WebSphere Event Broker

IBM

Configuration, Administration, and Security

Version 6 Release 0

WebSphere Event Broker

Configuration, Administration, and Security

Version 6 Release 0

IBM

Contents

About this topic collection

This PDF has been created from the WebSphere Event Broker Version 6.0 GA (September 2005) information center topics. Always refer to the WebSphere Event Broker online information center to access the most current information. The information center is periodically updated on the document update site and this PDF and others that you can download from that Web site might not contain the most current information.

The topic content included in the PDF does not include the "Related Links" sections provided in the online topics. Links within the topic content itself are included, but are active only if they link to another topic in the same PDF collection. Links to topics outside this topic collection are also shown, but these attempt to link to a PDF that is called after the topic identifier (for example, ac12340_.pdf) and therefore fail. Use the online information to navigate freely between topics.

Feedback: do not provide feedback on this PDF. Refer to the online information to ensure that you have access to the most current information, and use the Feedback link that appears at the end of each topic to report any errors or suggestions for improvement. Using the Feedback link provides precise information about the location of your comment.

The content of these topics is created for viewing online; you might find that the formatting and presentation of some figures, tables, examples, and so on are not optimized for the printed page. Text highlighting might also have a different appearance.

Part 1. Security

Security

The topics in this section help you to deal with security in WebSphere Event Broker:

- "Security overview"
- "Planning for security when you install WebSphere Event Broker" on page 8
- "Setting up broker domain security" on page 8
- "Setting up z/OS security" on page 29
- "Publish/subscribe security" on page 33
- "Securing the publish/subscribe domain" on page 41

Security overview

The role of the WebSphere Event Broker system administrator includes customizing and configuring, running utilities, performing problem determination, and collecting diagnostic materials. People involved in these activities need WebSphere Event Broker authorities. There is some security setup that you must do for WebSphere Event Broker to work properly. There is also some security setup that people typically do, for example, protecting queues against unauthorized use.

Security is split into several areas:
- "Authorization for configuration tasks"
- "Security for development resources" on page 4
- "Security for runtime resources" on page 4
- "Security exits" on page 5
- "SSL authentication" on page 5 (for the Real Time node only)
- "Tunneling" on page 7
- "Quality of protection" on page 7
- "Domain awareness" on page 8.

Authorization for configuration tasks

Authorization is the process of granting or denying access to a system resource. For WebSphere Event Broker, authorization is concerned with controlling who has permission to access WebSphere Event Broker resources, and ensuring that users who attempt to work with those resources have the necessary authorization to do so.

Examples of tasks that require authorization are:
- Configuring a broker using, for example, the mqsicreatebroker command.
- Accessing queues, for example, putting a message to the input queue of a message flow.
- Taking actions within the workbench, for example, deploying a message flow to an execution group.
- Publishing topics and subscribing to topics, as described in "Enabling topic-based security" on page 25

Authorization for existing resources

Your system administrator might decide to continue working with the local groups: **mqbrasgn**, **mqbrdevt**, **mqbrops**, and **mqbrtpic** to secure access to resources from the workbench. If you want to use the security facilities of Version

6.0, when you require access to a resource that already exists, the user ID with which you work must be included in the Access Control List (ACL) for that resource.

For authorization to create resources, for example brokers, you must be a member of the **Administrator** local group.

The security architecture of WebSphere Event Broker is platform independent. If you are running in a heterogeneous environment, ensure that you limit all the principals you define for WebSphere Event Broker tasks to eight characters or fewer. If you have a Windows-only environment, you can create principals of up to twelve characters, but only use these longer names if you are sure that you will not later include a Linux, UNIX, or z/OS system in your WebSphere Event Broker network.

Security for development resources

Security for development resources must be addressed and defined by the application development organization. No special facilities are provided by WebSphere Event Broker to address this environment over and above those available for a production environment.

Security for runtime resources

Runtime resources are WebSphere Event Broker objects that exist at run time. Runtime security controls your permission to take actions on those resources from the workbench. For example, the user ID that you use to deploy a message flow to an execution group must have permission to take that deploy action.

In WebSphere Event Broker, each runtime object has an Access Control List (ACL). The ACL for an object determines the view and modify permissions that a principal has for that object. ACLs provide more granularity and give you a greater degree of control over the permissions that principals have. ACLs also enable a single Configuration Manager in a single security domain to control brokers with incompatible access control, such as development, system test, and production.

WebSphere Event Broker allows you to control access by object as opposed to by group. For example, user JUNGLE\MPERRY might be given access to modify BROKERA, but have no access rights to BROKERB. In a further example the same user might have access to deploy to execution group EXEGRP1, but not to EXEGRP2, even though they are both members of BROKERA.

Access level control is based on the operation requested on any type of WebSphere Event Broker object, for example, brokers and execution groups. Access control enables permission assignment to principals (users and groups), and these permissions specify view and modify rights for deployment configuration resources. In WebSphere Event Broker, administrators can configure ACLs to use their organization's principals.

This model provides full object level access based on the action that needs to be performed, provided the necessary group and access entry has been created, and the system administrator can nest local and domain groups to help organize security requirements and users.

You can still operate with any existing group level security environments from earlier releases of WebSphere Event Broker without any migration issues. If you have any existing user role definition groups with members, you can continue to

use this model without making any additional changes. The object level security model is only invoked if the user role definition groups are either empty or do not exist. If you want to implement object level security, you must ensure that the user role definition groups are empty.

In the workbench, when you attempt an operation for which you require permission, the following information is passed to the Configuration Manager:
- Object type
- Object name
- Requested action
- Your user ID.

The Configuration Manager checks the ACL table. If your user ID is included in the ACL for the named object, you have permission to perform the operation.

Refer to Related reference information below for descriptions of the tools that system administrators use to control the ACLs.

Security exits

You can use security exit programs to verify that the partner at the other end of the channel is genuine.

When you start the Message Brokers Toolkit, it does not invoke a security exit to monitor its connection to the Configuration Manager by default. If you want to protect access to the Configuration Manager from client programs, you can use the WebSphere MQ channel security exit facility. This works in a standard way for the Configuration Manager end of the connection. You can find information about how to set up the security exit at the server end of the connection in the *WebSphere MQ Intercommunication* manual.

If you want to use a security exit, define when creating a new domain within the Domains view of the Message Brokers Toolkit to provide connection security. These security exits are standard WebSphere MQ security exits, written in Java, and implementing the WebSphere MQ exit interface.

For an overview of security exits, refer to the section "Channel security exit programs" in the chapter "Channel exit programs" in the *WebSphere MQ Intercommunication* manual on the WebSphere MQ library Web page.

SSL authentication

SSL authentication is available for the Real Time node, the HTTP listener, and the WebSphere MQ Java Client.

SSL authentication for the Real Time node

SSL authentication in WebSphere Event Broker supports an authentication protocol known as mutual challenge-response password authentication. This is a non-standard variant of the industry standard SSL protocol in which the public key cryptography called for by SSL is replaced by symmetric secret key cryptography. While this protocol is both secure and convenient to administer, it might be better to use the industry standard SSL protocol exactly as defined, especially if a public key cryptography infrastructure is already deployed for other purposes. There are two standardized versions of SSL which are:

Asymmetric SSL

This is used by most Web browsers. In this protocol, only the brokers have public/private key pairs and clients know the brokers' public keys. The SSL protocol establishes a secure connection in which the broker is authenticated to the client using public key cryptography, after which the client can send its password, encrypted by a secure session key, to authenticate itself to the broker.

Symmetric SSL

This is where both participants have public/private key pairs. The SSL protocol uses public key cryptography to accomplish mutual authentication.

In both instances, SSL authentication does not keep the SSL protocol up for the entire lifetime of a connection, because that would incur protection overheads on all messages. The SSL protocol remains in force long enough to accomplish mutual authentication and to establish a shared secret session key that can be used by message protection (see "Message protection" on page 40). Messages are then individually protected in accordance with the protection level specified for the given topic.

The SSL protocol implementation requires a Public-Key Cryptography Standards (PKCS) file, containing X.509 V3 certificates for the broker's private key, and possibly the public keys of clients and other brokers. This file, called the key ring file, must contain at least one certificate for the broker and for the trusted certification authority (CA) that issued and signed the broker's certificate. For the R form of SSL, the key ring file can also have the public keys of clients and other brokers that need to be authenticated, and the certificates supporting those public keys. However, the SSL protocol calls for the exchange of public keys and certificates, so key ring files do not need to be fully primed in this fashion, as long as there are enough commonly-trusted authorities to ensure that authentication completes.

By convention, key ring files are encrypted and protected by a passphrase, which is stored in a second file. The passphrase file requires careful protection using operating system mechanisms to ensure that it is not exposed to unauthorized observers. An observer who learns the passphrase can learn the private keys in the key ring file. However, only the passphrase file needs to be secure in this way and the key ring file is protected by the passphrase. Only private keys are sensitive. Other information in the key ring file, such as the broker's certificates, can be revealed without compromising security.

For more information on SSL authentication for the Real Time node, see "Enabling SSL for the Real Time node" on page 17.

SSL authentication for the HTTP listener

For information on SSL authentication for the HTTP listener, see "Enabling SSL for the HTTP listener" on page 16.

SSL authentication for the MQ Java Client

For information on SSL authentication for the MQ Java Client, see "Enabling SSL on the WebSphere MQ Java Client" on page 16.

Tunneling

When implementing WebSphere Event Broker, both the clients and their brokers can reside on different intranets, that is, separate organizational entities. This causes problems when a client attempts to connect to a broker. Tunneling addresses this problem where a broker's firewall has been configured to allow incoming connections from clients. Two options are provided for a client to connect through its own firewall to a broker with both methods achieving the same result, these are:

HTTP tunneling

This is suitable for applets where, due to sandbox security, an attempt to connect explicitly to an HTTP proxy server would be rejected. HTTP tunneling uses the Web support in Web browsers and connects through the proxy as if it were connecting to a Web site.

Activating HTTP tunneling support is configured on each node. Once a node has been configured to use HTTP tunneling, all client connections to that node must use this method of connection. Clients that don't will be rejected when an attempt to connect is made.

HTTP tunneling is not supported in conjunction with SSL authentication.

Connect via proxy

This is not suitable for applets. It is suitable for use where there are no sandbox security restrictions. It connects directly to the proxy and uses Internet protocols to request that the proxy forwards the connection to the broker. This option does not work in applets where the security manager rejects an explicit connection to the proxy.

Quality of protection

In Internet deployments, cryptographically-based protection of messages enhances security by preventing tampering and eavesdropping by hackers. The authentication services provided by WebSphere Event Broker ensure that only legitimate event broker servers and clients can connect to each other. However, a hacker might still be able to observe messages in transit or tamper with messages on established connections. Message protection provides security against these kinds of attacks.

Message protection consumes processor time and can slow system throughput. However, not all messages are equally sensitive, so message protection is configurable on a per-topic basis, so that you get only the protection you really need. Some topics might get no message protection at all, others might get channel integrity (making it impossible for hackers to insert or delete messages undetected), or message integrity (making it impossible for hackers to alter messages undetected), or message privacy (making it impossible for hackers to observe message contents). The protection levels are cumulative. For example, if you request message privacy you get message integrity and channel integrity as well. If you request message integrity you also get channel integrity. The higher levels of protection consume more resources than the lower levels.

You can also set message protection on internal system topics. Unlike user topics this must be either enabled on all topics, or on none.

Domain awareness

Enabling domain awareness means that access to the Message Brokers Toolkit is not restricted to users from one domain. With domain awareness enabled, users from different domains who are either trusted by the primary domain, or in a Windows 2000 transitive trust relationship, can perform Message Brokers Toolkit tasks, provided that they are in the correct user role definition groups or object level security groups.

When using domain awareness:
- Domain information is retrieved by the Configuration Manager.
- Membership of user role definition/object level security groups is not restricted to users from one domain.
- Nesting in user role definition/object level security groups is supported.

Planning for security when you install WebSphere Event Broker

On Linux and UNIX systems, you must complete security tasks before you install WebSphere Event Broker; these tasks are described in Installation Guide. On Windows systems, security tasks are completed during and after installation.

Always refer to the the Installation Guide for the latest information about installation tasks.

After installation, refer to "Creating user IDs" on page 9 for further security considerations.

Setting up broker domain security

This section introduces the steps that you must take when considering the security requirements for a broker configuration running on Windows, Linux, or UNIX platforms:
- "Creating user IDs" on page 9
- "Considering security for a broker" on page 12
- "Considering security for a Configuration Manager" on page 13
- "Considering security for the workbench" on page 10
- "Enabling topic-based security" on page 25
- "Using security exits" on page 28
- "Database security" on page 29

This section does not apply to z/OS. Refer to "Setting up z/OS security" on page 29 and "Summary of required access (z/OS)" on page 337 for information about setting up broker domain security on z/OS.

You can use these steps as a security checklist. Each step comprises a list of questions about the security tasks to consider for your broker configuration. The answers to the questions provide the security information that you need to configure your WebSphere Event Broker components and also give you information about other security controls that you might want to deploy.

The following steps show you what to do the first time you install WebSphere Event Broker.

1. If you plan to use WebSphere Event Broker within a domain environment, but are not running with domain awareness enabled, check that the global group Domain **mqbrkrs** exists. If necessary, create the global group.

2. Add your installation user ID, which must have **Administrator** authority, to the **mqbrkrs** local group.

3. When you migrate from Version 2.1 and you want to use access control lists (ACLs), ensure that you remove unauthorized user IDs from the **mqbrkrs** group.

To set up security for your broker domain, begin with "Planning for security when you install WebSphere Event Broker" on page 8 and follow links from there.

Creating user IDs

When you are planning the administration of your broker configuration, consider defining user IDs for the following roles:

- Administrator user IDs that can issue mqsi* commands. Refer to:
  - "Deciding which user accounts can execute broker commands" on page 12
  - "Deciding which user accounts can execute User Name Server commands" on page 26
  - "Deciding which user accounts can execute Configuration Manager commands" on page 14
- Service user IDs under which components run. Refer to:
  - "Deciding which user account to use for the broker service ID" on page 12
  - "Deciding which user account to use for the User Name Server service ID" on page 27
  - "Deciding which user account to use for the Configuration Manager service ID" on page 14

You must add broker service user IDs to the **mqbrkrs** local group.

- User IDs that access broker databases:
  1. Select the user IDs with which you intend to access the database used by your broker. Ensure that the user ID is not more than eight characters long. When you use a DB2 database, use a local user ID to access the database. If you create brokers in a domain environment where you use a domain user ID for the service user ID, set the database ID to a local ID that is authorized to access databases.
  2. Authorize your selected user IDs to access the broker and Configuration Manager databases. Refer to "Authorizing access to the databases" on page 84 for more information.
  3. When you create your broker, identify the selected user ID using the -u and -p options on the **mqsicreatebroker** command.
- Workbench users.
- Publishers and subscribers. Refer to "Enabling topic-based security" on page 25.

If you are running a Configuration Manager with one user ID and a broker with a different user ID on another system, you might see an error message when you deploy message flows to the broker. To avoid this, complete the following steps:

- Ensure that the broker's user ID is a member of the **mqm** and **mqbrkrs** groups.
- Define the broker's user ID on the system on which the Configuration Manager is running.
- Define the Configuration Manager's user ID on the system on which the broker is running.

- Ensure that all IDs are in lowercase so that they are compatible between computers.

Considering security for the workbench

During this task you consider the factors for deciding which users can take actions within the workbench.

Consider the following:

1. "Are you running with domain awareness enabled?"
2. "Are you running with domain awareness disabled?" on page 11
3. "Securing the channel between the workbench and the Configuration Manager" on page 11

Ensure that the IDs of the users who will run the workbench are not more than eight characters long.

Are you running with domain awareness enabled?

It is recommended that you run with domain awareness enabled. With this option, the domain information for a workbench user is flowed with the userid to the Configuration Manager for increased security. Assume that you are running the Configuration Manager on a computer named WKSTN1, which is a member of a domain named DOMAIN1. Users from DOMAIN2 also want to use the workbench. Perform the following steps:

1. Add any domain users or groups to the local group names that you will be using in your ACLs.
2. When you create the Configuration Manager use the -m option on the **mqsicreateaclentry**command to ensure that the domain is considered when verifying the user.

When you start the workbench, it automatically sends the domain information for your user ID to the Configuration Manager. Enable domain awareness in the Configuration Manager to access domain information.

Note: If you are running a Configuration Manager with one user ID and a broker with a different user ID on another computer, you might see an error message when trying to deploy message flows to the broker. To avoid this, do the following:

- Ensure that the broker's user ID is a member of the **mqm** and **mqbrkrs** groups.
- Define the broker's user ID on the computer where the Configuration Manager is running.
- Define the Configuration Manager's user ID on the computer where the broker is running.
- Ensure that all IDs are in lowercase so that they are compatible between computers.

Go to "Securing the channel between the workbench and the Configuration Manager" on page 11.

Are you running with domain awareness disabled?

You can set domain awareness to disabled, but running with this option means that the domain information for the workbench user is not flowed with the userid information, thus reducing security. It is therefore recommended that you run with domain awareness enabled.

You can use the -a option on the **mqsicreateaclentry**command to allow a user to be verified without considering the domain.

To set domain awareness to disabled, answer the following questions:
1. Are your workbench users drawn from a local domain?
    a. No: Go to the next question.
    b. Yes: Add any users to the local groups that you will be using in your ACLs.
       Go to "Securing the channel between the workbench and the Configuration Manager."
2. Are your workbench users drawn from another domain?
    a. Yes: Make the other domain a trusted domain of the Configuration Manager's computer then add the groups and users from the trusted domain to the local groups of the Configuration Manager.

For additional security, run with both domain awareness and security exits enabled. For more information about security exits, refer to "Security exits" on page 5.

Turning off the toolkit domain awareness

The toolkit sends the user and domain name to the Configuration Manager queue manager, regardless of the domain awareness setting on the Configuration Manager. This can cause problems connecting to the queue manager because of the security required to connect, put or get messages.

To turn off the domain awareness on the toolkit, run the toolkit in the following way:
1. Change to the <install_wbimb_dir>\eclipse directory.
2. Run the toolkit using the command **mqsistudio -vmargs -DDomainAware=0**.

Alternatively, modify the shortcut that runs the toolkit and add on **-vmargs -DDomainAware=0**.

Go to "Securing the channel between the workbench and the Configuration Manager"

Securing the channel between the workbench and the Configuration Manager
Create and enable a pair of security exits to run at the workbench and Configuration Manager ends of the connection. Use these exits to verify workbench users with the Windows security manager on the Configuration Manager computer.

For more information about creating and enabling security exits, refer to "Security exits" on page 5.

Considering security for a broker

Consider the factors for deciding which users can:
- Execute broker commands
- Control security for other broker resources.

Consider the following steps:
1. "Deciding which user accounts can execute broker commands"
2. "Deciding which user account to use for the broker service ID"
3. "Setting security on the broker's queues" on page 13
4. "Enabling topic-based security in the broker" on page 13

Deciding which user accounts can execute broker commands

Decide what permissions are required for the user IDs that:
- Create, change, list, delete, start, and stop brokers
- Display, retrieve, and change trace information.

Answer the following questions:
1. Is your broker installed on a Linux or UNIX operating system?
    a. No: Go to the next question.
    b. Yes: Go to "Deciding which user account to use for the broker service ID"
2. Are you executing broker commands under a Windows local account?
    a. No: Go to the next question.
    b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a broker, ensure that your user ID is defined in your local domain. When you create or start a broker, ensure that your user ID is a member of WKSTN1\Administrators.

        Go to "Deciding which user account to use for the broker service ID."
3. Are you executing broker commands under a Windows domain account?
    a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you create a broker using, for example, DOMAIN1\user1, ensure that DOMAIN1\user1 is a member of WKSTN1**Administrators**.

        Go to "Deciding which user account to use for the broker service ID."

Deciding which user account to use for the broker service ID

When you set the service ID with the -i option on the mqsicreatebroker or mqsichangebroker command, you determine the user ID under which the broker component process runs.

Answer the following questions:
1. Is your broker installed on a Linux or UNIX operating system?
    a. No: Go to the next question.
    b. Yes: Go to "Setting security on the broker's queues" on page 13
2. Do you want your broker to run under a Windows local account?
    a. No: Go to the next question.
    b. Yes: Ensure that your user ID is defined in your local domain and is a member of **mqbrkrs**.

Go to "Setting security on the broker's queues"

3. Do you want your broker to run under a Windows domain account?

   a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a broker using, for example, DOMAIN1\user1, ensure that: DOMAIN1\user1 is a member of DOMAIN1**Domain mqbrkrs** and DOMAIN1**Domain mqbrkrs** is a member of WKSTN1**mqbrkrs**.

      Go to "Setting security on the broker's queues"

Setting security on the broker's queues

When you run the mqsicreatebroker command, the **mqbrkrs** group gets access authority to the following queues:

   SYSTEM.BROKER.ADMIN.QUEUE

   SYSTEM.BROKER.CONTROL.QUEUE

   SYSTEM.BROKER.EXECUTIONGROUP.QUEUE

   SYSTEM.BROKER.EXECUTIONGROUP.REPLY

   SYSTEM.BROKER.INTERBROKER.QUEUE

   SYSTEM.BROKER.MODEL.QUEUE

- If you have created WebSphere Event Broker components to run on different queue managers, set the permissions for the transmission queues that you define to handle the message traffic between the queue managers. Grant `put` and `setall` authority to the local **mqbrkrs** group, or to the service user ID of the component supported by the queue manager on which you defined the transmission queue.
- Ensure that the broker's service user ID has authority to:
  1. Get messages from each input queue included in a message flow
  2. Put messages to any output, reply, and failure queues included in a message flow.
- Ensure that the user IDs under which applications that interact with the broker run have authority to:
  1. Put messages to each input queue included in a message flow
  2. Get messages from any output, reply, and failure queue included in a message flow.
- Go to "Enabling topic-based security in the broker"

Enabling topic-based security in the broker

Perform this task by responding to the following question:

Do you want to enable topic-based security in the broker?
1. Yes: Go to "Enabling topic-based security" on page 25
2. No: Go to "Considering security for a Configuration Manager"

Considering security for a Configuration Manager

During this task you consider the group membership that is required for:
- Users that execute Configuration Manager commands
- Service IDs
- Access to the Configuration Manager's queues.

An ACL is associated with the Configuration Manager itself. Users or groups that have full-control membership of the Configuration Manager's ACL implicitly have full-control membership of all other ACLs. Full-control membership of the Configuration Manager's ACL also allows users or groups to modify the ACLs for any object, including the Configuration Manager.

Consider the following:

1. "Deciding which user accounts can execute Configuration Manager commands"
2. "Deciding which user account to use for the Configuration Manager service ID"
3. "Setting security on the Configuration Manager's queues" on page 15
4. "Running the Configuration Manager in a domain environment" on page 15

Deciding which user accounts can execute Configuration Manager commands

During this task you decide what permissions are required for the user IDs that:
- Create, change, list, delete, start, and stop a Configuration Manager
- Display, retrieve, and change trace information.

Answer the following questions:

1. Are you executing Configuration Manager commands under a Windows local account?

   a. No: Go to the next question.

   b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a Configuration Manager, ensure that your user ID is defined in your local domain. When you create or start a Configuration Manager, ensure that your user ID is a member of WKSTN1**Administrators**.

   Go to "Deciding which user account to use for the Configuration Manager service ID"

2. Are you executing Configuration Manager commands under a Windows domain account?

   a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you create a Configuration Manager using, for example, DOMAIN1\user1, ensure that DOMAIN1\user1 is a member of WKSTN1**Administrators**.

   Go to "Deciding which user account to use for the Configuration Manager service ID"

Deciding which user account to use for the Configuration Manager service ID

When you set the service ID with the -i option on the mqsicreateconfigmgr or mqsichangeconfigmgr command, you determine the user ID under which the Configuration Manager component process runs.

Answer the following questions:

1. Do you want your Configuration Manager to run under a Windows local account?

   a. No: Go to the next question.

b. Yes: Ensure that your user ID is defined in your local domain, is a member of **mqbrkrs**, and is a member of **mqm**.

Go to "Setting security on the Configuration Manager's queues."

2. Do you want your Configuration Manager to run under a Windows domain account?

a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a Configuration Manager using, for example, DOMAIN1\user1, ensure that: DOMAIN1\user1 is a member of DOMAIN1**Domain mqbrkrs**, DOMAIN1\user1 is a member of WKSTN1**mqm** and DOMAIN1**Domain mqbrkrs** is a member of WKSTN1**mqbrkrs**. Alternatively, you can:

   1) Define user1 in DOMAIN1
   2) Add user1 to the DOMAIN1**Domain mqm** group
   3) Add the DOMAIN1**Domain mqm** group to the WKSTN1**mqm** group
   4) Add the DOMAIN1**Domain mqbrkrs** group to the WKSTN1**mqbrkrs** group

Go to "Setting security on the Configuration Manager's queues."

Setting security on the Configuration Manager's queues

When you run the command, the **mqbrkrs** group gets access authority to the following queues:

SYSTEM.BROKER.CONFIG.QUEUE
SYSTEM.BROKER.CONFIG.REPLY
SYSTEM.BROKER.ADMIN.REPLY
SYSTEM.BROKER.SECURITY.QUEUE
SYSTEM.BROKER.MODEL.QUEUE.

The broker and the User Name Server require access to the Configuration Manager's queues.

Each group or user for which you create access control lists (ACLs), gets access authority to the following queues:

SYSTEM.BROKER.CONFIG.QUEUE
SYSTEM.BROKER.CONFIG.REPLY.

Go to "Running the Configuration Manager in a domain environment."

Running the Configuration Manager in a domain environment

- If you want to enable domain awareness, go to "Are you running with domain awareness disabled?" on page 11.
- If you want to disable domain awareness, go to "Are you running with domain awareness enabled?" on page 10.

Changing the security domain for the Configuration Manager

You can change the security domain only if you are not using domain awareness. To change the domain currently in use for your Configuration Manager, use the "mqsichangeconfigmgr command" on page 228.

Changing the security domain for the User Name Server

You can change the security domain only if you are not using domain awareness. To change the security domain currently in use for your User Name Server, use the "mqsichangeusernameserver command" on page 250.

Implementing SSL authentication

This topic contains instructions for implementing SSL authentication in three ways:
- "Enabling SSL for the HTTP listener"
- "Enabling SSL on the WebSphere MQ Java Client"
- "Enabling SSL for the Real Time node" on page 17

Enabling SSL for the HTTP listener

To enable this feature, use the following method:

Use the "mqsicreateusernameserver command" on page 275 or "mqsichangeusernameserver command" on page 250 to enable the SSL authentication options.

Enabling SSL on the WebSphere MQ Java Client

The WebSphere MQ Java Client supports SSL-encrypted connections over the SVRCONN channel between the application and the queue manager. This topic tells you how to make use of this SSL support when communicating between the Configuration Manager Proxy (CMP) and the Configuration Manager.

For one-way authentication (with the client (Configuration Manager Proxy) authenticating the server (Configuration Manager) only) perform the following steps:

1. Generate or obtain all the appropriate keys and certificates. This includes a signed pkcs12 certificate for the server and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the pkcs12 certificate to the queue manager certificate store and assign it to the queue manager. Use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer).
3. Add the certificate of the certificate authority to the JSEE truststore of the Java Virtual Machine (JVM) at the Configuration Manager Proxy end using a tool such as Keytool.
4. Decide which cipher suite to use.
5. Change the properties on the SYSTEM.BKR.CONFIG channel, using WebSphere MQ Explorer, to specify the cipher suite to be used.
6. Add the required parameters (cipher suite, for example) to the Configuration Manager Proxy. If a truststore other than the default is used, its full path must be passed in via the truststore parameter.

When you have performed these steps, the Configuration Manager Proxy will connect to the Configuration Manager if it has a valid signed key that has been signed by a trusted certificate authority.

For two-way authentication (with the Configuration Manager also authenticating the Configuration Manager Proxy) perform the following additional steps:

1. Generate or obtain all the appropriate keys and certificates. This includes a signed pkcs12 certificate for the client and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the certificate of the certificate authority to the queue manager certificate store; use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer).
3. Set the SYSTEM.BKR.CONFIG channel to always authenticate. You can use WebSphere MQ Explorer to do this or SSLCAUTH(REQUIRED) in runmqsc.
4. Add the pkcs12 certificate to the JSEE keystore of the JVM at the Configuration Manager Proxy end using a tool such as Keytool.
5. If not using the default keystore, its full path must be passed into the Configuration Manager Proxy via the keystore parameter

When you have performed these steps, the Configuration Manager allows the Configuration Manager Proxy to connect only if the Configuration Manager Proxy has a certificate signed by one of the certificate authorities in its keystore.

Further restrictions can be made using the sslPeerName field; for example, you can allow connections only from certificate holders with a specific company or department name in their certificates. In addition, you can invoke a security exit for communications between the Configuration Manager Proxy and the Configuration Manager; see "Using security exits" on page 28.

Enabling SSL for the Real Time node

WebSphere Event Broker authentication services provide an optional facility that is supported between JMS clients and Real-timeInput nodes of WebSphere Event Broker.

In a default configuration, authentication services are disabled.

To configure the product to use the authentication services, complete the following steps.
- Configure and start a User Name Server in a broker domain
- Configure each Real-timeInput node to use authentication and set your chosen authentication protocol in each of the brokers that are to use the authentication services
- Edit a file that specifies client user Ids and passwords
- Specify the names of the files that are required to implement the SSL protocol

Configuring the User Name Server:

The User Name Server distributes to the brokers the information (specifically, passwords) that is required to support these authentication protocols.

To configure the User Name Server to support authentication, two parameters are provided for the **mqsicreateusernameserver** and **mqsichangeusernameserver** commands.

The first parameter, *AuthProtocolDataSource*, describes the location of an operating system file that contains the information that is required to support the authentication protocols.

The second parameter, the -jflag, indicates whether the file pointed to by the*AuthProtocolDataSource* parameter contains group and group membership information as well as password information.

- Set the -j flag if you want to support both authentication and publish/subscribe access control in your broker domain, and you want to draw user and group information from a flat file rather than from the operating system.
- Use the *AuthProtocolDataSource* parameter to specify the source of any protocol-related information. For example, you can specify the name of a file that contains user ID and password information. The user ID and password information in this file must exactly mirror the operating system user ID and password definitions. Make sure that you set the appropriate file system security for this password file.
- The default location of this file is the WebSphere Event Broker home directory. If you store the file somewhere else, give the full path definition of the location of the file.
- For the change to the User Name Server to take effect you must stop and then restart the User Name Server.

The **mqsichangeusernameserver** command also supports a-d flag to disable the option.

Configuring a broker:

Configure a broker to support WebSphere Event Broker authentication services. You need to specify two authentication and access control parameters and use the workbench to configure the appropriate Real-timeInput nodes and the sets of protocols that are to be supported on the broker.

The following steps show you how to do this.
- Switch to the Broker Administration perspective.
- For each message flow in the Message Flow Topology:
  1. Right-click the Real-timeInput node.
  2. Click properties.
  3. Check the Authentication box.
- For each broker in the Broker Topology:
  1. Right-click the broker.
  2. Click properties.
  3. Fill in the Authentication Protocol Type field

     Choose any combination of the options P, M, S, and R; for example, S, SR, RS, R, PS, SP, PSR, SRM, MRS, and RSMP are all valid combinations of options.

     The order in which you specify the options is significant; the broker chooses the first option that the client supports. If you want the broker to always support the strongest protocol that the client supports, choose RSMP.
  4. If you have chosen S or R as one of the options in the Authentication Protocol Type field, specify the SSL Key Ring File Name and the SSL Password File Name.
  5. Click OK.
  6. Use the **mqsicreatebroker** or **mqsichangebroker** command, with the following two parameters, to configure the broker:

UserNameServerQueueManagerName (-s)

This parameter specifies the name of the queue manager associated with the User Name Server. This parameter must be specified if you require authentication services, publish/subscribe access control services, or both types of service.

Publish/Subscribe Access Control Flag (-j)

This flag must be set, in addition to specifying the UserNameServerQueueManagerName parameter, if you want to use publish/subscribe access control services.

There is no corresponding flag for using the authentication services in the broker; it is enabled at the IP input node level.

Sample password files:

Two sample files, named password.dat and pwgroup.dat, are shipped with WebSphere Event Broker.

pwgroup.dat is a sample file that can be used when you set the -j flag.

password.dat is a sample file that can be used in the default case.

password.dat has the following layout:

```
# This is a password file.

# Each line contains two required tokens delimited by
# commas. The first is a user ID, the second is that user's
# password.

#USERNAME PASSWORD
========================
subscriber,subpw
admin,adminpw
publisher,pubpw
```

This file complements the user and group information drawn by the User Name Server from the operating system. User names that are defined in the file, but not the operating system, are treated as unknown by the broker domain. User names that are defined in the operating system, but are not defined in the password file, are denied access to the system.

pwgroup.dat contains group information as well as user and password information. Each user entry includes a list of group names that specify the groups that contain the user.

pwgroup.dat has the following layout:

```
#This is a password file.
#Each line contains two or more required tokens delimited by
#commas.The first is a user ID and the second is that user's
#password. All subsequent tokens
#specify the set of groups that the user belongs to.

#USERNAME PASSWORD  GROUPS
subscriber,subpw,group1,group2,group3
admin,adminpw,group2
publisher,pubpw,group2,group4
```

As mentioned above, this file can be used to provide the only source of user, group, and password information for the broker domain.

To deploy updated user and password information to the broker network if this information is drawn from an operating system file, stop the User Name Server and brokers, update the file, and then restart the User Name Server and brokers.

If passwords are drawn from the operating system, updates are automatically distributed to the brokers. Use normal operating system management tools to change users or passwords.

Authentication in the JMS client:

For client applications that use WebSphere MQ classes for Java Message Service Version 5.3 before CSD4, the client application always has an authentication protocol level of PM. The client application and broker negotiate on the choice of protocol for a session. Where the broker supports both protocols (that is, you have set PM or MP in the workbench definition of a broker), the first protocol specified in the workbench is chosen.

For client applications that use WebSphere MQ classes for Java Message Service Version 5.3, CSD 5 or later, the client application supports two levels of authentication.

A *TopicConnectionFactory* can be configured to support either a MQJMS_DIRECTAUTH_BASIC authentication mode or a MQJMS_DIRECTAUTH_CERTIFICATE authentication mode. The MQJMS_DIRECTAUTH_BASIC authentication mode is equivalent to a level of PM and the MQJMS_DIRECTAUTH_CERTIFICATE authentication mode is equivalent to a level of SR.

If authentication services have been successfully configured for a Real-timeInput node, a JMS client application needs to specify its credentials when creating a connection. To do this, the JMS client application supplies a user/password combination to the *TopicConnectionFactory.createTopicConnection* method; for example:

```
factory.createTopicConnection("user1", "user1pw");
```

If credentials are not specified, or are specified incorrectly, the application receives a JMS wrapped exception containing the MQJMS error text.

Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)

This task topic explains how to configure the HTTPInput and HTTPReply nodes to communicate with other applications using HTTP over SSL. The task covers the steps required for windows system but almost identical tasks are required for any other platform.

The steps required are as follows:
1. "Create a key store file to store the broker's certificates"
2. "Configuring the broker to use SSL on a particular port" on page 21
3. "Creating a message flow to process HTTPS requests" on page 22
4. "Testing your example" on page 22

Create a key store file to store the broker's certificates:

WebSphere Event Broker includes a Java Runtime Environment (JRE) that supplies a keystore manipulation program, which is called keytool. To invoke this command complete the following steps:

1. Select **Start → IBM WebSphere Message Brokers 6.0 → Command Console** to open the broker command console.

2. In the command console, type the following command:

   ```
   "%MQSI_FILEPATH%\jre\bin\keytool"
   ```

   This displays the help options and therefore validates that the command is working.

3. Use the keytool command to create the key store: In the command console, type the following command:

   ```
   "%MQSI_FILEPATH%\jre\bin\keytool" -genkey -keypass password -keystore keystore file -alias tom
   ```

   password
   > the password used for the keystore

   keystore file
   > the fully qualified name of the keystore file. This file is normally called .keystore and is usually located in the message broker users home directory.

   The command then prompts for some personal details that are used to generate the certificates. When all of the values have been entered, a key store should either be generated or be added to (if one already exists).

   These values can be set to any values required but the properties on the broker must be changed to reflect these values. The `-genkey` generates all the certificate files that are required to get HTTPS working but they are not official certificates and would not be advisable for use in a production system. You must purchase a real certificate from a certificate organization. Consult your system administrator to find out your company policy for certificate creation. To import a certificate generated by a certificate authority use the `-import` option instead of the `-genkey` option.

The keystore is now created and is ready for use by the broker.

Configuring the broker to use SSL on a particular port:

The broker requires several properties to be set in order to make use of HTTP over SSL. All of these properties can be changed using the **mqsichangeproperties** command. Change the properties as follows:

- Turn on SSL support in message broker, by setting a value for **enableSSLConnector**

  ```
  mqsichangeproperties broker name -b httplistener -o HTTPListener
  -n enableSSLConnector -v true
  ```

- Choose the keystore file to be used, by setting a value for **keystoreFile**

  ```
  mqsichangeproperties broker name -b httplistener -o HTTPSConnector
  -n keystoreFile -v fully qualified file path to keystore file
  ```

- Specify the password for the keystore file, by setting a value for **keystorePass**

  ```
  mqsichangeproperties broker name -b httplistener -o HTTPSConnector
  -n keystorePass -v password for keystore
  ```

- Specify the **port** on which WebSphere Event Broker should listen for HTTPS requests

  ```
  mqsichangeproperties broker name -b httplistener -o HTTPSConnector
  -n port -v Port to listen on for https
  ```

Ensure that sure all of these properties are set with correct values for your system. Only the **enableSSLConnector** property has to be set. The other three properties have default values. However, it is advisable to change these default values. The **mqsichangeproperties** command lists the default values for all the properties.

Creating a message flow to process HTTPS requests:

The most simple message flow that shows the HTTPS functionality working is one that contains an HTTPInput node connected directly to an HTTPReply node: The two important properties to set on the HTTPInput node are:
- **URL Selector**. For example /* or /testHTTPS
- **Use HTTPS**. Tick the box if you want to use HTTPS

/* means that the HTTPInput node will match against any request that is sent to the http listener on a designated port. This is useful for testing purposes, but is not recommended for production.

You can now deploy the message flow to the broker. If all other steps have been followed up to this point, a BIP3132 message should appear in the local system log (this is the event log on Windows) stating that the https listener has been started.

You can now test the system.

Testing your example:

The most simple method of testing whether HTTPS is configured correctly is to use a web browser in order to make a request to the broker over HTTPS.

Start a web browser and type the following URL:

```
https://localhost:7083/testHTTPS
```

Change any values in the URL to reflect changes you have made in your broker configuration. When a pop up window appears asking you to accept the certificate, select yes to any questions. The browser should then refresh and display the structure of an empty html page. In Mozilla browsers this will look like the following example:

```
<html>
  <body/>
</html>
```

and in Internet Explorer the following information should be displayed:

```
XML document must have a top level element. Error processing resource 'https://localhost:7083/testHT
```

These responses mean that a blank page was returned, showing that the set up worked correctly. To add content to the page that is returned, you can add a compute node to the flow.

