO_ALL_MQI.
3. *No check* means that no authorization checking is carried out.
4. *Not applicable* means that authorization checking is not relevant to this operation. For example, you cannot issue an MQPUT call to a process object.

Authorizations for MQSC commands in escape PCFs on i5/OS
These authorizations allow a user to issue administration commands as an escape PCF message. These methods allow a program to send an administration command as a message to a queue manager, for execution on behalf of that user.

Authorizations for PCF commands on i5/OS
These authorizations allow a user to issue administration commands as PCF commands. These methods allow a program to send an administration command as a message to a queue manager, for execution on behalf of that user.

Parent topic: Setting up security on i5/OS

16.3.1. Authorizations for MQSC commands in escape PCFs on i5/OS

These authorizations allow a user to issue administration commands as an escape PCF message. These methods allow a program to send an administration command as a message to a queue manager, for execution on behalf of that user.

This section 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 commands 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	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

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	
Service	

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 *object*

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 *object*

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 **GRTMQMAUT** command.
3. This option applies if the object to be replaced already exists. If it does not, the check is as for DEFINE *object* NOREPLACE.

Parent topic: Authorization specification tables for i5/OS

This build: January 26, 2011 11:20:30

Notices | Trademarks | Downloads | Library | Support | Feedback

16.3.2. Authorizations for PCF commands on i5/OS

These authorizations allow a user to issue administration commands as PCF commands. These methods allow a program to send an administration command as a message to a queue manager, for execution on behalf of that user.

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 the following authorizations:

- MQZAO_CHANGE
- MQZAO_CLEAR
- MQZAO_DELETE
- MQZAO_DISPLAY
- MQZAO_CONTROL
- MQZAO_CONTROL_EXTENDED

MQZAO_CREATE is not included as it 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
Topic	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_CHANGE
Service	MQZAO_CHANGE

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	
Service	

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

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

Stop 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

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 **GRTMQAUT** command.
3. For Create commands, MQZAO_DISPLAY authority is also needed for the appropriate SYSTEM.DEFAULT.* object.

4. This option 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: Authorization specification tables for i5/OS

16.4. Generic OAM profiles on i5/OS

Object authority manager (OAM) generic profiles enable you to set the authority a user has to many objects at once, rather than having to issue separate **GRTMQMAUT** commands against each individual object when it is created. Using generic profiles in the **GRTMQMAUT** command enables you to set a generic authority for all future objects created that fit that profile.

The rest of this section describes the use of generic profiles in more detail:

- Using wildcard characters
- Profile priorities

Using wildcard characters

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 created 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` would apply 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` would apply to the objects `ABC.DEF.JKL`, and `ABC.GHI.JKL`. (Note that it would **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` would apply 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 the keyword `OBJTYPE (*PRC)` 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.

Profile priorities

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:

```
GRTMQMAUT OBJ(AB.*) OBJTYPE(*Q) USER(FRED) AUT(*PUT) MQMNAME(MYQMGR)
GRTMQMAUT OBJ(AB.C*) OBJTYPE(*Q) USER(FRED) AUT(*GET) MQMNAME(MYQMGR)
```

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 GRTMQMAUT 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 than 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: Setting up security on i5/OS

16.5. Specifying the installed authorization service on i5/OS

You can specify which authorization service component to use.

The parameter **Service Component name** on **GRTMQMAUT** and **RVKMQMAUT** allows you to specify the name of the installed authorization service component.

Selecting **F24** on the initial panel, followed by **F9=All parameters** on the next panel of either command, allows you to specify either the installed authorization component (*DFT) or the name of the required authorization service component specified in the Service stanza of the queue manager's qm.ini file.

DSPMQMAUT also has this extra parameter. This parameter allows you to search all the installed authorization components (*DFT), or the specified authorization-service component name, for the specified object name, object type, and user

Parent topic: Setting up security on i5/OS

16.6. Working with and without authority profiles on i5/OS

Use this information to learn how to work with authority profiles and how to work without authority profiles.

You can work with authority profiles, as explained in Working with authority profiles, or without them, as explained here:

To work without authority profiles, use `*NONE` as an Authority parameter on **GRTMQMAUT** to create profiles without authority. This leaves any existing profiles unchanged.

On **RVKMQMAUT**, use `*REMOVE` as an Authority parameter to remove an existing authority profile.