You can use another HTTPS client to process HTTPS requests. Read the documentation for the client to find out how it should be configured to make client connections over SSL.

Another HTTPS client, such as a Java or .net client, could be used instead of the web browser. Depending on the type of client, the certificate that was created with keytool might have to be exported from the http listener's keystore file and then

imported into that client's own keystore. Consult the client documentation to find out how you should configure the client to make client connections over SSL.

Related concepts

"SSL authentication" on page 5

"Authentication services" on page 38

Related tasks

"Implementing SSL authentication" on page 16

"Configuring an HTTPRequest node to use SSL (HTTPS)"

Related reference

"mqsichangeproperties command" on page 236

"mqsireportproperties command" on page 316

HTTPInput node

HTTPReply node

HTTPRequest node

Configuring an HTTPRequest node to use SSL (HTTPS)

This task topic explains how to configure the HTTPRequest node to communicate with other applications using HTTP over SSL. The task covers the steps required for windows system but almost identical tasks are required for any other platform. To complete the HTTPRequest task, an HTTPS server application is required. For simplicity, the only details that are given here are for using the HTTPInput node for SSL as the server application. However, the same details also apply when you are using any other server application.

The steps required are as follows:

1. "Adding certificates to the cacerts file"
2. "Creating a message flow to make HTTPS requests" on page 24
3. "Testing your example" on page 25

Adding certificates to the cacerts file:

The certificate for the server application to be called must be added to the cacerts file for WebSphere Event Broker. This is located in the JRE security directory. To find the cacerts file on Windows, complete the following steps:

1. Select **Start** → **IBM WebSphere Message Brokers 6.0** → **Command Console** to open a broker command console.
2. In the command console, type the following command to change directory to where the cacerts files is located:

   ```
   cd "%MQSI_FILEPATH%\jre\lib\security"
   ```

To modify the cacerts file, you must use the keytool command.

Importing a certificate into the cacerts file

1. Select **Start** → **IBM WebSphere Message Brokers 6.0** → **Command Console** to open a broker command console.
2. In the command console, type the following command:

   ```
   "%MQSI_FILEPATH%\jre\bin\keytool" -import -alias mykey -file name of certificate file -keystor
   ```

   name of certificate file
   > the fully qualified name of the certificates file. This file is normally found in the message broker users home directory.

changeit
> the default password for the cacerts file. You should change this
> password as soon as possible. You can use keytool to change the
> password.

Extracting a certificate from another keystore

1. Select **Start** → **IBM WebSphere Message Brokers 6.0** → **Command Console** to
   open a broker command console.
2. In the command console, type the following command:

```
"%MQSI_FILEPATH%\jre\bin\keytool" -export -alias tomcat -file name of certificate file -keystore
```

name of certificate file
> the fully qualified name of the certificates file. This file is normally
> called .keystore and is usually located in the message broker users
> home directory.

keystore file
> the fully qualified name of the keystore file. This file is normally found
> in the message broker users home directory.

changeit
> the default password for the cacerts file. You should change this
> password as soon as possible. You can use keytool to change the
> password.

It is important to ensure that the correct certificate has been imported into the
cacerts. The correct certificate is the certificate that the HTTP server should use.

Creating a message flow to make HTTPS requests:

The following message flow creates a generic message flow for converting an
WebSphere MQ message into an HTTPRequest.:

1. Create a message flow with the nodes MQInput->HTTPRequest->Compute-
   >MQOutput.
2. For the MQInput node, set the queue name to HTTPS.IN1 and create the
   MQSeries queue.
3. For the MQOutput node, set the queue name to HTTPS.OUT1 and create the
   MQSeries queue.
4. For the HTTPRequest node, set the Web Service URL to point to the HTTP
   server to call. For calling the HTTPInput task
   use https://localhost:7083/testHTTPS.
5. For the HTTPRequest node, set the advance properties to use
   OutputRoot.BLOB as the *Response location in tree*.
6. In the compute node add in the following esql:

```
CREATE COMPUTE MODULE test_https_Compute
    CREATE FUNCTION Main() RETURNS BOOLEAN
    BEGIN
        -- CALL CopyMessageHeaders();
        CALL CopyEntireMessage();
        set OutputRoot.HTTPResponseHeader = null;
        RETURN TRUE;
    END;

    CREATE PROCEDURE CopyMessageHeaders() BEGIN
        DECLARE I INTEGER;
        DECLARE J INTEGER;
        SET I = 1;
```

```
                SET J = CARDINALITY(InputRoot.*[]);
                WHILE I < J DO
                    SET OutputRoot.*[I] = InputRoot.*[I];
                    SET I = I + 1;
                END WHILE;
            END;

            CREATE PROCEDURE CopyEntireMessage() BEGIN
                SET OutputRoot = InputRoot;
            END;
        END MODULE;
```

The message flow is now ready to be deployed to the broker and tested.

Testing your example:

To test that the example works, complete the following steps

1. Follow all of the instructions given in "Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)" on page 20, including testing the example.
2. Deploy the HTTPRequest message flow.
3. Put a message to the MQSeries queue HTTPS.IN1. A message should appear on the output queue. If it fails, an error appears in the local error log (event log on windows).

   Related concepts

   "SSL authentication" on page 5

   "Authentication services" on page 38

   Related tasks

   "Implementing SSL authentication" on page 16

   "Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)" on page 20

   Related reference

   "mqsichangeproperties command" on page 236

   "mqsireportproperties command" on page 316

   HTTPInput node

   HTTPReply node

   HTTPRequest node

Enabling topic-based security

If your applications use the publish/subscribe services of a broker, you can apply an additional level of security to the topics on which messages are published and subscribed. This topic-based security is managed by the User Name Server.

Complete the following steps:

1. Before you create a User Name Server, refer to "Considering security for a User Name Server" on page 26.
2. Create a User Name Server. Refer to "Creating a User Name Server" on page 114.
3. Select the -j flag and set the -s parameter to the name of the queue manager for the User Name Server on the mqsicreatebroker or mqsichangebroker command.
4. Set the -s parameter on the mqsicreateconfigmgr or mqsichangeconfigmgr command to the name of the queue manager for the User Name Server.

5. Create ACLs for the topics that require additional security. For more information, see "Security for runtime resources" on page 4 (a workbench task).
6. Ensure that the broker's service user ID has authority to:
   a. Get messages from each input queue included in a message flow
   b. Put messages to any output, reply, and failure queues included in a message flow.
7. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put to and get from message flow queues:
   a. Authorize publish applications to put messages to the input queue of the message flow.
   b. Authorize applications that register subscriptions to put to the SYSTEM.BROKER.CONTROL.QUEUE queue.
   c. Authorize subscribe applications to get from the queue to which messages are published.
   d. Authorize publish and subscribe applications to get from the reply queue.

If you are issuing publish/subscribe requests from a JMS client, additional security options are available. Refer to "SSL authentication" on page 5, "Quality of protection" on page 7, and "Authentication services" on page 38.

Go to "Considering security for a Configuration Manager" on page 13.

Considering security for a User Name Server

Complete this task by answering the following question:

Have you enabled topic-based security in your broker?
1. No: Go to "Considering security for a Configuration Manager" on page 13.
2. Yes: You need a User Name Server. Go to "Deciding which user accounts can execute User Name Server commands."

Deciding which user accounts can execute User Name Server commands

During this task you decide what permissions are required for the user IDs that:
• Create, change, list, delete, start, and stop a User Name Server
• Display, retrieve, and change trace information.

Answer the following questions:
1. Is your User Name Server installed on a Linux or UNIX operating system?
   a. No: Go to the next question.
   b. Yes: The information about executing User Name Server commands on Linux and UNIX is not yet available.

      Go to "Deciding which user account to use for the User Name Server service ID" on page 27.
2. Are you executing User Name Server commands under a Windows local account?
   a. No: Go to the next question.
   b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a User Name Server, ensure that your user ID is defined in your local domain. When you create or start a User Name Server, ensure that your user ID is a member of WKSTN1**Administrators**.

Go to "Deciding which user account to use for the User Name Server service ID."

3. Are you executing User Name Server commands under a Windows domain account?

   a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you create a User Name Server using, for example, DOMAIN1\user1, ensure that DOMAIN1\user1 is a member of WKSTN1**Administrators**.

   Go to "Deciding which user account to use for the User Name Server service ID."

Deciding which user account to use for the User Name Server service ID

When you set the service ID with the -i option on the mqsicreateusernameserver or mqsichangeusernameserver command, you determine the user ID under which the User Name Server component process runs.

Answer the following questions:

1. Is your User Name Server installed on a Linux or UNIX operating system?

   a. No: Go to the next question.

   b. Yes: The information about running your User Name Server on Linux and UNIX is not yet available.

   Go to "Setting security on the User Name Server's queues"

2. Do you want your User Name Server to run under a Windows local account?

   a. No: Go to the next question.

   b. Yes: Ensure that your user ID is defined in your local domain and is a member of **mqbrkrs**.

   Go to "Setting security on the User Name Server's queues"

3. Do you want your User Name Server to run under a Windows domain account?

   a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a User Name Server using, for example, DOMAIN1\user1, ensure that:DOMAIN1\user1 is a member of DOMAIN1**Domain mqbrkrs** and DOMAIN1**Domain mqbrkrs** is a member of WKSTN1**mqbrkrs**.

   Go to "Setting security on the User Name Server's queues."

Setting security on the User Name Server's queues

When you run the **mqsicreateusernameserver** command, the **mqbrkrs** group gets access authority to the following queues:

   SYSTEM.BROKER.SECURITY.QUEUE
   SYSTEM.BROKER.MODEL.QUEUE

Only the broker and the Configuration Manager require access to the User Name Server's queues.

Running the User Name Server in a domain environment

When the users that issue publish and subscribe commands are domain users, set the -d option on the mqsicreateusernameserver command to the domain those users come from. All users that issue publish and subscribe commands must come from the same domain.

Using security exits

To invoke a security exit for communications between the Message Brokers Toolkit and the Configuration Manager, one must be defined when the domain connection is created.

1. Click **File** → **New** → **Domain**. The Domain view appears.
2. Enter the queue manager name, host, and port that you wish to use.
3. Optional: Enter the security exit name. This must be a valid Java class name.
4. Optional: Enter the location of the JAR file for the Security Exit require to connect to the WebSphere MQ queue manager. Use the **Browse** button to find the file location.

   Note: The location of the JAR file is required if a Security Exit class is entered.

When the connection is attempted, the security exit is invoked.

Alternatively, you can use SSL when communicating between the Configuration Manager Proxy (CMP) and the Configuration Manager; see "Implementing SSL authentication" on page 16.

Implementing HTTP tunneling

HTTP tunneling support in the broker is activated by selecting the Use HTTP Tunneling option within the properties for the Real-timeInput node or Real-timeOptimizedFlow node. Optional settings are also configured using the "mqsichangeproperties command" on page 236.

Once set, HTTP tunneling is administered using the "mqsichangeproperties command" on page 236 with the httpProtocolTimeout and httpDispatchThreads options.

Implementing quality of protection

The enableQoPSecurity option of the mqsichangeproperties command enables quality of protection. By default, quality of protection is enabled if any of the quality of protection settings have been changed from n or none. The levels of message protection are defined using the sysQoPLevel and isysQoPLevel options also in the "mqsichangeproperties command" on page 236.

From the broker properties window, you can set the values for temporary topics using:
- Temporary Topic Quality of Protection
- Sys Topic Quality of Protection
- ISys Topic Quality of Protection .

The default value is none, or you can select one of the following values from each of the drop-down lists:

- Channel Integrity (preventing hackers from inserting or deleting messages without being detected)
- Message Integrity (preventing hackers from changing the content of a message without being detected)
- Encrypted for Privacy (preventing hackers from looking at the content of a message).

The value selected is the same for all users and is independent of the user currently selected.

Database security

The service user ID for the brokers must be authorized to access databases:
- Each broker service user ID must be authorized for create and update tasks on the database that contains the broker internal tables.

  In addition to the broker service user ID, this also applies to any data source user ID if defined (using "mqsicreatebroker command" on page 257).

Database security is defined and modified using various commands dependent upon the database that is being used. See the following:
- "mqsicreatebroker command" on page 257
- "mqsichangebroker command" on page 221

For information relating to specific databases see:
- "Oracle: granting and revoking user access to databases"
- "SQL Server: granting and revoking user access to databases"
- "Sybase: granting and revoking user access to databases"

Oracle: granting and revoking user access to databases

The Oracle database administrator controls user access to Oracle databases. The WebSphere Event Broker does not provide any special commands to administer an Oracle database.

SQL Server: granting and revoking user access to databases

The SQL database administrator controls user access to Microsoft SQL Server databases. WebSphere Event Broker does not provide any special commands to administer a Microsoft SQL Server database.

Sybase: granting and revoking user access to databases

The Sybase database administrator controls user access to Sybase databases. WebSphere Event Broker does not provide any special commands to administer a Sybase database.

Setting up z/OS security

You need to complete some security configuration tasks before WebSphere Event Broker can work correctly. The steps you need to follow are described in this topic and also in the following topics:
- "Setting up DB2" on page 31
- "Setting up WebSphere MQ" on page 32
- "Setting up workbench access on z/OS" on page 33

- "Creating Publish/Subscribe user IDs" on page 33

Decide on the started task names of the broker, Configuration Manager, and User Name Server. These names are used to set up started task authorizations, and to manage your system performance.

Decide on a data set naming convention for your WebSphere Event Broker PDSEs. A typical name might be `WMQI.MQP1BRK.CNTL` or `MQS.MQP1UNS.BIPCNTL`, where `MQP1` is the queue manager name. You need to give the WebSphere Event Broker, WebSphere MQ, DB2, and z/OS administrators access to these data sets. You can give these professionals control access in several ways, for example:
- Give each user individual access to the specific data set.
- Define a generic data set profile, defining a group that contains the user IDs of the administrators. Grant the group control access to the generic data set profile.

If you intend to use Publish/Subscribe, define a group called MQBRKRS and connect the started task user IDs to this group. Define an OMVS group segment for this group so that the User Name Server can extract information from the External Security Manager (ESM) database to enable you to use Publish/Subscribe security.

Each broker needs a unique ID for its DB2 tables. This can be:
- A unique started task user ID; you could use the broker name as the started task user ID.

  A unique group for the broker (for example MQP1GRP) which has defined all necessary DB2 authorities. The broker started task user ID and the WebSphere Event Broker administrator are both members of this group.
- A shared started task user ID and a unique group specified to identify the DB2 tables to be used with the ODBC interface. Use the broker name as the group name.

Define an OMVS segment for the started task user ID and give its home directory sufficient space for any WebSphere Event Broker dumps. Consider using the started task procedure name as the started task user ID. Check that your OMVS segment is defined by using the following TSO command:

```
LU userid OMVS
```

The command output includes the OMVS segment, for example:

```
USER=MQP1BRK NAME=SMITH, JANE OWNER=TSOUSER
CREATED=99.342 DEFAULT-GROUP=TSOUSER PASSDATE=01.198
PASS-INTERVAL=30
......
OMVS INFORMATION
----------------
UID=0000070594
HOME=/u/MQP1BRK
PROGRAM=/bin/sh
CPUTIMEMAX=NONE
ASSIZEMAX=NONE
FILEPROCMAX=NONE
PROCUSERMAX=NONE
THREADSMAX=NONE
MMAPAREAMAX=NONE
```

The command:

```
df -P /u/MQP1BRK
```

displays the amount of space used and available, where /u/MQP1BRK is the value from HOME above. This command shows you how much space is currently available in the file system. Check with your data administrators that this is sufficient. You need a minimum of 400 000 blocks free; this is needed if a dump is taken.

Associate the started task procedure with the user ID to be used. For example, you can use the STARTED class in RACF®. The WebSphere Event Broker and z/OS administrators must agree on the name of the started task.

WebSphere Event Broker administrators need an OMVS segment and a home directory. Check the setup described above.

The started task user IDs and the WebSphere Event Broker administrators need access to the install processing files, the component specific files, and the home directory of the started task. During customization the file ownership can be changed to alter group access. This might require super user authority.

When the service user ID is root, all libraries loaded by the broker, including all user-written plug-in libraries and all shared libraries that they might access, also have root access to all system resources (for example, file sets). Review and assess the risk involved in granting this level of authorization.

Setting up DB2

This is part of the larger task of setting up security on z/OS.

The user ID of the person running the DB2 configuration jobs needs UPDATE access to the component PDSE, READ/EXECUTE access to the installation directory, and READ/WRITE/EXECUTE access to the broker-specific directory.

A user needs SYSADM or SYSCTRL authority to run the DB2 configuration jobs.

You cannot share DB2 tables between brokers; each broker needs its own DB2 tables. The format of the table names is:

table_owner.table_name

where table_owner is known as the table owner.

When the broker starts up, the started task user ID is used to connect to DB2 using ODBC. The ODBC statement Set current SQLID is used to set the ID to table_owner; the table owner ID specifies which tables to use. You have two options in setting up the IDs:

1. Make the table owner the same as the started task user ID. This means that each broker must have a different user ID. You need to check that the started task user ID specified has access to SYSIBM tables. From a TSO user with no system administration authority, use SPUFI to issue the following commands:

   ```
   select *      from SYSIBM.SYSTABLES;
   select *      from SYSIBM.SYSSYNONYMS;
   select *      from SYSIBM.SYSDATABASE;
   ```

   and resolve any problems.
2. Make the table owner different from the started task user ID. For this to work the started task needs to be able to issue the Set current SQLID request. The easiest way to do this is to create a RACF group with the same name as the table owner, and connect the started task user ID to this group.

Check that the group ID specified has access to SYSIBM tables. From a TSO user with no system administration authority, use SPUFI to issue the following commands:

```
SET    CURRENT SQLID='WMQI';
select * from    SYSIBM.SYSTABLES;
select * from    SYSIBM.SYSSYNONYMS;
select * from    SYSIBM.SYSDATABASE;
```

and resolve any problems (WMQI is the name of the group). You might need to connect the TSO user IDs of the DB2 administrators to the group.

If you have a unique group for each broker (and not a unique started task user ID), the started task user ID must be connected to the group for the ODBC request set currentsqlid to work successfully.

The DB2 administrator user ID must have access to one of the programs DSNTEP2 or DSNTIAD, or equivalent.

The started task user ID must be authorized to connect to DB2. The started task user ID needs a minimum of READ access to the subsystem.RRSAF profile in the DSNR class, if present. In this case, subsystem is the DB2 subsystem name. For example, the following RACF command lists all the resources in DSNR class:

```
RLIST DSNR *
```

The started task user ID needs EXECUTE authority to the DSNACLI plan or equivalent.

The DB2 subsystem started task user ID needs authority to create data sets with the high level qualifier specified in the DB2_STOR_GROUP_VCAT value.

Setting up WebSphere MQ

This is part of the larger task of setting up security on z/OS.

The user ID of the person running the create component (BIPCBRK, BIPCRCM, and BIPCRUN) jobs needs UPDATE access to the component PDSE, READ/EXECUTE access to the installation directory, and READ/WRITE/EXECUTE access to the component directory. If you do not use queue manager security, you do not need to read the rest of this topic. Topic "Creating the broker component" on page 101 provides detailed statements on how to protect your queues.

The broker, Configuration Manager, and the User Name Server need to be able to connect to the queue manager.

By default, the broker's internal queues, which all have names of the form:

```
 SYSTEM.BROKER.*
```

should be protected. Restrict access to the broker, Configuration Manager, andUser Name Server started task user IDs, and to WebSphere Event Broker administrators.

If you are using Publish/Subscribe, subscribers need to PUT to SYSTEM.BROKER.CONTOL.QUEUE.

You can control which applications can use queues used by message flows. Applications need to be able to PUT and GET to queues defined in any nodes.

Setting up workbench access on z/OS

Access to the workbench is controlled from Windows or Linux (x86 platform). The workbench must run on Windows or Linux (x86 platform). See "Considering security for a Configuration Manager" on page 13 for further information.

Creating Publish/Subscribe user IDs

This is part of the larger task of setting up security on z/OS.

A User Name Server can only see user IDs that have been enabled to use UNIX System Services facilities. To ensure that your user or group information can be seen by your User Name Server, you must comply with the following instructions:

- Define user IDs with `OMVS(UID(nnnnn)...)` and their default group with `OMVS(GID(nnnnn))`.
- Define groups with `OMVS(GID(nnnnn))`.

Publish/subscribe security

Three publish/subscribe security services are provided:

- Topic-based security

  Access to messages on particular topics is controlled using access control lists (ACLs).
- Authentication services

  An authentication protocol is used by a broker and a client application to confirm that they are both valid participants in a session.
- Message protection

  Message protection provides security options to prevent messages from being read or modified while in transit.

These services operate independently of each other, but before you can use any of these services you must have a User Name Server.

To use one of these services, use the mqsicreatebroker or mqsichangebroker command to configure a broker with the -s parameter to specify the name of the User Name Server's queue manager.

In addition, to use topic-based security, specify the publish/subscribe access control flag (-j parameter) on the mqsicreatebroker or mqsichangebroker command.

Topic-based security

Use topic-based security to control which applications in your publish/subscribe system can access information on which topics.

For each topic to which you want to restrict access, you can specify which principals (user IDs and groups of user IDs) can publish to the topic, and which principals can subscribe to the topic. You can also specify which principals can request persistent delivery of messages.

Any principal can publish, subscribe, and request persistent delivery of, messages on any topic whose access you do not explicitly restrict.

Topic-based security is managed by a User Name Server that uses the access control lists (ACLs) that you create to decide which authorizations are applied.

Principals and the User Name Server

The User Name Server in WebSphere Event Broker manages the set of principals that are already defined in your network, on behalf of the brokers and the Configuration Manager, for use in publish/subscribe. On Windows, this list of users is taken from the domain specified on the mqsicreateusernameserver command.

The User Name Server is made known to both the broker and the Configuration Manager by specifying the User Name Server queue manager on the mqsicreatebroker and mqsicreateconfigmgr commands.

Message brokers within the broker domain interact with the User Name Server to retrieve the total set of users and groups from which the access control lists are built and against which the publish/subscribe requests are validated. The Configuration Manager interacts with the User Name Server to display the users and user groups in the ACLs that are created using the Topics Hierarchy Editor that is provided in the Broker Administration perspective of the workbench.

Access control lists

Access control lists are used to define, for any topic and principal, the right of that principal to publish on, or subscribe to, that topic, or to request persistent delivery of a publication on that topic.

You can also use the ACL to define the level of message protection that you want to apply to each topic.

Specify these definitions using the Topics Hierarchy Editor in the Broker Administration perspective of the workbench.

Access control can be set explicitly for each individual topic. However, if no explicit ACL is defined for a topic, access control is inherited from an ancestor or parent topic, as defined by the hierarchical structure of the topic tree. If no topic in the hierarchy up to the topic root has an explicit ACL, the topic inherits the ACL of the topic root.

Any defined principal that is known to the User Name Server can be associated with a topic in this way.

Resolving ACL conflicts

If the principals in your broker domain include one or more users in more than one group, the explicit or inherited ACL values might conflict. The following rules indicate how a conflict is resolved:

- If the user has an explicit user ACL on the topic of interest, this always takes priority and the broker verifies the current operation on that basis.
- If the user does not have an explicit user ACL on the topic of interest, but has explicit user ACLs against an ancestor in the topic tree, the closest ancestor ACL for that user takes priority and the broker verifies the current operation on that basis.
- If there are no explicit user ACLs for the user on the topic of interest or its ancestors, the broker attempts to verify the current operation on the basis of group ACLs:

- If the user is a member of a group that has an explicit group ACL on the topic of interest, the broker verifies the current operation on the basis of that group ACL.
- If the user is not a member of a group that has an explicit group ACL on the topic of interest, but is a member of a group with explicit group ACLs against an ancestor in the topic tree, the closest ancestor ACL takes priority and the broker verifies the current operation on that basis.
- If, at a particular level in the topic tree, the user ID is contained in more than one group with an explicit ACL, permission is granted if any of the specifications are positive; otherwise it is denied.

You cannot associate ACLs with topics that include one or more wildcard characters. However, access from your client application is resolved correctly when the subscription registration is made, even when that application specifies a wildcard character in the topic.

PublicGroup authorizations

In addition to the groups that you define, WebSphere Event Broker provides an implicit group, PublicGroup, to which all users automatically belong. This implicit group simplifies the specification of ACLs in a topic tree. In particular, this group is used in the specification of the ACL for the topic root. Note that the default setting of the topic root allows publish and subscribe operations for the PublicGroup. You can view and change this ACL using the workbench, but you cannot remove it. It determines the default permissions for the entire topic tree. You can specify ACLs for the PublicGroup elsewhere in the topic tree, wherever you want to define permissions for all users.

If you have a principal named *Public* defined in your existing security environment, you cannot use this for topic-based security. If you specify this principal within an ACL, it is equated to PublicGroup and therefore always allows global access.

mqbrkrs authorizations

WebSphere Event Broker grants special publish/subscribe access control privileges to members of the mqbrkrs group, and to the corresponding Domain mqbrkrs global group if appropriate.

Brokers need special privileges to perform internal publish and subscribe operations in networks where there is access control. When you create a broker in such a network, you must specify a user ID that belongs to the group mqbrkrs as the service user ID for the broker. The mqbrkrs group is given implicit privileges so that its members can publish, subscribe and request the persistent delivery of messages on the topic root (″″). All other topics inherit these permissions. If you attempt to configure an ACL for the mqbrkrs group using the workbench, this ACL is ignored by WebSphere Event Broker.

ACLs and system topics

Messages that are used for internal publish and subscribe operations are published throughout the broker domain using system topics, which begin with the strings ″$SYS″ and ″$ISYS″. These topics must be published and subscribed to only by members of the mqbrkrs group, except for the following two scenarios:

1. If you are migrating topics from WebSphere MQ Publish/Subscribe, you can configure ACLs on topics that begin with the string ″$SYS/STREAM″.

2. Clients can subscribe to topics that begin with the string "$SYS"; this means that applications that provide a management function can subscribe to the broker for administrative events.

Do not configure ACLs on topics that begin with the string "$ISYS". You are not prevented from doing so, but they are ignored.

Setting access control on topics

All members of the group mqbrtpic, or a user or member of a group that has an object-level security ACL that gives at least deploy permission to the root topic object, can define and manipulate the ACLs that define which principals can publish on, and subscribe to, which topics. ACLs can also limit delivery of persistent messages, and define the level of message protection. All defined principals can be associated with any topic; the permissions that can be set are shown in the following table:

Option	Description
Publish	Permits or denies the principal to publish messages on this topic.
Subscribe	Permits or denies the principal to subscribe to messages on this topic.
Persistent	Specifies whether the principal can receive messages persistently. If the principal is not permitted, all messages are sent non-persistently. Each individual subscription indicates whether the subscriber requires persistent messages.
QoP Level	Specifies the level of message protection that is enforced. One of the following four values can be chosen: • None • Channel Integrity • Message Integrity • Encrypted The default value is 'None'.

Persistent access control behavior is not identical to the publish and subscribe control:

• Clients that are denied Publish access have their publication messages refused.

• Clients that are denied Subscribe access do not receive the publication.

• The persistent access control does not deny the message to subscribers, but denies them persistence, so denied subscribers always receive messages, subject to their subscribe access control, but always have the message sent to them non-persistently, regardless of the persistence of the original message.

Inheritance of security policies

Typically, topics are arranged in a hierarchical tree. The ACL of a parent topic can be inherited by some or all of its descendent topics that do not have an explicit ACL. Therefore, it is not necessary to have an explicit ACL associated with each and every topic. Every topic has an ACL policy which is that of its parent. If all parent topics up to the root topic do not have explicit ACLs, that topic inherits the ACL of the root topic.

For example, in the topic tree shown below, the topic root is not shown but is assumed to have an ACL for PublicGroup whose members can publish, subscribe, and receive persistent publications. (The symbol "¬" means "not".)

Inheriting ACLs in a topic tree
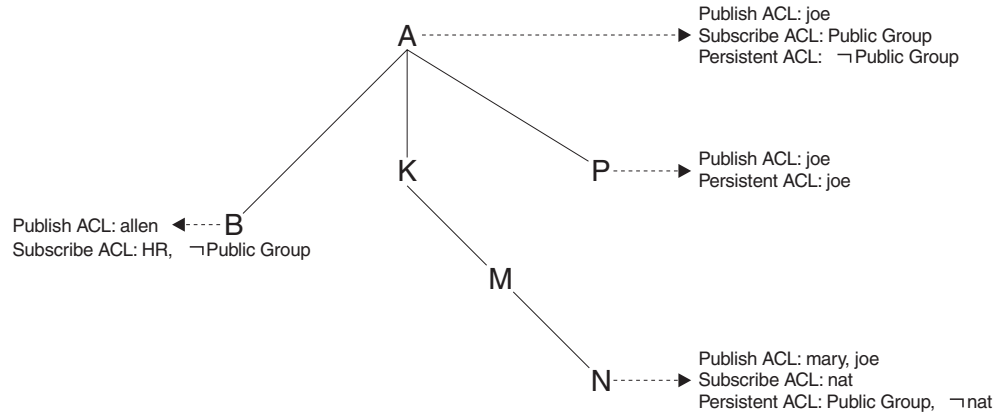
The following table shows the ACLs, inherited in some cases, that result from the topic tree shown in the figure:

Topic	Publishers	Subscribers	Persistent
A	only joe	everyone	no-one
A/P	only joe	everyone	only joe
A/K	only joe	everyone	no-one
A/K/M	only joe	everyone	no-one
A/K/M/N	only mary, joe	everyone	everyone except nat
A/B	allen, joe	HR	no-one

Dynamically created topics

Topics that are not explicitly created by the system administrator, but are created dynamically when a client publishes or subscribes to messages, are treated in the same way as those that are created by the system administrator, but they do not have explicitly defined ACLs. That is, the ACLs for dynamically created topics are inherited from the closest ancestor in the topic tree that has an explicit policy. It is therefore not necessary to define leaf topics in the tree if they do not have explicit ACLs.

ACLs and wildcard topics

With WebSphere Event Broker you cannot associate an explicit security policy with a wildcard topic. For example, you cannot associate an ACL with topic "A/+", which represents a two level hierarchy and includes "A/B", "A/K", and "A/P".

However, WebSphere Event Broker does guarantee correct access mediation when a client application subscribes to a wildcard topic.

For example, the topic "A/+" does not, and cannot, have a security policy explicitly associated with it. Therefore, "A/+" inherits its policy from "A". Any user can subscribe to "A/+" because the subscribe ACL includes everyone.

When a message is published on "A/P" or "A/K", the broker delivers it to the user who subscribed to "A/+". However, when a message is published to "A/B", that message is only delivered to subscribers who are in the HR group.

If the system administrator changes the subscribe ACL of any topic that matches "A/+", the broker correctly enforces the ACL when the message is delivered.

Subscribing to a wildcard topic has the semantics to deliver messages on all topics that match the wild card, and for which the subscriber has authorization to receive that message.

ACLs and subscription resolution

The broker enforces access control through the topic of the message to be delivered. Messages are delivered only to those clients that have not had subscribe access to that topic denied, either explicitly or through inheritance. Because a subscription can contain a wildcard character, the actual match against the topic namespace, and hence the topic ACLs, cannot be made when the subscription is received. The decision to deliver a message to a subscriber is made only when a specific message with a topic is being processed by the message broker.

Activating topic ACL updates

Updates to a topic ACL do not become active until deployed and activated across the broker domain from the WebSphere Event Broker workbench. You must be a member of the group mqbrops to deploy ACLs, or a user or member of a group that has an object-level security ACL that gives at least deploy permission to the root topic object.

Authentication services

Authentication services are supported only between client applications that use the WebSphere MQ Real-time Transport and WebSphere Event Broker Real-timeInput and Real-timeOptimizedFlow nodes.

The WebSphere Event Broker authentication services verify that a broker and a client application are who they claim they are, and can therefore participate in a publish/subscribe session.

Each participant in the session uses an authentication protocol to prove to the other that they are who they say they are, and are not an intruder impersonating a valid participant.

The WebSphere Event Broker product supports the following four protocols:
- P - simple telnet-like password authentication
- M - mutual challenge-response password authentication
- S - asymmetric SSL
- R - symmetric SSL

The first two of these protocols and their infrastructure requirements are described in "Simple telnet-like password authentication" on page 39 and "Mutual challenge-response password authentication" on page 40 respectively. Asymmetric and symmetric SSL protocols are described in "SSL authentication" on page 5.

The protocols vary in strength, in terms of providing protection against participants that are not valid participants in the session; P is the weakest and R is the strongest.

Configuring authentication protocols

The set of protocols that can be supported by a specific broker in the broker domain can be configured using the workbench. One or more protocols can be specified for each broker. Use the workbench to enable or disable authentication on each Real-timeInput node that is defined for a particular broker. When

authentication is enabled at a Real-timeInput node, that node supports the full set
of protocols specified for its corresponding broker. The configuration options are
illustrated in the following diagrams:

Overview of authentication configuration
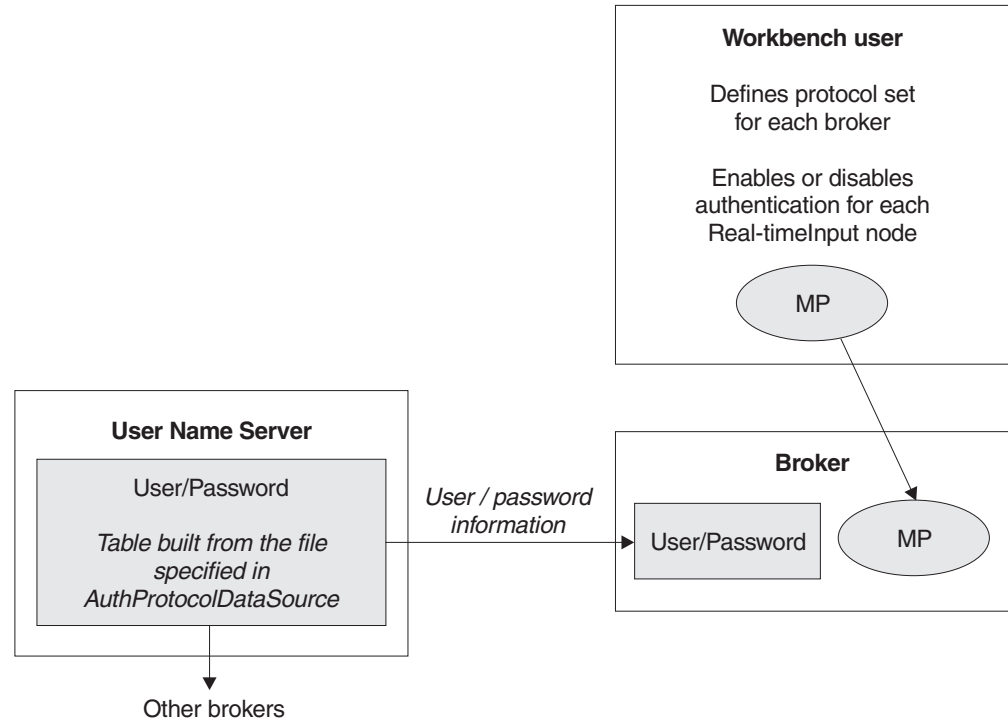
Two-stage runtime authentication process

Stage 1 - Determine protocol for session
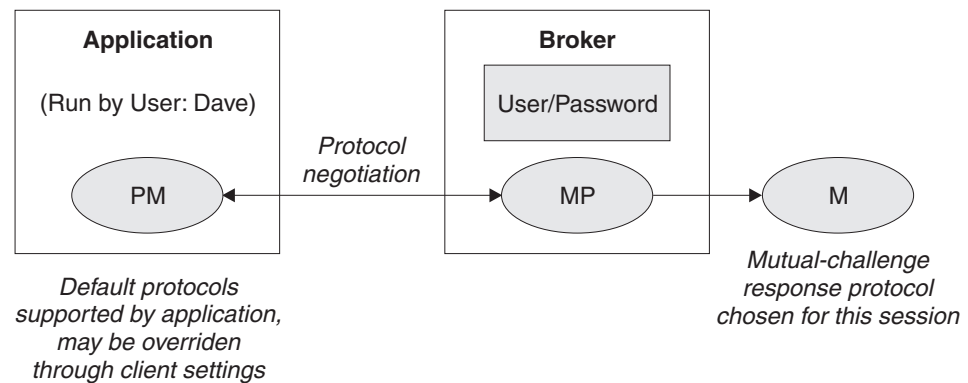
Stage 2 - Drive chosen protocol
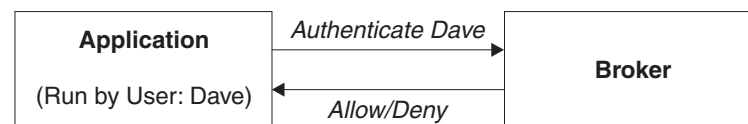
Simple telnet-like password authentication

This protocol can also be described as password in the clear because the password
passes un-encrypted over the network. The client application connects to the

Real-timeInput node using TCP/IP. The input node requests that the client identify itself. The client sends its "userid" and its password.

This simple protocol relies on both the client and the broker knowing the password associated with a user ID. In particular, the broker needs access to a repository of user and password information. The user ID and password information is distributed by theUser Name Server to all the brokers in a WebSphere Event Broker product's domain. The User Name Server extracts user and password information from an operating system file.

The User Name Server approach allows for the centralized maintenance of the source of users and passwords, with automatic distribution of the information to brokers, and automatic refreshes of the information if required. It also provides availability benefits, because user and password information is maintained persistently at each broker.

Each client application must know its own user ID and keep its password secret. When creating a connection, a client specifies its credentials as a name/password combination.

This protocol provides relatively weak security. It does not compute a session key, and should only be used in environments where there are no "eavesdroppers" and no untrusted "middle-men".

In the case where user and password information is stored in a flat file on the User Name Server system, the passwords are stored and distributed "in-the-clear".

The computational load on the client and server is very light.

Mutual challenge-response password authentication

This is a more sophisticated and secure protocol that involves the generation of a secret session key. Both the client and the server compute this key using the client's password. They prove to each other that they know this secret through a challenge and response protocol.

The client must satisfy the server's challenge before the server satisfies the client's challenge. This means that an attacker impersonating a client can gather no information to mount an "offline" password guessing attack. Both the client and the server prove to each other that they know the password, so this protocol is not vulnerable to "impersonation" attacks.

As in the case of the simple telnet-like password protocol, the broker must have access to user and password information. Information about the user ID and password is distributed by the User Name Server to all the brokers in the domain. The User Name Server extracts user and password information from an operating system file.

Each client application must know its own user ID and keep its password secret. When creating a connection, a client specifies its credentials as a name/password combination.

Computational demands on both client and server are fairly modest.

Message protection

The authentication services provided by WebSphere Event Broker ensure that only legitimate message brokers and client applications can connect to each other.

However, a hacker might still be able to observe messages in transit or interfere with messages on established connections. Message protection provides security options to protect your messages against such activities.

You cannot use message protection if you are using 'simple telnet-like password authentication'.

Because the use of message protection can have an adverse affect on the performance of your publish/subscribe system, and because security is not equally important for all messages, you might want to define different levels of message protection for different messages. You do this by assigning a Quality of Protection (QoP) value to each topic in your publish/subscribe system.

There a four QoP values. They give the following levels of protection:

n   This is the default value. It gives no message protection.

c   This provides channel integrity. With this level of protection, hackers are unable to insert or delete messages without being detected.

m   This provides message integrity. With this level of protection, hackers are unable to change the content of a message without being detected.

e   This provides message encryption. With this level of protection, hackers are unable to look at the content of a message.

The protection levels are cumulative. For example, if you specify message encryption, you also get message integrity and channel integrity; if you request message integrity, you also get channel integrity.

If any QoP settings are made, all clients that connect to the broker must use a security level that supports message integrity or message encryption.

Securing the publish/subscribe domain

The following topics describe how to secure your publish/subscribe domain.
- "Enabling topic-based security" on page 25
- "Creating ACL entries" on page 44
- "Enabling SSL for the Real Time node" on page 17
- "Using message protection" on page 44
- "Securing WebSphere MQ resources" on page 44
-

Enabling topic-based security

If your applications use the publish/subscribe services of a broker, you can apply an additional level of security to the topics on which messages are published and subscribed. This topic-based security is managed by the User Name Server.

Complete the following steps:

1. Before you create a User Name Server, refer to "Considering security for a User Name Server" on page 26.
2. Create a User Name Server. Refer to "Creating a User Name Server" on page 114.

3. Select the -j flag and set the -s parameter to the name of the queue manager for the User Name Server on the mqsicreatebroker or mqsichangebroker command.

4. Set the -s parameter on the mqsicreateconfigmgr or mqsichangeconfigmgr command to the name of the queue manager for the User Name Server.

5. Create ACLs for the topics that require additional security. For more information, see "Security for runtime resources" on page 4 (a workbench task).

6. Ensure that the broker's service user ID has authority to:
   a. Get messages from each input queue included in a message flow
   b. Put messages to any output, reply, and failure queues included in a message flow.

7. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put to and get from message flow queues:
   a. Authorize publish applications to put messages to the input queue of the message flow.
   b. Authorize applications that register subscriptions to put to the SYSTEM.BROKER.CONTROL.QUEUE queue.
   c. Authorize subscribe applications to get from the queue to which messages are published.
   d. Authorize publish and subscribe applications to get from the reply queue.

If you are issuing publish/subscribe requests from a JMS client, additional security options are available. Refer to "SSL authentication" on page 5, "Quality of protection" on page 7, and "Authentication services" on page 38.

Go to "Considering security for a Configuration Manager" on page 13.

Considering security for a User Name Server

Complete this task by answering the following question:

Have you enabled topic-based security in your broker?
1. No: Go to "Considering security for a Configuration Manager" on page 13.
2. Yes: You need a User Name Server. Go to "Deciding which user accounts can execute User Name Server commands" on page 26.

Deciding which user accounts can execute User Name Server commands

During this task you decide what permissions are required for the user IDs that:
• Create, change, list, delete, start, and stop a User Name Server
• Display, retrieve, and change trace information.

Answer the following questions:
1. Is your User Name Server installed on a Linux or UNIX operating system?
   a. No: Go to the next question.
   b. Yes: The information about executing User Name Server commands on Linux and UNIX is not yet available.

      Go to "Deciding which user account to use for the User Name Server service ID" on page 27.

2. Are you executing User Name Server commands under a Windows local account?

a. No: Go to the next question.

b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a User Name Server, ensure that your user ID is defined in your local domain. When you create or start a User Name Server, ensure that your user ID is a member of WKSTN1**Administrators**.

Go to "Deciding which user account to use for the User Name Server service ID" on page 27.

3. Are you executing User Name Server commands under a Windows domain account?

a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you create a User Name Server using, for example, DOMAIN1\user1, ensure that DOMAIN1\user1 is a member of WKSTN1**Administrators**.

Go to "Deciding which user account to use for the User Name Server service ID" on page 27.

Deciding which user account to use for the User Name Server service ID

When you set the service ID with the -i option on the mqsicreateusernameserver or mqsichangeusernameserver command, you determine the user ID under which the User Name Server component process runs.

Answer the following questions:

1. Is your User Name Server installed on a Linux or UNIX operating system?

a. No: Go to the next question.

b. Yes: The information about running your User Name Server on Linux and UNIX is not yet available.

Go to "Setting security on the User Name Server's queues" on page 27

2. Do you want your User Name Server to run under a Windows local account?

a. No: Go to the next question.

b. Yes: Ensure that your user ID is defined in your local domain and is a member of **mqbrkrs**.

Go to "Setting security on the User Name Server's queues" on page 27

3. Do you want your User Name Server to run under a Windows domain account?

a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a User Name Server using, for example, DOMAIN1\user1, ensure that:DOMAIN1\user1 is a member of DOMAIN1**Domain mqbrkrs** and DOMAIN1**Domain mqbrkrs** is a member of WKSTN1**mqbrkrs**.

Go to "Setting security on the User Name Server's queues" on page 27.

Setting security on the User Name Server's queues

When you run the **mqsicreateusernameserver** command, the **mqbrkrs** group gets access authority to the following queues:

SYSTEM.BROKER.SECURITY.QUEUE
SYSTEM.BROKER.MODEL.QUEUE

Only the broker and the Configuration Manager require access to the User Name Server's queues.

Running the User Name Server in a domain environment

When the users that issue publish and subscribe commands are domain users, set the -d option on the mqsicreateusernameserver command to the domain those users come from. All users that issue publish and subscribe commands must come from the same domain.

Creating ACL entries

Using message protection

To use message protection (sometimes known as Quality of Protection (QoP)), set the enableQopSecurity parameter of the **mqsichangeproperties** command to true. The default value of this parameter is false.

To define a level of message protection for $SYS topics, use the sysQopLevel parameter of the **mqsichangeproperties** command.

To define a level of message protection for $ISYS topics, use the isysQopLevel parameter of the **mqsichangeproperties** command.

Choose one of the following values for these parameters:

n   This is the default value. It gives no message protection.

c   This provides channel integrity. With this level of protection, hackers cannot insert or delete messages without being detected.

m   This provides message integrity. With this level of protection, hackers cannot change the content of a message without being detected.

e   This provides message encryption. With this level of protection, hackers cannot look at the content of a message.

Securing WebSphere MQ resources

Secure the WebSphere MQ resources that your configuration requires.

This section does not apply to z/OS.

WebSphere Event Broker depends on a number of WebSphere MQ resources to operate successfully. You must control access to these resources to ensure that the product components can access the resources on which they depend, and that these same resources are protected from other users.

Some authorizations are granted on your behalf when commands are issued. Others depend on the configuration of your broker domain.

* When you issue the command **mqsicreatebroker**, it grants put and get authority on your behalf to the group **mqbrkrs** for the following queues:
  – SYSTEM.BROKER.ADMIN.QUEUE
  – SYSTEM.BROKER.CONTROL.QUEUE
  – SYSTEM.BROKER.EXECUTIONGROUP.QUEUE

- SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- SYSTEM.BROKER.INTERBROKER.QUEUE
- SYSTEM.BROKER.MODEL.QUEUE

- When you issue the command **mqsicreateconfigmgr**:

  1. It grants put and get authority on your behalf to the group **mqbrkrs** for the following queues:
     - SYSTEM.BROKER.CONFIG.QUEUE
     - SYSTEM.BROKER.CONFIG.REPLY
     - SYSTEM.BROKER.ADMIN.REPLY
     - SYSTEM.BROKER.SECURITY.REPLY
     - SYSTEM.BROKER.MODEL.QUEUE

  2. It grants put and get authority on your behalf to the groups **mqbrdevt**, **mqbrasgn**, **mqbrops**, and **mqbrtpic** for the following queues:
     - SYSTEM.BROKER.CONFIG.QUEUE
     - SYSTEM.BROKER.CONFIG.REPLY

- When you issue the command **mqsicreateusernameserver**, it grants put and get authority on your behalf to the group **mqbrkrs** for the following queues:
  - SYSTEM.BROKER.SECURITY.QUEUE
  - SYSTEM.BROKER.MODEL.QUEUE

- When you issue the command mqsicreateaclentry, it grants put and get authority on your behalf to the resource or user that you have specified for the command parameters -p or -u for the following queues:
  - SYSTEM.BROKER.CONFIG.QUEUE
  - SYSTEM.BROKER.CONFIG.REPLY

- If you have created WebSphere Event Broker components to run on different queue managers, the transmission queues that you define to handle the message traffic between the queue managers must have put and setall authority granted to the local **mqbrkrs** group, or to the service user ID of the component supported by the queue manager on which the transmission queue is defined.

- When you start the workbench, it connects to the Configuration Manager using a WebSphere MQ client/server connection. For details of WebSphere MQ channel security refer to ″Setting up WebSphere MQ client security″ in the *WebSphere MQ Clients* book.

- When you create and deploy a message flow, grant:

  1. get and inq authority to each input queue identified in an MQInput node, for the broker's ServiceUserID.

  2. put and inq authority to each output queue identified in an MQOutput node, or by an MQReply node, for the broker's ServiceUserID.

  3. get authority to each output queue identified in an MQOutput node or an MQReply node to the user ID under which a receiving or subscribing client application runs.

  4. put authority to each input queue identified in an MQInput node to the user ID under which a sending or publishing client application runs.

Part 2. Configuring the broker domain

Configuring the broker domain

Before you start:

To configure a broker domain, you need the authorization and permission to access WebSphere Event Broker resources. Refer to "Authorization for configuration tasks" on page 3.

A broker domain is a collection of brokers that share a common configuration, together with a single Configuration Manager that controls them. A domain can also include a User Name Server that interfaces with operating system facilities to determine valid users and groups for publish/subscribe.

Design a broker domain before creating and configuring its components. Consider the factors described in "Designing a broker domain" to create a broker domain that best suits its components and the required functionality. Also, review the conceptual information below for an overview of the broker domain, and broker domain components.

Creating the broker domain is a two-stage process. You create the physical components using the command line, and then configure the physical components using the workbench or programmatically using the Configuration Manager Proxy Java API.

The following set of tasks takes you through the process for configuring the broker domain (depending on the configuration that you require, and the platform that you are using).

- "Designing a broker domain"
- "Customizing the z/OS environment" on page 57
- "Configuring broker domain components" on page 66
- "Configuring a broker domain in the workbench" on page 146
- "Setting up broker domain security" on page 8
- "Configuring a publish/subscribe topology" on page 160

If you require a default broker domain configuration, you can use the Default Configuration wizard. The Default Configuration wizard creates all the components you need to start exploring WebSphere Event Broker and run the supplied samples. See "Using the Default Configuration wizard" on page 128.

Designing a broker domain

When you are designing a broker domain consider the following elements.

1. **Brokers:** The number of brokers that you need in your domain depends on the following factors:
    a. What is the required message throughput? (Refer to "Optimizing message flow throughput" on page 56)
    b. What is the size of the messages that are being processed? Larger messages take longer to process.

49

c. Is the performance of the broker domain important? A small number of brokers handling many messages might impact the broker domain's performance.

d. Do you need to isolate applications from each other? You might want to separate applications that serve different functions, for example personnel and finance.

e. Do the brokers need to handle publish/subscribe? (Refer to Developing publish/subscribe applications)

2. **User Name Server:** Consider the following if you have a User Name Server in your broker domain:

a. Performance: If you have a large number of brokers in your broker domain, the requests that they send the User Name Server can be handled more quickly if there is more than one User Name Server. More than one User Name Server might also be beneficial (in terms of network traffic) if your broker domain is complex.

b. Resilience: Although no standby mechanism is provided by WebSphere Event Broker, you might want to be able to redirect requests to a second User Name Server if a system error occurs on the system of your first User Name Server.

3. **Configuration Manager:** This acts as an interface between the configuration repository and the set of brokers in the domain and the workbench. It uses WebSphere MQ messages to communicate with the brokers, and thus a large number of brokers in a broker domain (if poorly designed) can cause congestion at the Configuration Manager. To solve this, consider dividing the brokers into more than one domain where related brokers are kept together. Then you can establish connections with each domain ("Creating a domain connection" on page 147).

For more information about how your design affects performance, see "Considering performance in the domain" on page 55.

Considering resource naming conventions

When you plan a new WebSphere Event Broker network, one of your first tasks must be to establish a convention for naming the resources that you create within this network. There are three aspects to this:

- Product component naming conventions
- WebSphere MQ naming conventions
- Database naming conventions

Product component naming conventions

A naming convention for WebSphere Event Broker resources throughout your network ensures that names are unique, and that users creating new resources can be confident of not introducing duplication or confusion.

The resources you must create and name within an WebSphere Event Broker network are:

Configuration Managers
When you create a Configuration Manager, give it a name that is unique on your system. Names must be unique between Configuration Managers and between Configuration Managers and brokers. Configuration Manager names are case sensitive on UNIX systems.

Brokers

When you create a broker, give it a name that is unique within your broker domain. You must use the same name for that broker when you create it on the system in which it is installed (using the command **mqsicreatebroker**) and when you create a reference to that broker in the broker domain topology in the workbench. The latter is a representation of the physical broker (created by **mqsicreatebroker**) in the configuration repository, and this single name links the two. Broker names are case-sensitive except on Windows platforms.

Execution groups

Each execution group name must be unique within a broker.

The workbench component is described in "Configuring a broker domain in the workbench" on page 146.

Message flows and message processing nodes

Each message processing node must be unique within the message flow it is assigned to. For example, if you include two MQOutput nodes in a single message flow, provide a unique name for each.

Message flow names must be unique within the broker domain. Any reference to that name within the broker domain is always to the same message flow. You can assign the same message flow to many brokers.

Message sets and messages

Each message name must be unique within the message set to which it belongs.

Message set names must be unique within the broker domain. Any reference to that name within the broker domain is always to the same message set. You can therefore assign the same message set to many brokers.

The User Name Server is not allocated names when you create it. It is identified only by the name of the WebSphere MQ queue manager that hosts the services it provides.

There are a few restrictions for naming resources: see "Naming conventions for WebSphere Event Broker for z/OS" on page 335.

WebSphere MQ naming conventions

All WebSphere Event Broker resources have dependencies on WebSphere MQ services and objects. You must therefore also consider what conventions to adopt for WebSphere MQ object names. If you already have a WebSphere MQ naming convention, use a compatible extension of this convention for WebSphere Event Broker resources.

When you create a broker or a Configuration Manager, you must specify a queue manager name. This queue manager is created for you if it does not already exist. Because the broker and Configuration Manager each use a unique set of WebSphere MQ queues, they can share one queue manager, if appropriate. However, every broker must have a dedicated queue manager.

If you set up a User Name Server in your broker domain, this also uses a unique set of WebSphere MQ queues. The User Name Server can therefore also share a queue manager with a broker, or the Configuration Manager, or both.

Ensure that every queue manager name is unique within your network of interconnected queue managers, whether or not every queue manager is in your WebSphere Event Broker network. This ensures that each queue manager can unambiguously identify the target queue manager to which any given message must be sent, and that WebSphere Event Broker applications can also interact with basic WebSphere MQ applications.

WebSphere MQ supports a number of objects defined to queue managers. These objects (queues, channels, and processes) also have naming conventions and restrictions.

In summary, the restrictions are:
- All names must be a maximum of 48 characters in length (channels have a maximum of 20 characters).
- The name of each object must be unique within its type (for example, queue or channel).
- Names for all objects starting with the characters SYSTEM. are reserved for use by IBM.

There are a few restrictions for naming resources: see "Naming conventions for WebSphere Event Broker for z/OS" on page 335.

Database naming conventions

Consider the naming conventions you use for databases, both for databases that you create for WebSphere Event Broker product and for databases that you create for application use.

Database tables used for brokers can be unique and local to the broker, or can be *shared*, because the rows of the tables specific to each individual broker incorporate the name of the broker. You might need to align the naming of all these databases with other databases that are in use in your broker domain.

See "Creating the databases" on page 67 for details of creating the database tables for WebSphere Event Broker.

Also ensure that the databases used for application data (accessed through message flows) are uniquely named throughout your network, so that there is no opportunity for confusion or error.

Designing the WebSphere MQ infrastructure

WebSphere Event Broker depends on the WebSphere MQ transport services to support internally-generated communications between components. Some of these resources are created for you, when you create WebSphere Event Broker components that depend on them. Others depend on the exact setup of your broker domain; you must create these resources yourself.

Communications between WebSphere Event Broker components are protocol-independent, with the exception of the connection between every instance of the workbench and the Configuration Manager. This must be a TCP/IP connection, as must connections to the WebSphere MQ Everyplace and SCADA nodes. Other connections can use any of the protocols supported by the WebSphere MQ messaging product for the operating system for your WebSphere Event Broker product.

Except for WebSphere MQ Everyplace and SCADA applications, applications that use broker services must also use WebSphere MQ to send and receive all messages. The resources required by your applications (queues and client connection and server connection channels) are application specific; you must create these resources yourself.

The information here concentrates on the specific requirements that WebSphere Event Broker imposes on a WebSphere MQ network. For a full description of designing and connecting a WebSphere MQ network, see *WebSphere MQ Intercommunication*, which covers the basics, such as setting up transmission queues and channels, in detail.

For further information see:
- "WebSphere MQ resources for brokers"
- "WebSphere MQ resources for Configuration Managers" on page 54
- "WebSphere MQ resources for the User Name Server" on page 54

WebSphere MQ resources for brokers

Each broker depends on a number of WebSphere MQ resources; some are always required; others depend on the broker domain setup:

1. Each broker must be associated with a queue manager to host its services. Specify a queue manager name when you create the broker. If this queue manager does not exist, it is created for you. (For WebSphere Event Broker on z/OS, you must create a queue manager in WebSphere MQ for your broker. See "Creating a broker on z/OS" on page 94 for more details.)

   The broker cannot share a queue manager with any other broker. It can share a queue manager with the Configuration Manager, or the User Name Server, or both if the broker is on a Windows platform.

   The broker and its queue manager can share the same name, subject to naming restrictions for both products.

2. Each broker must have a number of fixed-name queues on its queue manager. These allow it to exchange information with other components in the broker domain. These queues are defined for you when the broker is created. The use of these fixed-name queues dictates that each broker must be hosted by a unique queue manager.

3. Each broker must communicate with the Configuration Manager. If the broker and the Configuration Manager do not share a queue manager, you must define the channels and transmission queues that support communications between the two queue managers.

4. If you have included a User Name Server in your broker domain, each broker must communicate with it. If the broker and the User Name Server do not share a queue manager, you must define transmission queues and channels that support two-way communications between the two queue managers.

5. The broker's queue manager must have a listener to receive messages from other components that do not share its queue manager, and from clients on other physical systems. Create a listener for every protocol used for connections to the broker. If any connection uses the TCP/IP protocol, you must decide which port the listener must listen on.

6. If the broker is connected to other brokers, either in a collective or to communicate with another collective, the queue manager needs transmission queues and channel definitions to support two-way communications with each of the other brokers' queue managers.

WebSphere MQ resources for Configuration Managers

Each Configuration Manager depends on a number of WebSphere MQ resources: some must be available, others depend on the broker domain setup:

1. Each Configuration Manager must be associated with a queue manager to host its services. Specify a queue manager name when you create a Configuration Manager. If this queue manager does not exist, it is created for you except on z/OS. The Configuration Manager can share a queue manager with a component (broker, or the User Name Server, or both).

2. The Configuration Manager must have a number of fixed-name queues on its queue manager. These allow it to exchange information with other components in the broker domain. These queues are defined for you when the Configuration Manager is created.

3. The Configuration Manager must communicate with every broker in the broker domain. Define transmission queues and channels to support two-way communications between the Configuration Manager and every broker, except the one (if defined) that shares its queue manager.

4. If you have included a User Name Server in your broker domain, the Configuration Manager must communicate with it. If the Configuration Manager and the User Name Server do not share a queue manager, you must define transmission queues and channels to support two-way communications between the two queue managers.

5. The Configuration Manager's queue manager must have a listener to receive messages from the workbench, and from other components and clients that do not share its queue manager. You must create a listener for every protocol used for the intercomponent connections. If the connection is TCP/IP, you must also decide which port the listener must listen on; no other listener must be active on this port.

6. The Configuration Manager's queue manager must have a server connection. This is defined for you when the Configuration Manager is created. Every workbench client can use this single definition.

WebSphere MQ resources for the User Name Server

The User Name Server depends on a number of WebSphere MQ resources: some must be available; others depend on the broker domain setup:

1. The User Name Server must be associated with a queue manager to host its services. You must specify a queue manager name when you create the User Name Server. If this queue manager does not exist, it is created for you. On z/OS, you must create a queue manager in WebSphere MQ for your broker. See "Creating a broker on z/OS" on page 94 for more details.

   The User Name Server can share a queue manager with a component (broker, or the Configuration Manager, or both) if they are on Windows platforms.

2. The User Name Server must have a number of fixed-name queues on its queue manager. These allow it to exchange information with other components in the broker domain. These queues are defined for you when the User Name Server is created.

3. The User Name Server must communicate with the Configuration Manager. If the two do not share a queue manager, you must define the transmission queues and channels to support two-way communications between the two queue managers.

4. The User Name Server must communicate with every broker in the broker domain. Define transmission queues and channels to support two-way

communications between the User Name Server and every broker except the one (if defined) that shares its queue manager.

5. The User Name Server's queue manager must have a listener to receive messages from other components that do not share its queue manager. Create a listener for every protocol used for connections to the User Name Server. If you create a TCP/IP listener, you must also decide which port it must listen on.

Considering performance in the domain

When you design your broker domain, and the resources associated with its components, there are several areas where decisions that you make can affect the performance of your brokers and applications:

Message flows

A message flow includes an input node that receives a message from an application over a particular protocol (for example WebSphere MQ).

You need to consider how you split your business logic; how much work should the application do, and how much should the message flow do? Every interaction between an application and a message flow involves I/O and message parsing and therefore adds to processing time. Design your message flows, and design or restructure you applications, to minimize these interactions.

For more information about these factors, see Optimizing message flow response times.

Messages

The type, format, and size of the messages that are processed can have a significant effect on the performance of a message flow. For example, if you process persistent messages, these have to be stored for safekeeping.

For more information about these factors, see Optimizing message flow response times

Broker configuration and domain topology

You can configure your broker domain to include multiple brokers, multiple systems, multiple execution groups, and so on. These can play a part in how message flows perform, and how efficiently messages can be processed.

For more information about these factors, see "Optimizing message flow throughput" on page 56 and Performance considerations for Real-time transport.

All these factors are examined in more detail in the Designing for Performance SupportPac (IP04).

For a description of common performance scenarios, review Resolving problems with performance.

For further articles about WebSphere Event Broker and performance, review these sources:

- The Business Integration Zone on developerWorks. This has a search facility; enter "performance" and review the links that are returned.
- The developerWorks article on message flow performance.
- The developerWorks article on monitoring resource use.

- A developerWorks article that describes other available resources including best practices, tools, and performance data.

Optimizing message flow throughput

Each message flow that you design must provide a complete set of processing for messages received from a certain source. However, this might result in very complex message flows that include large numbers of nodes and cause a performance overhead and potential bottlenecks. Increasing the number of message flows that process your messages provides the opportunity for parallel processing and therefore improved throughput.

You can also consider the way in which the actions taken by the message flow are committed, and the order in which messages are processed.

Consider the following options for optimizing message flow throughput:

Multiple threads processing messages in a single message flow
When you deploy a message flow, the broker automatically starts an instance of the message flow for each input node that it contains. This is the default behavior. However, if you have a message flow that handles a very large number of messages, the input source (for example, a WebSphere MQ queue) might become a bottleneck.

You can update the *Additional Instances* property of the deployed message flow in the bar file: the broker starts additional copies of the message flow on separate threads, providing parallel processing. This is the most efficient way of handling this situation, if you are not concerned about the order in which messages are processed.

If the message flow receives messages from a WebSphere MQ queue, you can influence the order in which messages are processed to some extent by setting the *Order Mode* property of the MQInput node:

- If you set *Order Mode* to By User ID, the node ensures that messages from a specific user (identified by the UserIdentifier field in the MQMD) are processed in guaranteed order. A second message from one user is not processed by an instance of the message flow if a previous message from this user is currently being processed by another instance of the message flow.
- If you set *Order Mode* to By Queue Order, the node processes one message at a time to preserve the order in which the messages are read from the queue. Therefore, this node behaves as though you have set the *Additional Instances* property of the message flow to zero.

For publish/subscribe applications that communicate with the broker over any supported protocol, messages for any given topic are published by brokers in the same order as they are received from publishers (subject to reordering based on message priority, if applicable). This normally means that each subscriber receives messages from a particular broker, on a particular topic, from a particular publisher, in the order that they are published by that publisher.

However, it is possible for messages, occasionally, to be delivered out of order. This can happen, for example, if a link in the network fails and subsequent messages are routed by another link.

If you need to ensure the order in which messages are received, you can use either the *SeqNum* (sequence number) or *PubTime* (publish time stamp) parameter on the **Publish** command for each published message, to calculate the order of publishing.

For more information about the techniques recommended for all MQI and AMI users, see the *WebSphere MQ Application Programming Guide* for programs written to the MQI, and *WebSphere MQ Application Messaging Interface* (available as SupportPac MA0F from the WebSphere MQ SupportPacs Web page) for programs written to the AMI.

WebSphere MQ Everyplace and SCADA applications use a different method of message ordering as described in WebSphere MQ Mobile Transport and WebSphere MQ Telemetry Transport respectively. The broker does not provide message ordering for messages received across WebSphere MQ Real-time Transport or WebSphere MQ Multicast Transport.

Multiple copies of the message flow in a broker
You can also deploy several copies of the same message flow to different execution groups in the same broker. This has similar effects to increasing the number of processing threads in a single message flow, although usually provides less noticeable gains.

This option also removes the ability to determine the order in which the messages are processed. This is because, if there is more than one copy of the message flow active in the broker, each copy can be processing a message at the same time, from the same queue. The time taken to process a message might vary, and multiple message flows accessing the same queue could therefore read messages from the input source in a random order. The order of messages produced by the message flows might not correspond to the order of the original messages.

Ensure that the applications that receive message from these message flows can tolerate out-of-order messages.

Copies of the message flow in multiple brokers
You can deploy several copies of the same message flow to different brokers. This solution requires changes to your configuration, because you must ensure that applications that supply messages to the message flow can put their messages to the right input queue or port. You can often make these changes when you deploy the message flow by setting the message flow's configurable properties.

Customizing the z/OS environment

Although you might be installing only one broker initially, you should consider how it will be used in your organization in a few years' time. Planning ahead makes developing your WebSphere Event Broker configuration easier.

You should consider creating the Configuration Manager on z/OS to manage the broker domain. If you are migrating from an earlier version of WebSphere Event Broker you should consider migrating the Configuration Manager previously on Windows.

If you are using Publish/Subscribe with security, you also need a User Name Server. The User Name Server can be on z/OS or on another platform. The queue

managers need to be interconnected so that information from the User Name Server can be distributed to the brokers on other queue managers.

A broker requires access to a queue manager and to DB2. A Configuration Manager and User Name Server requires access to a queue manager only. A broker cannot share its queue manager with another broker, but a broker can share a queue manager with a Configuration Manager and User Name Server.

You cannot use WebSphere MQ shared queues to hold data related to WebSphere Event Broker as SYSTEM.BROKER queues, but you can use shared queues for your message flow queues.

You can find details of the DB2 database user tables and the WebSphere MQ queues created and used by WebSphere Event Broker on z/OS in the topic "mqsicreatebroker command" on page 257

You need to create started task procedures for the broker, User Name Server, and Configuration Managers that you plan to use. These procedures need to be defined, in the started task table, with an appropriate user ID.

You need to decide on your recovery strategy. As part of your systems architecture, you should have a strategy for restarting systems if they end abnormally. Common solutions are to use automation products like NetView or the Automatic Restart Manager (ARM) facility. You can configure WebSphere Event Broker to use ARM.

You must also plan for corequisite products, including UNIX System Services, Resource Recovery Services, DB2, WebSphere MQ, and Java.

z/OS customization overview

After using SMP/E to install WebSphere Event Broker for z/OS, the installed executable code is located inside the file system. JCL samples are located in the PDS <hlq>.SBIPSAMP, the JCL procedures are located in the PDS <hlq>.SBIPPROC, and load module for synchronizing statistics with SMF are located in the PDS <hlq>.SBIPAUTH. The diagram below gives an overview of the post-installation process.

The customization is performed by updating and submitting the required JCL. All necessary JCL is supplied to create the runtime environments of your broker, Configuration Manager, and User Name Server. One of these JCL files is required to start the broker, Configuration Manager, or User Name Server. This JCL is run as a started task.

Installation   Customization

Installation
Directory

SBIPSAMP
PDSE

SMP/E
Tape

SBIPPROC
PDSE

SBIPAUTH
PDSE

Component
PDSE

Component
Directory

Started Task
JCL

DB2

WebSphere
MQ
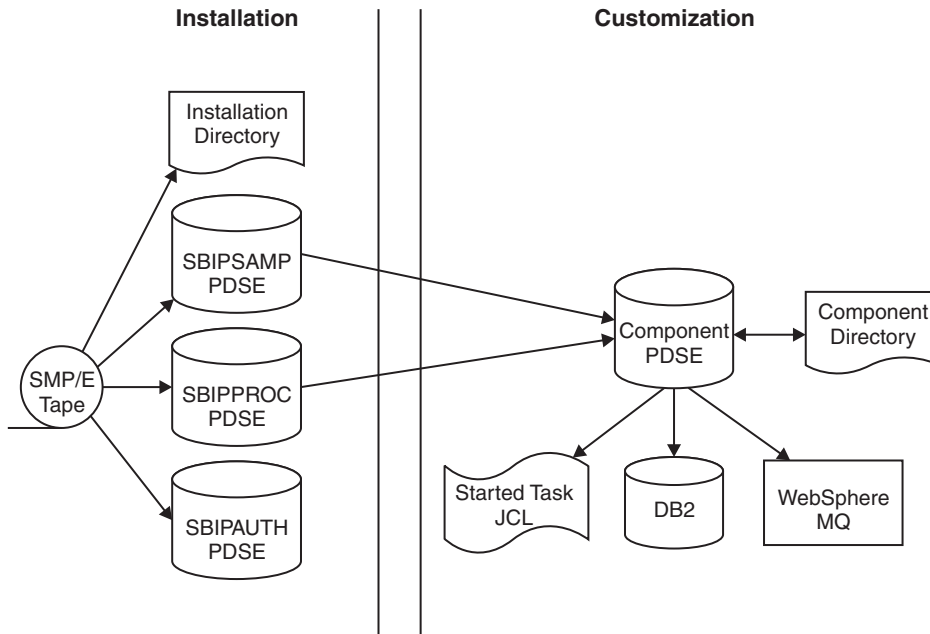
Installation directory (z/OS)

On z/OS, the SMP/E installation places all the product executable files into a directory of a file system under UNIX System Services (USS).

Components (z/OS)

A *component* is a broker, a Configuration Manager, or a User Name Server.

A broker processes messages, a Configuration Manager acts as an interface between the configuration repository and the set of brokers in the domain, and a User Name Server extracts information from a security product and makes it available to brokers and the workbench.

A broker that is running has a control address space and one additional address space for each deployed execution group. When the control address space is started, the broker component is started automatically. This behavior can be changed by an optional start parameter in the started task.

A Configuration Manager and User Name Server each have a single address space. When the address space is started, the components are started automatically. This behavior can be changed by an optional start parameter in the started task.

The *component name* is the external name of the component and is used, for example, in the WebSphere Event Broker workbench.

Each component requires a name, which is usually the name of the started task that runs the component. This is typically the queue manager name with a suffix of the facility; for example

- `MQP1BRK` for the broker
- `MQP1UNS` for the User Name Server
- `MQP1CMGR` for the Configuration Manager

You only need a User Name Server if you are using Publish/Subscribe security. This User Name Server can exist anywhere in the network, including z/OS.

Each component has its own runtime environment in UNIX System Services and needs its own WebSphere MQ queue manager, although a broker, Configuration Manager, and User Name Server can share a single queue manager. A broker component also needs access to a database.

Component directory

The *component directory* is the root directory of the component's runtime environment.

It is also referred to as `ComponentDirectory`. Both the WebSphere Event Broker administrator and the component require read and write access to the component directory, for example:
- `/mqsi/V6R0M0/brokers/MQP1BRK` for the broker
- `/mqsi/V6R0M0/uns/MQP1UNS` for the User Name Server
- `/mqsi/V6R0M0/configmgrs/MQP1CMGR` for the Configuration Manager

Component PDSE

The *component PDSE* contains jobs to:
- Define resources to:
  - DB2
  - WebSphere MQ
  - The WebSphere Event Broker started task on z/OS
- Run WebSphere Event Broker commands and utilities.

All component user IDs require read access to their component PDSE at runtime.

XPLink

XPLink is a z/OS technology used by the C and C++ compilers to reduce the cost of function calling for programs written in these languages.

Many products, including WebSphere Event Broker for z/OS, use XPLink technology to improve their performance. To ensure the highest possible performance gains, WebSphere Event Broker requires as many as possible of the software components it uses to be XPLink-compliant. These include the broker, Java runtime, ODBC, and z/OS Language Environment.

The WebSphere Event Broker broker has been compiled by IBM to use XPLink technology and has been link-edited within the SMP/E environment to call the appropriate XPLink routines of the software components it uses. Normally, these XPLink-enabled components are configured during their customization, and the broker needs only to locate the appropriate libraries to become XPLink-enabled.

Binding a DB2 plan to use data-sharing groups

During customization, you can specify which plan name to use, or use the default `DSNACLI`. If you are using XPLINK, the default plan is called `DSNACLX`. If you want your broker to access DB2 data-sharing groups other than its own, the `DSNACLI` plan must be bound in a special way. If the broker uses one data sharing group, but might want to access tables on `DSNONE` and `DSNTWO`, which are in different data-sharing groups, amend the DB2 supplied job `DSNTIJCL` to do the following:

```
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIQR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIQR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIQR)
BIND PLAN(DSNACLI)-
PKLIST(*.DSNAOCLI.DSNCLICS -
*.DSNAOCLI.DSNCLINC -
*.DSNAOCLI.DSNCLIRR -
*.DSNAOCLI.DSNCLIRS -
*.DSNAOCLI.DSNCLIUR -
*.DSNAOCLI.DSNCLIC1 -
*.DSNAOCLI.DSNCLIC2 -
*.DSNAOCLI.DSNCLIF4 -
*.DSNAOCLI.DSNCLIMS -
*.DSNAOCLI.DSNCLIQR )
```

Customizing UNIX System Services

WebSphere Event Broker requires the setup of some UNIX System Services system
characteristics. You can use the SETOMVS operator command for dynamic changes or
the BPXPRMxx PARMLIB member for permanent changes. Refer to the *z/OS UNIX
System Services Planning* manual for more information.

Do not include the broker addresses if you use the IEFUSI exit to limit the region
size of OMVS address spaces.

Set the UNIX System Services characteristics as follows:

Description	Parameter	Value
The maximum core dump file size (in bytes) that a process can create. Allow an unlimited size.	MAXCORESIZE	2 147 483 647

The CPU time (in seconds) that a process is allowed to use. Allow an unlimited CPU time	MAXCPUTIME	2 147 483 647
The address space region size. Set to the size of the biggest address space.	MAXASSIZE	> 1 GB A minimum of 375 MB is required.
Specifies the maximum number of threads that a single process can have active. Depends on the definitions of message flows.	MAXTHREADS MAXTHREADTASKS	The value of MAXTHREADS and MAXTHREADTASKS depends on your application. You can calculate the value needed for WebSphere Event Broker as follows: For each message flow: 1. Multiply the number of input nodes by the number of instances (additional threads +1). 2. Sum the values of all message flows and add 10 to the resultant sum. 3. Add in the number of threads used for each HTTP listener.

DB2 planning (z/OS)

This is part of the larger task of customizing your z/OS environment.

WebSphere Event Broker for z/OS accesses DB2 tables using ODBC. To connect to DB2 using ODBC, the location name of the DB2 subsystem is used. See the *DB2 UDB for OS/390® and z/OS V7 Data Sharing: Planning and Administration* manual for more details.

When your DB2 system starts up there should be a message DSNL004I DDF START COMPLETE. The location name is displayed just after this message. When you customize a broker component on z/OS you create a dsnaoini file called BIPDSNAO in the broker PDSE. It contains necessary information to establish the ODBC connection. See the *DB2 UDB for OS/390 and z/OS V7 ODBC Guide and Reference* manual for more details.

You should avoid using a data source name that is the same as the subsystem ID or data sharing ID. If the same name is used, this might affect the granularity of directives on connection with the database.

If you choose to use the same value for the data source name and subsystem ID, you must edit BIPDSNAO in the broker PDSE so that the Datasource and Subsystem keywords are in one section.

See the *DB2 UDB for OS/390 and z/OS V7 ODBC Guide and Reference* manual for more information on customizing this file.

During customization you can specify which plan name to use, or use the default DSNACLI. If you want your broker to access DB2 data-sharing groups other than its own, the DSNACLI plan must be bound in a special way. Check the wildcard location is specified by using SPUFI and issuing the following command:

```
select * from SYSIBM.SYSPACKLIST where planname ='DSNACLI';
```

You should rebind if the location column is blank and not '*'.

You should also check that DSNACLI is in the SYSIBM.SYSPLAN table.

You will get significant performance benefits from using the CACHE DYNAMIC SQL facility of DB2, because this eliminates the need to reprocess DB2 statements. See CACHEDYN=YES in the *DB2 UDB for OS/390 and z/OS V7 Installation Guide*.

If your user database is configured to use a comma as a decimal separator using the DSNHDECP module, you will find there is a restriction. If there is a mismatch between DB2 and the locale settings of the user ID under which the broker runs (specifically LC_NUMERIC), your user database updates can be unpredictable. LC_NUMERIC is set through the LC_ALL setting in the BIPBPROF member, and therefore the environment file. The following list details the four possibilities:

- If DB2 is configured to use a period as a decimal point and LC_NUMERIC is set to a value that indicates a period decimal point; user database updates should work correctly.
- If DB2 is configured to use a comma as a decimal point and LC_NUMERIC is set to a value that indicates a comma decimal point; user database updates should work correctly.
- If DB2 is configured to use a period as a decimal point and LC_NUMERIC is set to a value that indicates a comma decimal point; user database updates can lead to unpredictable behavior.
- If DB2 is configured to use a comma as a decimal point and LC_NUMERIC is set to a value that indicates a period decimal point; user database updates can lead to unpredictable behavior.

The broker uses DB2 tables in a database to store information. The configuration of these requires someone with the following authorities:

- CREATESG to create a storage group
- CREATEDB to create the database
- Grant use of buffer pools

WebSphere MQ planning (z/OS)

This is part of the larger task of customizing your z/OS environment.

You are required to have a separate WebSphere MQ queue manager for each broker, User Name Server (although you would typically have only one User Name Server in your environment), and Configuration Manager. A broker, User Name Server, and Configuration Manager however, can share the same queue manager. All WebSphere Event Broker for z/OS system queues are defined during customization.