Working with authority profiles

There are two commands associated with authority profiling:

- **WRKMQMAUT**
- **WRKMQMAUTD**

You can access these commands directly from the command line, or from the WRKMQM panel by:

1. Typing in the queue manager name and pressing the `Enter` key to access the **WRKMQM** results panel.
2. Selecting `F23=More options` on this panel.

Option 24 selects the results panel for the **WRKMQMAUT** command and option 25 selects the **WRKMQMAUTI** command, which is used with the SSL bindings layer.

WRKMQMAUT

This command allows you to work with the authority data held in the authority queue.

Note: To run this command you must have `*connect` and `*admdsp` authority to the queue manager. However, to create or delete a profile, you need QMQMADM authority.

If you output the information to the screen, a list of authority profile names, together with their types, is displayed. If you print the output, you receive a detailed list of all the authority data, the registered users, and their authorities.

Entering an object or profile name on this panel, and pressing ENTER takes you to the results panel for **WRKMQMAUT**.

If you select `4=Delete`, you go to a new panel from which you can confirm that you want to delete all the user names registered to the generic authority profile name you specify. This option runs **RVKMQMAUT** with the option *REMOVE for all the users, and applies **only** to generic profile names.

If you select `12=Work with profile` you go to the **WRKMQMAUTD** command results panel, as explained in WRKMQMAUTD.

WRKMQMAUTD

This command allows you to display all the users registered with a particular authority profile name and object type. To run this command you must have `*connect` and `*admdsp` authority to the queue manager. However, to grant, run, create, or delete a profile you need QMQMADM authority.

Selecting `F24=More keys` from the initial input panel, followed by option `F9=All Parameters` displays the Service Component Name as for **GRTMQMAUT** and **RVKMQMAUT**.

Note: The `F11=Display Object Authorizations` key toggles between the following types of authorities:

- Object authorizations
- Context authorizations
- MQI authorizations

The options on the screen are:

2=Grant
: Takes you to the **GRTMQMAUT** panel to add to the current authorities.

3=Revoke
: Takes you to the **RVKMQMAUT** panel to remove some of the current definitions

4=Delete
: Takes you to a panel that allows you to delete the authority data for specified users. This runs **RVKMQMAUT** with the option *REMOVE.

5=Display
: Takes you to the existing **DSPMQMAUT** command

F6=Create
: Takes you to the **GRTMQMAUT** panel that allows you to create a profile authority record.

Parent topic: Setting up security on i5/OS

16.7. Object Authority Manager guidelines for i5/OS

Additional hints and tips for using the Object Authority Manager (OAM)

Limit access to sensitive operations

Some operations are sensitive; limit them 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 copying application queues

Queue manager directories

The directories and libraries containing queues and other queue manager data are private to the product. Do not use standard operating system commands to grant or revoke authorizations to MQI resources.

Queues

The authority to a dynamic queue is based on, but is not necessarily the same as, that of the model queue from which it is derived.

For alias queues and remote queues, the authorization is that of the object itself, not the queue to which the alias or remote queue resolves. It is possible to authorize a user profile to access an alias queue that resolves to a local queue to which the user profile has no access permissions.

Limit the authority to create queues to privileged users. If you do not, users can bypass the normal access control simply by creating an alias.

Alternate-user authority

Alternate-user authority controls whether one user profile can use the authority of another user profile when accessing a WebSphere® MQ object. This technique is essential where a server receives requests from a program and the server 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:

- A server program running under user profile PAYSERV retrieves a request message from a queue that was put on the queue by user profile 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 profile (PAYSERV) to authorize opening the reply-to queue, the server can specify some other user profile, 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 profile when it opens the reply-to queue.

The alternate-user profile is specified on the *`AlternateUserId`* field of the object descriptor.

Note: You can use alternate-user profiles on any WebSphere MQ object. Use of an alternate-user profile does not affect the user profile used by any other resource managers.

Context authority

Context is information that applies to a particular message and is contained in the message descriptor, MQMD, which is part of the message.

For descriptions of the message descriptor fields relating to context, see the WebSphere MQ Application Programming Reference.

For information about the context options, see the WebSphere MQ Application Programming Guide.

Remote security considerations

For remote security, consider:

Put authority

For security across queue managers, you can specify the put authority that is used when a channel receives a message sent from another queue manager.

Specify the channel attribute PUTAUT as follows:

DEF

Default user profile. This is the QMQM user profile under which the message channel agent is running.

CTX

The user profile in the message context.

Transmission queues

Queue managers automatically put remote messages on a transmission queue; no special authority is required. However, putting a message directly on a transmission queue requires special authorization.