Your queue manager needs a dead-letter queue. Check this by using the WebSphere MQ command:

```
+cpf  DIS QMGR DEADQ
```

Check the queue exists by using the command:

```
 +cpf DIS QL(name) STGCLASS
```

Then use the:

```
+cpf DIS STGCLASS(...)
```

to check the STGCLASS value is valid. If the queue manager does not have a valid dead-letter queue, you must define one.

Set up your channel initiator to use distributed queuing. You need channels between the z/OS queue manager and the queue manager of your Configuration Manager (if not on z/OS). If you are using Publish/Subscribe security, you also need access to the queue manager used by the User Name Server, which can be on z/OS or on another platform. You should be able to successfully start channels between the various queue managers before you can test that the broker is working. When configuring the transmission queues between the brokers on z/OS and the Configuration Manager, ensure you set the maximum message size of the queues to 100 MB. This allows large reply messages concerning deployment to be returned to the Configuration Manager. See "Creating a domain connection" on page 147 for details.

Creating and deleting components on z/OS requires the command server on the WebSphere MQ queue manager to be started, which is normally done automatically (refer to the *WebSphere MQ for z/OS System Administration Guide* for more details).

This requires a reply-to queue based on SYSTEM.COMMAND.REPLY.MODEL; by default, this model queue is defined as permanent dynamic. However, if you leave the queue defined in this way, each time you run a create or delete component command these reply-to queues remain defined to the queue manager. To avoid this you can set the SYSTEM.COMMAND.REPLY.MODEL queue as temporary dynamic.

Resource Recovery Service planning

This is part of the larger task of customizing your z/OS environment.

WebSphere Event Broker for z/OS uses Resource Recovery Service (RRS) to coordinate changes to WebSphere MQ and DB2 resources. Ensure it is configured and active on your system, because your broker cannot connect to DB2 unless RRS is active.

Refer to the following manuals for detailed information about RRS: *z/OS V1R5.0 MVS Setting Up a Sysplex* and *z/OS V1R5.0 MVS Programming: Resource Recovery SA22-7616*.

Defining the started tasks to z/OS Workload Manager (WLM)

This is part of the larger task of customizing your z/OS environment.

If you are running z/OS in Workload mode, change the classification rules to add the started task names of the brokers, User Name Server, and Configuration Manager to the Started Task Control (STC) subsystem types, for example MQP1BRK, MQP1UNS, and MQP1CMGR to the OMVS subsystem types.

If you are running in compatibility mode, add the broker and User Name Server address spaces to the IEAICSxx member in SYS1.PARMLIB. For example, for a broker MQP1BRK, use the commands:

```
SUBSYS=STC
  ..
TRXNAME=MQP1BRK,PGN=35,RPGN=2010
  ..
SUBSYS=OMVS
 ..
TRXNAME=MQP1BRK%(1),PGN=35,RPGN=2011
  ../* Broker address spaces*/
```

The %(1) represents jobs beginning with MQP1BRK with one character following it. Define the priority of the broker lower than DB2 and WebSphere MQ.

Using the file system on z/OS

This is part of the larger task of customizing your z/OS environment.

If you have more than one MVS™ image, consider how you will use the file system. You can share files in a file system across different members of a sysplex. The file system is mounted on one MVS image and requests to the file are routed to the owning system using XCF from systems which do not have it mounted.

Moving a broker, User Name Server, or Configuration Manager from one image to another is straightforward and the files for each component can be shared.

However, there is a performance overhead when using files shared between images in a file system because the data flows through the Coupling Facility (this is true for trace and other diagnostic data).

Space requirements

For details of the disk space required, see "Disk space requirements (z/OS)" on page 340.

Security considerations

This is part of the larger task of customizing your z/OS environment.

The role of the WebSphere Event Broker administrator includes customizing and configuring, running utilities, performing problem determination, and collecting diagnostic materials. People involved in these activities need WebSphere Event Broker authorities. You must set up some security for WebSphere Event Broker to work properly. The information that you need to do this is in "Setting up z/OS security" on page 29.

Checking the level of Java

This is part of the larger task of customizing your z/OS environment.

To check your version of Java, enter `java -version` from an OMVS window. You obtain a response similar to the following:

```
java version "1.4.2"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2)
Classic VM (build 1.4.2 J2RE 1.4.2 IBM z/OS Persistent Reusable VM build cm142-20040917
 (JIT enabled: jitc))
```

This indicates that the level of Java is 1.4.2, which is the minimum required level for this platform.

Checking APF attributes of bipimain on z/OS

This is part of the larger task of setting up your z/OS environment.

Use the `extattr` command to display the attributes of the object `bipimain`. For example:

```
extattr /usr/lpp/wmqi/bin/bipimain
```

It should show `APF authorized = YES`. If it does not, use `extattr +a bipimain` to set this attribute. For example:

```
extattr +a /usr/lpp/wmqi/bin/bipimain
```

Note that you need the appropriate authorization to issue this command.

Configuring broker domain components

Before you start:

Ensure that:
- Your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385.
- On Windows platforms, you have created a new user ID, the service user ID. This is specified during component creation and is used to run the component (the Configuration Manager, broker, and User Name Server).
- Refer to "Planning for security when you install WebSphere Event Broker" on page 8 for more information on user ID authorization and creation.
- You have created and configured the required component databases and prepared a database user ID. The database user ID is specified during component creation and is used by the component to read and update its database. Refer to "Configuring component databases" on page 67.

To create, modify, or delete a WebSphere Event Broker component:
- Select the task for the component and action that you require from the following list.
- Select the platform that you require from within the task.

or
- From within the Table of Contents, select the top-level container for the component and action that you require.
- Expand the container.
- Select the platform that you require from the list.

When you have created your physical components you can configure the broker domain using either the workbench, or programmatically using the Configuration Manager Proxy Java API.

The following set of tasks describes creating, modifying, and deleting the physical broker domain components using the workbench. For information on the using the Configuration Manager Proxy Java API, see Developing applications using the CMP.

This collection of tasks uses specific resource names, user IDs, and so on. These names are examples only; if you want to use your own names, make a note of the changes that you want to make, and remember to apply them. Follow existing naming conventions for WebSphere MQ and other resources.

- "Creating a broker" on page 86
- "Creating a Configuration Manager" on page 103
- "Creating a User Name Server" on page 114
- "Enabling a User Name Server" on page 114
- "Modifying a broker" on page 131
- "Modifying a Configuration Manager" on page 134
- "Modifying a User Name Server" on page 138
- "Modifying component database access" on page 140
- "Moving from WebSphere Event Broker on a distributed platform to z/OS" on page 140
- "Deleting a broker" on page 141
- "Deleting the Configuration Manager" on page 143
- "Deleting a User Name Server" on page 145
- Developing applications using the CMP

Configuring component databases

Review and complete these tasks to configure the databases required by your broker domain components:

- "Creating the databases"
- "Customizing DB2 databases" on page 73
- "Connecting to the databases" on page 74
- "Authorizing access to the databases" on page 84
- "Using retained publications with a Sybase database" on page 85

Creating the databases

WebSphere Event Broker requires a database for each broker. This topic provides information about these databases and links to topics that tell you how to create them in your chosen database system.

The broker database

A broker stores configuration and control information in its database. You must define the database before you create the broker (because creating the broker creates tables within the specified database), and you must authorize access to the database for specific users. The broker database is also known as the broker's local persistent store. Choose a unique name for the broker database, for example WBRKBKDB, and keep a note of it for when you create the broker.

If you create a broker on UNIX, you can create the broker database in DB2, Oracle, or Sybase. On Windows, you can create the broker database in DB2, Oracle, Sybase, SQL Server, or Derby. For related information, see "Database commands on Windows" on page 72.

The Configuration Manager database
> The Configuration Manager also stores configuration and control information in its database, which is known as the internal configuration repository.It is created and managed automatically by the Configuration Manager.

Database sharing
> You can create a database for each broker, or you can use one database for multiple brokers if the platforms are compatible. The tables for each broker are identified with the broker name; this separates the data for each broker.

Database size
> There is no fixed size requirement for the broker database; the size required depends on the complexity of your message flows. If you develop message flows that support many publishers or subscribers, you might need to increase your initial sizings.

Database schema
> When you create a broker, the database tables required by that component are created in the default schema associated with the user ID used to access that database. You specify this user ID on the create command (mqsicreatebroker).
>
> - For DB2 and Oracle, the default behavior is for the schema name to default to the user ID used to access the database.
> - For Sybase and SQLServer, the typical behavior is to use the database owning schema (dbo).
>
> WebSphere Event Broker does not require a particular schema or set of tablespaces; you can configure the database and access privileges of the user ID to choose your own values.

1. On UNIX systems, complete the setup procedures defined in "Preparing databases on UNIX systems" before you create the database.
2. Create the database. Instructions are provided for DB2 in "Creating a DB2 database on Windows" on page 69 and "Creating a DB2 database on UNIX systems" on page 70. Explicitly creating a Derby database is described in "mqsicreatedb command" on page 270. For another database, refer to the documentation for that database.
3. If your database is DB2, customize it for WebSphere Event Broker use. This is described in "Customizing DB2 databases" on page 73.
4. Establish connections to the database. Refer to "Connecting to the databases" on page 74.
5. Authorize access to the database. This task is described in "Authorizing access to the databases" on page 84.

Preparing databases on UNIX systems:

If you are creating a database on a UNIX system, you must complete setup tasks before you create the database. You are recommended to complete this setup after you have installed your database, but you can complete it before database installation if you correctly specify the intended database installation directory during this setup.

You do not have to complete this task if your database is on Windows or Linux.

1. Issue the following command:

```
mqsi_setupdatabase database_type <database_install_directory>
```

where database_type is db2, oracle9, oracle10, or sybase12_5 and <database_install_directory> is the directory in which the database is (or will be) installed (for example, /usr/lpp/db2_08_01 or /opt/IBM/db2).

2. You can now create the broker database. For DB2, see "Creating a DB2 database on Windows" or "Creating a DB2 database on UNIX systems" on page 70. For Oracle and Sybase, refer to the database product documentation for information about how to do this.

When you issue the command that creates the broker, tables are created within the database to hold the information required.

If you are creating a Sybase database for your broker, and you expect heavy use of retained publications with multiple topics, you are recommended to apply row-level locking to the retained publications table in the database. See "Using retained publications with a Sybase database" on page 85.

Creating a DB2 database on Windows:

You can use broker database "Commands" on page 202 to create a DB2or Derby database on Windows for the broker. Alternatively, you can use the DB2 Control Center to create a DB2 database on Windows for the broker. If you prefer, you can use any other method supported by DB2 (including command line or batch files); refer to the DB2 documentation for details of how to do this.

When you create a broker , you specify the user ID and password that are used to connect to the database. The user ID must be known to the database and must own the tables that are being accessed. When you create the database, ensure that you create them for the right user ID. DB2 authenticates the user ID using operating system user management, you do not have to define the user ID to DB2 itself.

1. Start the DB2 Control Center.
2. For each database you want to create:
   a. Expand the Object tree in the DB2 Control Center until you find Databases. Right-click Databases and select **Create** → **Database Using Wizard**.
   b. Enter a name and alias for your database. If you have a naming convention for databases, choose a compatible name. The alias name can be the same as the database name. Database names are limited to eight characters. For example, enter WBRKBKDB.
   c. Click **Done**.
3. You must increase the database heap size to ensure it is sufficient for the broker. This task is described in "Customizing DB2 databases" on page 73.
4. When you have completed these steps for every database you have created, click **OK**.

If you use the DB2 command line to create the databases, or if you are using DB2 Version 7.2 plus FixPak 9, you must also bind the db2cli package to the broker database. (You do not have to do this if you used either the Create Database Wizard, or the "Preparing the samples for first use" cheat sheet and the Default Configuration wizard.)

1. Open a DB2 Command Line Processor window.
2. Connect to the broker database:
   db2cmd db2 CONNECT to <YourBrokerDatabaseName>

3. Enter the following commands, where c:\ is the drive on which you installed DB2. You must enter your full DB2 installation path; do not use spaces or quotes.

```
db2 bind C:\SQLLIB\BND\@db2ubind.lst GRANT PUBLIC
db2 bind C:\SQLLIB\BND\@db2cli.lst GRANT PUBLIC
```

4. Repeat the previous two steps for every broker database.

Creating a DB2 database on UNIX systems:

When you create a broker, you specify the user ID and password that are used to connect to the database. The user ID must be known to the database and must own the tables that are being accessed. When you create the database, ensure that you create them for the right user ID. DB2 authenticates the user ID using operating system user management, you do not have to define the user ID to DB2 itself.

1. Logon as root.
2. Create a database instance. Use the commands shown here for guidance for the different platforms.

   a. On AIX:

   ```
   /usr/lpp/db2_07_01/instance/db2icrt -u <fence userID> <username>
   ```

   b. On Linux, Solaris, or HP-UX:

   ```
   /opt/IBM/db2/V7.1/instance/db2icrt -u <fence userID> <username>
   ```

   The <username> that you specify on this command determines the nominated owner of the database instance. You are recommended to log on as this user whenever you perform any actions against the database instance (for example, creating or modifying a database). The command examples that are used in this help assume that you are logged on as <username>, and use the tilde (~) character to indicate this in the DB2 commands issued.

   If you are not logged on as the user that owns the database instance, you must modify the commands shown to specify explicit ownership by specifying the owner user ID <username> following the ~ character wherever it is used in the examples.

   For further explanation of database ownership, refer to the DB2 library, accessible on the Web.

3. Logon as <username>
4. Create a database (in this example called WBRKBKDB) using the following commands (on some platforms, an explicit path name is required).

   Note: You must insert a space between the starting period and the tilde character in the first command shown here:

   ```
   . ~/sqllib/db2profile
   db2start
   db2 create database WBRKBKDB
   db2 connect to WBRKBKDB
   db2 bind ~/sqllib/bnd/@db2cli.lst grant public CLIPKG 5
   ```

5. You must increase the database heap size to ensure it is sufficient for the broker. This task is described in "Customizing DB2 databases" on page 73.

When you issue the command that creates the broker, tables are created within the database to hold the information required.

Derby databases on Windows: This topic describes the processes, services, IP ports, and database files used to support Derby on Windows.

Security

With the current implementation of Derby in WebSphere Event Broker, no security controls are available, and no optimizations have been performed. For these reasons, it is strongly recommended that you do not use Derby in a production environment.

DB2 Run-Time Client use

WebSphere Event Broker uses ODBC to connect to Derby databases. Derby is a native Java database engine without ODBC support. The DB2 Run-Time Client provides the drivers that allow ODBC to access Derby databases. The DB2 Run-Time Client is used only for providing and managing the ODBC connection between WebSphere Event Broker and the Derby databases. It does not provide a DB2 database and therefore does not consume the resources that a full DB2 installation normally requires.

Database Instance manager (managing, creating, deleting and running databases)

A running network server is required for WebSphere Event Broker to be able to access Derby databases through ODBC. The network server provides access to Derby databases from external programs. When you create the first Derby database using the mqsicreatedb command, a Windows service is also created. The service is called IBM MQSeries Broker DatabaseInstanceMgr6 and starts automatically when Windows starts, under the user name that you supplied with the mqsicreatedb command. The service is referred to as DatabaseInstanceMgr component, and starts the network server.

All Derby databases that you create using the mqsicreatedb command are served by one instance of the DatabaseInstanceMgr and network server. Before the network server can function, it requires a TCP/IP port number. The default port number for Derby is 1527 (use this when creating the Derby database). You can specify a different port number when you issue the mqsicreatedb command to create a Derby database for the first time. However, you cannot subsequently change the port number after a network server has been set up, without first using mqsideletedb to remove all Derby databases.

You can remove the DatabaseInstanceMgr and the network server after the last Derby database has been deleted, using the mqsideletedb command. If the password of the user name under which the Windows service runs is changed, use the mqsichangedbimgr command to update the service with the new password. You can also use the mqsichangedbimgr command to change the user name of the service. Use the mqsistart and mqsistop commands to start and stop the DatabaseInstanceMgr component.

Multiplicity (brokers, dbiMgrs, installations, databases)

There is no limit to the number of databases that you can create with the mqsicreatedb command. A maximum of one DatabaseInstanceMgr is created irrespective of how many databases have been created. If there are multiple simultaneous installations of WebSphere Event Broker Version 6.0, these use a single instance of the DatabaseInstanceMgr component.

Uninstall broker description, and considerations

Use the mqsideletedb command to clear all resources created by the mqsicreatedb command. When the last Derby database is deleted, the DatabaseInstanceMgr and network server are also stopped and removed. If the database files cannot be deleted using the mqsideletedb command, it is possible to delete them manually.

Database commands on Windows: On Windows, WebSphere Event Broker provides commands for creating and deleting databases for use by a broker, or by applications. Only DB2 and Derby databases are supported. The commands are:
- "mqsicreatedb command" on page 270
- "mqsideletedb command" on page 285
- "mqsichangedbimgr command" on page 231

The mqsilist command lists the databases that have been created using the mqsicreatedb command. Only databases created using the mqsicreatedb command can be deleted using the mqsideletedb command.

The Default Configuration wizard and the Prepare Samples wizard use the mqsicreatedb command to create the databases for the broker and the samples, using the default database engine. Therefore, you can list these databases using the mqsilist DatabaseInstanceMgr command.

The mqsisetdbparms command is used to manage the access security for user databases only. It has no effect on Derby databases, which have no access security protection, nor on broker databases in general, which are governed by the access security settings in the broker itself. The rest of this page applies only to the mqsicreatedb, mqsideletedb, and mqsichangedbimgr commands.

Supported database engines

If DB2 version 8.1 Fix Pak 7 or later is installed, both DB2 and Derby databases can be created and used. If DB2 Run-time Client Version 8.2 is installed, only Derby databases are supported. If an earlier version of DB2 is installed, only DB2 databases can be created.

The mqsicreatedb command has an option to select the database engine to use (either DB2 or Derby). The default for this option depends on which database engines are installed. If DB2 Run-time Client Version 8.2 is installed, the default is Derby, otherwise the default is DB2.

Database Instance manager

The databases that are created by mqsicreatedb are managed by a component called the Database Instance manager. This component exists only on Windows. The component stores a list of all the databases created and which database engine is used for each database. Normally there is no process or Windows service required for the Database Instance manager component, and if you start the component it will not be recognized.

The first time a Derby database is created, a Windows service called IBM MQSeries Broker DatabaseInstanceMgr6 is created and started. This service is required in order to access Derby databases. This service can be started or stopped by the mqsistart and mqsistop commands, and automatically starts when Windows is started, if necessary. The service is deleted when the last Derby database is deleted.

There is at most one Database Instance manager Windows service, even if you install WebSphere Event Broker Version 6.0 more than once on your Windows computer (multiple installed instances).

The database commands affect all the databases created in any installed instance on your Windows computer, regardless of the instance under which they are created. For example, the mqsilist DatabaseInstanceMgr6 command lists all the databases that have been created using the mqsicreatedb command on this Windows computer. The mqsichangedbimgr command is used to change the user name and password under which the Database Instance manager Windows service is run. This is required only if passwords change or if user names are updated after the initial installation and configuration. For more information, see "Derby databases on Windows" on page 70.

Creating and deleting databases

The command mqsicreatedb is used to create databases for broker use or for application use. The Prepare Samples wizard and the Default Configuration wizard, for example, use the mqsicreatedb command to create their databases on Windows. Not only is the database itself created (in either DB2 or Derby), but the ODBC data source name (DSN) is also created (with the same name).

Because the data source names and the instance manager component are system wide, you cannot create two databases with the same name, on the same Windows computer, even if they are for brokers on different installed instances of WebSphere Event Broker Version 6.0. The mqsicreatedb command warns you if this is attempted. Any database created using the mqsicreatedb command can be deleted by the mqsideletedb command, even if that database is in use by a broker. See the links to the command descriptions for more information.

Customizing DB2 databases

If you create broker databases in DB2, you must ensure that the Database Heap Size is set to 900 (4 KB pages).

To change the Heap Size on Windows:
1. Start the DB2 Control Center.
2. For each database you have created:
   a. Right-click the database name in the object tree and select **Configure Parameters**.
   b. In the window that opens, select **Performance → Database heap size**. You can reset this within the range 32 to 60000, but are recommended to choose a value of at least 900.
   c. Click **OK**.

To change the Heap Size on UNIX platforms:
1. Open a command window.
2. Enter the db2 command with the parameters shown, specifying the number of pages required:
   ```
   db2 update database configuration for WBRKBKDB using dbheap 900
   ```

Connecting to the databases

The broker uses ODBC to access its database and you must set up an ODBC connection on each broker system. This topic describes the connections that a broker database needs, and the ODBC parameters that you must tailor for the connections:
- "Broker database connections"
- "ODBC.ini Parameters" on page 75

If you are using the Default Configuration Wizard or the database commands to create a broker or a database on Windows, the ODBC connections are automatically created for you.

To define the connection explicitly, link to the topic for your operating system and connection type.
- "Defining an ODBC connection on Windows" on page 76
- "Defining a 32-bit ODBC connection on UNIX systems" on page 77
- "Defining a 64-bit ODBC connection on UNIX systems" on page 80 if you are using WebSphere MQ Version 6 and XA coordination.

Broker database connections:

The number of connections required by the broker to its database depend to some extent on the actions of the message flows it is processing. The following connections are required for each broker:
- Five by internal broker threads.
- One for each Publish/Subscribe neighbor, if the topology has been deployed.
- One for each message flow thread that contains a publication node.
- One for each message flow thread that parses MRM messages.
- A further number if you are using SCADA nodes with WebSphere MQ Everyplace. The exact number to add depends on whether thread pooling is being used (determined by the Use Thread Pooling property of the SCADAInput node):
  - If Use Thread Pooling is not selected (the default setting) add the number of SCADA clients that will connect to the SCADAInput node.
  - If Use Thread Pooling is selected, add the value in the Max Threads property of the SCADAInput node. The default value is 500.

If you are using the same database for several brokers, you must take account of all brokers in your calculations.

The connections for neighbors and publication nodes are only required if you are using retained publications.

When you start a broker, it opens all connections that it requires to the broker database for its own operation. When you stop the broker, it releases all current database connection handles.

If you are using DB2 for your database, the default action taken by DB2 is to limit the number of concurrent connections to a database to the value of the *maxappls* configuration parameter. The default for *maxappls* is 40. If you believe the connections that the broker might require exceed the value for *maxappls*, increase this and the associated parameter *maxagents* to new values based on your calculations.

ODBC.ini Parameters:

Configure the template files by updating the following list of configurable parameters. Some parameters are not applicable to all databases, as shown.

[ODBC Data Sources]
> This stanza describes the data source names (DSNs) for the databases that are configured in the `.odbc.ini` file. You must configure this stanza to show all your databases, and you must include a stanza for each one.

Driver
> - If you are using DB2, specify the location of the driver library that matches your DB2 installation.
> - If you are using Oracle or Sybase, use the path exactly as shown.

Description
> You can change this if you choose. This is a text field for information only and has no bearing on operation.

Database (DB2, Sybase)
> - If you are using DB2, specify the database alias. If you are using a remote DB2 database, you must set up your client-server connection to resolve this alias to the correct database. For more information, see the DB2 documentation.
> - If you are using Sybase, specify the name of the database to which you want to connect by default. If you do not specify a value, the default is the database defined by your system administrator for each user.

NetworkAddress (Sybase only)
> Specify the network address of your Sybase server (this is required for local and remote databases). Specify an IP address as follows:
>
> `<servername or IP address>, <portnumber>`
>
> For example `Sybaseserver, 5000`. You can also specify the IP address directly, for example `199.226.224.34, 5000`. You can find the port number in the Sybase interfaces file which is typically named `interfaces`, `interfac`, or `sql.ini`, depending on the operating system.

ServerName (Sybase, Oracle)
> - If you are using Oracle, specify the "Service Name" that resolves to a "Connect Descriptor", for example through a mapping in the TSNAMES.ORA file.
> - If you are using Sybase, specify the name of the Sybase database server that you have defined on the server computer.

The following parameters are available for Oracle on Linux only, for which a later driver level is installed. Leave the default settings unchanged.

ApplicationUsingThreads

ArraySize

CatalogIncludesSynonyms

CatalogOptions

DefaultLongDataBuffLen

DescribeAtPrepare

EnableDescribeParam

```
EnableNcharSupport

EnableScrollableCursors

EnableStaticCursorsForLongData

EnableTimestampWithTimeZone

LocalTimeZoneOffset

LockTimeOut

LogonID

OptimizePrepare

OptimizeLongPerformance

Password

ProcedureRetResults

UseCurrentSchema
```

Defining an ODBC connection on Windows:

Configure an ODBC data source using the ODBC Data Source Administrator. Select
Start → **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources (ODBC)**,
and select the **System DSN** tab. When you define a new data source, select the
appropriate driver for your database and complete the dialog that is displayed.
Refer to your relevant database product documentation for more information.

Setup considerations specific to WebSphere Event Broker are described below for
the supported databases.

DB2 UDB

When you define a data source for DB2 UDB you must choose the driver
named `IBM DB2 ODBC DRIVER`. Enter the data source name (DSN) and
description, and select the correct database alias from the drop-down list.
You must register the data source as a system data source.

You might find it easier to use the Configuration Assistant:
1. Open the DB2 Configuration Assistant.
2. Right-click the database, and select **Change Database**.
3. Select **Data Source**.
4. Select **Register this database for ODBC** and select the system data
   source option.
5. Click **Finish**.
6. The **Test Connection** dialog opens automatically and you can test the
   various connections.

Microsoft SQL Server

When you define a data source for Microsoft SQL Server you must choose
the driver named `SQL Server`. The driver level must be Version 3.60 or
later. Specify a name and description, and select the correct server from the
drop-down list.

Oracle When you define a data source for Oracle, choose the driver:
• `MQSeries DataDirect Technologies 5.00 32-BIT Oracle`

Complete these steps to configure the driver:

1. Enter the DSN name, description, and server name (where the server name is the "Service Name" that resolves to a "Connect Descriptor", for example through a mapping in the `TSNAMES.ORA` file).
2. Select the appropriate Oracle client version from the drop-down list.
3. Ensure that the **Enable SQLDescribeParam** box is checked. This parameter is on the **Advanced** tab of the driver settings.
4. Create a new registry subkey for each of your DSNs that reference an Oracle database. Use `REGEDIT` to navigate to the correct location:

```
HKEY_LOCAL_MACHINE
     SOFTWARE
         ODBC
             ODBC.INI
```

5. Right-click the DSN, and select **New** → **String Value**. Specify **WorkArounds** for the string, and set the value to 536870912.

Sybase Adaptive Server Enterprise

When you define a data source for Sybase Adaptive Server Enterprise, choose the driver:

- `MQSeries DataDirect Technologies 5.00 32-BIT Sybase Wire Protocol`

Complete these steps to configure the driver:

1. Enter the DSN name, description, network address of the server (for a description of the format of this address, see the explanation for NetworkAddress= in "ODBC.ini Parameters" on page 75).
2. Ensure the **Enable Describe Parameter** box is checked. This parameter is on the **Advanced** tab.
3. Ensure the **Prepare Method** setting is 1 - `Partial`. This parameter is on the **Performance** tab.

Defining a 32-bit ODBC connection on UNIX systems:

In the UNIX environment, there is no ODBC Administrator or Driver Manager. To configure an ODBC data source name (DSN) definition, you must edit the required system information, which is held in a plain text file called `odbc.ini`.

Set the `ODBCINI` environment variable to point to the file, specifying a full path and file name. If you do not set this variable, the path defaults to `<install_dir>/merant/odbc.ini` for a 32-bit execution group, which is created when WebSphere Event Broker is installed. The file must have file permissions of `mqm:mqbrkrs`.

.A sample template is provided as the file `<install_dir>/merant/odbc.ini` which contains examples of how to configure a DSN residing in each of the databases supported by WebSphere Event Broker The sample template files for the supported platforms are displayed below. They show the recommended configuration for the supported databases.

The entries in the stanzas that you must configure to match your local requirements are shown in italics (descriptions of these entries are provided in "ODBC.ini Parameters" on page 75). You must retain all the entries shown, but you can remove a complete stanza if you do not require it for your configuration (for example, if you are using an Oracle database, you can remove the stanzas shown for DB2 and Sybase).

Make a copy of the default file, set `ODBCINI` to point to this copy, and make any changes to the copied file rather than to the default.

Ensure that the appropriate library search path environment variable (LD_LIBRARY_PATH on Solaris and Linux (x86 platform)) is set to reflect the database products to be used. Refer to your database product documentation for more details.

On AIX, if you are using DB2 as your database, a maximum of 10 connections using shared memory are allowed from a single process. This means that if you deploy more than one or two message flows at the same time, you might see connection failures characterized by the DB2 error message SQL1224N. The connection errors are reported in the system log from the broker's execution group. To avoid this problem, you can use TCP/IP mode to connect to DB2. For detailed instructions on how to do this, see DB2 error message SQL1224N is issued when you connect to DB2.

AIX

```
[ODBC Data Sources]
WBRKBKDB=IBM DB2 ODBC Driver
MYDB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.0 Oracle
SYBASEDB=DataDirect 5.0 Sybase Wire Protocol

[WBRKBKDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.a
Description=WBRKBKDB DB2 ODBC Database
Database=WBRKBKDB

[MYDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.a
Description=MYDB DB2 ODBC Database
Database=MYDB

[ORACLEDB]
Driver=<Your installation directory>/merant/lib/UKor820.so
Description=DataDirect 5.0 Oracle
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
ProcedureRetResults=1
WorkArounds=536870912


[SYBASEDB]
Driver=<Your installation directory>/merant/lib/UKase20.so
Description=DataDirect 5.0 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<YourServerName,YourPortNumber>
SelectUserName=1

[ODBC]
Trace=0
TraceFile=<A directory with plenty of free space>/odbctrace.out
TraceDll=<Your installation directory>/merant/lib/odbctrac.so
InstallDir=<Your installation directory>/merant
UseCursorLib=0
IANAAppCodePage=4
```

HP-UX

```
[ODBC Data Sources]
WBRKBKDB=IBM DB2 ODBC Driver
MYDB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.0 Oracle
```

```
SYBASEDB=DataDirect 5.0 Sybase Wire Protocol

[WBRKBKDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.sl
Description=WBRKBKDB DB2 ODBC Database
Database=WBRKBKDB

[MYDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.sl
Description=MYDB DB2 ODBC Database
Database=MYDB

[ORACLEDB]
Driver=<Your installation directory>/merant/lib/UKor820.sl
Description=DataDirect 5.0 Oracle
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
ProcedureRetResults=1
WorkArounds=536870912

[SYBASEDB]
Driver=<Your installation directory>/merant/lib/UKase20.sl
Description=DataDirect 5.0 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<YourServerName,YourPortNumber>
SelectUserName=1

[ODBC]
Trace=0
TraceFile=<A directory with plenty of free space>/odbctrace.out
TraceDll=<Your installation directory>/merant/lib/odbctrac.sl
InstallDir=<Your installation directory>/merant
UseCursorLib=0
IANAAppCodePage=4
```

Linux (x86 platform)

```
[ODBC Data Sources]
WBRKBKDB=IBM DB2 ODBC Driver
MYDB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.0 Oracle
SYBASEDB=DataDirect 5.0 Sybase Wire Protocol

[WBRKBKDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.so
Description=WBRKBKDB DB2 ODBC Database
Database=WBRKBKDB

[MYDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.so
Description=MYDB DB2 ODBC Database
Database=MYDB

[ORACLEDB]
Driver=<Your installation directory>/merant/lib/UKor820.so
Description=DataDirect 5.0 Oracle
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
ProcedureRetResults=1
WorkArounds=536870912

[SYBASEDB]
Driver=<Your installation directory>/merant/lib/UKase20.so
```

```
Description=DataDirect 5.0 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<YourServerName,YourPortNumber>
SelectUserName=1

[ODBC]
Trace=0
TraceFile=<A Directory with plenty of free space>/odbctrace.out
TraceDll=<Your installation directory>/merant/lib/odbctrac.so
InstallDir=<Your installation directory>/merant
UseCursorLib=0
IANAAppCodePage=4
```

Solaris

```
[ODBC Data Sources]
WBRKBKDB=IBM DB2 ODBC Driver
MYDB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.0 Oracle
SYBASEDB=DataDirect 5.0 Sybase Wire Protocol

[WBRKBKDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.so
Description=WBRKBKDB DB2 ODBC Database
Database=WBRKBKDB

[MYDB]
Driver=/u/db2inst1/sqllib/lib/libdb2.so
Description=MYDB DB2 ODBC Database
Database=MYDB

[ORACLEDB]
Driver=<Your installation directory>/merant/lib/UKor820.so
Description=DataDirect 5.0 Oracle
ServerName=<YourServerName>
EnableDescribeParam=1
ProcedureRetResults=1
OptimizePrepare=1
WorkArounds=536870912

[SYBASEDB]
Driver=<Your installation directory>/merant/lib/UKase20.so
Description=DataDirect 5.0 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<YourServerName>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<YourServerName,YourPortNumber>
SelectUserName=1

[ODBC]
Trace=0
TraceFile=<A Directory with plenty of free space>/odbctrace.out
TraceDll=<Your installation directory>/merant/lib/odbctrac.so
InstallDir=<Your installation directory>/merant
UseCursorLib=0
IANAAppCodePage=4
```

Defining a 64-bit ODBC connection on UNIX systems:

You need to define a 64-bit ODBC connection if you are using WebSphere MQ
Version 6 and XA coordination. The XA-coordinated databases must appear in the

ODBCINI64 stanza file so that WebSphere MQ may perform the XA coordination. WebSphere MQ Version 6 supports a 64-bit environment.

In the UNIX environment, there is no ODBC Administrator or Driver Manager. To configure an ODBC data source name (DSN) definition, you must edit the required system information, which is held in a plain text file called `odbc64.ini`.

Set the `ODBCINI64` environment variable to point to the file, specifying a full path and file name. If you do not set this variable, the path defaults to `<install_dir>/DD64/odbc64.ini`, which is created when WebSphere Event Broker is installed. The file must have file permissions of `mqm:mqbrkrs`.

A sample template is provided as the file `<install_dir>/DD64/odbc64.ini` which contains examples of how to configure a DSN residing in each of the databases supported by WebSphere Event Broker The sample template files for the supported platforms are displayed below. They show the recommended configuration for the supported databases.

The entries in the stanzas that you must configure to match your local requirements are shown in italics (descriptions of these entries are provided in "ODBC.ini Parameters" on page 75). You must retain all the entries shown, but you can remove a complete stanza if you do not require it for your configuration (for example, if you are using an Oracle database, you can remove the stanzas shown for DB2 and Sybase).

Make a copy of the default file, set `ODBCINI64` to point to this copy, and make any changes to the copied file rather than to the default.

Ensure that the appropriate library search path environment variable (LD_LIBRARY_PATH on Solaris) is set to reflect the database products to be used. Refer to your database product documentation for more details.

On AIX, if you are using DB2 as your database, a maximum of 10 connections using shared memory are allowed from a single process. This means that if you deploy more than one or two message flows at the same time, you might see connection failures characterized by the DB2 error message SQL1224N. The connection errors are reported in the system log from the broker's execution group. To avoid this problem, you can use TCP/IP mode to connect to DB2. For detailed instructions on how to do this, see DB2 error message SQL1224N is issued when you connect to DB2.

AIX

```
[ODBC Data Sources]
WBRKBKDB=IBM DB2 ODBC Driver
MYDB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.0 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.0 64bit Sybase Wire Protocol

[WBRKBKDB]
Driver=libdbWrapper64.so
Description=WBRKBKDB DB2 ODBC Database
Database=WBRKBKDB

[MYDB]
Driver=libdbWrapper64.so
Description=MYDB DB2 ODBC Database
Database=MYDB

[ORACLEDB]
```

```
                    Driver=<Your installation directory>/DD64/lib/UKora20.so
                    Description=DataDirect 5.0 Oracle Wire Protocol
                    HostName=<Your Oracle Server Machine Name>
                    PortNumber=<Port on which Oracle is listening on HostName>
                    SID=<Your Oracle SID>
                    CatalogOptions=0
                    ProcedureRetResults=1
                    EnableStaticCursorForLongData=0
                    ApplicationUsingThreads=1
                    ServerName=<YourServerName>
                    EnableDescribeParam=1
                    OptimizePrepare=1
                    WorkArounds=536870912


                    [SYBASEDB]
                    Driver=<Your installation directory>/DD64lib/UKase20.so
                    Description=DataDirect 5.0 Sybase Wire Protocol
                    Database=<sybasedb>
                    ApplicationsUsingThreads=1
                    EnableDescribeParam=1
                    OptimizePrepare=1
                    SelectMethod=0
                    NetworkAddress=<YourServerName,YourPortNumber>
                    SelectUserName=1

                    [ODBC]
                    Trace=0
                    TraceFile=<A directory with plenty of free space>/odbctrace.out
                    TraceDll=<Your installation directory>/DD64/lib/odbctrac.so
                    InstallDir=<Your installation directory>/DD64
                    UseCursorLib=0
                    IANAAppCodePage=4
```

HP-UX

```
                    [ODBC Data Sources]
                    WBRKBKDB=IBM DB2 ODBC Driver
                    MYDB=IBM DB2 ODBC Driver
                    ORACLEDB=DataDirect 5.0 64bit Oracle Wire Protocol
                    SYBASEDB=DataDirect 5.0 64bit Sybase Wire Protocol

                    [WBRKBKDB]
                    Driver=libdb2Wrapper64.sl
                    Description=WBRKBKDB DB2 ODBC Database
                    Database=WBRKBKDB

                    [MYDB]
                    Driver=libdb2Wrapper64.sl
                    Description=MYDB DB2 ODBC Database
                    Database=MYDB

                    [ORACLEDB]
                    Driver=<Your installation directory>/DD64/lib/UKora20.sl
                    Description=DataDirect 5.0 Oracle Wire Protocol
                    HostName=<Your Oracle Server Machine Name>
                    PortNumber=<Port on which Oracle is listening on HostName
                    SID=<Your Oracle SID>
                    CatalogOptions=0
                    ProcedureRetResults=1
                    EnableStaticCursorForLongData=0
                    ApplicationUsingThreads=1
                    ServerName=<YourServerName>
                    EnableDescribeParam=1
                    OptimizePrepare=1
                    WorkArounds=536870912

                    [SYBASEDB]
```

```
            Driver=<Your installation directory>/DD64/lib/UKase20.sl
            Description=DataDirect 5.0 Sybase Wire Protocol
            Database=<sybasedb>
            ApplicationUsingThreads=1
            EnableDescribeParam=1
            OptimizePrepare=1
            SelectMethod=0
            NetworkAddress=<YourServerName,YourPortNumber>
            SelectUserName=1

            [ODBC]
            Trace=0
            TraceFile=<A directory with plenty of free space>/odbctrace.out
            TraceDll=<Your installation directory>/DD64/lib/odbctrac.sl
            InstallDir=<Your installation directory>/DD64
            UseCursorLib=0
            IANAAppCodePage=4
```

Solaris
```
            [ODBC Data Sources]
            WBRKBKDB=IBM DB2 ODBC Driver
            MYDB=IBM DB2 ODBC Driver
            ORACLEDB=DataDirect 5.0 64bit Oracle Wire Protocol
            SYBASEDB=DataDirect 5.0 64bit Sybase Wire Protocol

            [WBRKBKDB]
            Driver=libdb2Wrapper64.so
            Description=WBRKBKDB DB2 ODBC Database
            Database=WBRKBKDB

            [MYDB]
            Driver=libdb2Wrapper64.so
            Description=MYDB DB2 ODBC Database
            Database=MYDB

            [ORACLEDB]
            Driver=<Your installation directory>/DD64/lib/UKor820.so
            Description=DataDirect 5.0 Oracle Wire Protocol
            HostName=<Your Oracle Server Machine Name>
            PortNumber=<Port on which Oracle is listening on HostName
            SID=<Your Oracle SID>
            CatalogOptions=0
            ProcedureRetResults=1
            EnableStaticCursorForLongData=0
            ApplicationUsingThreads=1
            ServerName=<YourServerName>
            EnableDescribeParam=1
            OptimizePrepare=1
            WorkArounds=536870912

            [SYBASEDB]
            Driver=<Your installation directory>/DD64/lib/UKase20.so
            Description=DataDirect 5.0 Sybase Wire Protocol
            Database=<sybasedb>
            ServerName=<YourServerName>
            ApplicationUsingThreads=1
            EnableDescribeParam=1
            OptimizePrepare=1
            SelectMethod=0
            NetworkAddress=<YourServerName,YourPortNumber>
            SelectUserName=1

            [ODBC]
            Trace=0
            TraceFile=<A Directory with plenty of free space>/odbctrace.out
```

```
TraceDll=<Your installation directory>/DD64/lib/odbctrac.so
InstallDir=<Your installation directory>/DD64
UseCursorLib=0
IANAAppCodePage=4
```

Authorizing access to the databases

When you have created a broker database, you must authorize the owning
component to access it. When you invoke the mqsicreatebroker command, you
must specify at least one ID for runtime authorization; you can specify a second ID
specifically for database connect authorization. If you do not specify the second ID,
the component uses its runtime ID (the service user ID) for database access as well.

You specify the service user ID and its password with the -i and -a flags, and the
optional database connection ID and password with the -u and -p flags.

The way that you authorize access depends on the database you are using, and the
platform on which you have created it.

DB2 authorization:

To authorize access to a DB2 database on Windows:
1. Start the DB2 Control Center, if it is not already active.
2. Expand the object tree until you find the database you created for the broker.
3. Expand the tree under this database and select the **User and Group Objects**
   folder. The **DB Users** and **DB Groups** folders are displayed in the right pane.
4. Right-click the **DB Users** folder in the right pane and select **Add**. The Add User
   notebook opens.
5. Select the user ID you want to authorize to access the database (for example,
   mqsiuid) from the drop-down list. This must be the user ID that you specify to
   be used for database access when you create the broker. The drop-down list
   contains all user IDs that are defined to Windows. If you cannot see the user ID
   you specified in the command, you must define it before you can give it
   authority.
6. Select the appropriate options from the choices in the box labelled **Choose the**
   appropriate authorities to grant to the selected user for the database. These
   are:
   • Connect database
   • Create tables
   • Create packages
   • Register functions to execute in database manager's process
7. Click **OK**. The authorities are granted. The dialog closes.
8. Close the DB2 Control Center.

If you prefer, you can use the command line commands shown below for UNIX
platforms (for a broker database); the commands are the same on Windows.

To authorize access to a DB2 database on UNIX:
1. Connect to the database with a user ID that has DB2 system administration
   (SYSADM or DBADM) authority (substitute the correct database and ID in this
   command):
   ```
   db2 connect to <broker_db> user <SysAd_id>
   ```

2. Issue the following command to grant the required privileges to the user ID that the broker will use to connect to the database (substitute the correct ID for your broker in this command if you are not using the sample mqsiuid):

    ```
    db2 grant connect, createtab, bindadd, create_external_routine on database to user mqsiuid
    ```

If you need further guidance about any of these tasks, use the online help facility of the DB2 Control Center. Your database administrator might also be able to offer advice and assistance.

Oracle authorization:

To authorize access to an Oracle database:

1. Logon to the database using SQL*Plus as the Oracle DBA (database administrator).
2. Modify the privileges of the user ID that you have specified for database connection to ensure that the broker can successfully access the database. The user ID needs quota in its tablespaces and sufficient privilege to allow the creation of, and updates to, the broker tables:

    ```
    GRANT CREATE SESSION TO dbid;
    GRANT CREATE TABLE TO dbid;
    ```

For further information, refer to the Oracle documentation, or consult your database administrator.

Using retained publications with a Sybase database

If you have created a broker that uses a Sybase database, and you expect extensive use of retained publications with multiple topics, you are recommended to apply row-level locking to the retained publications table in the database. If you do not plan to use retained publications, or expect their use to be infrequent, you do not have to make this change.

If you do not apply row-level locking, and your use of retained publications is too great, the broker will experience deadlock problems.

To apply row-level locking:

1. At a command prompt enter the following command:

    ```
    isql -Umqsiuid -Pmqsipw
    ```

    If you have authorized another user ID and password for broker access to this database, substitute your values for mqsiuid and mqsipw in that command.
2. Connect to the broker database with this command:

    ```
    use WBRKBKDB
    ```

    If you have created your broker database with a different name, substitute your name for WBRKBKDB in that command.
3. Update the retained publications table to use row-level locking with this command:

    ```
    alter table mqsiuid.BRETAINEDPUBS lock datarows
    ```

    If the owner of this database instance is not mqsiuid, substitute the correct schema name in that command.
4. Apply the change with this command:

    ```
    go
    ```

You can check that the change has been successfully applied by entering the commands:

```
sp_help BRETAINEDPUBS
go
```

The locking scheme is displayed: `lock scheme datarows`.

If the change has **not** completed, it is displayed as: `lock scheme allpages`.

Creating a broker

Before you start:

You must complete the following tasks:
- Ensure that your ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385.
- Create and configure the broker database. Configure ODBC to allow the broker to access its database. Refer to "Configuring component databases" on page 67 for information on all these tasks.
- Create a new user ID to access the broker database tables (persistent store).
- If you want to configure a broker on Windows, UNIX systems, or Linux as a WebSphere MQ trusted application, refer to "Using WebSphere MQ trusted applications" before creating the broker. Using trusted (fastpath) applications allows the broker and the WebSphere MQ queue manager agent to run in the same process, thus improving overall performance.
- If you want to add a User Name Server to your broker domain, create the broker with the additional -s and -j parameters on the **mqsicreatebroker** command. These parameters allow the broker to communicate with the User Name Server. See "Enabling a User Name Server" on page 114 for details of the parameters to specify.

You create a broker using the command line on the system where the broker component is installed.

The broker must be given a name that is unique within the broker domain. Broker names are case-sensitive on all supported platforms, except the Windows platforms. The broker's name and its queue manager name are required when creating the broker reference in the workbench.

To create a broker, follow the link for the appropriate platform:
- "Creating a broker on AIX" on page 88
- "Creating a broker on HP-UX" on page 89
- "Creating a broker on Linux" on page 90
- "Creating a broker on Solaris" on page 91
- "Creating a broker on Windows" on page 92
- "Creating a broker on z/OS" on page 94

Using WebSphere MQ trusted applications

Before you start:

You must complete the following tasks:

- Ensure that your user ID is a member of the **mqm** group. On HP-UX and Solaris, specify the user ID **mqm** as the service user ID when you create the broker. On Windows, use any service user ID that is a member of **mqm**. Refer to "Security requirements for administrative tasks" on page 385.
- Review the restrictions that WebSphere MQ places on trusted applications that apply to your environment. See the section "Connection to a queue manager using the MQCONNX call" in the *WebSphere MQ Application Programming Guide*, available on the WebSphere MQ library Web page.

You cannot configure a broker as a trusted application on AIX and z/OS.

You can configure a broker to run as a trusted (fastpath) application on the other platforms that are supported. It runs in the same process as the WebSphere MQ queue manager agent, and thereby improves the overall system performance.

A broker does not run as a trusted application by default; you either create a trusted application using the **"mqsicreatebroker command" on page 257**, or modify an existing broker using the **"mqsichangebroker command" on page 221**.

The Configuration Manager is configured to run as a WebSphere MQ trusted application for performance reasons, and you cannot change the configuration.

Configuring a broker as a trusted application does not affect the operation of WebSphere MQ channel agents or listeners. For more information about running these as trusted applications, see the section "Running channels and listeners as trusted applications" in *WebSphere MQ Intercommunication*, available on the WebSphere MQ library Web page.

Take care when deploying user-defined nodes or parsers. Because a trusted application (the broker) runs in the same operating system process as the queue manager, a user-defined node or parser might compromise the integrity of the queue manager. Consider fully the restrictions that apply to your environment and test user-defined nodes and parsers in a non-trusted environment before deploying them in a trusted broker.

You can either configure a new broker to run as a trusted application, or modify an existing broker.
- To configure a new broker, on a command line run the **mqsicreatebroker** command with the -t flag,, which specifies that the broker is created as a trusted application.

  For example, enter the following command to create a broker called WBRK_BROKER as a trusted application:

  ```
  mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
  -q WBRK_QM -n WBRKBKDB -u wbrkuid -p wbrkpw -t
  ```

  Refer to "Creating a broker" on page 86 for more detailed information about how to create a broker for your platform.
- To modify an existing broker:
  1. Run the **mqsistop** command on the command line to stop the broker.
  2. Run the **mqsichangebroker** command with the -t flag. For example, enter the following command to modify a broker called WBRK_BROKER to run as a trusted application:

     ```
     mqsichangebroker WBRK_BROKER -t
     ```

     You might need to change the service user ID and password if you did not originally create the broker to use an appropriate service user ID.

Configuring the broker domain **87**

Refer to "Modifying a broker" on page 131 for more detailed information on how to modify a broker for your platform.

3. Restart the broker using the **mqsistart** command. The broker restarts with fastpath set.

Creating a broker on AIX

If you expect your broker to handle a heavy load (running a large number of message flows, or interacting with a number of databases), it might exceed the default memory limits that you configured during AIX installation. If the broker experiences memory problems, check the maximum size of the data segment for the user under which your broker is running (the `ServiceUserID`), and consider increasing its value. The size is defined in `/etc/security/limits`.

To create a broker:

1. Log on using your service user ID. If you use the `su` command to switch user you must ensure that you invoke that user's profile.

   a. Enter `su - <user ID>` to run a different user's profile.

2. Run `'. <install_dir>/bin/mqsiprofile'` to source the `mqsiprofile` script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

3. Ensure that the broker database instance profile has been run. Invoke the SQL profile that was created when the broker database was created.

4. Enter the following command to create the broker:

   ```
   mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
   -q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
   ```

   If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

   In the command above:

   WBRK_BROKER
   > Is the broker name.

   wbrkuid
   > Is the service user ID that is used to run the broker.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_QM
   > Is the name of the WebSphere MQ queue manager that the broker will use. This queue manager is created if it does not exist.

   WBRKBKDB
   > Is the name of the existing broker database, the broker tables will be created within this database.

   dbuid Is the user ID that has read, write and create access permissions for the database. This is the user ID that will be used to read and update the broker's persistent store.

   dbpw Is the password that is associated with the database user ID, `dbuid`.

On completion of this task, you have:

- Created a broker called WBRK_BROKER.
- Created and started a WebSphere MQ queue manager called WBRK_QM.

- Created and set up the WebSphere MQ resources required by the broker, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreatebroker** command.

  Note: Note that this command does **not** start the listener

- Ensured that all database tables required by the broker have been created in the database WBRKBKDB.

Now that you have created a physical broker, you are ready to:

1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

2. Add the broker to the broker domain.
   - To add the broker using the workbench, see "Adding a broker to a broker domain" on page 150.
   - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

Creating a broker on HP-UX

To create a broker:

1. Log on using your service user ID. If you use the `su` command to switch user ensure that you invoke that user's profile.

   a. Enter su - <user ID> to run a different user's profile.

2. Run '. <install_dir>/bin/mqsiprofile' to source the `mqsiprofile` script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

3. Ensure that the broker database instance profile has been run. Invoke the SQL profile that was created when the broker database was created.

4. Enter the following command to create the broker:

   ```
   mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
   -q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
   ```

   If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

   In the command above:

   WBRK_BROKER
   > Is the broker name.

   wbrkuid
   > Is the service user ID that is used to run the broker.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_QM
   > Is the name of the WebSphere MQ queue manager that the broker will use. This queue manager is created if it does not exist.

   WBRKBKDB
   > Is the name of the existing broker database, the broker tables will be created within this database.

dbuid Is the user ID that has read, write and create access permissions for the database. This is the user ID that will be used to read and update the broker's persistent store.

dbpw Is the password that is associated with the database user ID, dbuid.

On completion of this task, you have:
- Created a broker called WBRK_BROKER.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the broker, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreatebroker** command.

  Note: Note that this command does **not** start the listener
- Ensured that all database tables required by the broker have been created in the database WBRKBKDB.

Now that you have created a physical broker, you are ready to:
1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Add the broker to the broker domain.
   - To add the broker using the workbench, see "Adding a broker to a broker domain" on page 150.
   - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

Creating a broker on Linux

Follow the steps detailed in this task for creating a broker on either Linux (zSeries platform) and Linux (x86 platform).

To create a broker:
1. Log on using your service user ID. If you use the su command to switch user you must ensure that you invoke that user's profile.
   a. Enter su - <user ID> to run a different user's profile.
2. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
3. Ensure that the broker database instance profile has been run. Invoke the SQL profile that was created when the broker database was created.
4. Enter the following command to create the broker:

   ```
   mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
   -q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
   ```

   Note: The service user ID you use on the mqsicreatebroker command line **must** be the user ID you use to invoke the mqsicreatebroker command or root.

   If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

In the command above:

WBRK_BROKER
Is the broker name.

wbrkuid
Is the service user ID that is used to run the broker.

wbrkpw
Is the password for the service user ID.

WBRK_QM
Is the name of the WebSphere MQ queue manager that the broker will use. This queue manager is created if it does not exist.

WBRKBKDB
Is the name of the existing broker database, the broker tables will be created within this database.

dbuid Is the user ID that has read, write and create access permissions for the database. This is the user ID that will be used to read and update the broker's persistent store.

dbpw Is the password that is associated with the database user ID, dbuid.

On completion of this task, you have:
- Created a broker called WBRK_BROKER.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the broker, and defined these on the queue manager. This includes the default dead letter-queue (DLQ), which is automatically enabled by running the **mqsicreatebroker** command.

  Note: Note that this command does **not** start the listener
- Ensured that all database tables required by the broker have been created in the database WBRKBKDB.

Now that you have created a physical broker, you are ready to:
1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130
2. Add the broker to the broker domain.
   - To add the broker using the workbench, see "Adding a broker to a broker domain" on page 150.
   - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

Creating a broker on Solaris

To create a broker:
1. Log on using your service user ID. If you use the su command to switch user you must ensure that you invoke that user's profile.
   a. Enter su - <user ID> to run a different user's profile.

2. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

3. Ensure the broker database instance profile has been run. Invoke the SQL profile that was created when the broker database was created.

4. Enter the following command to create the broker: If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
-q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
```

where:
- WBRK_BROKER is the broker name.
- wbrkuid is the service user ID that is used to run the broker.
- wbrkpw is the password for the service user ID.
- WBRK_QM is the name of the WebSphere MQ queue manager that the broker will use. This queue manager is created if it does not exist.
- WBRKBKDB is the name of the existing broker database, the broker tables will be created within this database.
- dbuid is the user ID that has read, write and create access permissions for the database. This is the user ID that will be used to read and update the broker's persistent store.
- dbpw is the password that is associated with the database user ID, dbuid.

On completion of this task, you have:
- Created a broker called WBRK_BROKER.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the broker, and defined these on the queue manager. This includes the default dead letter queue (DLQ) which is automatically enabled by running the **mqsicreatebroker** command.

   Note: Note that this command does **not** start the listener
- Ensured that all database tables required by the broker have been created in the database WBRKBKDB.

Now that you have created a physical broker, you are ready to:
1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers and Configuration Manager). This allows components in your broker domain, that are supported by different queue managers, to exchange messages and communicate effectively. Refer to "Connecting components" on page 130
2. Add the broker to the broker domain.
   - To add the broker using the workbench, see "Adding a broker to a broker domain" on page 150.
   - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

Creating a broker on Windows

To create a broker:
1. Open a WebSphere Event Broker command prompt for the desired runtime.

2. Enter the following command to create the broker:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
-q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
```

If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

In the command above:

WBRK_BROKER
> Is the broker name.

wbrkuid
> The broker runs as a Windows service. A service needs to log on with some credentials; "wbrkuid" is the user id for the credentials.

wbrkpw
> Is the password for the service user ID.

WBRK_QM
> Is the name of the WebSphere MQ queue manager that the broker will use. This queue manager is created if it does not exist.

WBRKBKDB
> Is the name of the existing broker database, the broker tables will be created within this database.

dbuid Is the user ID that has read, write and create access permissions for the database. This is the user ID that will be used to read and update the broker's persistent store.

dbpw Is the password that is associated with the database user ID, dbuid.

On completion of this task, you have:
- Created a broker called WBRK_BROKER.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the broker, and defined these on the queue manager. This includes the default dead letter-queue (DLQ), which is automatically enabled by running the **mqsicreatebroker** command.

  Note: Note that this command does **not** start the listener
- Ensured that all database tables required by the broker have been created in the database WBRKBKDB.

Now that you have created a physical broker, you are ready to:
1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130
2. Add the broker to the broker domain.
   - To add the broker using the workbench, see "Adding a broker to a broker domain" on page 150.
   - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

Creating a broker on z/OS

To create your broker, perform the following steps in order.

1. "Information required to create a broker"
2. "Creating the broker PDSE" on page 96
3. "Creating the broker directory on z/OS" on page 96
4. "Customizing the broker component data set" on page 97
5. "Customizing the broker JCL" on page 98
6. "Creating the environment file" on page 100
7. "Prime DB2" on page 100
8. "Creating the broker component" on page 101
9. "Copy started task to the procedures library" on page 101
10. "Starting the component" on page 102.

Information required to create a broker:

This is part of the larger task of creating a broker on z/OS.

You need to complete the information in each of the following tables before continuing:

- "Installation information - Broker and User Name Server"
- "DB2 information"
- "Component information - broker" on page 95

Installation information - Broker and User Name Server:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
Fully qualified name of the product's SBIPPROC dataset	N/A.	<hlq>.SBIPPROC	
Fully qualified name of the product's SBIPSAMP dataset	N/A	<hlq>.SBIPSAMP	
File system directory where the product has been installed.	++INSTALL++	/usr/lpp/mqsi	
LE high-level-qualifier	++LEHLQ++	PP.ADLE370.ZOS140	
WebSphere MQ high-level-qualifier.	++WMQHLQ++	MQM.V530	
Location of Java installation.	++JAVA++	/usr/lpp/java/IBM/J1.4	
Locale of environment where commands are run by submitting JCL.	++LOCALE++	C	
Time zone of environment where commands are run by submitting JCL.	++TIMEZONE++	GMT0BST	

DB2 information:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
DB2 high-level qualifier	++DB2HLQ++.	SYS2.DB2.V710	
DB2 run library value	++DB2RUNLIB++	DSN710PK.RUNLIB.LOAD	
DB2subsystem identifier	++DB2SUBSYSTEM++	DFK4	
DB2 plan name	++DB2DSNACLIPLAN++	DSNACLI	
DB2 program value	++DB2SAMPLEPROGRAM++	DSNTEP2	
DB2 plan value	++DB2SAMPLEPROGRAMPLAN++	DSNTEP71	
DB2 location value of the DB2 subsystem	++DB2LOCATION++	DSN710PK	
Specifies the DB2converter	++DB2CONVERSION++	SINGLE	
DB2 user ID for the component and commands.	++DB2CURRENTSQLID++	MQPIBRK	
DB2 table owner user ID	++DB2TABLEOWNER++	MQP1BRK	

Component information - broker:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
The broker's user ID HFS home directory	++HOME++	/u/mqp1brk	
Queue Manager associated with the broker.	++QUEUEMANAGER++	MQP1	
HFS directory where the broker is to exist.	++COMPONENTDIRECTORY++	/var/wmqi/MQP1BRK	
Broker name	++COMPONENTNAME++	MQP1BRK	
The dataset where all JCL relevant to the broker is saved.	++COMPONENTDATASET++	TESTDEV.A.V6.MQP1BRK	
Profile name	++COMPONENTPROFILE++	BIPBPROF, BIPUPROF, or BIPCPROF	**BIPBPROF**
Name of the Started Task JCL; can be a maximum of 8 characters.	++STARTEDTASKNAME++	MQP1BRK	
The name of the broker DB2 database.	++DB2DATABASE++	DMQP1BRK	
The name of the broker DB2 storage group.	++DB2STORAGEGROUP++	MQP1STOR	
The name of the broker DB2 bufferpool.	++DB2BUFFERPOOL++	BP0	

mqsicreatebroker options	++OPTIONS++	Any additional optional parameters for the **mqsicreatebroker** command	

Creating the broker PDSE:

This is part of the larger task of creating a broker on z/OS.

Each broker requires a PDSE or a PDS. A PDSE is preferable to a PDS because free space is available without the need to compress the data set.

Create the broker PDSE, for example using option 3.2 on ISPF. The name of the PDSE must be the same as the JCL variable ++COMPONENTDATASET++. Allocate a data set with:
- Eight directory blocks
- 15 tracks (or 1 cylinder) of 3390 DASD with a record format of fixed blocked 80
- A suitable block size (for example 27920)
- Data set type of library

Creating the broker directory on z/OS:

This is part of the larger task of creating a broker on z/OS .

Before you start

Before starting this step, you must have completed "Information required to create a broker" on page 94 and "Creating the broker PDSE."

Use the TSO command OMVS to get into OMVS. Create the broker root directory using:

```
 mkdir -p <ComponentDirectory>
```

The name of the directory must be the same as the JCL variable ++COMPONENTDIRECTORY++.

Use the following command to display the contents of the directory, which is currently empty:

```
ls -dl /var/wmqi/MQP1BRK
```

Use the ls -l /var/wmqi/MQP1BRK command to display the permissions on the directory.

Ensure that the user ID of the person doing the customization has a group that matches the group of the directory. Issue the following command, where userid is the ID you want to check:

```
id userid
```

Check that the directory has a valid group and that the group has rwx permissions. If they do not, issue the following command to set the group of the directory:

```
chgrp group pathname
```

For example:

```
chgrp WMQI /var/wmqi/MQP1BRK
```

You have to be the owner of the group or have superuser authority to issue this command.

To give the group READ, WRITE, and EXECUTE access, issue:
```
chmod g=rwx <pathname>
```

For example:
```
chmod g=rwx /usr/wmqi/MQP1BRK
```

To display the amount of space used and available, use the command:
```
df -P /var/wmqi/MQP1BRK
```

Creating the broker runtime environment on z/OS:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed the following task:
- "Creating the broker directory on z/OS" on page 96

Check that you have access to the installation libraries. For example, use the following command to display information about the directory structure:
```
ls -l /usr/lpp/wmqi
```

This command displays lines like the following example:
```
drwxr-xr--   2 A8LIGH2  MQM        8192 Aug 20 16:22 lib
```

In this example, MQM is the group associated with the file or directory structure. Those who will be using the files, for example the WebSphere Event Broker administrators, and the started task user IDs, should be in the group of the files. Use the `id userid` command to display the groups a user ID is in. If a user ID is not in the group, you can:
- Add the user ID to the group using standard ESM facilities.
- Change the group of the file using the example command below:
  ```
  chgrp -R WMQI /usr/lpp/wmqi
  ```

You might need super user authority to issue this command.

Customizing the broker component data set:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Information required to create a broker" on page 94.
1. Create the broker data set as described in "Creating the broker PDSE" on page 96; this is identified by ++COMPONENTDATASET++. Copy all the following JCL to the broker data set you have created so that you can customize the source.

   Note: This then forms the source for the broker.

2. Copy the following JCL from `<hlq>.SBIPSAMP` to `++COMPONENTDATASET++`.
   - BIPBPROF (broker profile). "Sample BIPBPROF file" on page 348 lists the shipped BIPBPROF file.
   - BIPDSNAO (DB2 dsnaoini). "Sample BIPDSNAO file" on page 369 lists the shipped BIPDSNAO file.
3. Copy the following members from `<hlq>.SBIPPROC` to `++COMPONENTDATASET++`. Note that you copy only the following files.
   - BIPALDB - job to alter DB2 table spaces and tables.
   - BIPALMQ - job to alter WebSphere MQ table spaces and tables.
   - BIPBRWS - job to run the mqsibrowse command.
   - BIPCHBK - job to issue the mqsichangebroker command.
   - BIPCHMS - job to issue the mqsichangeflowstats command.
   - BIPCHPR - job to issue the mqsichangeproperties command.
   - BIPCLMP - job to issue the mqsiclearmqpubsub command.
   - BIPCRBK - job to create a broker.
     "Sample BIPCBRK file" on page 352 lists the shipped BIPCRBK file.
   - BIPCRDB - job to create the DB2 storage group, database and table spaces.
     "Sample BIPCRDB file" on page 360 lists the shipped BIPCRDB file.
   - BIPDLBK - job to issue the mqsideletebroker command.
   - BIPDLDB - job to delete the DB2 storage group, database and table spaces.
   - BIPEDIT - JCL customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDBK.
   - BIPFMLG - job to format the log.
   - BIPGEN - generate ENVFILE.
     "Sample BIPGEN file" on page 370 lists the shipped BIPGEN file.
   - BIPJNMP - job to issue the mqsijoinmqpubsub command.
   - BIPLIST - job to issue the mqsilist command.
   - BIPLSMP - job to issue the mqsilistmqpubsub command.
   - BIPMGTB - job for changing DB2 definitions when migrating from Version 2.1 to Version 6.0.
   - BIPRELG - job to read logs.
   - BIPRPMS - job to issue the mqsireportflowstats command.
   - BIPRPPR - job to issue the mqsireportproperties command.
   - BIPSDBP - job to define a data source, user ID, and password for user data sources.
   - BIPBRKP (started task). Rename BIPBRKP to **the same as the ++STARTEDTASKNAME++**.
     "Sample BIPBRKP file" on page 375 lists the shipped BIPBRKP file.

Customizing the broker JCL:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the broker component data set" on page 97.

All JCL has a standard header, comprising:

1. A brief description of its function.
2. A description where further information can be found, relating to the function of the JCL.
3. If appropriate, a topic number.
4. The section listing the JCL variables themselves.
5. Each JCL file defines its own STEPLIB
6. Some JCL files, for example, BIPRELG might require DB2 defined in the STEPLIB for a broker component. This must be removed from the JCL if the component is either a Configuration Manager or User Name Server, as it is not required.

You can customize the files using an ISPF edit macro that you have to tailor, or you can make changes to each of the PDSE members manually.

BIPEDIT is a REXX program that can be used to assist you in customizing your JCL. Once you have customized BIPEDIT you can run this REXX program against the other JCL files to change their JCL variables.

First of all, customize the renamed BIPEDIT file using the information you collected in:
- "Installation information - Broker and User Name Server" on page 94
- "DB2 information" on page 94
- "Component information - broker" on page 95

Next, you must activate the renamed BIPEDIT file before you customize any other JCL files. Do this by running the following TSO command:

    ALTLIB ACTIVATE APPLICATION(EXEC) DA('COMPONENTDATASET')

where 'COMPONENTDATASET' is identical to ++COMPONENTDATASET++.

This command is active for the local ISPF session for which it was issued. Note, that if you have split screen sessions, the other sessions are not able to use this. If you use ISPF option 6 to issue the command, use ISPF option 3.4 to edit the data set; this enables you to use the edit command.

You can then edit each JCL file and run the renamed BIPEDIT exec by typing its name on the command line (for example MQ01EDBK). You might want to use View instead of Edit on a member until any problems in your REXX program are resolved. Alternatively, you can Cancel the Edit session instead of using Save.

You **must** set a value for all the variables listed in the JCL; if you do not do so, the JCL will not work correctly.

Some JCL files include ++OPTIONS++ for a command, these **must** be replaced with additional optional parameters specific to the command on z/OS, or removed. It is likely that you will have to do this in addition to running BIPEDIT. If you do not require any additional options, remove ++OPTIONS++ using the following command:

    "c ++OPTIONS++ '' all"

where ' ' represents two single quotation marks.

Save the edit macro and run this macro against all the members except the edit macro itself.

If the user ID submitting the BIPCBRK command has the appropriate DB2 and WebSphere MQ authorities, you can ignore the optional **mqsicreatebroker** parameters -1, -2, and -3.

If it is your intention to have different administrators create the DB2 and WebSphere MQ resources, you can consider using one of these optional parameters; see the "mqsicreatebroker command" on page 257 for further information.

When you update BIPBPROF (the broker profile), the changes are not accessible until you run BIPGEN to copy the profile to the HFS and create the ENVFILE. You must do this each time you update BIPBPROF for the changes to take effect.

Note: You need to be aware that another process might be using the current ENVFILE, so you need to consider whether updating the current ENVFILE in the HFS will have any impact.

Creating the environment file:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the broker JCL" on page 98

1. Review the BIPBPROF member. If you define parameters for all users, you can configure BIPPROF to use these parameters, rather than specify them.

   For example, if the time zone option TZ is set as a system wide parameter for all users, you can remove it from BIPBPROF.

2. Submit member BIPGEN. Review the job output and make sure the environment file in the output contains the parameters you expect.

   If you change BIPBPROF, or system-wide parameters, you must submit BIPGEN again to pick up the changes.

Prime DB2:

This is part of the larger task of creating a broker on z/OS. The broker uses DB2 tables to hold its internal data. These tables are defined in table spaces, which in turn, are defined within a DB2 database.

Data from tables is accessed through in-memory buffer pools, and typically you have different buffer pools for:
- Different applications
- Indexes
- Tables
- LOB tables

The DASD volumes that can be used by a database are defined using a storage group (STOGROUP) and your DB2 systems administrator will tell you which buffer pools and storage groups to use.

The BIPCRDB job issues commands that require certain authorities, which are:
- CREATESG - to create a storage group
- CREATEDBA - to create the database

Note that you might also need authority to grant use of a buffer pool.

If you have any problems with this job, you can edit and customize member BIPDLDB to delete the database. You can run the BIPCRDB again when you have resolved the problems.

Before you start

Before starting this step, you must have completed "Creating the environment file" on page 100

1. Edit the BIPCRDB job.
2. Review and change the BUFFERPOOL and INDEXP buffer pools on the Create Database statement
3. Change the buffer pool specification on the CREATE LOB TABLESPACE statements to a suitable value.
4. Submit BIPCRDB from your broker PDSE. You need the authority described earlier to submit the DB2 job.

   BIPCRDB creates the DB2 StorageGroup, Database, and Table Spaces but does not create any tables or indexes.

   The steps in the BIPCRDB job must complete with return code zero.

Creating the broker component:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Prime DB2" on page 100

If the user ID submitting the BIPCBRK command has the appropriate DB2 and WebSphere MQ authorities, you can ignore the optional **mqsicreatebroker** parameters -1, -2, and -3.

If it is your intention to have different administrators create the DB2 and WebSphere MQ resources, you can consider using one of these optional parameters; see the "mqsicreatebroker command" on page 257 for further information.

1. Submit job BIPCRBK with option -1. This job creates the files and directories which are placed in the default storage group. You must run this job first, and to do this you need authority to access the broker root directory.
2. Edit BIPCRBK and submit the job with option -2. This job creates the WebSphere MQ queues. If you do not have the requisite authority, ask your WebSphere MQ system administrator to run the job.
3. Edit BIPCRBK and submit the job with option -3. This job creates the DB2 queues. If you do not have the requisite authority, ask your DB2 system administrator to run the job.

If you encounter any problems, edit and configure job BIPDLBK. Run job BIPDLBK with the same option, or options, that caused the problems when you ran the BIPCRBK job. Note that you need the appropriate authority to run the jobs.

Correct the problems and run the BIPCRBK job again.

Copy started task to the procedures library:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Creating the broker component" on page 101

Ensure that the broker started task user ID is defined and that the broker procedure is associated with the user ID.

If you are using a security manager, for example RACF, update the STARTED class for your broker. See "Setting up z/OS security" on page 29 and "Summary of required access (z/OS)" on page 337 for more information.

Copy the Started Task JCL to the procedures library, for example, USER.PROCLIB.

Starting the component:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this step, you must have completed "Copy started task to the procedures library" on page 101.

From SDSF, start the component, for example, /S MQP1BRK This command produces, for example, the following output:

```
+BIP914W MQP1BRK 0 The component was started
+BIP2001I MQP1BRK 0 The WebSphere Event Broker service has started, process ID 50528622.
```

Adding an execution group to a broker

Before you start:

You must complete the following tasks:
- "Adding a broker to a broker domain" on page 150

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly.

For more details about why you might want to create multiple execution groups, see Execution groups.

Instead of employing the Message Brokers Toolkit, you can use this task as an alternative method of creating an execution group .

For more details on creating an execution group from the Message Brokers Toolkit see "Adding an execution group to a broker" on page 155

To add an execution group to a broker using the command line on Windows, UNIX systems, and Linux:
1. Open a command prompt that has the environment configured for WebSphere Event Broker Version 6.0.
2. Enter the following command to add the execution group:
   ```
   mqsicreateexecutiongroup -i host -p 1414 -q QMGR -b BROKER —e EG1
   ```

where

host   Is the host name or IP address of the Configuration Manager for the
        domain on which the broker resides.

1414   Is the port on which the Configuration Manager's queue manager is
        listening.

QMGR
        Is the name of the Configuration Manager's queue manager.

BROKER
        Is the name of the broker.

EG1   Is the name of the execution group that you want to create.

On completion of this task, you have requested that the Configuration Manager
creates an execution group on the broker when it is next deployed.

Adding an execution group to a broker on z/OS

Before you start:

You must complete the following tasks:
- "Adding a broker to a broker domain" on page 150

When you create a broker, it has a default execution group. If you want additional
execution groups, you must create them explicitly.

For more details about why you might want to create multiple execution groups,
see Execution groups.

Instead of employing the Message Brokers Toolkit, you can use this task as an
alternative method of creating an execution group .

For more details on creating an execution group from the Message Brokers Toolkit
see "Adding an execution group to a broker" on page 155

To add an execution group to a broker on z/OS:

Configure and run the `BIPCREG` job to create an execution group. Note that you
need to create the broker in the Configuration Manager before you run the `BIPCREG`
job.

On completion of this task, you have requested that the Configuration Manager
creates an execution group on the broker when it is next deployed.

Creating a Configuration Manager

Before you start:

You must complete the following tasks:
- Ensure that your user ID has the correct authorizations to perform the task.
  Refer to "Security requirements for administrative tasks" on page 385.

You create a Configuration Manager using the command line on the system where
the Configuration Manager component is installed.

To create a Configuration Manager, follow the link for the appropriate platform.
- "Creating a Configuration Manager on AIX"
- "Creating a Configuration Manager on HP-UX" on page 105
- "Creating a Configuration Manager on Linux" on page 105
- "Creating a Configuration Manager on Solaris" on page 106
- "Creating a Configuration Manager on Windows" on page 107
- "Creating a Configuration Manager on z/OS" on page 108

Creating a Configuration Manager on AIX

The following steps show you how to create a Configuration Manager.

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
2. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
3. Enter the following command to create the Configuration Manager:

   mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   CMGR01
   > Is the name of the Configuration Manager that you are creating.

   wbrkuid
   > Is the service user ID that is used to run the Configuration Manager.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_QM
   > Is the name of the WebSphere MQ queue manager that will host the Configuration Manager. This is created if it does not exist.

On completion of this task, you have:
- Created a Configuration Manager.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the Configuration Manager, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateconfigmgr** command.
- Set up the authorizations that the Configuration Manager requires to access WebSphere MQ resources.

Now that you have created the Configuration Manager, you are ready to:

1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 147.

Creating a Configuration Manager on HP-UX

The following steps show you how to create a Configuration Manager.

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
2. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
3. Enter the following command to create the Configuration Manager:

   mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   CMGR01
   > Is the name of the Configuration Manager that you are creating.

   wbrkuid
   > Is the service user ID that is used to run the Configuration Manager.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_QM
   > Is the name of the WebSphere MQ queue manager that will host the Configuration Manager. This is created if it does not exist.

On completion of this task, you have:

- Created a Configuration Manager.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the Configuration Manager, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateconfigmgr** command.
- Set up the authorizations that the Configuration Manager requires to access WebSphere MQ resources.

Now that you have created the Configuration Manager, you are ready to:

1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 147.

Creating a Configuration Manager on Linux

Follow the steps detailed in this task for creating a Configuration Manager on either Linux (zSeries platform) or Linux (x86 platform).

To create a Configuration Manager.

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

2. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
3. Enter the following command to create the Configuration Manager:

   mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   CMGR01
   > Is the name of the Configuration Manager that you are creating.

   wbrkuid
   > Is the service user ID that is used to run the Configuration Manager.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_QM
   > Is the name of the WebSphere MQ queue manager that will host the Configuration Manager. This is created if it does not exist.

On completion of this task, you have:

- Created a Configuration Manager.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the Configuration Manager, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateconfigmgr** command.
- Set up the authorizations that the Configuration Manager requires to access WebSphere MQ resources.

Now that you have created the Configuration Manager, you are ready to:

1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 147.

Creating a Configuration Manager on Solaris

The following steps show you how to create a Configuration Manager.

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
2. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
3. Enter the following command to create the Configuration Manager:

   mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

CMGR01

Is the name of the Configuration Manager that you are creating.

wbrkuid

Is the service user ID that is used to run the Configuration Manager.

wbrkpw

Is the password for the service user ID.

WBRK_QM

Is the name of the WebSphere MQ queue manager that will host the Configuration Manager. This is created if it does not exist.

On completion of this task, you have:

- Created a Configuration Manager.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the Configuration Manager, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateconfigmgr** command.
- Set up the authorizations that the Configuration Manager requires to access WebSphere MQ resources.

Now that you have created the Configuration Manager, you are ready to:

1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 147.

Creating a Configuration Manager on Windows

You create a Configuration Manager using the command line. Create the Configuration Manager on the system where the Configuration Manager component is installed.

Use the **mqsicreateconfigmgr** command. The parameters on this command provide the Configuration Manager with all the additional information it requires to be ready for action as soon as it is started.

To create a Configuration Manager:

1. Open a WebSphere Event Broker command prompt for the desired runtime.
2. Enter the following command to create the Configuration Manager:

   ```
   mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_QM
   ```

   If you are using different names or values for any parameter on this command, you **must** replace the appropriate values with your own.

   In the command above:

   CMGR01

   Is the name of the Configuration Manager that you are creating. This is an optional parameter; if you do not specify it, the default is 'ConfigMgr'.

wbrkuid
>>is the service user ID used to run the Configuration Manager.

>>This ID must be a member of the `mqm`, `mqbrkrs` and `Administrators` groups.

wbrkpw
>>Is the password for the service user ID.