Channel exits

Channel exits can be used for added security.

For more information about remote security, see WebSphere MQ Intercommunication.

Protecting channels with SSL

You can use SSL to protect channel commands.

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 security you want, such as the encryption algorithm you want to use.

SSL support in WebSphere MQ uses the queue manager *authentication information object* and various CL and MQSC commands and queue manager and channel parameters that define the SSL support required in detail.

The following CL commands support SSL:

WRKMQMAUTI

Work with the attributes of an authentication information object.

CHGMQMAUTI

Modify the attributes of an authentication information object.

CRTMQMAUTI
Create an authentication information object.

CPYMQMAUTI
Create an authentication information object by copying an existing one.

DLTMQMAUTI
Delete an authentication information object.

DSPMQMAUTI
Displays the attributes for a specific authentication information object.

For an overview of channel security using SSL, see WebSphere MQ Security.

For details of PCF commands associated with SSL, see WebSphere MQ Programmable Command Formats and Administration Interface.

Parent topic: Setting up security on i5/OS

17. Keeping clusters secure

This chapter discusses the following security considerations:

- Stopping unauthorized queue managers sending messages to your queue manager
- Stopping unauthorized queue managers putting messages on your queues
- Authorizing putting messages on remote cluster queues
- Preventing queue managers joining a cluster
- Forcing unwanted queue managers to leave a cluster
- Using SSL

Stopping unauthorized queue managers sending messages to your queue manager

Stopping unauthorized queue managers putting messages on your queues

Authorizing putting messages on remote cluster queues
On z/OS set up authorization to a cluster queue using RACF. On other platforms, authorize access to cluster queues on each of the queue managers in the cluster.

Preventing queue managers joining a cluster

Forcing unwanted queue managers to leave a cluster

Using SSL

Parent topic: Security

17.1. Stopping unauthorized queue managers sending messages to your queue manager

To prevent selected queue managers from sending messages to your queue manager, define a channel security exit program on the CLUSRCVR channel definition. Write a program that authenticates queue managers trying to send messages on your cluster-receiver channel and denies them access if they are not authorized. Channel security exit programs are called at MCA initiation and termination. See the WebSphere MQ Intercommunication book for more information.

Clustering has no effect on the way security exits work. You can restrict access to a queue manager in the same way as you would in a distributed queuing environment.

Parent topic: Keeping clusters secure

17.2. Stopping unauthorized queue managers putting messages on your queues

To prevent certain queue managers from putting messages on a queue, use the security facilities available on your platform. For example:

- RACF or other external security managers on WebSphere® MQ for z/OS®
- The Object Authority Manager (OAM) on WebSphere MQ for i5/OS®, WebSphere MQ on UNIX systems, and WebSphere MQ for Windows, and on ›WebSphere MQ for HP NonStop Server v5.3‹ Compaq Tru64 UNIX, V5.1, ›WebSphere MQ for HP NonStop Server v5.3‹ Compaq OpenVMS Alpha, Version 5.1, and MQSeries® for Compaq NonStop Kernel, V5.1

In addition, you can use the PUT authority (PUTAUT) attribute on the CLUSRCVR channel definition. The PUTAUT attribute allows you to specify what user IDs are to be used to establish authority to put a message to a queue. The options on the PUTAUT attribute are:

DEF

Use the default user ID. On z/OS this might involve using both the user ID received from the network and that derived from MCAUSER.

CTX

Use the user ID in the context information associated with the message. On z/OS this might involve using either the user ID received from the network, or that derived from MCAUSER, or both. Use this option if the link is trusted and authenticated.

ONLYMCA (z/OS only)

As for DEF, but any user ID received from the network will not be used. Use this option if the link is not trusted and you want to allow only a specific set of actions on it, which are defined for the MCAUSER.

ALTMCA (z/OS only)

As for CTX, but any user ID received from the network will not be used.

For more information about using the PUTAUT attribute on a channel definition, see the WebSphere MQ Intercommunication book or see the WebSphere MQ Script (MQSC) Command Reference book.

Parent topic: Keeping clusters secure

This build: January 26, 2011 11:20:36

Notices | Trademarks | Downloads | Library | Support | Feedback

17.3. Authorizing putting messages on remote cluster queues

On z/OS® set up authorization to a cluster queue using RACF®. On other platforms, authorize access to cluster queues on each of the queue managers in the cluster.

About this task

The commands you use to manage authorization are different on WebSphere® MQ for z/OS from other platforms.

WebSphere MQ for z/OS

Use RACF to control whether your queue manager puts messages to a remote queue. With RACF, you can set up permissions for a named queue regardless of whether that queue exists on your system.

Platforms other than z/OS

On these platforms, you cannot control access to individual queues that do not exist on your queue manager. You can give an application put access to all queues in the cluster, subject to channel and local security settings at the remote queue manager, by giving `+put` access to the cluster transmission queue. This technique has the disadvantage that any application with this access can put to any queue in the cluster, which is probably undesirable. To control access to individual queues use aliases at the queue manager local to the putting application that resolve to queues in the cluster. Give them the appropriate authority for access, instead of the cluster transmission queue.

Procedure

- For z/OS, issue the following commands:

  ```
  RDEFINE MQQUEUE QMgrName.QUEUE.ObjectProfile UACC(NONE)
  PERMIT QMgrName.QUEUE.ObjectProfile CLASS(MQADMIN) ID(GroupName) ACCESS(UPDATE)
  ```

- On platforms other than z/OS, to give control access to all the queues in a cluster, issue the following commands:
  - For UNIX and Windows systems, issue the following commands:

    ```
    setmqaut -m QMgrName -t qmgr -g GroupName +connect +inq
    setmqaut -m QMgrName -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t queue -g GroupName +put
    ```

  - For i5/OS®, issue the following commands:

    ```
    GRTMQMAUT OBJ('QMgrName') OBJTYPE(*MQM) USER(GroupName) AUT(*CONNECT *INQ)
    GRTMQMAUT OBJ(SYSTEM.CLUSTER.TRANSMIT.QUEUE) OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
    ```

- On platforms other than z/OS, to give access to individual queues in the cluster, issue the following commands:
  1. Define an alias queue on your local queue manager, naming the cluster queue as the base queue.

     ```
     DEFINE QALIAS(AliasName) TARGET(BaseQueue)
     ```

  2. Give the group authority to connect to its local queue manager, and to open the alias queue for put. For UNIX and Windows systems, issue the following commands:

     ```
     setmqaut -m QMgrName -t qmgr -g GroupName +connect
     setmqaut -m QMgrName -t queue -n AliasName -g GroupName -all +put
     ```

     For i5/OS, issue the following commands:

     ```
     GRTMQMAUT OBJ('QMgrName') OBJTYPE(*MQM) USER(GroupName) AUT(*CONNECT)
     GRTMQMAUT OBJ('AliasName') OBJTYPE(*Q) USER(GroupName) AUT(*PUT) MQMNAME('QMgrName')
     ```

  The user can put messages only to the specified cluster queue, and no other cluster queues.
  The variable names have the following meanings:

  QMgrName

  The name of the queue manager. On z/OS, this value can also be the name of a queue-sharing group.

  ObjectProfile

  The name of the queue or generic profile for which to change authorizations. This is the local name of the remote queue or the name of the transmission queue, as appropriate.

  GroupName

  The name of the group to be granted access.

  AliasName

  Local name of the queue.

  BaseQueue

  The name of the queue being aliased.

What to do next

If you specify a reply-to queue when you put a message on a cluster queue, the consuming application must have authority to send the reply. If you trust the application, grant it `+put` authority to SYSTEM.CLUSTER.TRANSMIT.QUEUE. Otherwise, create a reply-to queue alias and grant `+put` authority to that.

Parent topic: Keeping clusters secure

Related concepts
Profiles for command security
Profiles for command resource security
Profiles to control queue-sharing group or queue manager level security

Related information
setmqaut
GRTMQMAUT
Security controls and options (z/OS)
Reply-to queue aliases

17.4. Preventing queue managers joining a cluster

If you want to ensure that only certain authorized queue managers attempt to join a cluster, you must either use a security exit program on the cluster-receiver channel, or write an exit program to prevent unauthorized queue managers from writing to SYSTEM.CLUSTER.COMMAND.QUEUE. Do not restrict access to SYSTEM.CLUSTER.COMMAND.QUEUE such that no queue manager can write to it, or you would prevent any queue manager from joining the cluster.

It is difficult to stop a queue manager that is a member of a cluster from defining a queue. Therefore, there is a danger that a rogue queue manager can join a cluster, learn what queues are in it, define its own instance of one of those queues, and so receive messages that it must not be authorized to receive.