WBRK_QM
>>Is the name of the WebSphere MQ queue manager that will host the Configuration Manager. This is created if it does not exist.

On completion of this task, you have:
- Created a Configuration Manager and added its Windows service to the Services (viewable from the Control Panel).
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the Configuration Manager, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateconfigmgr** command.
- Set up the authorizations that the Configuration Manager requires to access WebSphere MQ resources.
- Defined a service user ID wbrkuid, and database password wbrkpw.

Now that you have created the Configuration Manager, you are ready to:
1. Create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 147.

Creating a Configuration Manager on z/OS

To create your Configuration Manager, perform the following steps in order.
1. "Installation information - Configuration Manager" on page 109
2. "Component information - Configuration Manager" on page 109
3. "Creating the Configuration Manager PDSE" on page 110
4. "Creating the Configuration Manager directory on z/OS" on page 110
5. "Customizing the Configuration Manager component data set" on page 110
6. "Customizing the Configuration Manager JCL" on page 111
7. "Creating the Configuration Manager component" on page 113
8. "Copy the started task to the procedures library" on page 113
9. "Starting the Configuration Manager component" on page 113

Information required to create a Configuration Manager on z/OS:

This is part of the larger task of creating a Configuration Manager on z/OS.

You need to complete the information in each of the following tables before continuing:
- "Installation information - Configuration Manager" on page 109

- "Component information - Configuration Manager"

Installation information - Configuration Manager:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
Fully qualified name of the product's SBIPPROC dataset	N/A.	<hlq>.SBIPPROC	
Fully qualified name of the product's SBIPSAMP dataset	N/A	<hlq>.SBIPSAMP	
HFS directory where the product has been installed.	++INSTALL++	/usr/lpp/mqsi	
LE high-level-qualifier	++LEHLQ++	PP.ADLE370.ZOS140	
WebSphere MQ high-level-qualifier.	++WMQHLQ++	MQM.V530	
Location of Java installation.	++JAVA++	/usr/lpp/java/IBM/J1.4	
Locale of environment where commands are run by submitting JCL.	++LOCALE++	C	
Time zone of environment where commands are run by submitting JCL.	++TIMEZONE++	GMT0BST	
WebSphere MQ HFS install directory.	++MQPATH++	/usr/lpp/mqm	

Component information - Configuration Manager:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
The Configuration Manager name	++COMPONENTNAME++	MQP1CMGR	
The Configuration Manager's user ID HFS home directory	++HOME++	/u/mqp1cmgr	
HFS directory where the Configuration Manager is to exist.	++COMPONENTDIRECTORY++	/mqsi/V6R0M0/cmgr/MQP1BRK	
The dataset where all JCL relevant to the Configuration Manager is saved.	++COMPONENTDATASET++	TESTDEV.A.V6.MQP1CMGR	
Profile name	++COMPONENTPROFILE++	BIPBPROF, BIPUPROF, or BIPCPROF	**BIPCPROF**

Name of the Started Task JCL; can be a maximum of 8 characters.	++STARTEDTASKNAME++	MQP1CMGR	
mqsicreateconfigmgr options	++OPTIONS++	Not needed, remove from the command in JCL	

Creating the Configuration Manager PDSE:

This is part of the larger task of creating a Configuration Manager on z/OS.

Create the Configuration Manager PDSE, for example using option 3.2 on ISPF. The name of the PDSE must be the same as the JCL variable ++COMPONENTDATASET++. Allocate a data set with:
- Eight directory blocks
- 15 tracks (or 1 cylinder) of 3390 DASD with a record format of fixed blocked 80
- A suitable block size (for example 27920)
- Data set type library

Creating the Configuration Manager directory on z/OS:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

To complete this task, you must have completed the following task:
- "Information required to create a Configuration Manager on z/OS" on page 108
- "Creating the Configuration Manager PDSE"

Create the Configuration Manager directory manually using:
```
mkdir <ComponentDirectory>
```

The name of the directory must be the same as the JCL variable ++COMPONENTDIRECTORY++.

Use the chmod command to set the required authorizations. See "Creating the broker directory on z/OS" on page 96 for more information.

Customizing the Configuration Manager component data set:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

Before starting this step, you must have completed "Information required to create a Configuration Manager on z/OS" on page 108.
1. Create the Configuration Manager data set in TSO identified by ++COMPONENTDATASET++. Copy all the following JCL to the Configuration Manager data set you have created so that you can customize the source.

   Note: This then forms the source for the Configuration Manager.
2. Copy the following JCL from <hlq>.SBIPSAMP to ++COMPONENTDATASET++.
   - BIPCPROF (Configuration Manager profile).

"Sample BIPCPROF file" on page 354 lists the shipped `BIPCPROF` file.

3. Copy the following JCL from `<hlq>.SBIPPROC` to `++COMPONENTDATASET++`.
    - `BIPALMQ` - job to alter WebSphere MQ table spaces and tables.
    - `BIPBUCM` - job to issue the mqsibackupconfigmgr command.
    - `BIPCHCM` - job to issue the mqsichangeconfigmgr command.
    - `BIPCRACL` - job to create an Access Control List (ACL) entry by issuing the mqsicreateaclentry command.
    - `BIPCRCM` - job to create a Configuration Manager.

      "Sample BIPCRCM file" on page 357 lists the shipped `BIPCRCM` file.
    - `BIPCREG` - job to issue the mqsicreateexecutiongroup command.
    - `BIPDLACL` - job to delete an Access Control List (ACL) entry by issuing the mqsideleteaclentry command.
    - `BIPDLCM` - job to issue the mqsideleteconfigmgr command
    - `BIPDLEG` - job to issue the mqsideleteexecutiongroup command.
    - `BIPDPLY` - job to issue the mqsideploy command.
    - `BIPEDIT` - edit macro for customization. Rename `BIPEDIT` to a unique name that identifies it to the current component; for example, `MQ01EDCM`.
    - `BIPFMLG` - job to format the log.
    - `BIPGEN` - job to generate the `ENVFILE`.

      "Sample BIPGEN file" on page 370 lists the shipped `BIPGEN` file.
    - `BIPLIACL` - job to list the Access Control List entries by issuing the mqsilistaclentry command.
    - `BIPLIST` - job to issue the mqsilist command.
    - `BIPRELG` - job to read the logs.
    - `BIPRSCM` - job to issue the mqsirestoreconfigmgr command.
    - `BIPSPMF` - job to issue the mqsistopmsgflow command.
    - `BIPSTMF` - job to issue the mqsistartmsgflow command.
    - `BIPCMGRP` - started task). Rename `BIPCMGRP` to **the same as the ++STARTEDTASKNAME++**.

      "Sample BIPCMGRP file" on page 378 lists the shipped `BIPCMGRP` file.

Customizing the Configuration Manager JCL:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the Configuration Manager component data set" on page 110.

All JCL has a standard header, comprising:
1. A brief description of its function.
2. A description where further information can be found, relating to the function of the JCL.
3. If appropriate, a topic number.
4. The section listing the JCL variables themselves.
5. Each JCL file defines its own `STEPLIB`

6. Some JCL files, for example, `BIPRELG` might require DB2 defined in the `STEPLIB` for a broker component. This must be removed from the JCL if the component is either a Configuration Manager or User Name Server, as it is not required.

You can customize the files using an ISPF edit macro that you have to tailor, or you can make changes to each of the PDSE members manually.

`BIPEDIT` is a REXX program that can be used to assist you in customizing your JCL. Once you have customized `BIPEDIT` you can run this REXX program against the other JCL files to change their JCL variables.

First of all customize the renamed `BIPEDIT` file using the information you collected in:
- "Installation information - Configuration Manager" on page 109
- "Component information - Configuration Manager" on page 109

Next, you must activate the renamed `BIPEDIT` file before you customize any other JCL files. Do this by running the following TSO command:

```
ALTLIB ACTIVATE APPLICATION(EXEC) DA('COMPONENTDATASET')
```

where `'COMPONENTDATASET'` is identical to `++COMPONENTDATASET++`.

This command is active for the local ISPF session for which it was issued. Note, that if you have split screen sessions, the other sessions are not able to use this. If you use ISPF option 6 to issue the command, use ISPF option 3.4 to edit the data set; this enables you to use the edit command.

You can then edit each JCL file and run the renamed `BIPEDIT` exec by typing its name on the command line (for example `MQ01EDCM`). You might want to use `View` instead of `Edit` on a member until any problems in your REXX program are resolved. Alternatively, you can `Cancel` the `Edit` session instead of using `Save`.

You **must** set a value for all the variables listed in the JCL; if you do not do so, the JCL will not work correctly.

Some JCL files include `++OPTIONS++` for a command, these **must** be replaced with additional optional parameters specific to the command on z/OS, or removed. It is likely that you will have to do this in addition to running `BIPEDIT`. If you do not require any additional options, remove `++OPTIONS++` using the following command:

```
"c ++OPTIONS++ '' all"
```

where ' ' represents two single quotation marks.

Save the edit macro and run this macro against all the members except the edit macro itself.

Do not set either of the optional pass parameters (-1 or -2) in `BIPCRCM` at this time because you want to create the registry and the WebSphere MQ queues.

When you update `BIPCPROF` (the Configuration Manager profile), the changes are not accessible until you run `BIPGEN` to copy the profile to the HFS and create the `ENVFILE`. You must do this each time you update `BIPCPROF` for the changes to take effect, which happens when you restart the Configuration Manager.

Note: You need to be aware that another process might be using the current `ENVFILE`, so you need to consider whether updating the current `ENVFILE` in the file system will have any impact.

Creating the Configuration Manager component:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the Configuration Manager JCL" on page 111

If the user ID submitting the `BIPCRCM` command has the appropriate WebSphere MQ authorities, you can ignore the optional **mqsicreateconfigmgr** parameters -1 and -2.

If it is your intention to have a different administrator create the WebSphere MQ resources, you can consider using one of these optional parameters; see the "mqsicreateconfigmgr command" on page 266 for further information.

1. Submit job `BIPCRCM` with option -1. This job creates the Configuration Manager together with the files and directories which are placed in the registry. You must run this job first, and to do this you need authority to access the Configuration Manager.

2. Edit `BIPCRCM` and submit the job with option -2. This job creates the WebSphere MQ queues. If you do not have the requisite authority, ask your WebSphere MQ system administrator to run the job.

If you encounter any problems, edit and configure job `BIPDLCM`. Run job `BIPDLCM` with the same option, or options, that caused the problems when you ran the `BIPCRCM` job. Note that you need the appropriate authority to run the jobs.

Correct the problems and run the `BIPCRCM` job again.

Copy the started task to the procedures library:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

Before starting this step, you must have completed "Creating the Configuration Manager component"

Copy the Started Task JCL to the procedures library, for example, `USER.PROCLIB`.

Starting the Configuration Manager component:

This is part of the larger task of creating a Configuration Manager on z/OS.

Before you start

Before starting this step, you must have completed "Copy the started task to the procedures library."

From SDSF, start the component, for example, `/S MQ01CMGR`

Enabling a User Name Server

If you require a User Name Server as a component of your broker domain, you must create the necessary connections between the broker, Configuration Manager, and User Name Server, so that they can communicate effectively.

Specify additional parameters on the **mqsicreatebroker** and **mqsicreateconfigmgr** commands, before you create the User Name Server. The following steps show you how to do this.

1. Create a broker with the additional -s and -j parameters on the **mqsicreatebroker** command. These parameters allow the broker to communicate with the WebSphere MQ queue manager for the User Name Server, and also enable the broker for publish/subscribe access control.

   If you have created the broker without these parameters, modify the broker, defining the -s and -j parameters. Refer to "Modifying a broker" on page 131.

2. Create a Configuration Manager with the additional -s parameter on the **mqsicreateconfigmgr** command. This parameter allows the Configuration Manager to communicate with the WebSphere MQ queue manager for the User Name Server.

   If you have created the Configuration Manager without this parameter, modify the Configuration Manager, defining the -s parameter. Refer to "Modifying a Configuration Manager" on page 134.

Now that you have made the required changes to the broker and Configuration Manager, you can create the User Name Server, and thus enable publish/subscribe services.

Creating a User Name Server

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385.
- Create a broker and Configuration Manager, with the additional parameters on the **mqsicreatebroker** and **mqsicreateconfigmgr** commands to allow them to communicate with the User Name Server. Refer to "Enabling a User Name Server."

You create a User Name Server using the command line on the system where the User Name Server component is installed.

To create a User Name Server, follow the link for the appropriate platform.
- "Creating a User Name Server on AIX"
- "Creating a User Name Server on HP-UX" on page 115
- "Creating a User Name Server on Linux" on page 116
- "Creating a User Name Server on Solaris" on page 117
- "Creating a User Name Server on Windows" on page 117
- "Creating a User Name Server on z/OS" on page 118

Creating a User Name Server on AIX

The following steps show you how to create a User Name Server.

1. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
2. Enter the following command to create the User Name Server:

   mqsicreateusernameserver -i wbrkuid -a wbrkpw -q WBRK_UNS_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   wbrkuid

   > Is the service user ID that is used to run the User Name Server.

   wbrkpw

   > Is the password for the service user ID.

   WBRK_UNS_QM

   > Is the name of the WebSphere MQ queue manager for the User Name Server. This is created if it does not exist.

On completion of this task, you have:

- Created a User Name Server.
- Created a default startup status of manual for the User Name Server.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the User Name Server, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateusernameserver** command.

Now that you have created the User Name Server, you are ready to create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

Creating a User Name Server on HP-UX

The following steps show you how to create a User Name Server.

1. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
2. Enter the following command to create the User Name Server:

   mqsicreateusernameserver -i wbrkuid -a wbrkpw -q WBRK_UNS_QM

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   wbrkuid

   > Is the service user ID that is used to run the User Name Server.

   wbrkpw

   > Is the password for the service user ID.

   WBRK_UNS_QM

   > Is the name of the WebSphere MQ queue manager for the User Name Server. This is created if it does not exist.

On completion of this task, you have:

- Created a User Name Server.
- Created a default startup status of manual for the User Name Server.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the User Name Server, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateusernameserver** command.

Now that you have created the User Name Server, you are ready to create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

Creating a User Name Server on Linux

Follow the steps detailed in this task for creating a User Name Server on either Linux (zSeries platform) or Linux (x86 platform).

1. Log on using your user ID. If you use the su command to switch user from root, enter `su - <user ID>` to run your user profile.
2. Enter the following command to create the User Name Server:

   `mqsicreateusernameserver -i wbrkuid -a wbrkpw -q WBRK_UNS_QM`

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   wbrkuid
   > Is the service user ID that is used to run the User Name Server.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_UNS_QM
   > Is the name of the WebSphere MQ queue manager for the User Name Server. This is created if it does not exist.

On completion of this task, you have:
- Created a User Name Server.
- Created a default startup status of manual for the User Name Server.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the User Name Server, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateusernameserver** command.

Now that you have created the User Name Server, you are ready to create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

Creating a User Name Server on Solaris

The following steps show you how to create a User Name Server.

1. Log on using your user ID. If you use the su command to switch user from root, enter su - <user ID> to run your user profile.
2. Enter the following command to create the User Name Server:

   `mqsicreateusernameserver -i wbrkuid -a wbrkpw -q WBRK_UNS_QM`

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   wbrkuid
   > Is the service user ID that is used to run the User Name Server.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_UNS_QM
   > Is the name of the WebSphere MQ queue manager for the User Name Server. This is created if it does not exist.

On completion of this task, you have:

- Created a User Name Server.
- Created a default startup status of manual for the User Name Server.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the User Name Server, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateusernameserver** command.

Now that you have created the User Name Server, you are ready to create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

Creating a User Name Server on Windows

The following steps show you how to create a User Name Server.

1. Open a command prompt.
2. Enter the following command to create the User Name Server:

   `mqsicreateusernameserver -i wbrkuid -a wbrkpw -q WBRK_UNS_QM`

   If you are using different names or values for any parameter on this command, you must replace the appropriate values with your own.

   In the command above:

   wbrkuid
   > Is the service user ID that is used to run the User Name Server.

   wbrkpw
   > Is the password for the service user ID.

   WBRK_UNS_QM
   > Is the name of the WebSphere MQ queue manager for the User Name Server. This is created if it does not exist.

On completion of this task, you have:

- Created a User Name Server.
- Created a default startup status of manual for the User Name Server.
- Created and started a WebSphere MQ queue manager called WBRK_QM.
- Created and set up the WebSphere MQ resources required by the User Name Server, and defined these on the queue manager. This includes the default dead-letter queue (DLQ), which is automatically enabled by running the **mqsicreateusernameserver** command.

Now that you have created the User Name Server, you are ready to create and start the WebSphere MQ queue manager channels that are required to connect WebSphere Event Broker components (brokers, User Name Servers, and Configuration Manager). This allows components in your broker domain that are supported by different queue managers to exchange messages and communicate effectively. Refer to "Connecting components" on page 130.

Creating a User Name Server on z/OS

To create your User Name Server, perform the following steps in order.

1. "Information required to create a User Name Server on z/OS"
2. "Creating the User Name Server PDSE" on page 119
3. "Creating the User Name Server directory on z/OS" on page 119
4. "Creating the User Name Server runtime environment on z/OS" on page 120
5. "Customizing the User Name Server component data set" on page 121
6. "Customizing the User Name Server JCL" on page 121
7. "Creating the User Name Server component" on page 123
8. "Copy the started task to the procedures library" on page 123
9. "Starting the component" on page 123

Information required to create a User Name Server on z/OS:

This is part of the larger task of creating a User Name Server on z/OS.

You need to complete the information in each of the following tables before continuing:

- "Installation information - Broker and User Name Server" on page 94
- "Component information - User Name Server" on page 119

Installation information - Broker and User Name Server:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
Fully qualified name of the product's SBIPPROC dataset	N/A.	<hlq>.SBIPPROC	
Fully qualified name of the product's SBIPSAMP dataset	N/A	<hlq>.SBIPSAMP	
File system directory where the product has been installed.	++INSTALL++	/usr/lpp/mqsi	

LE high-level-qualifier	++LEHLQ++	PP.ADLE370.ZOS140	
WebSphere MQ high-level-qualifier.	++WMQHLQ++	MQM.V530	
Location of Java installation.	++JAVA++	/usr/lpp/java/IBM/J1.4	
Locale of environment where commands are run by submitting JCL.	++LOCALE++	C	
Time zone of environment where commands are run by submitting JCL.	++TIMEZONE++	GMT0BST	

Component information - User Name Server:

Collect the information explained in the Description column and complete the values you require for your particular system. A complete list of variables you can customize is given in "z/OS JCL variables" on page 345.

Description	JCL variable	Typical installation variable	Your installation variable
The User Name Server's user ID HFS home directory	++HOME++	/u/mq06uns	
HFS directory where the User Name Server is to exist.	++COMPONENTDIRECTORY++	/mqsi/V6R0M0/uns/MQ01BRK	
The dataset where all JCL relevant to the User Name Server is saved.	++COMPONENTDATASET++	TESTDEV.A.V6.MQ06UNS	
Profile name	++COMPONENTPROFILE++	BIPBPROF, BIPUPROF, or BIPCPROF	**BIPUPROF**
Name of the Started Task JCL; can be a maximum of 8 characters.	++STARTEDTASKNAME++	MQ06UNS	
mqsicreateusernameserver options	++OPTIONS++	Not needed, remove from the command in JCL	

Creating the User Name Server PDSE:

This is part of the larger task of creating a User Name Server on z/OS.

Create the User Name Server PDSE, for example using option 3.2 on ISPF. The name of the PDSE must be the same as the JCL variable ++COMPONENTDATASET++. Allocate a data set with:
• Eight directory blocks
• 15 tracks (or 1 cylinder) of 3390 DASD with a record format of fixed blocked 80
• A suitable block size (for example 27920)
• Data set type library

Creating the User Name Server directory on z/OS:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

To complete this task, you must have completed the following task:
- "Information required to create a User Name Server on z/OS" on page 118
- "Creating the User Name Server PDSE" on page 119

Create the User Name Server directory manually using:

```
 mkdir <ComponentDirectory>
```

The name of the directory must be the same as the JCL variable
++COMPONENTDIRECTORY++.

Use the chmod command to set the required authorizations. See "Creating the broker directory on z/OS" on page 96 for more information.

Creating the User Name Server runtime environment on z/OS:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

To complete this task, you must have completed the following task:
- "Creating the User Name Server directory on z/OS" on page 119.

Invoke the **mqsicreateusernameserver** command to create a User Name Server and its runtime environment. The command syntax is:

```
mqsicreateusernameserver -q QueueManagerName  [-r RefreshInterval] [-1] [-2]
```

You will be asked to confirm that the parameters you enter are correct. Enter Y to confirm, or N to change the parameters. If you choose to change the parameters, invoke **mqsicreateusernameserver** again.

Check that you are using the correct installation path, particularly if you are customizing the system after applying maintenance to an alternate set of WebSphere MQ Event Broker libraries. If you are applying service to the broker, you might want a different installation path.

The functionality of this command is not the same as on distributed platforms, because no WebSphere MQ-related definitions are performed. You have to follow further steps to complete the User Name Server creation and customization.

Required parameters:

-q *QueueManagerName*
    The name of the WebSphere MQ queue manager associated with your User Name Server, for example MQP1.

Optional parameters:

-r *RefreshInterval*
    The interval, specified in seconds, at which the User Name Server interrogates the security subsystem for changes to user or group attributes. If an interval is not specified, the User Name Server default interval of 60 seconds is used.

-1 The registry pass; creates only the User Name Server registry.

-2 The WebSphere MQ pass; creates only the User Name Server WebSphere MQ queues.

Customizing the User Name Server component data set:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

Before starting this step, you must have completed "Information required to create a User Name Server on z/OS" on page 118.

1. Create the User Name Server data set in TSO identified by `++COMPONENTDATASET++`. Copy all the following JCL to the User Name Server data set you have created so that you can customize the source.

   Note: This then forms the source for the User Name Server.

2. Copy the following JCL from `<hlq>.SBIPSAMP` to `++COMPONENTDATASET++`.
   - `BIPUPROF` - User Name Server profile.
     "Sample BIPUPROF file" on page 372 lists the shipped `BIPUPROF` file.

3. Copy the following JCL from `<hlq>.SBIPPROC` to `++COMPONENTDATASET++`.
   - `BIPALMQ` - job to alter WebSphere MQ table spaces and tables.
   - `BIPCHUN` - job to issue the mqsichangeusernameserver command.
   - `BIPCRUN` - create User Name Server.
     "Sample BIPCRUN file" on page 368 lists the shipped `BIPCRUN` file.
   - `BIPDLUN` - job to issue the mqsideleteusernameserver command.
   - `BIPEDIT` - JCL customization. Rename `BIPEDIT` to a unique name that identifies it to the current component; for example, `MQ01EDUN`.
   - `BIPFMLG` - job to format the log.
   - `BIPGEN` - generate `ENVFILE`.
     "Sample BIPGEN file" on page 370 lists the shipped `BIPGEN` file.
   - `BIPLIST` - job to issue the mqsilist command.
   - `BIPRELG` - job to read the logs.
   - `BIPUNSP` - started task. Rename `BIPUNSP` to **the same as the ++STARTEDTASKNAME++**.
     "Sample BIPUNSP file" on page 380 lists the shipped `BIPUNSP` file.

Customizing the User Name Server JCL:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the User Name Server component data set."

All JCL has a standard header, comprising:

1. A brief description of its function.
2. A description where further information can be found, relating to the function of the JCL.
3. If appropriate, a topic number.
4. The section listing the JCL variables themselves.

5. Each JCL file defines its own `STEPLIB`
6. Some JCL files, for example, `BIPRELG` might require DB2 defined in the `STEPLIB` for a broker component. This must be removed from the JCL if the component is either a Configuration Manager or User Name Server, as it is not required.

You can customize the files using an ISPF edit macro that you have to tailor, or you can make changes to each of the PDSE members manually.

`BIPEDIT` is a REXX program that can be used to assist you in customizing your JCL. Once you have customized `BIPEDIT` you can run this REXX program against the other JCL files to change their JCL variables.

First of all customize the renamed `BIPEDIT` file using the information you collected in:
- "Installation information - Broker and User Name Server" on page 94
- "Component information - User Name Server" on page 119

Next, you must activate the renamed `BIPEDIT` file before you customize any other JCL files. Do this by running the following TSO command:

```
ALTLIB ACTIVATE APPLICATION(EXEC) DA('COMPONENTDATASET')
```

where `'COMPONENTDATASET'` is identical to `++COMPONENTDATASET++`.

This command is active for the local ISPF session for which it was issued. Note, that if you have split screen sessions, the other sessions are not able to use this. If you use ISPF option 6 to issue the command, use ISPF option 3.4 to edit the data set; this enables you to use the edit command.

You can then edit each JCL file and run the renamed `BIPEDIT` exec by typing its name on the command line (for example MQ01EDUN). You might want to use `View` instead of `Edit` on a member until any problems in your REXX program are resolved. Alternatively, you can `Cancel` the `Edit` session instead of using `Save`.

You **must** set a value for all the variables listed in the JCL; if you do not do so, the JCL will not work correctly.

Some JCL files include `++OPTIONS++` for a command, these **must** be replaced with additional optional parameters specific to the command on z/OS, or removed. It is likely that you will have to do this in addition to running `BIPEDIT`. If you do not require any additional options, remove `++OPTIONS++` using the following command:

```
"c ++OPTIONS++ '' all"
```

where ' ' represents two single quotation marks.

Save the edit macro and run this macro against all the members except the edit macro itself.

Do not set either of the optional pass parameters (-1 or -2) in `BIPCRUN` at this time because you want to create the registry and the WebSphere MQ queues.

When you update `BIPUPROF` (the User Name Server profile), the changes are not accessible until you run `BIPGEN` to copy the profile to the HFS and create the `ENVFILE`. You must do this each time you update `BIPUPROF` for the changes to take effect.

Note: You need to be aware that another process might be using the current
ENVFILE, so you need to consider whether updating the current ENVFILE in
the HFS will have any impact.

Creating the User Name Server component:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

Before starting this step, you must have completed "Customizing the User Name
Server JCL" on page 121

If the user ID submitting the BIPCRUN command has the appropriate WebSphere
MQ authorities, you can ignore the optional **mqsicreateusernameserver** parameters
-1 and -2.

If it is your intention to have a different administrator create the WebSphere MQ
resources, you can consider using one of these optional parameters; see the
"mqsicreateusernameserver command" on page 275 for further information.

1. Submit job BIPCRUN with option -1. This job creates the User Name Server
   together with the files and directories which are placed in the registry. You
   must run this job first, and to do this you need authority to access the User
   Name Server.
2. Edit BIPCRUN and submit the job with option -2. This job creates the WebSphere
   MQ queues. If you do not have the requisite authority, ask your WebSphere
   MQ system administrator to run the job.

If you encounter any problems, edit and configure job BIPDLUN. Run job BIPDLUN
with the same option, or options, that caused the problems when you ran the
BIPCRUN job. Note that you need the appropriate authority to run the jobs.

Correct the problems and run the BIPCRUN job again.

Copy the started task to the procedures library:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

Before starting this step, you must have completed "Creating the User Name
Server component"

Copy the Started Task JCL to the procedures library, for example, USER.PROCLIB.

Starting the component:

This is part of the larger task of creating a User Name Server on z/OS.

Before you start

Before starting this step, you must have completed "Copy the started task to the
procedures library."

From SDSF, start the component, for example, /S MQ01UNS

Connecting the User Name Server to the WebSphere Event Broker network

This is part of the larger task of creating a User Name Server.

Before you start

On z/OS, to complete this task, you must have completed the following task:
- "Starting the component" on page 123.

To enable communication between all the components, define and start WebSphere MQ channels between the following components:
- Configuration Manager, queue manager and the User Name Server queue manager.
- Configuration Manager and the broker queue manager.
- User Name Server queue manager and the broker queue managers.
- All brokers used in publish/subscribe.

You connect a User Name Server to another component in the same way as you connect the Configuration Manager to another component. This task is described in detail in "Connecting components" on page 130.

The Configuration Manager requests user IDs and group information from the User Name Server. The WebSphere Event Broker administrator defines Access Control Lists (ACLs) on the workbench. These ACLs are sent to each broker using WebSphere MQ channels following a deploy.

For further details of connecting your User Name Server to a broker and enabling Publish/Subscribe, refer to "Configuring Publish/Subscribe security."

Configuring Publish/Subscribe security:

Refer to the following tasks:
- "Connecting the User Name Server to a broker"
- "Connecting the User Name Server to a broker on z/OS" on page 125
- "Starting the WebSphere MQ channels and listeners" on page 126
- "Enabling applications to use Publish/Subscribe" on page 127
- "Enabling applications to use Publish/Subscribe security on z/OS" on page 127

Connecting the User Name Server to a broker:

Before you start:

To complete this task, you must have completed the following task:
- "Creating a User Name Server" on page 114

You need to make the broker known to the User Name Server. You can do this using either of the following methods.
- Create a broker and specify s=UserNameServerQueueManagerName on the **mqsicreatebroker** command.
- Change an existing broker using the **mqsichangebroker** command.

If the User Name Server is not connected to the broker's queue manager, you need channels between the broker's queue manager and the User Name Server's queue manager. You need channels between the Configuration Manager queue manager and the broker's queue manager to receive message flows and Access Control Lists.

Change the Configuration Manager to use the queue manager name used by the User Name Server. You can use the **mqsicreateconfigmgr** or **mqsichangeconfigmgr**commands to set this value.

On the Topics panel in the workbench, you can view user information sent from the User Name Server.

Check that the User Name Server has registered the Configuration Manager. For more information on implementing topic-based security using the workbench, see "Enabling topic-based security" on page 25.

Example startup messages:

When a broker starts for the first time, and a User Name Server queue manager has been specified, and no response has ever been received from the User Name Server, you will receive the following message:

```
+BIP9141W  UserNameServer 0 The component was started.
```

When the User Name Server starts and indicates that it has registered with the broker, you will receive the following message:

```
18:17:18.54 BIP9141W: The component was started.
18:17:18.57 BIP2001I: The WebSphere Event Broker service has started, process  ID 196827.
18:17:24.31 BIP8201I: User Name Server starting with refresh interval 60.
18:17:28.21 BIP8204I: User Name Server is registering a client with UUID
             12345678-1234-1234-1234-123456789abc, and cache version 0.
```

Connecting the User Name Server to a broker on z/OS:

Before you start:

To complete this task, you must have completed the following task:
- "Creating a User Name Server on z/OS" on page 118

You need to make the broker known to the User Name Server. You can do this using either of the following methods.
- Create a broker and specify -s UserNameServerQueueManagerName on the **mqsicreatebroker** command.
- Change an existing broker using the **mqsichangebroker** command.

If the User Name Server is not connected to the broker's queue manager, you need channels between the broker's queue manager and the User Name Server's queue manager. You need channels between the Configuration Manager queue manager and the broker's queue manager to receive message flows and Access Control Lists.

Check the z/OS console for the message BIP8204, which is issued when the User Name Server has successfully registered a client.

Change the Configuration Manager to use the queue manager name used by the User Name Server on z/OS, or another supported platform. You can use the **mqsicreateconfigmgr** or **mqsichangeconfigmgr**commands to set this value.

On the Topics panel in the workbench, you can view user information sent from the User Name Server.

Check that the User Name Server has registered the Configuration Manager. Also, check the z/OS console for message BIP8204, which is issued when the User Name Server has successfully registered a client. For more information on implementing topic-based security using the workbench, see "Enabling topic-based security" on page 25.

Example startup messages:

When a broker starts for the first time, and a User Name Server queue manager has been specified, and no response has ever been received from the User Name Server, you will receive the following message:

```
+BIP9141W  UserNameServer 0 The component was started.
```

When the User Name Server starts and indicates that it has registered with the broker, you will receive the following message:

```
18:17:18.54 BIP9141W: The component was started.
18:17:18.57 BIP2001I: The WebSphere Event Broker service has started, process  ID 196827.
18:17:24.31 BIP8201I: User Name Server starting with refresh interval 60.
18:17:28.21 BIP8204I: User Name Server is registering a client with UUID
            12345678-1234-1234-1234-123456789abc, and cache version 0.
```

Starting the WebSphere MQ channels and listeners:

This topic tells you how to start the channels and listeners on Linux, UNIX systems and Windows platforms.

To complete the connection between two components that are supported by different queue managers, start the server channels that you created in "Connecting the User Name Server to a broker on z/OS" on page 125.

Before you can do that, you need to start the WebSphere MQ listeners that are to receive the messages sent out from these channels.

Note: All the examples use port 1414, the default WebSphere MQ port. You **must** ensure that you use the port space in the channel definition and that this port is not in use by another application.

Linux and UNIX systems:
1. To start a listener enter the following command on the command line:
   ```
   runmqlsr -t tcp -p 1414 -m WBRK_QM
   ```
2. To start a sender channel, enter the following command on the command line:
   ```
   runmqchl -c BROKER.CONFIG -m WBRK_QM
   ```

Windows platforms:
1. To start a listener as a background task (the preferred option) use WebSphere MQ Services:
   a. Click *Start -> Programs -> IBM WebSphere MQ -> WebSphere MQ Services*

b. Expand the left pane and select the queue manager WBRK_CONFIG_QM to display its services in the right pane.

   c. If the listener is displayed, right-click it and click *All Tasks -> Start*. This starts the listener. If the listener is not displayed:

   d. Right-click the queue manager and click *New -> Listener*. A new listener is created with (default) transport type TCP and (default) port number 1414.

   e. Right-click the new listener and click *Start*. This starts the listener.

2. To start a listener as a foreground task, enter the following command on the command line:

   ```
   runmqlsr -t tcp -p 1414 -m WBRK_CONFIG_QM
   ```

   To start a sender channel, use WebSphere MQ Explorer. This starts the channels as background tasks.

3. If you prefer, you can enter the commands:

   ```
   runmqchl -m WBRK_UNS_QM -c WBRK_UN_TO_BR
   ```

   ```
   runmqchl -m WBRK_QM -c WBRK_BR_TO_UN
   ```

Enabling applications to use Publish/Subscribe:

If WebSphere MQ queue security is enabled, users who want to subscribe need UPDATE authority to put to the SYSTEM.BROKER.CONTROL.QUEUE on the User Name Server's queue manager. Publish/Subscribe users also need UPDATE authority to allow them to use input and output queues in message flow nodes.

Enabling applications to use Publish/Subscribe security on z/OS:

This topic lists the steps that you need to complete to enable applications to use Publish/Subscribe security on z/OS.

- For the User Name Server on z/OS to extract user ID and group information from the External Security Manager (ESM) database, user IDs and groups must have an OMVS segment defined.

- To use publish/subscribe security, you need to have an ESM group defined called **mqbrkrs**. This group needs to have an OMVS segment defined. The user ID of the started task needs to be in this group.

- If you are using RACF, use the LG group OMVS command. For example:

  ```
  LG MQBRKRS OMVS
  ```

User IDs:

- If you have suitable authorization, you can use the following RACF command to display OMVS information about a user:

  ```
  LU id OMVS
  ```

- To give a user ID an OMVS segment, you can use the following RACF command if you have suitable authorization:

  ```
  ALTUSER id  OMVS(UID(xxx))
  ```

Groups:

- You can use the following RACF command to display OMVS information about a group if you have suitable authorization:

  ```
  LG group OMVS
  ```

- To give a group an OMVS segment you can use the following RACF command if you have suitable authorization:

```
ALTGROUP id  OMVS(GID(xxx))
```

Refer to the *OS/390 Security Server (RACF) Security Administrator's Guide* (or the appropriate documentation for an external security manager installed on the system) for details.

If an application tries to use publish/subscribe with security, and the user ID is not found by the User Name Server (either because the user ID does not exist or the user ID does not have an OMVS segment), the message BIP7017W is written to the SYSLOG.

If an application tries to use publish/subscribe with security, and the user ID is found by the User Name Server, but an access control list denies access to the topic, either of the following messages are written to SYSLOG:

```
BIP7025 User does not have permission to subscribe to a topic.
BIP7026 User does not have publish permission on a topic.
```

Using the Default Configuration wizard

Before you run the Default Configuration wizard for the first time:
- You must have administration privileges.

The Default Configuration wizard allows you to set up a simple default configuration on your local machine so that you can explore the product and run the samples that are supplied. It also allows you to remove an existing default configuration that has been set up on your logon account.

The wizard creates the following resources:
- A broker domain
- A sample broker
- A database to be used by the broker. The wizard defaults to whatever database is available and you are informed of the database used after it has been created in the **Default Configuration Summary** page. The text message also confirms which database manager will be used (details of the default database manager are also written to the wizard's log file.
- A WebSphere MQ queue manager.

This topic contains the following sections:
- "Creating a default configuration"
- "Removing a default configuration" on page 129

Creating a default configuration

1. The Default Configuration wizard launches automatically at the end of the installation of WebSphere Event Broker. You can launch the wizard manually from Getting started or from the **Welcome** page for the Message Brokers Toolkit application, which appears after you launch the Message Brokers Toolkit.
2. The **Welcome** panel of the wizard describes what is about to happen. Enter your password to log on and click **Next** to continue. You can click **Cancel** at any time to cancel the creation of the default configuration.

   The wizard performs a verification test to check that the default configuration is not already installed.
3. The **Default Configuration Summary** panel lists the resources that will be created. The information field in this panel confirms whether or not Derby has

been set as the default broker database. It also suggests an alternative option of installing and configuring an enterprise database server instead. Click **Next** to continue.

4. The **Default Configuration Progress** panel lists the background configuration actions as they happen and indicates successful completion. You can cancel the creation of the default configuration at this point by clicking **Cancel**. The wizard backs out all configuration tasks and the displays the progress and success of the process. The configuration process is written to a log file in the Eclipse workspace directory (TOOLKIT_WORKSPACE_dir/.metadata/DefaultConfigurationWizard.log).

   If the default configuration is set up successfully, you see an appropriate message. If there are errors during the setup of the default configuration, you see an appropriate message and the wizard backs out all configuration tasks. If an error occurs during the back out process, the wizard displays a list of resources that you must remove manually.

5. You can use the samples to verify the default configuration. The **Launch Samples Wizard when finished** check box is ticked by default. Click **Finish** to launch the sample preparation wizard.

   If you do not need to launch the sample preparation wizard, clear the **Launch Samples Wizard when finished** check box before clicking **Finish**.

   If you are viewing the information center within the Message Brokers Toolkit, you can launch the samples manually by clicking Pager samples. Alternatively, click **Help** → **Samples Gallery** then expand **Application samples** and **Message Broker - Installation verification samples**.

Removing a default configuration

1. Launch the wizard from Getting started or from the **Welcome** page for the Message Brokers Toolkit application, which appears after you launch the Message Brokers Toolkit.

2. The **Welcome** panel of the wizard describes what is about to happen. Enter your password to log on and click **Next** to continue. You can click **Cancel** at any time to cancel the removal of the default configuration.

   The wizard performs a verification test to check that the default configuration is already installed.

3. The **Remove Default Configuration Summary** panel lists the resources that will be removed. Click **Next** to continue.

4. The **Default Configuration Progress** panel lists the removal actions as they happen and indicates successful completion. The removal process is written to a log file in the Eclipse workspace directory (TOOLKIT_WORKSPACE_dir/.metadata/DefaultConfigurationWizard.log).

5. If the default configuration is removed successfully, you see an appropriate message. Click **Finish** to close the wizard. If errors occur during the removal of the default configuration, the wizard displays the errors and also write them to the log file.

Verifying components

To verify that the WebSphere Event Broker components that you have created exist, use the **mqsilist** command. On the command line type:

```
mqsilist
```

Note, that on z/OS it is not possible to display all the components.

If you do not specify any parameters when you issue this command, a list of components and queue manager names is displayed for each component created on this system, in the form:

```
BIP8099I: Broker: brokername - queuemanagername
BIP8099I: UserNameServer: UserNameServer - queuemanagername
BIP8099I: ConfigMgr: configmgrname - queuemanagername
BIP8071I: Successful command completion
```

Connecting components

Before you start:

To complete this task, you must have completed the following tasks:
- "Creating a Configuration Manager" on page 103
- "Creating a broker" on page 86
- "Creating a User Name Server" on page 114

Complete the steps below to make connections between the Configuration Manager, brokers, and the User Name Server.

If components in your broker domain are supported by different queue managers, establish WebSphere MQ connections between those queue managers to enable messages to be exchanged. It is important that each broker is able to exchange messages with the User Name Server that provides user name services for the broker.

If your broker domain components all run on the same system, and use a single queue manager, you do not need to create any WebSphere MQ connections between your brokers.

To achieve the required connection, complete the following steps. All the steps here are illustrated with MQSC examples. You can use any appropriate method for defining these resources. These examples assume that the queue managers are called COMP1 and COMP2.

1. Define a transmission queue on each component's queue manager. These queues will collect messages ready for transmission between components. Give the transmission queue the same name as the queue manager to which it transmits messages (that is COMP1 and COMP2 for this example).

   For example, on queue manager COMP1:

   `define qlocal('COMP2') usage(XMITQ) replace`

   and on queue manager COMP2:

   `define qlocal('COMP1') usage(XMITQ) replace`

2. Define the channels for the connection. Use sender-receiver pairs of channels for all two-way communications between queue managers that host WebSphere Event Broker components.

   a. Define the sender channel on the first component's queue manager (Sender(3)). This will transport messages sent by the first component to the second component.

   Allocate connection names according to your WebSphere MQ network conventions, and specify the protocol that you are using for this connection and the port on which the listener is listening.

   For example, on queue manager COMP1:

```
            define channel('COMP1_TO_COMP2') chltype(sdr) trptype(tcp)
            conname('WBRKSYS1(1415)') xmitq('COMP2') replace
```

b. Define a receiver channel on the first component's queue manager
   (`Receiver(2)`). Messages sent by the second component to the first will be
   received by this channel.

   This receiver channel must have the same name as the sender channel on
   COMP2, defined in Step 2c. For example, on queue manager COMP1:

```
   define channel('COMP2_TO_COMP1') chltype(rcvr) trptype(tcp) replace
```

c. Define the sender channel on the second component's queue manager
   (`Sender(1)`). This will transport messages sent by the second component to
   the first component.

   Allocate connection names according to your WebSphere MQ network
   conventions, and you must specify the protocol you are using for this
   connection.

   For example, on queue manager COMP2:

```
   define channel('COMP2_TO_COMP1') chltype(sdr) trptype(tcp)
   conname('WBRKSYS1(1414)') xmitq('COMP1') replace
```

d. Define a receiver channel on the second component's queue manager
   (`Receiver(4)`). Messages sent by the first component to the second will be
   received by this channel.

   This receiver channel must have the same name as the sender channel on
   COMP2, defined in Step 2a. For example, on queue manager COMP2:

```
   define channel('COMP1_TO_COMP2') chltype(rcvr) trptype(tcp) replace
```

3. Create and start a listener for each protocol in use.
4. Start the sender channels (1) and (3) on the respective queue managers. You can
   set up channel initiators for these channels. This reduces overhead by allowing
   the channels to stop when there is no message traffic, but ensures automatic
   startup when there are messages to transport.

You can set up a single receiver channel on the Configuration Manager's queue
manager to support all sender channels created for the brokers. This requires a
single definition on the Configuration Manager and a single sender definition on
each broker, which must have the same name on each broker. You can also use this
receiver channel on the Configuration Manager to support communications from
the User Name Server.

All WebSphere MQ connections between WebSphere Event Broker components,
and between clients and WebSphere Event Broker components, can be set up using
any of the communications protocols supported by WebSphere MQ (TCP/IP and
SNA on all operating systems; also, NetBIOS and SPX on Windows).

Modifying a broker

Before you start:

You must complete the following tasks:
- Ensure that your user ID has the correct authorizations to perform the task.
  Refer to "Security requirements for administrative tasks" on page 385
- "Creating a broker" on page 86

Modify a broker using the command line on the system where the broker
component is installed.

The parameters you can change on the broker affect the physical broker, created using the command line.

You can also modify the broker reference in the workbench, where it is possible to change broker properties, such as multicast properties.

Follow the link for the appropriate platform.
- "Modifying a broker on Linux and UNIX systems"
- "Modifying a broker on Windows" on page 133
- "Modifying a broker on z/OS" on page 134

Modifying a broker on Linux and UNIX systems

To modify a broker on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:

1. Stop the broker using the **mqsistop** command.
2. Enter the **mqsichangebroker** command with the parameters that you want to change: mqsichangebroker brokername <<-i ServiceUserID> -a ServicePassword> <-p DatabaseSourcePassword> <-s UserNameServerQueueManagerName> <-j | -d> <-t | -n> <-l UserLilPath> <-g ConfigurationTimeout> <-k ComfigurationDelayTimeout> <-v StatisticsMajorInterval> <-P HttpPort> where:

   brokername
   Is the broker name.

   -i   Is the service user ID that is used to run the broker.

   -a   Is the password for the broker user ID.

   -p   Is the password for the broker's database user ID.

   -s   Is the WebSphere MQ queue manger for the User Name Server

   -j   Indicates that publish/subscribe access control is to be enabled for this broker.

   -d   Indicates that publish/subscribe access control is to be disabled for this broker.

   -t   Indicates that the broker runs as a WebSphere MQ trusted application (not supported on AIX)

   -n   Indicates that the broker should cease to run as a WebSphere MQ trusted application.

   -l   Indicates from where LIL (loadable implementation libraries) files are loaded.

   -g   Is the maximum time (in seconds) to allow a broker to process a deployed message.

   -k   Is the maximum time (in seconds) to allow a broker to process a minimum size deployed message.

   -v   Is the time (in minutes) for the duration of the interval for collecting statistics archive records.

   -P   Is the port that the broker HTTP listener will use.

   For example, to change the user ID that is used to run the broker, enter the following command at the command prompt:

   ```
   mqsichangebroker WBRK_BROKER -i wbrkuid -a wbrkpw
   ```

3. Restart the broker using the mqsistart command. The broker restarts with the new properties.

If you cannot change a property using **mqsichangebroker**, delete the broker and then create a new one with the new properties.

Modifying a broker on Windows

To modify a broker on Windows:.

1. Stop the broker using the **mqsistop** command.
2. Enter the **mqsichangebroker** command with the parameters that you want to change:

```
mqsichangebroker brokername <<-i ServiceUserID> -a ServicePassword>
<-p DatabaseSourcePassword> <-s UserNameServerQueueManagerName>
<-j | -d> <-t | -n> <-l UserLilPath> <-g ConfigurationTimeout>
<-k ComfigurationDelayTimeout> <-v StatisticsMajorInterval> <-P HttpPort>
```

where:

brokername
> Is the broker name.

-i   Is the service user ID that is used to run the broker.

-a   Is the password for the broker user ID.

-p   Is the password for the broker's database user ID.

-s   Is the WebSphere MQ queue manger for the User Name Server

-j   Indicates that publish/subscribe access control is to be enabled for this broker.

-d   Indicates that publish/subscribe access control is to be disabled for this broker.

-t   Indicates that the broker runs as a WebSphere MQ trusted application.

-n   Indicates that the broker should cease to run as a WebSphere MQ trusted application.

-l   Indicates from where LIL (loadable implementation libraries) files are loaded.

-g   Is the maximum time (in seconds) to allow a broker to process a deployed message.

-k   Is the maximum time (in seconds) to allow a broker to process a minimum size deployed message.

-v   Is the time (in minutes) for the duration of the interval for collecting statistics archive records.

-P   Is the port that the broker HTTP listener will use.

For example, to change the user ID that is used to run the broker, enter the following command at the command prompt:

```
mqsichangebroker WBRK_BROKER -i wbrkuid -a wbrkpw
```

3. Restart the broker using the **mqsistart** command. The broker restarts with the new properties.

If you cannot change a property using **mqsichangebroker**, delete the broker and then create a new one with the new properties.

Modifying a broker on z/OS

Before you start:

To complete this task, you must have completed the following task:
- "Creating a broker on z/OS" on page 94

To modify a broker:
1. Ensure that the broker is running
2. Stop the broker components by issuing the command/F BROKERNAME, PC.
3. When it has stopped, use the MVS MODIFY command with the changebroker parameters that you want to change. For example:

   `/F <BROKERNAME>,cb g=100,k=200`
4. Restart the broker components by issuing the following command/F BROKERNAME, SC

   The broker now uses the changed parameters.

You cannot change all the parameters with which you created a broker. If you cannot modify a parameter that you need to change using the changebroker command, delete the broker and then create a new one. This will allow you to redefine all the parameters.

The parameters that you can change are:

g ConfigurationTimeOut

k ConfigurationDelayTimeout

s UserNameServerQueueManagerName

l UserLilPath

v StatisticsArchiveInterval

P HTTPPort

See "mqsichangebroker command" on page 221 for further information on these parameters.

Modifying a Configuration Manager

Before you start:

You must complete the following tasks:
- Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385
- "Creating a Configuration Manager" on page 103

Parameters that are required in order to start, stop and migrate the Configuration Manager (such as the service user ID and password, and the connection parameters to a configuration database for migration) can be modified only by using the command line on the system where the Configuration Manager component is installed.

Parameters that control a running Configuration Manager or domain (such as the set of broker references stored in the Configuration Manager) can be modified

using the Message Brokers Toolkit or a Configuration Manager Proxy application, which might or might not be on the same machine as the Configuration Manager component.

Follow the link for the appropriate platform.
- "Modifying a Configuration Manager on Linux and UNIX systems"
- "Modifying a Configuration Manager on Windows"
- "Modifying a Configuration Manager on z/OS" on page 136

If you need to transfer the Configuration Manager onto another queue manager, follow the steps described in "Moving the Configuration Manager to a new queue manager" on page 137.

Modifying a Configuration Manager on Linux and UNIX systems

The following steps show you how to modify a Configuration Manager's service user ID, service password, database password, User Name Server queue manager, and the maximum JVM heap size, on Linux and UNIX systems:

1. Stop the Configuration Manager using the **mqsistop** command.
2. Enter the **mqsichangeconfigmgr** with the parameters you want to change:

   ```
   mqsichangeconfigmgr configmgrName <<-i ServiceUserID> -a ServicePassword>
   <-p DatabasePassword> <-s UserNameServerQueueManagerName> <-j  MaxJVMHeapSize>
   ```

   where:

   configmgrName
   > Is the Configuration Manager name.

   -i   Is the service user ID that is used to run the Configuration Manager.

   -a   Is the password for the Configuration Manager user ID.

   -p   If an existing DB2 database from a pervious version of the product has not yet been migrated, use this option to set the password used to access the database.

   -s   Is the WebSphere MQ queue manger for the User Name Server.

   -j   Is the maximum Java virtual machine heap size, in megabytes (minimum 64).

   For example, to modify the Configuration Manager so that it can communicate with the User Name Server, enter the following command at the command prompt:

   ```
   mqsichangeconfigmgr CMGR01 -s WBRK_UNS_QM
   ```

3. Restart the Configuration Manager using the **mqsistart** command. The Configuration Manager restarts with the new properties.

If you cannot change a property, delete the Configuration Manager then create a new one with the new property. Creating a new Configuration Manager does not cause any loss of data as long as the previous Configuration Manager's database tables were not deleted (for example, by specifying the **-n** parameter on the **mqsideleteconfigmgr** command).

Modifying a Configuration Manager on Windows

The following steps show you how to modify a Configuration Manager's service user ID, service password, database password, User Name Server queue manager, and the maximum JVM heap size, on Windows:

1. Stop the Configuration Manager using the **mqsistop** command.
2. Enter the **mqsichangeconfigmgr** with the parameters you want to change:

```
mqsichangeconfigmgr configmgrName <<-i ServiceUserID> -a ServicePassword>
<-p DatabasePassword> <-s UserNameServerQueueManagerName> <-j  MaxJVMHeapSize>
```

   where:

   configmgrName
   Is the Configuration Manager name. This is optional.

   -i   Is the service user ID that is used to run the Configuration Manager.

   -a   Is the password for the Configuration Manager user ID.

   -p   If an existing DB2 database from a pervious version of the product has not yet been migrated, use this option to set the password used to access the database.

   -s   Is the WebSphere MQ queue manger for the User Name Server.

   -j   Is the maximum Java virtual machine heap size, in megabytes (minimum 64).

   For example, to modify the Configuration Manager so that it can communicate with the User Name Server, enter the following command at the command prompt:

```
mqsichangeconfigmgr CMGR01 -s WBRK_UNS_QM
```

3. Restart the Configuration Manager using the **mqsistart** command. The Configuration Manager restarts with the new properties.

If you cannot change a property, delete the Configuration Manager then create a new one with the new property. Creating a new Configuration Manager does not cause any loss of data as long as the previous Configuration Manager's database tables were not deleted (for example, by specifying the **-n** parameter on the **mqsideleteconfigmgr** command).

Modifying a Configuration Manager on z/OS

Before you start:

To complete this task, you must have completed the following task:
- "Creating a Configuration Manager on z/OS" on page 108

The following steps show you how to modify a Configuration Manager's database password, User Name Server queue manager, and the maximum JVM heap size:

1. At the command prompt, issue the stopcomponent command to stop the Configuration Manager.
2. When it has stopped, use the MODIFY command with the changeconfigurationmanager parameters that you want to change. For example:

```
MODIFY <configurationmanagername>,changeconfigurationmanager s=WBRK_UNS_QM
```

3. At the command prompt issue the startcomponent command.

   The Configuration Manager now uses the changed parameters.

You cannot change all the parameters with which you created a Configuration Manager. If you cannot modify a parameter that you need to change using the changeconfigurationmanager command, delete the Configuration Manager and then create a new one. This will allow you to redefine all the parameters.

The parameters that you can change are:

p=	If an existing DB2 database from a pervious version of the product has not yet been migrated, use this option to set the password used to access the database.
s=	Is the WebSphere MQ queue manger for the User Name Server.
j=	Is the maximum Java virtual machine heap size, in megabytes (minimum 64).

See "mqsichangeconfigmgr command" on page 228 for further information on these parameters.

Moving the Configuration Manager to a new queue manager

The following steps show you how to move the Configuration Manager to a new queue manager that is on the same computer or on a different computer:

1. Use the **mqsicreateconfigmgr** command to create a new Configuration Manager that uses the new queue manager. Do not specify a database name.
2. If possible, stop all brokers in the domain using the **mqsistop** command.
3. Stop the original Configuration Manager using the **mqsistop** command.
4. Back up the original Configuration Manager using the **mqsibackupconfigmgr** command.
5. On the computer that contains the new Configuration Manager, use the **mqsirestoreconfigmgr** command to overwrite the new Configuration Manager's repository with the one that you backed up.
6. Start the new Configuration Manager using the **mqsistart** command.
7. Perform a complete deployment of the topology, using the Message Brokers Toolkit, the **mqsideploy** command, or the Configuration Manager Proxy. This tells all the brokers in the domain to associate themselves with the new Configuration Manager.
8. If you stopped the brokers in the domain in Step 2, start them using the **mqsistart** command as soon as deployment is initiated, so that the deployments can now be processed.
9. If it was not possible to stop the brokers in Step 2, ensure that any messages on the original Configuration Manager's queue manager's SYSTEM.BROKER.ADMIN.QUEUE are transferred manually to the new Configuration Manager's queue manager's SYSTEM.BROKER.ADMIN.QUEUE. This is the queue that brokers use to communicate their status to the Configuration Manager and if any status change event occurred between stopping the original Configuration Manager in Step 3 and the complete deployment in Step 7, any messages that report a change in status will have been sent to the old Configuration Manager.

   Related tasks

   "Modifying a Configuration Manager" on page 134

   Recovering after the Configuration Manager's queue manager fails

   Recovering after a shared queue manager fails

   Related reference

   "mqsicreateconfigmgr command" on page 266

   "mqsistop command" on page 328

   "mqsibackupconfigmgr command" on page 220

   "mqsirestoreconfigmgr command" on page 321

   "mqsistart command" on page 323

Modifying a User Name Server

Before you start:

You must complete the following tasks:
- Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385
- "Creating a User Name Server" on page 114

Modify the User Name Server using the command line on the system where the User Name Server component is installed.

Follow the link for the appropriate platform.
- "Modifying a User Name Server on Linux and UNIX systems"
- "Modifying a User Name Server on Windows" on page 139
- "Modifying a User Name Server on z/OS" on page 139

Modifying a User Name Server on Linux and UNIX systems

To modify a User Name Server on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:

1. Stop the User Name Server using the **mqsistop** command.
2. Enter the **mqsichangeusernameserver** command with the parameters that you want to change: mqsichangeusernameserver <<-i ServiceUserID> <-a ServicePassword> <-d SecurityDomainName> <-r RefreshInterval> <-g AuthProtocolDataSource> <-j`` | -o> where:

   -i   Is the service user ID that is used to run the User Name Server

   -a   Is the password for the User Name Server user ID.

   -d   Is the security domain that the User Name Server uses on the Windows platform.

   -r   Is the number of seconds between each refresh of the User Name Server internal cache.

   -j   Indicates that groups and group memberships are defined in the data source for the authentication protocol, rather than being drawn from the operating system.

   -o   Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.

   -g   Is the name of the data source required by the authentication protocol.

   For example, to change the number of seconds between each refresh of the User Name Server internal cache, enter the following command at the command prompt:
   ```
   mqsichangeusernameserver -r 2000
   ```
3. Restart the User Name Server using the **mqsistart** command. The User Name Server restarts with the new properties.

If you cannot change a property using **mqsichangeusernameserver**, delete the User Name Server and then create a new one with the required properties.

Modifying a User Name Server on Windows

To modify a User Name Server:

1. Stop the User Name Server using the **mqsistop** command.
2. Enter the **mqsichangeusernameserver** command with the parameters you want to change:

   ```
   mqsichangeusernameserver <<-i ServiceUserID> <-a ServicePassword>
   <-d SecurityDomainName> <-r RefreshInterval> <-k AuthProtocoltype>
   <-j AuthProtocolModule> <-g AuthProtocolDataSource>
   ```

   where:

   -i   Is the service user ID that is used to run the User Name Server

   -a   Is the password for the User Name Server user ID.

   -d   Is the security domain that the User Name Server uses.

   -r   Is the number of seconds between each refresh of the User Name Server internal cache.

   -k   Indicates that the authentication protocol is supported by brokers.

   -j   Indicates that the authentication services product library is to be used.

   -g   Indicates the name and location of the password file used to source any protocol related information.

   For example, to change the number of seconds between each refresh of the User Name Server internal cache, enter the following command at the command prompt:

   ```
   mqsichangeusernameserver -r 2000
   ```

3. Restart the User Name Server using the **mqsistart** command. The User Name Server restarts with the new properties.

If you cannot change a property using **mqsichangeusernameserver**, delete the User Name Server and then create a new one with the new properties.

Modifying a User Name Server on z/OS

Before you start:

To complete this task, you must have completed the following task:

- "Creating a User Name Server on z/OS" on page 118

To modify a User Name Server.

1. At the command prompt issue the stopcomponent command to stop the User Name Server.
2. When it has stopped, use the MODIFY command with the changeusernameserver parameters that you want to change. For example:

   ```
   MODIFY <usernameserver>,changeusernameserver r=2000
   ```

3. At the command prompt issue the startcomponent command.

   The User Name Server now uses the changed parameters.

You cannot change all the parameters with which you created the User Name Server: if you cannot modify a parameter that you need to change using the changeusernameserver command, delete the User Name Server and then create a new one. This will allow you to redefine all the parameters.

Modifying component database access

Set the user ID through which a broker accesses its database tables by using the DataSourceUserID parameter on the **mqsicreatebroker** command.

You cannot modify this parameter once the component has been created. Therefore you must save the database tables, delete the component, and create a new one, specifying the new user ID. The new component uses the data in the saved tables, with the updated database user ID.

To ensure that you preserve the data held in these databases, follow the steps detailed in the following task.

"Modifying broker database access"

Modifying broker database access

Set the user ID through which the broker accesses its database tables by using the DataSourceUserID parameter on the **mqsicreatebroker** command.

You cannot modify this parameter using the **mqsichangebroker** command. You have to delete the broker and create a new one, specifying the new user ID.

To ensure that you do not lose any of the data in the broker database when performing this task, complete the following steps.
- Back up the database tables. These are listed in "mqsicreatebroker command" on page 257.
- Delete the broker, following the "Deleting a broker" on page 141 task.
- Restore the database tables to the new location.
- Create a new broker, following the "Creating a broker" on page 86 task. Specify the new user ID when you create the new broker.

  The new broker uses the database tables with the new user ID

Moving from WebSphere Event Broker on a distributed platform to z/OS

Recommendations on how to define resources for WebSphere Event Broker for z/OS are given in the following topics:
- "z/OS customization overview" on page 58
- "Customizing the z/OS environment" on page 57
- "Creating a broker on z/OS" on page 94
- "Creating a User Name Server on z/OS" on page 118
- "Creating a Configuration Manager on z/OS" on page 108
- "Administration in z/OS" on page 333

Taking these into account, recreate your broker and User Name Server resources on z/OS and deploy your message flows and execution groups to a WebSphere Event Broker for z/OS broker. If you have extended WebSphere Event Broker in a distributed environment with plug-in parsers or message processing nodes, port these plug-ins to run under z/OS.

Also consider the following points:

- Floating point conversion: z/OS runs under z/OS floating point format, so floating point operations on z/OS run in a different range and accuracy from distributed platforms.
- Administration commands are partially implemented as console commands and partially as JCL commands.
- Event log messages: All address spaces have a JOBLOG where messages appear. In addition to this, all messages appear on the SYSLOG, with important operator messages being filtered to the console through MPF (Message Processing Facility).

Moving user applications

You can write your own applications to work with WebSphere Event Broker. If these applications use the common subset of functionality of all WebSphere Event Broker brokers, no migration is necessary. If you are using functionality that is available on some WebSphere Event Broker platforms only, for example message segmentation and WebSphere MQ message groups, be aware that WebSphere Event Broker for z/OS does not provide support for this migration.

Deleting a broker

Before you start:

You must complete the following tasks:
- Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385
- Remove the broker from the broker domain in the workbench ("Removing a broker from a broker domain" on page 153).

Delete a broker using the command line on the system where the broker component is installed.

You can remove the broker from the broker topology using the workbench, but the physical broker will not be deleted until the broker is physically deleted from the command line.

Follow the link for the appropriate platform.
- "Deleting a broker on Linux and UNIX systems"
- "Deleting a broker on Windows" on page 142
- "Deleting a broker on z/OS" on page 142

Deleting a broker on Linux and UNIX systems

To delete a broker on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:.
1. Stop the broker using the **mqsistop** command.
2. Enter the following command to delete the broker:

   `mqsideletebroker WBRK_BROKER`

   where:

   `WBRK_BROKER` is the broker name.

On completion of this task, you have:
- Removed the broker's data from the database.

- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the **mqsilist** command.

Deleting a broker on Windows

To delete a broker:

1. Stop the broker using the **mqsistop** command.
2. Enter the following command to delete the broker:

   `mqsideletebroker WBRK_BROKER`

   where:

   `WBRK_BROKER` is the broker name.

On completion of this task, you have:
- Stopped the Windows service that runs the broker.
- Removed the broker's data from the database.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the **mqsilist** command.

Deleting a broker on z/OS

To delete a broker:

1. Remove the broker from the broker domain, in the workbench. Refer to "Removing a broker from a broker domain" on page 153.
2. Stop the broker, by stopping the started task.
3. Customize and submit the following delete jobs in your component PDSE to delete WebSphere MQ and DB2 definitions:

Delete jobs	Description
BIPDLBK	Delete component including WebSphere MQ broker queues and channels and rows in the DB2 database
BIPDLDB	Drop the broker DB2 database

Note that not all files are deleted from the component directory in the HFS.

Deleting an execution group from a broker using the command line

Before you start:

You must complete the following tasks:
- "Adding an execution group to a broker" on page 102

Instead of employing the Message Brokers Toolkit, you can use this task as an alternative method of deleting an execution group .

For more details on deleting an execution group from the Message Brokers Toolkit see "Deleting an execution group" on page 156

To delete an execution group from a broker using the command line:

1. Open a command prompt that has the environment configured for WebSphere Event Broker Version 6.0.
2. Enter the following command to delete the execution group:

   `mqsideleteexecutiongroup -i host -p 1414 -q QMGR -b BROKER –e EG1`

   where

   host   Is the host name or IP address of the Configuration Manager for the domain on which the broker resides.

   1414   Is the port on which the Configuration Manager's queue manager is listening.

   QMGR
       Is the name of the Configuration Manager's queue manager.

   BROKER
       Is the name of the broker.

   EG1   Is the name of the execution group that you want to delete.

On completion of this task, the execution group is no longer running on the specified broker. In addition, any message flows that were running on the execution group are no longer running.

Deleting the Configuration Manager

Before you start:

Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385

Follow the link for the appropriate platform:
- "Deleting a Configuration Manager on Linux and UNIX systems"
- "Deleting a Configuration Manager on Windows" on page 144
- "Deleting a Configuration Manager on z/OS" on page 144

Deleting a Configuration Manager on Linux and UNIX systems

You delete the Configuration Manager using the command line. The Configuration Manager can be deleted only from the system where the Configuration Manager component is installed.

You can delete a Configuration Manager without also deleting your domain connection parameters in the workbench. If you want to delete a Configuration Manager and create a new one, you can keep your connection parameters in the workbench, even if you specify different parameters when creating the new Configuration Manager. When you reconnect to your domain in the workbench your new settings are displayed.

To delete a Configuration Manager on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:
1. Stop the Configuration Manager using the "mqsistop command" on page 328command.
2. Delete the Configuration Manager using the "mqsideleteconfigmgr command" on page 283 command.

On completion of this task, you have:

- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the "mqsilist (list resources) command" on page 296 command.
- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the "mqsideleteconfigmgr command" on page 283 command.

Deleting a Configuration Manager on Windows

You delete the Configuration Manager using the command line. The Configuration Manager can be deleted only from the system where the Configuration Manager component is installed.

You can delete a Configuration Manager without also deleting your domain connection parameters in the workbench. If you want to delete a Configuration Manager and create a new one, you can keep your connection parameters in the workbench, even if you specify different parameters when creating the new Configuration Manager. When you reconnect to your domain in the workbench your new settings will be displayed.

To delete a Configuration Manager:

1. Stop the Configuration Manager using the "mqsistop command" on page 328 command.
2. Delete the Configuration Manager using the "mqsideleteconfigmgr command" on page 283 command.

On completion of this task, you have:

- Stopped the Windows service that runs the Configuration Manager.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the "mqsilist (list resources) command" on page 296 command.
- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the "mqsideleteconfigmgr command" on page 283 command.

Deleting a Configuration Manager on z/OS

To delete a Configuration Manager:

1. Stop the Configuration Manager, by stopping the started task.
2. Customize and submit the following delete jobs in your component PDSE to delete WebSphere MQ definitions:

Delete jobs	Description
BIPDLCM	Delete component including WebSphere MQ broker queues and channels.

Note that not all files are deleted from the component directory in the file system.

Disabling a User Name Server

When you delete a User Name Server you disable publish/subscribe services within the broker domain.

To delete a User Name Server from the broker domain, remove the connections between the broker, Configuration Manager, and User Name Server. This ensures that the broker and Configuration Manager do not continue to communicate with the User Name Server.

Modify the broker and Configuration Manager using the **mqsichangebroker** and **mqsichangeconfigmgr** commands, before you delete the User Name Server. The following steps show you how to do this.

- Modify the broker by removing the reference to the queue manager for the User Name Server. Use the **mqsichangebroker** command to modify the -s and -d parameters:
  1. Specify an empty string (two double quotation marks, ″″) on the -s parameter.
  2. Specify the -d parameter to disable publish/subscribe access for the broker.

  This ensures that the broker does not try to communicate with the User Name Server.

- Modify the Configuration Manager by removing the reference to the queue manager for the User Name Server. Use the command to modify the -s parameter, by specifying an empty string (two double quotation marks, ″″). This ensures that the Configuration Manager does not try to communicate with the User Name Server.

Now that you have made the required changes to the broker and Configuration Manager, you can delete the User Name Server, and thus disable publish/subscribe services.

Deleting a User Name Server

Before you start:

You must complete the following task:
- Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385
- Modify the broker and Configuration Manager so that they do not continue to communicate with the User Name Server. See "Disabling a User Name Server" on page 144 for details of the changes you must make.

Delete a User Name Server using the command line on the system where the User Name Server component is installed.

Follow the link for the appropriate platform.
- "Deleting a User Name Server on Linux and UNIX systems"
- "Deleting a User Name Server on Windows" on page 146
- "Deleting a User Name Server on z/OS" on page 146

Deleting a User Name Server on Linux and UNIX systems

To delete a User Name Server on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:
1. Stop the User Name Server using the **mqsistop** command.
2. Enter the following command to delete the User Name Server:

   mqsideleteusernameserver

On completion of this task, you have:

- Deleted the queue associated with the User Name Server on the local queue manager
- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the **mqsilist** command.

Deleting a User Name Server on Windows

To delete a User Name Server:

1. Stop the User Name Server using the **mqsistop** command.
2. Enter the following command to delete the User Name Server:

   ```
   mqsideleteusernameserver
   ```

On completion of this task, you have:

- Stopped the Windows service that runs the User Name Server.
- Deleted the queue associated with the User Name Server on the local queue manager.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the **mqsilist** command.

Deleting a User Name Server on z/OS

To delete a User Name Server:

1. Stop the User Name Server, by stopping the started task.
2. Customize and submit the following delete job manually to delete WebSphere MQ definitions:

Delete jobs	Description
BIPDLUN	Deletes components WebSphere MQ queues and channels

Note that not all files are deleted from the component directory in the file system.

Configuring a broker domain in the workbench

Before you start:

You must have created the physical broker domain components. Refer to "Configuring broker domain components" on page 66.

Configuring a broker domain using the workbench is only possible on Windowsand Linux (x86 platform). It is the second part of the two-stage process to create and configure your broker domain.

The workbench allows you to configure and administer the broker domain components as you require. You can launch the workbench in any of the following ways:

- Using the shortcut added to your desktop during installation, labelled 'WebSphere Message Brokers Toolkit'.

- Using the following commands in a command prompt from their location in either, the RAD root directory (if Rational products such as Rational Application Developer were installed on your system before WebSphere Event Broker was installed), or the WebSphere Event Broker installation directory (if no Rational products were installed on your system before WebSphere Event Broker was installed).

  On Windows:

  `wmbt.exe`

  On Linux (x86 platform):

  `wmbt.bin`

- From the Start Menu (Windows only) **Start>Programs>IBM WebSphere Message Brokers 6.0>WebSphere Message Brokers Toolkit**

See the following tasks for instructions on how to configure a broker domain in the workbench:

- "Creating a domain connection"
- "Modifying domain connection properties" on page 149
- "Deleting a domain connection" on page 150
- "Adding a broker to a broker domain" on page 150
- "Modifying broker properties" on page 152
- "Removing a broker from a broker domain" on page 153
- "Removing deployed children from a broker" on page 154
- "Adding an execution group to a broker" on page 155
- "Deleting an execution group" on page 156
- "Removing deployed children from an execution group" on page 157

When configuring your broker domain, you are prompted to deploy all changes to the Configuration Manager. It is possible to set your user preferences to not prompt you to deploy after each change. Refer to "Changing Broker Administration preferences" on page 157.

When you make changes to the broker domain, and deploy to the Configuration Manager, there can be a short delay until the workspace is updated and the Configuration Manager tells you that the deploy has worked. The delay depends on the network configuration, or the number of changes to make to the configuration of the broker domain during the deployment.

Creating a domain connection

Before you start:

You must complete the following tasks:
- "Creating a Configuration Manager" on page 103
- Create and start a listener for the Configuration Manager

Create a domain connection in the workbench using the Create a New Domain Connection wizard. Enter a set of parameters to create a `.configmgr` file. The parameters contained within the `.configmgr` file are used to connect to the Configuration Manager, where you can view and edit your broker domain.

To create a domain connection:

1. Switch to the Broker Administration perspective.
2. Right-click in the Domains view, and click **New** → **Domain**
3. In the Create a New Domain Connection wizard, enter:
   a. The name of the WebSphere MQ queue manager that the Configuration Manager is using. This property is mandatory.
   b. The host name or IP address of the machine on which the Configuration Manager is running (the default is `localhost`). This property is mandatory.
   c. The TCP port on which the WebSphere MQ queue manager is listening (the default is 1414). This property must be a valid positive number.
   d. Optional: The name of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java classname, but you can leave this box blank if it does not apply to your domain connection.
   e. Optional: The location of the JAR file for the Security Exit required to connect to the WebSphere MQ queue manager. Use the Browse button to find the file location. You can leave this box blank if it does not apply to your domain connection.

      Note: The location of the JAR file is required if a Security Exit class is entered.
   f. Optional: The cipher suite, distinguished names, CRL name list, key store, and trust store parameters required when enabling SSL (see "Implementing SSL authentication" on page 16 for more information). The cipher suite field displays available cipher suites. Click **More** to configure a custom cipher suite in the Broker Administration preferences window. If a cipher suite is not specified, all other fields in the SSL section are unavailable.

      You can configure several domain connections in your workspace. Each domain connection has to address a different Configuration Manager, which needs to have a different WebSphere MQ queue manager, host name, or TCP port number. An error message is displayed in the wizard, if you try to create a second broker domain using the same queue manager, host name, and port number.
4. Click **Next** to begin the domain connection to the Configuration Manager.

   You cannot connect to a Version 2.1 Configuration Manager. An error is thrown, and you will not be able to populate the domain with brokers or topics, or connect to it using the workbench.
5. If you click **Cancel**, the Create a New Domain Connection wizard closes and a forced disconnection from the domain occurs.
6. After the domain connection has been made, enter:
   a. The name of your Server Project. This is the container for your domain connection. If you have not already created a server project, you can specify the name of a new project here. The server project is created with the domain connection.
   b. The Connection Name. This is the name you give to the `.configmgr` file that contains the parameters to connect to the Configuration Manager.
7. Click **Finish** to create the domain connection.

   The new domain connection is added to the Broker Administration Navigator view, under Domain Connections. Here the server project holds the `.configmgr` domain connection file.

   The view of the broker domain is displayed in the Domains view.

Modifying domain connection properties

Before you start:

To complete this task, you must have completed the following tasks:
- "Creating a domain connection" on page 147
- Disconnect from the broker domain. Refer to "Connecting to and disconnecting from the broker domain" on page 185.

Modify the parameters of the `.configmgr` domain connection file that are used to connect to the Configuration Manager.

To modify the domain connection properties.
1. Switch to the Broker Administration perspective.
2. In the Broker Administration Navigator view, expand Domain Connections, and open your server project.
3. Right-click the `.configmgr` domain connection file and click **Open With → Domain Connection Editor**. From here you can change:
   a. The name of the WebSphere MQ queue manager that the Configuration Manager is using. This property is mandatory.
   b. The host name or IP address of the machine on which the Configuration Manager is running. This property is mandatory.
   c. The TCP port on which the WebSphere MQ queue manager is listening. This property must be a valid positive number.
   d. Optional: The name of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java class name, but it is not mandatory and you can leave it blank, if it does not apply to your domain connection.
   e. Optional: The location of the JAR file for the Security Exit required to connect to the WebSphere MQ queue manager. Use the Browse button to find the file location. You can leave this box blank if it does not apply to your domain connection.

      Note: The location of the JAR file is required if a Security Exit class is entered.
   f. Optional: The cipher suite, distinguished names, CRL name list, key store, and trust store parameters required when enabling SSL (see "Implementing SSL authentication" on page 16 for more information). The cipher suite field displays available cipher suites. Click **More** to configure a custom cipher suite in the Broker Administration preferences window. If a cipher suite is not specified, all other fields in the SSL section are unavailable.
4. Close the editor.
5. You are prompted to save the changes; click **Yes**. If the broker domain is connected, you are prompted to disconnect before you can save your changes. The `.configmgr` domain connection file is updated with the new parameters.

You can also change the domain connection parameters for a disconnected domain, from the Domains view. Right-click the disconnected domain and click **Edit Parameters**. This opens the Domain Connection editor. Follow steps 3 to 5 above to make the changes you require.

Deleting a domain connection

Before you start:

You must complete the following task:
- "Creating a domain connection" on page 147

To delete a domain connection, delete the corresponding `.configmgr` file from your server project.

The configuration of the broker domain is stored in the Configuration Manager and is not affected by deleting the domain connection. If you create a new domain connection to the same Configuration Manager, the broker domain will be configured and available for use.

The following steps show you how to delete a domain connection.
1. Switch to the Broker Administration perspective.
2. In the Broker Administration Navigator view:
   a. Expand Domain Connections
   b. Open the servers project (called Servers by default).
3. Right-click the domain connection file(`<connection name>.configmgr`) and click **Delete**.
4. Click **OK** at the prompt, to confirm that you want to delete the domain connection.

   If the broker domain in the Domains view is connected, you are prompted to disconnect from the broker domain before it can be deleted. Click **Yes** to disconnect from the broker domain. Clicking **Cancel** at this prompt, cancels deletion of the domain connection.

On completion of this task:
- The `.configmgr` file is removed from the server project.
- The view of the broker domain and its hierarchy is removed from the Domains view.

Adding a broker to a broker domain

Before you start:

You must have completed the following tasks:
- Create a physical broker. On Windows and UNIX platforms, you do this using the command line. Refer to "Creating a broker" on page 86. To create a physical broker on z/OS, refer to "Creating a broker on z/OS" on page 94.
- Connect to the broker domain. Refer to "Connecting to and disconnecting from the broker domain" on page 185.

Adding a broker to the broker topology creates a reference to the broker in the configuration repository; it does not create the physical broker. When you add a broker, be sure to use the same name that you used to create the broker.

To add a broker to a broker domain:
1. Switch to the Broker Administration perspective.
2. Right-click in the Domains view, and click **New** → **Broker**.

3. In the **Create a Broker** wizard:
   a. Select the broker domain to which you want to add the broker. If the selected broker domain is not connected, you are prompted to connect to the domain. Click **OK**.
   b. Enter the name of the broker.
   c. Enter the name of the WebSphere MQ queue manager that the broker uses.