To prevent a queue manager receiving messages, you can write:

- A channel exit program on each cluster-sender channel, which uses the connection name to determine the suitability of the destination queue manager to be sent the messages.
- A cluster workload exit program, which uses the destination records to determine the suitability of the destination queue and queue manager to be sent the messages
- A channel auto-definition exit program, which uses the connection name to determine the suitability of defining channels to the destination queue manager

Using security exits on cluster channels

Parent topic: Keeping clusters secure

17.4.1. Using security exits on cluster channels

When a cluster-sender channel is first started, it uses attributes defined manually by a system administrator. When the channel is stopped and restarted, it picks up the attributes from the corresponding cluster-receiver channel definition. The original cluster-sender channel definition is overwritten with the new attributes, including the SecurityExit attribute. Note the following:

1. You must define a security exit on both the cluster-sender end and the cluster-receiver end of a channel, in order for it to be effective. Even though the security exit name is sent over from the cluster-receiver definition, the initial connection must be made with a security-exit handshake.
2. In addition to the normal security-message handshake, the security exit must validate the PartnerName in the MQCXP structure. The exit must allow the channel to start only if the partner queue manager is authorized.
3. Design the security exit on the cluster-receiver definition to be *receiver initiated*. If you design it as *sender initiated*, an unauthorized queue manager without a security exit can join the cluster because no security checks are performed. Not until the channel is stopped and restarted can the SCYEXIT name be sent over from the cluster-receiver definition and full security checks made. Refer to the WebSphere MQ Intercommunication book for information about sender-initiated and receiver-initiated security exits.
4. To view the cluster-sender channel definition that is currently in use, use the command:
   ```
   DISPLAY CLUSQMGR(queue manager) ALL
   ```
   This displays the attributes that have been sent across from the cluster-receiver definition. To view the original definition, use the command:
   ```
   DISPLAY CHANNEL(channel name) ALL
   ```
5. If the queue managers are on different platforms, you might need to define a channel auto-definition exit (CHADEXIT) on the cluster-sender queue manager to set the SecurityExit attribute to an appropriate format for the target platform. For details of this, see Auto-definition of channels.
6. On z/OS® the security-exit load module must be in the data set specified in the CSQXLIB DD statement of the channel-initiator address-space procedure. On Windows the security-exit and channel auto-definition exit DLLs must be in the path specified in the SCYEXIT attribute of the channel definition or the CHADEXIT attribute of the queue manager definition respectively, or in the Registry.

Parent topic: Preventing queue managers joining a cluster

17.5. Forcing unwanted queue managers to leave a cluster

You can force an unwanted queue manager to leave a cluster. You might need to do this to tidy up, if for example, a queue manager is deleted but its cluster-receiver channels are still defined to the cluster.

Only full repository queue managers are authorized to eject a queue manager from a cluster. For example, to eject the queue manager OSLO from the cluster NORWAY, the full repository queue manager issues the command:

```
    RESET CLUSTER(NORWAY) QMNAME(OSLO) ACTION(FORCEREMOVE)
```

or

```
    RESET CLUSTER(NORWAY) QMID(qmid) ACTION(FORCEREMOVE)
```

The queue manager that is force removed does not change, it believes it is still in the cluster. All other queue managers will believe it has gone.

For more information on RESET CLUSTER see RESET CLUSTER.

Parent topic: Keeping clusters secure

17.6. Using SSL

This topic is discussed in the *WebSphere® MQ Security* book. The advice there is generally applicable to cluster channels, but you might want to give some special consideration to the following:

Note: SSL is available on the WebSphere MQ products only.

In a WebSphere MQ cluster a particular CLUSRCVR channel definition is frequently propagated to many other queue managers where it is transformed into an auto-defined CLUSSDR. Subsequently the auto-defined CLUSSDR is used to start a channel to the CLUSRCVR. If the CLUSRCVR is configured for SSL connectivity the following considerations apply:

- All queue managers that want to communicate with this CLUSRCVR must have access to SSL support. This SSL provision must support the CipherSpec for the channel.
- The different queue managers to which the auto-defined CLUSSDRs have been propagated will each have a different distinguished name associated. If distinguished name peer checking is to be used on the CLUSRCVR it must be set up so all of the distinguished names that can be received are successfully matched.
  For example, let us assume that all of the queue managers that will host CLUSSDRs which will connect to a particular CLUSRCVR, have certificates associated. Let us also assume that the distinguished names in all of these certificates define the country as UK, organization as IBM®, the organization unit as WebSphere MQ Development, and all have common names in the form DEVT.QMxxx, where xxx is numeric.
  In this case an SSLPEER value of `C=UK, O=IBM, OU=WebSphere MQ Development, CN=DEVT.QM*` on the CLUSRCVR will allow all the required CLUSSDRs to connect successfully, but will prevent unwanted CLUSSDRs from connecting.
- If custom CipherSpec strings are used, be aware that the custom string formats are not allowed on all platforms. An example of this is that the CipherSpec string RC4_SHA_US has a value of 05 on i5/OS® but is not a valid specification on UNIX or Windows. So if custom SSLCIPH parameters are used on a CLUSRCVR, all resulting auto-defined CLUSSDRs should reside on platforms on which the underlying SSL support implements this CipherSpec and on which it can be specified with the custom value. If you cannot select a value for the SSLCIPH parameter that will be understood throughout your cluster you will need a channel auto definition exit to change it into something the platforms being used will understand. Use the textual CipherSpec strings where possible (for example RC4_MD5_US).

An SSLCRLNL parameter applies to an individual queue manager and is not propagated to other queue managers within a cluster.

Upgrading clustered queue managers and channels to use SSL

Disabling SSL on clustered queue managers and channels

Parent topic: Keeping clusters secure

17.6.1. Upgrading clustered queue managers and channels to use SSL

1. The first step is to upgrade all queue managers in the cluster to V5.3 or higher, if they are not already at these levels, and to distribute the certificates and keys so that SSL works from each of them.
2. You then switch the CLUSRCVR channels to SSL in any order you like (the changes flow in the opposite direction over channels which are not changed to SSL). Make sure that you do not change the reverse path until the changes for the current channel have been distributed throughout the cluster. In other words, change one CLUSRCVR at a time, and allow the changes to flow through the cluster before changing the next.
3. When the CLUSRCVR changes are complete, you should change all manual CLUSSDR channels to SSL. This does not have any effect on the operation of the cluster, unless you use the REFRESH CLUSTER command with the REPOS(YES) option.

Parent topic: Using SSL

17.6.2. Disabling SSL on clustered queue managers and channels

If you want to disable SSL in your cluster, follow these steps:

1. Set the value of the SSLCIPH parameter to ' ' (an empty string in single quotes), or *NONE on i5/OS®.
2. Run the **RESET CLUSTER** command (**RSTMQMCL CLUSTER** on i5/OS) against each queue manager pointing to the other queue managers in the cluster. That is, if you have two queue managers in your cluster, run the RSTMQMCL command four times in total, twice for each queue manager.
   The RESET CLUSTER command replaces the auto-defined cluster send channels with the newer version. For example:
   On QM1:

```
RESET CLUSTER(clustername) QMNAME(QM1) ACTION(FORCEREMOVE)QUEUES(NO)
RESET CLUSTER(clustername) QMNAME(QM2) ACTION(FORCEREMOVE)QUEUES(NO)
```

On QM2:

```
RESET CLUSTER(clustername) QMNAME(QM2) ACTION(FORCEREMOVE)QUEUES(NO)
RESET CLUSTER(clustername) QMNAME(QM1) ACTION(FORCEREMOVE)QUEUES(NO)
```

On i5/OS, the equivalent command is:

```
RSTMQMCL CLUSTER(CLUSTER.NAME) QMNAME(QM1)ACTION(*FRCRMV) MQMNAME(QM1)
QUEUES(*NO)
RSTMQMCL CLUSTER(CLUSTER.NAME) QMNAME(QM2)ACTION(*FRCRMV) MQMNAME(QM1)
QUEUES(*NO)
RSTMQMCL CLUSTER(CLUSTER.NAME) QMNAME(QM1)ACTION(*FRCRMV) MQMNAME(QM2)
QUEUES(*NO)
RSTMQMCL CLUSTER(CLUSTER.NAME) QMNAME(QM2)ACTION(*FRCRMV) MQMNAME(QM2)
QUEUES(*NO)
```

3. Run the **REFRESH CLUSTER** command (**RFRMQMCL CLUSTER** on i5/OS) to reinstate the queue managers into the cluster. For example:
   On QM1:

```
REFRESH CLUSTER(clustername) REPOS(NO)
```

On QM2:

```
REFRESH CLUSTER(clustername) REPOS(NO)
```

On i5/OS, the equivalent command is:

```
RFRMQMCL CLUSTER(CLUSTER.NAME) MQMNAME(QM1) REPOS(*NO)
RFRMQMCL CLUSTER(CLUSTER.NAME) MQMNAME(QM2) REPOS(*NO)
```

Parent topic: Using SSL