      Note:
      1) If the WebSphere MQ queue manager is on a separate machine, make sure that you have performed the steps listed in "Connecting components" on page 130.
      2) You can associate a WebSphere MQ queue manager with only one broker, even if the brokers are in different broker domains.
4. Click **Next**.
5. Optional: Enter a short or long description for the broker.
6. Click **Finish** to add the broker to the broker domain.
7. You are, by default, prompted to deploy the updated publish/subscribe topology configuration.

   You only need to deploy the topology (either **Complete** or **Delta**) if you are using publish/subscribe and want to share publications or subscriptions.

   If you are *not* using publish/subscribe, click **None**. A deployment now (to associate the broker with the Configuration Manager), is not necessary; the broker is automatically associated with the Configuration Manager the first time a bar file is deployed.

   You can set user preferences so that you are not prompted to deploy the publish/subscribe topology. Instead, you can choose for either a complete or delta deployment to be performed automatically. Alternatively, if you are not using publish/subscribe, you might prefer to set the preference so that *no* automatic deployment takes place.

In the Domains view, the broker is added to the broker domain and a default execution group is added to the broker.

Adding a broker to the broker topology creates security ACL groups, which give the user ID full control of the broker and its default execution group. These ACL groups exist until this broker is removed from the broker domain. This means that the user ID can be removed from the **mqbr*** groups, but the user still has the full control access level for the broker and its default execution group.

Next:

Continue to add further execution groups to the broker if necessary. Then create, modify, or reuse message flows, message sets and other required files, and add them to the broker archive for deploying to the broker.

Copying a broker

Before you start:

You must complete the following task:
- "Adding a broker to a broker domain" on page 150

You can copy a broker that you have previously added to a broker domain. The new broker reference can be pasted within the same broker domain only.

The new broker inherits the same short and long description as the original broker, and the same execution groups are added. All other broker properties and multicast properties are not inherited by the new broker.

The new broker is automatically given a unique broker name, and queue manager name.

The following steps show you how to copy a broker.

1. Switch to the Broker Administration perspective.
2. In the Domains view, open the broker domain and right-click the broker that you want to copy. Click **Copy**.
3. In the broker domain right-click the Broker Topology and click **Paste**.
4. You are prompted to deploy the updated topology configuration. Click **Delta** to perform a delta deploy. If user preferences are not set to *prompt*, the topology deploy is automatic.

   The new broker reference is added to the broker domain. It inherits the short and long description and the same execution groups as the original broker.

   The new broker is given a unique name in the broker domain, and a new queue manager name.

The newly-created broker does not necessarily reference a physical broker correctly. The workbench does not check if a physical broker exists for the new broker reference.

To ensure that the broker reference does not contain any errors, rename the broker and the broker's queue manager to be exactly the same names as those specified on the **mqsicreatebroker** command when the physical broker was created:

- To rename the broker, refer to "Renaming a broker" on page 153.
- To rename the broker's queue manager, refer to "Modifying broker properties."

Modifying broker properties

Before you start:

To complete this task, you must have completed the following tasks:
- "Adding a broker to a broker domain" on page 150
- Start the broker. Refer to "Starting and stopping a broker" on page 187.

You can modify broker properties by adding a long or short description and , customizing the broker or multicast properties.

To modify the broker properties:

1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the broker that you want to modify, and click **Properties**. In the properties window you can modify:
   a. The broker properties.

      Click **Broker** in the left panel of the properties window. Refer to "Broker properties" on page 201 for details of all the broker properties.
   b. The multicast properties.

Click **Multicast** in the left panel of the properties window and expand it.

Click **Advanced** to access the advanced multicast properties.

Refer to "Setting up a multicast broker" on page 166 for details of all the multicast properties.

c. The description for the broker.

Click **Description** in the left panel of the properties window. Add a short or long description for the broker.

3. Click **OK** to save all modifications to the broker.

An automatic broker configuration deploy is immediately performed to take into account the broker properties changed, except for the broker queue manager name and the short and long descriptions, for which this deploy is not needed to update the broker properties.

Renaming a broker

Before you start:

You must complete the following tasks:

- "Adding a broker to a broker domain" on page 150

You might need to rename a broker (and possibly its queue manager; refer to "Modifying broker properties" on page 152) if your original attempt at creating a broker reference contained an error.

Renaming a broker is simpler than deleting and re-creating it.

The following steps show you how to rename a broker:

1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the broker and click **Rename**. The **Rename Broker** dialog is displayed.
3. In the **New name** field, type the new name of the broker.

   The name must be exactly the same name as that specified on the **mqsicreatebroker** command.
4. Click **Finish** to rename the broker.

The broker's name is updated in the Domains view, and in the Topology editor.

Removing a broker from a broker domain

Before you start:

You must complete the following task:

- "Adding a broker to a broker domain" on page 150

Removing a broker from a broker domain deletes its broker reference in the configuration repository. The broker is not deleted from the system when you perform this task; it simply marks the broker as logically deleted from the configuration repository.

If you want to move a broker from one topology to another, you need to delete and recreate the broker physically (using the **mqsideletebroker** and

mqsicreatebroker commands) even if the Configuration Manager in both domains are at the same product and service release level. See "Deleting a broker" on page 141 for further information.

The following steps show you how to remove a broker from a broker domain.
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the broker domain to reveal the broker that you want to remove.

   To remove more than one broker from the same broker domain, select each broker while holding down the Ctrl key.
3. Right-click the broker and click **Delete**.
4. Click **OK** at the prompt to confirm that you want to remove the broker from the broker domain.
5. You are, by default, prompted to deploy the updated publish/subscribe topology configuration.

   You only need to deploy the topology if you are using publish/subscribe and want to share publications or subscriptions.

   If you are *not* using publish/subscribe, click **None**, as a deployment is not necessary; when removing a broker from the domain, the Configuration Manager automatically requests the broker component to stop message flows that are running and to tidy up any resources being used. (Although if this operation fails for any reason, you can again request the broker to tidy up by deploying a delta topology.)

   You can set user preferences so that you are not prompted to deploy the publish/subscribe topology. Instead, you can choose for either a complete or delta deployment to be performed automatically. Alternatively, if you are not using publish/subscribe, you might prefer to set the preference so that *no* automatic deployment takes place.

The broker and its execution groups are removed as components of the broker domain. Confirmation of the broker's deletion is in two places:
- The broker is removed from the Domains view.
- The broker icon is removed from the Broker Topology editor. If the broker was connected to another broker, this connection is also removed.

To delete the physical broker after you have removed the broker from the domain, refer to "Deleting a broker" on page 141.

Removing deployed children from a broker

Remove the deployed children from a broker if the synchronization between the broker and the Configuration Manager falls into an inconsistent state. Removing the deployed children removes all the message flows and message sets from all execution groups on the specific broker. It also deletes all execution groups, except the default execution group, which is automatically re-created.

The message flows and message sets are not deleted by removing deployed children from a broker. They remain in the development workspace, but their reference is removed from the broker.

To remove the deployed children from a broker:
1. Switch to the Broker Administration perspective.

2. In the Domains view, expand the broker domain to reveal the broker with which you want to work.
3. Right-click the broker, and click **Remove Deployed Children**.
4. Click **OK** at the prompt to confirm that you want to delete all execution groups on the broker.

   This will remove all message flows and message sets from all execution groups, and delete all execution groups. An automatic broker configuration deploy is immediately performed for the broker to save the changes.

A BIP08921 information message is displayed, to show that the request was received by the Configuration Manager. Verify the results of the deployment by opening the Event Log.

Adding an execution group to a broker

Before you start:

You must complete the following tasks:
- "Adding a broker to a broker domain" on page 150
- Connect to the broker domain. Refer to "Connecting to and disconnecting from the broker domain" on page 185.

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly.

For more details about why you might want to create multiple execution groups, see Execution groups.

Instead of employing this method, you can create an execution group using the command line; see "Adding an execution group to a broker" on page 102.

To add an execution group to a broker:
1. Switch to the Broker Administration perspective.
2. Right-click in the Domains view, and click **New** → **Execution Group**.
3. In the Create an Execution Group wizard:
   a. Select the broker to which you want to add the execution group.

      If the selected broker domain is not connected, a Confirm Connection dialog prompts you to connect to the domain. If you click **OK**, the domain is connected and populated with the defined brokers. If you click **Cancel** at this prompt, the wizard remains open.
   b. Enter the name of the execution group.
4. Click **Next**.
5. Optional: Enter a short or long description for the execution group.
6. Click **Finish** to add the execution group to the broker.

In the Domains view, the execution group is added to the appropriate broker.

Modifying execution group properties

You can add a long or short description to an execution group. This can be an execution group that you have added to the broker, or the default execution group.

The following steps show you how to modify execution group properties.
1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the execution group broker that you want to modify, and click **Properties**. The **Execution Group Properties** dialog is displayed.
3. Add a long or short description to the execution group.
4. Click **OK** to add the description.

Renaming an execution group

Before you start:

You must complete the following task:
• "Adding an execution group to a broker" on page 155

You can rename any execution group that you have added to a broker.

To rename an execution group:
1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the execution group and click **Rename**. The **Rename Execution Group** dialog is displayed.
3. In the **New name** field, type the new name of the execution group.
4. Click **Finish** to rename the execution group.

The name of the execution group is updated in the Domains view.

Deleting an execution group

Before you start:

You must complete the following task:
• "Adding an execution group to a broker" on page 155

You can delete an execution group from the broker to which it belongs.

A broker must always have at least one execution group; you cannot delete the last group belonging to a broker.

Instead of employing this method, you can delete an execution group using the command line; see "Deleting an execution group from a broker using the command line" on page 142.

The following steps show you how to delete an execution group from a broker.
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain to reveal the execution group that you want to delete.
   To delete more than one execution group from the same broker domain, select each execution group while holding down the Ctrl key.
3. Right-click the execution group and click **Delete**.
4. Click **OK** at the prompt to confirm that you want to delete the execution group from the broker.

An automatic broker configuration deploy is immediately performed for the broker parent.

A BIP08921 information message is displayed to show that the request was received by the Configuration Manager. Verify the results of the deployment by opening the Event Log.

No message flows or message sets are deleted from the development workspace. The execution group and its assigned message flows and message sets are deleted from the Domains view. However, the messages flows and message sets remain in the Broker Administration Navigator view.

Their assignment reference to the execution group is removed from the configuration repository.

Removing deployed children from an execution group

Removing deployed children from an execution group removes all message flows and message sets without deleting them from the development workspace.

To remove the deployed children from a broker:
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the broker domain to reveal the execution group with which you want to work.
3. Right-click the execution group, and click **Remove Deployed Children**.
4. Click **OK** at the prompt to confirm that you want to remove all message flows and message sets from the execution group.

   An automatic broker configuration deploy is immediately performed for the broker parent.

   A BIP08921 information message is displayed to show that the request was received by the Configuration Manager. Verify the results of the deployment by opening the Event Log.

Configuring the workbench

The following topics show you how to configure aspects of the workbench. A minimum display resolution of at least 1024 x 768 is required for some dialogs, such as the Preferences dialog.
- "Changing Broker Administration preferences"
- "Changing palette preferences" on page 158
- "Configuring CVS to run with the Message Brokers Toolkit" on page 159

Changing Broker Administration preferences

You can change the following Broker Administration preferences:
- Show empty projects in the Navigator views.
- Show ACL restricted objects in the Navigator views.
- Warn before deleting alerts.
- Prompt to perform a topology deploy after changes. You can change this to never deploy, always a delta deploy, or always a complete deploy.
- Prompt to perform a topics deploy after changes. You can change this to never deploy, always a delta deploy, or always a complete deploy.

To change Broker Administration preferences:

1. Switch to the **Broker Administration perspective**.
2. Click **Window>Preferences**.
3. Expand the **Broker Administration** category in the left pane.
4. Make your selections.
5. Click **OK**.

You can also change Configuration Manager proxy and trace preferences. For more information see Changing trace settings from the workbench.

Changing palette preferences

You can change the palette preferences in the Message Brokers Toolkit . The changes that you can make are described in the following topics.

- "Changing the palette layout"
- "Changing the palette settings"
- "Customizing the palette" on page 159

Changing the palette layout

You can change the layout of the palette in the Message Flow editor and the Broker Topology editor using the Message Brokers Toolkit.

1. Right-click the palette to display the pop-up menu.
2. Click **Layout**.
3. Click one of the available views:

   Columns
   Displays named icons in one or more columns. Change the number of columns by clicking on the right edge of the palette and dragging.

   List Displays named icons in a single-column list. This is the default layout.

   Icons Only
   Displays a list of icons only.

   Details
   Displays a detailed list of icons.

Changing the palette settings

In Message Brokers Toolkit you can change the palette settings in the Message Flow editor and Broker Topology editor using the **Palette Settings** dialog.

1. Right-click the palette to display the pop-up menu.
2. Click **Settings**. The **Palette Settings** dialog opens.
3. Use the dialog to change the appropriate setting:
   - Click the **Change** button to change the font on the palette.
   - Click the **Restore Default** button to restore the default palette settings.
   - In the **Layout** list, tick the appropriate radio button to change the palette layout. (See "Changing the palette layout" for more information.)
   - Tick the **User large icons** check box to toggle between large and small icons in the palette.
   - In the **Drawer options** list, tick the appropriate radio button to change the way that drawers are handled in the palette. A drawer is a container for a

list of icons, such as the **Built-in nodes** drawer on the Message Flow editor's palette or the **Entity** drawer on the Broker Topology editor's palette.

Customizing the palette

In Message Brokers Toolkit you can customize the palette for the Message Flow editor using the **Customize Palette** dialog.

1. Right-click the palette to display the pop-up menu.
2. Click **Customize**. The **Customize Palette** dialog opens.
3. To change the order of entries and drawers in the palette, click the appropriate item in the list to highlight it, then click the **Move Down** or **Move Up** button.
4. To hide an entry or drawer, click the appropriate item in the list to highlight it, then tick the **Hide** check box.
5. To create a new separator, click **New → Separator**.
6. To create a new drawer:
   a. Click **New → Drawer**.
   b. Type a name and description for the drawer.
   c. If required, tick the **Open drawer at start-up** check box.
   d. If required, tick the **Pin drawer open at start-up** check box.

Configuring CVS to run with the Message Brokers Toolkit

Install CVS as a normal program by following the usual prompts. Not all versions of *CVSNT* are supported by Eclipse.

1. Configure CVS by carrying out the following tasks:
   a. Create a directory on your computer, for example, on Windows - c:\CVSRepository.
   b. Start the *CVSNT* control panel. Select `Start->Programs->CVSNT` to see the icon on the desktop.
   c. Stop both the `CVS Service` and the `CVS Lock Service`.
   d. Select the `Repositories` tag, click **Add** and create a new repository. Note that no entry appears on the screen the first time that you do this.
   e. Use the `...` button on the next window to select the directory that you created in step 1a and click **OK**. Note that when CVS has finished formatting its repository the backslash in the directory name is changed to a forward slash.
   f. Select the `Service Status` tab and restart both the `CVS Service` and the `CVS Lock Service`.
2. Enable the CVS Revision tag to be populated in the Eclipse Version fields in the Message Brokers Toolkit. To do this on Windows:
   a. Select `Window->Preferences`
   b. Expand the `Team` section and click **CVS**
   c. Use the drop down in the **Default keyword substitution:** field and set the value to `ASCII with keyword expansion(-kkv)`
3. Add the WebSphere Event Broker file types to the Eclipse CVS configuration. To do this:
   a. Select **File Content** in the `Team` section of the window you opened in step 2
   b. Click **Add** and add `msgflow` as an allowable file extension. Ensure that the value is set to ASCII.

c. Repeat the above procedure for the following file extensions that the broker uses:
   - esql
   - mset
   - mxsd

   If you use CVS to store other file types, for example, COBOL copybooks add the appropriate file types as well.

d. Click **OK** when you have finished.

Configuring a publish/subscribe topology

To configure a publish/subscribe topology:

1. Design and configure your broker domain.

   For further information refer to "Designing a broker domain" on page 49 and "Configuring broker domain components" on page 66

2. Define the topic trees that you require.

   For further information refer to Topics and "Adding a new topic" on page 174.

3. Decide which security options to use.

   For further information refer to "Publish/subscribe security" on page 33 and "Securing the publish/subscribe domain" on page 41.

Setting up the broker domain for publish/subscribe

Refer to the topics listed below.

Publish/subscribe topologies

A publish/subscribe topology is the brokers, the collectives, and the connections between them, that support publish/subscribe applications in the broker domain.

A publish/subscribe application can consist of a network of brokers connected together. The brokers can all be on the same physical system, or they can be distributed over several physical systems. By connecting brokers together, publications can be received by a client on any broker in the network.

This provides the following benefits:

- Client applications can communicate with a nearby broker rather than a distant one, thereby getting better response times.
- By using more than one broker, more subscribers can be supported.

Publications are sent only to brokers that have subscribers that have expressed an interest in the topics being published. This helps to optimize network traffic.

Broker networks: There are three ways of connecting brokers together to make a broker domain:

- Brokers can be simply joined together.
- Brokers can be grouped together into collectives, where a collective is a set of one or more brokers that are directly connected to each other.
- Collectives can be joined together; this is a combination of the previous two ways of grouping brokers together.

The following diagram shows a network of six collectives.
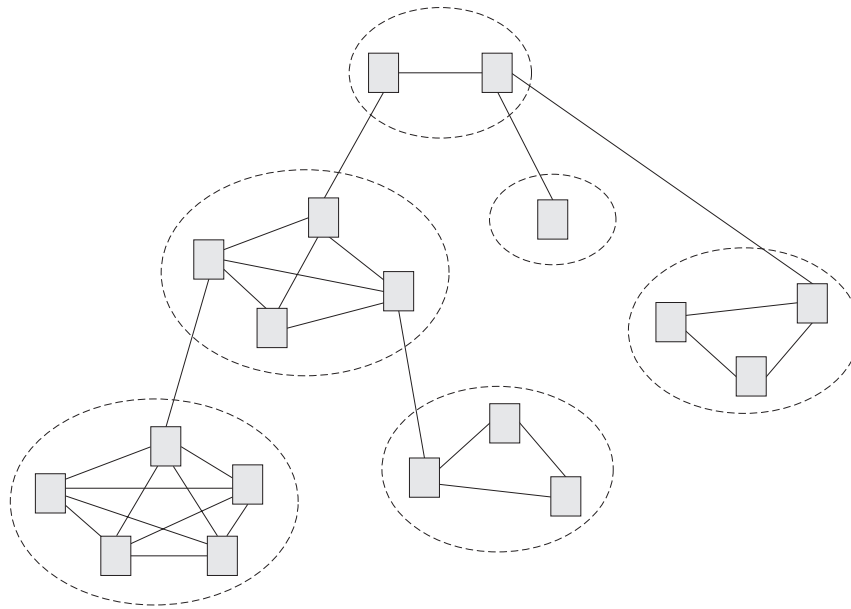
Collectives: A collective is a set of brokers that are fully interconnected and form part of a multi-broker network for publish/subscribe applications.

A broker cannot belong to more than one collective. Brokers within the same collective can exist on physically separate computers. However, a collective cannot span more than one broker domain.

Each pair of broker queue managers must be connected together by a pair of WebSphere MQ channels.

The following figure shows a simple collective of three brokers:
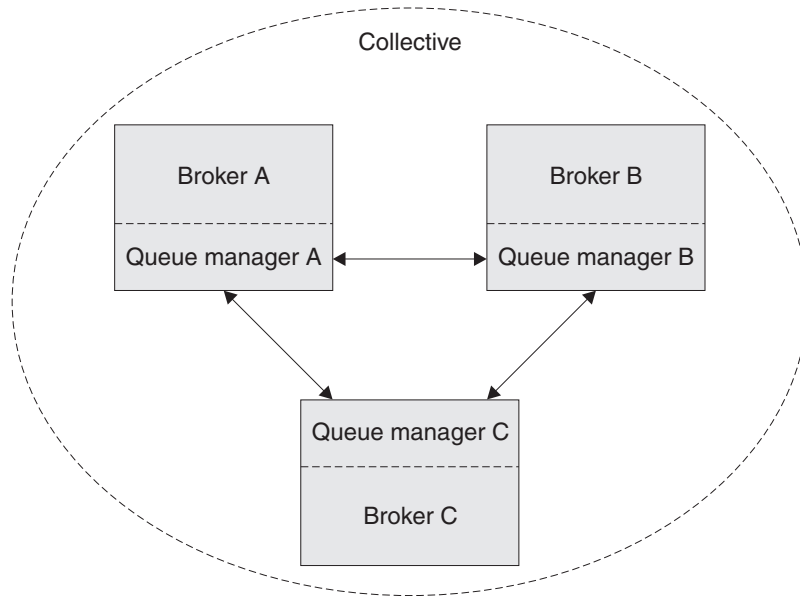
A collective provides the following benefits:

- Messages destined for a specific broker in the same collective are transported directly to that broker and do not need to pass through an intermediate broker. This improves broker performance and optimizes inter-broker publish/subscribe traffic, in comparison with a hierarchical tree configuration.
- If your clients are geographically dispersed, you can set up a collective in each location, and connect the collectives (by joining a single broker in each collective) to optimize the flow of publications and subscription registrations through the network.
- You can group clients according to the shared topics that they publish and to which they subscribe.

  Clients that share common topics can connect to brokers within a collective. The common publications are transported efficiently within the collective, because they pass through only brokers that have at least one client with an interest in those common topics.
- A client can connect to its nearest broker, to improve its own performance. The broker receives all messages that match the subscription registration of the client from all brokers within the collective.

  The performance of a client application is also improved for other services that are requested from this broker, or from this broker's queue manager. A client application can use both publish/subscribe and point-to-point messaging.
- The number of clients per broker can be reduced by adding more brokers to the collective to share workload within that collective.

When you create a collective, the workbench ensures that the connections that you make to other collectives and brokers are valid. You are prevented from making connections that would cause messages to cycle forever within the network. You are also prevented from creating a collective of brokers that does not have the required WebSphere MQ connections already defined.

The queue manager of each broker in a collective must connect to every other queue manager in the collective by a pair of WebSphere MQ channels.

Each broker in the collective maintains a list of its neighbors.

A neighbor can be one of the following:
- a broker in the same collective
- a broker outside its collective to which it has an explicit connection; that is, for which it is acting as a gateway

The complete list of neighboring brokers forms a broker's neighborhood.

Multicast publish/subscribe: In a publish/subscribe system there are client applications, some of which are publishers and some of which are subscribers, that are connected to a network of message brokers that receive publications on a number of topics, and send the publications on to the subscribers for those topics.

Normally, a separate message is sent to each subscriber of a publication. However, with multicast, regardless of how many subscribers to a topic there are on a subnet, only one message is sent. This improves network utilization.

The more subscribers there are in your publish/subscribe system, the greater the improvement to network utilization there might be if you use multicast.

The subscriber must be a JMS client if you want to use Multicast publish/subscribe.

To use multicast, you must change some of the properties of the broker. Some of these properties can be defined as applying to specific topics, but some properties apply to all Multicast messages controlled by that broker.

For each topic you can define whether the topic can be multicast, and the IP address to which Multicast messages are sent.

You can also change those properties in the broker that define, for example, the following:
- The multicast protocol type
- The port that is used for Multicast messages
- A 'Time To Live (TTL)' setting that determines how far from its source a Multicast packet can be sent
- The size of a Multicast packet
- Whether there is a maximum transmission rate and, if there is, its value
- What interface to use for Multicast transmissions

These properties apply to all Multicast messages.

Cloned brokers: A cloned broker is a broker for which you have defined one or more clones; the subscription table of a cloned broker is replicated to all other brokers with which it is cloned.

When a subscriber requests a subscription from a cloned broker, the subscription is also sent to each of the clones of that broker.

Use cloned brokers to improve the availability of your publish/subscribe system. By defining cloned brokers on different computers, you make sure that a publication is delivered to a subscriber even when one of the computers is unavailable.

The diagram shows what happens when Subscriber 1 sends a subscription to Broker 1, but Broker 1 becomes unavailable; because Broker 1 and Broker 2 have been defined as clones, the subscription is redirected to Broker 2 and Subscriber 1 gets the publication from Broker 2.
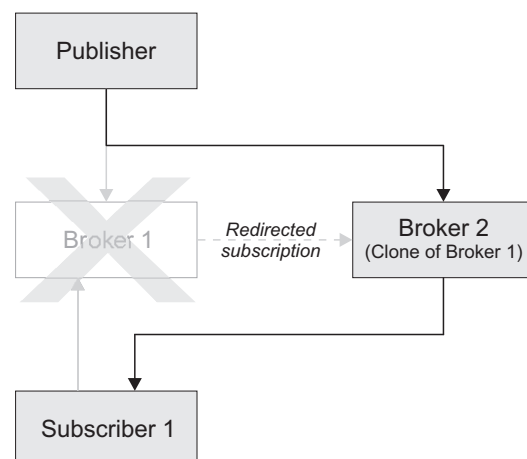
If two brokers are clones within a collective, duplicate messages might be sent to subscribers registered with brokers inside that collective.

mqsichangeproperties command: The following parameters on the mqsichangeproperties command support cloned brokers:

clonedPubSubBrokerList
> This is a set of two pairs. The first item in each pair is the name of a broker, and the second item in each pair is the name of that broker's queue manager.
> - Value type — String
> - Initial value — null

Migrated topologies: If you have a WebSphere MQ Publish/Subscribe broker network, you can continue to use this network unchanged. The introduction of WebSphere Event Broker to your environment, and the creation of brokers in that broker domain, does not affect your WebSphere MQ Publish/Subscribe broker domain until you take specific action to connect the two networks.

If you want to have two separate, independent networks, you do not have to take any specific actions. You can retain your existing WebSphere MQ Publish/Subscribe network, and install and configure a WebSphere Event Broker network, without any interaction.

Heterogeneous networks: A heterogeneous network is a network of brokers, some of which form a WebSphere MQ Publish/Subscribe network and some of which belong to the WebSphere Event Broker product.

With the WebSphere Event Broker product, there are two ways in which a broker can be joined to the WebSphere MQ Publish/Subscribe network; it can be joined either as a leaf node, or as a parent node.

Leaf node: When a broker is joined as a leaf node, it is joined as a child broker of another broker in the WebSphere MQ Publish/Subscribe network.

Adding the broker as a leaf node rather than as a parent node causes the new broker to receive only some of the WebSphere MQ Publish/Subscribe message traffic that is directed to the brokers for which this new broker is a child broker.

Parent node: When a broker is joined as a parent node, it is joined as a parent broker of one or more brokers in the WebSphere MQ Publish/Subscribe network.

Adding the broker as a parent node rather than as a leaf node causes the new broker to receive all the WebSphere MQ Publish/Subscribe message traffic that is directed to the child brokers for which this new broker is the parent broker.

Changing Broker Topology editor properties

After you have launched the Broker Topology editor in the editor area, you can change or remove the default background image displayed in the editor area.

The following steps show you how to change properties.
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain to display its contents.
3. Double-click **Broker Topology** to launch the Broker Topology editor.

4. Right-click the editor, then click **Properties** to display the Broker Topology editor properties.

5. On the **Editor** page, you can change the background image file, and modify its scale factor in a range of 1 to 5. The default value is 3. Alternatively, you can choose not to display a background image.

6. Optional: On the Description page, provide a description for the background image file.

7. Click **OK** to save your changes and close the Properties dialog.

Any changes you made to the background image are displayed when the Properties dialog closes.

> **Related reference**
>
> Broker Topology editor
>
> Broker Administration perspective

Connecting brokers in a collective

A collective is a set of brokers that are fully interconnected and form part of a multi-broker network for publish/subscribe applications.

You connect brokers in a collective by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.CollectiveProxy.

To connect brokers in a collective:

1. Define the WebSphere MQ channels between the queue managers of each pair of the brokers in the collective; use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer).

2. Assign the brokers as members of the collective using the Broker Topology editor in the workbench; they do not have to be connected together using the connect function.

Tip: Compare the use of collectives with the use of WebSphere MQ cluster queues, as described in Developing applications using the CMP.

Deleting a collective

You can delete a collective by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.CollectiveProxy.

To delete a collective:

1. Switch to the Broker Administration perspective.

2. In the Domains view, expand the appropriate broker domain.

3. Double click the Topology item to open the Broker Topology editor.

4. Right click the collective to be deleted and select Delete, or select the collective to be deleted and click the Delete key, or select Delete from Edit menu.

   The collective is deleted locally, but the delete operation is not completed until you save or close the editor.

Connecting a broker to a collective

You can connect a broker to a collective by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.CollectiveProxy.

To connect a broker to a collective:
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double click the Topology item to open the Broker Topology editor.
4. In the Broker Topology editor, click on Connection tool.
5. Click the broker to be connected and then click the collective that you want to connect the broker to.

   The connection is added locally, but the connection is only effective after you have saved, or closed the editor.

Removing a broker from a collective

You can remove a broker from a collective by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.CollectiveProxy.

The following steps show you how to remove a broker from a collective.
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double click the Topology item to open the Broker Topology Editor.
4. Right click the connection that you want to delete and select Delete.

Setting up a multicast broker

You can set up a multicast broker by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.BrokerProxy.MulticastParameterSet.

To make a broker capable of handling multicast requests, do the following tasks from within the workbench:
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double-click on the Topology item to open the Broker Topology editor.
4. In the Broker Topology editor, right-click the broker that you want to modify, and select **Properties**.
5. In the left panel of the properties window, select **Multicast**.
6. Select the **Multicast Enabled** check box.
7. Optional: Modify the properties that are listed below; any properties that are not modified take the default value.

Protocol Type

The multicast protocol type.

Valid values are "PTL", "PGM/IP", and "UDP encapsulated PGM". The default value is "PTL".

See "Multicast protocol types" on page 171 for an explanation of these multicast protocol types.

Min Address

The lowest IP address that the broker can use for its multicast transmissions.

This must be in the range 224.0.0.0 through 239.255.255.255. The default value is 224.0.0.0.

Max Address

The highest IP address that the broker can use for its multicast transmissions.

This must be in the range 224.0.0.0 through 239.255.255.255, and must not be lower than the value of Min Address. The default value is 239.255.255.255.

Data Port

The UDP data port through which multicast packets are sent and received.

The default value is 34343.

Broker Packet Size

The size, in bytes, of multicast packets.

This can be in the range 500 through 32000. The default value is 7000.

Broker Heartbeat Timeout

The broker sends a control packet periodically, approximately every second, to each client. This packet is used to send various control information, and to keep the heartbeat. The heartbeat timeout value is made known to the clients to help the clients detect a transmitter or network failure. If a control packet does not arrive within a number, defined as twice the value specified by this parameter, of seconds of the previous control packet's arrival, a client can suspect that there has been a transmitter failure or a network failure.

The default value is 20.

Broker Multicast TTL

The maximum number of hops that a multicast packet can make between the client and the broker. This value is one more than the maximum number of routers that there can be between the client and the broker.

The default value is 1, which means that the multicast packet must remain local to its originator and does not pass through any routers. The maximum value is 255.

Broker Network Interface

The name of the network interface over which multicast packets are transmitted. This is only relevant when the broker is running on a host with more than one network interface.

This can be a host name or an IP address. The default is 'None'. If the default value is chosen, the network interface that is used is operating system dependent.

Overlapping Multicast Topic Behavior

Choose Accept, Reject, or Revert.

The Overlapping Multicast Topic Behavior property controls the behavior of the broker when a client requests a multicast subscription for a topic that is part of a topic hierarchy containing topics that are explicitly disabled for multicast.

For example, consider a topic hierarchy where multicast is a topic with two children, *foo* that is enabled for multicast, and *bar* that is not enabled for multicast.

The three possible settings are:

Accept

A matching multicast subscription is accepted and all publications matching the topic, except those that are specifically excluded, are multicast. In the example shown above, a multicast subscription to *multicast/#* receives messages published on *foo* over multicast, but does not receive any messages published on *bar*.

Reject A multicast subscription to a topic with children that are disabled for Multicast is rejected by the broker. Subscriptions to *multicast/#* are rejected.

Revert Subscriptions to a topic that is disabled for multicast, or has children that are disabled for multicast, result in unicast transmission. A multicast subscription to *multicast/#* receives messages published on *foo* and *bar*, but the messages are sent unicast rather than multicast.

The default value is *Accept*.

Maximum Key Age

The maximum age, in minutes, of a topic encryption key before it must be redefined.

The default value is 360.

8. Optional: Click the + next to Multicast and click **Advanced**. You can now modify the following additional parameters:

Broker Transmission Rate Limit Activation

Use the Broker Transmission Rate Limit Activation property in conjunction with Broker Transmission Rate Limit Value to control network congestion. Choose one of the following values from the drop-down menu:

Disabled

Multicast data is transmitted as fast as possible. If the rate at which messages are submitted to be multicast exceeds the machine or network limits (that is, the speed of Ethernet or the host CPU becomes the bottleneck), these limits define the maximum transmission rate, and message submissions are stopped until all previously submitted messages have been sent.

Static   The transmission rate is limited by the value that is specified in Broker Transmission Rate Limit Value.

Dynamic
The limit on the transmission rate can vary during run time, depending on congestion conditions and data losses reported by clients. But the rate never exceeds the Broker Transmission Rate Limit Value.

The default is *Disabled*. If you choose *Static*, you can also choose a value for the parameter Broker Transmission Rate Limit Value.

Broker Transmission Rate Limit Value
This limits the overall transmission rate, in kilobits per second, of multicast packets. This parameter is effective only if the Broker Transmission Rate Limit Activation property is *Static*. This parameter must not exceed the capabilities of the machine or network.

This value can be in the range 10 through 1,000,000.

Client NACK Back Off Time
The maximum time, in milliseconds, that a client listens for another's NACKs before sending its own NACK.

This value can be in the range 0 through 1000. The default value is 100.

Client NACK Check Period
The time, in milliseconds, between periodic checks of reception status and sequence gap detection for NACK building.

This value can be in the range 10 through 1000. The default value is 300.

Client Packet Buffer Number
The number of memory buffers that are created at startup for packet reception. Having a high number of buffers available improves the reception performance and minimizes packet loss at high delivery rates, but requires increased memory use. Each buffer is 33 KB; having 500 buffers (the default value) uses approximately 15 MB of main memory.

If memory use is important, try using different values for this parameter and look at the effect on the overall performance of your application when transmission rates are high.

This value can be in the range 1 through 5000. The default value is 500.

Client Socket Buffer Size
The size, in kilobytes, of the client's socket receiver buffer. Increasing this value reduces the number of data packets that might be dropped by the client receiver.

This value can be in the range 65 through 10000. The default value is 3000.

Broker History Cleaning Time
The time, in seconds, that is defined for cleaning the retransmission buffer.

This value can be in the range 1 through 20. The default value is 7.

Note: This property is not used in Version 6.

Broker Minimal History Size

The minimum size, in kilobytes, of a buffer that is allocated as an archive for all transmitted packets. This buffer is shared by all reliable topics, and can be used to recover lost packets.

This value can be in the range 1000 through 1,000,000. The default value is 60,000.

Broker NACK Accumulation Time

The time, in milliseconds, that NACKs are aggregated in the broker before recovered packets are sent.

This value can be in the range 50 through 1000. The default value is 500.

Maximum Client Memory Size

The maximum amount of memory, in kilobytes, that can be used by reception buffers in the client.

This parameter is applicable only to PGM multicast protocols. The default value is 262,144 which represents 256 MB.

9. Click OK.

10. Restart the broker; you must do this for the changes that you have made to take affect.

Before you can use multicast, you must define some topics as capable of being multicast.

The recommended way of changing the broker's multicast configuration is to use the workbench. However, you can also use the command **mqsichangeproperties** to change the broker's properties.

The following table relates the properties described above to the corresponding names of the parameters on the **mqsichangeproperties** command that support multicast. Full details of the **mqsichangeproperties** command is in "mqsichangeproperties command" on page 236.

Property name	mqsichangeproperties parameter
Multicast Enabled	multicastEnabled
Protocol Type	multicastProtocolType
Min Address	multicastAddressRangeMin
Max Address	multicastAddressRangeMax
Data Port	multicastDataPort
Broker Packet Size	multicastPacketSizeBytes
Broker Heartbeat Timeout	multicastHeartbeatTimeoutSec
Broker Multicast TTL	multicastMCastSocketTTL
Broker Network Interface	multicastMulticastInterface
Overlapping Multicast Topic Behavior	multicastOverlappingTopicBehavior
Maximum Key Age	multicastMaxKeyAge
Broker Transmission Rate Limit Activation	multicastLimitTransRate
Broker Transmission Rate Limit Value	multicastTransRateLimitKbps
Client NACK Back Off Time	multicastBackoffTimeMillis

Property name	mqsichangeproperties parameter
Client NACK Check Period	multicastNackCheckPeriodMillis
Client Packet Buffer Number	multicastPacketBuffers
Client Socket Buffer Size	multicastSocketBufferSizeKbytes
Broker History Cleaning Time (deprecated in V6)	N/A
Broker Minimal History Size	multicastMinimalHistoryKBytes
Broker NACK Accumulation Time	multicastNackAccumulationTimeMillis
Maximum Client Memory Size,	multicastMaxMemoryAllowedKBytes

To enable multicast for the broker WBRK_BROKER use the following command:

```
mqsichangeproperties WBRK_BROKER -o DynamicSubscriptionEngine -n multicastEnabled -v true
```

This enables the broker for multicast, but does not change any other properties of the broker.

To enable multicast for the broker WBRK_BROKER, and to restrict the transmission rate to 50,000 kilobits per second, use the following command:

```
mqsichangeproperties WBRK_BROKER -o DynamicSubscriptionEngine -n multicastEnabled,
  multicastLimitTransRate,multicastTransRateLimitKbps -v true,Static,50000
```

None of the other properties of the broker are changed.

Note the use of commas to separate the properties that are being changed, and also their values.

For the changes to be effective, you must restart the broker.

Warning: Any changes to the broker configuration made using **mqsichangeproperties** are overwritten with the configuration that is held in the Configuration Manager whenever the broker configuration is deployed.

Multicast protocol types: IBM® WebSphere® Event Broker supports three different types of multicast protocol. These are:
* PTL (Packet Transfer Layer)
* PGM/IP
* PGM UDP encapsulated

PTL provides backward compatibility with Version 5, where it is the only multicast protocol that is supported. But for new multicast deployments, either of the two PGM multicast protocols is recommended.

The broker supports two implementations of the PGM multicast protocol, PGM/IP and PGM UDP encapsulated. Which of the two PGM protocol types that you should choose, depends on the complexity of your network topology.

If your network topology consists of two or more subnets with many receiver clients in each subnet, use PGM/IP. PGM/IP takes advantage of PGM router assist support.

For a simpler network topology, use the PGM UDP encapsulated implementation which does not use PGM router assist.

Important: To use PGM/IP, both the broker and the client applications must run with superuser authority. Because of the security risks that are associated with running with superuser authority, it is recommended that no other work should be run on the broker.

Making topics multicast:

To make individual topics, or groups of topics, capable of being multicast you need to make changes to the topic hierarchy:

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double-click on the Topics item to open the Topics Hierarchy editor.
4. In the Topics Hierarchy editor, right-click the topic, or group of topics, that you want to make capable of being multicast, and select properties.
5. In the left panel of the properties window, select **Multicast**.
6. Choose the **Multicast Enabled** required.

   For the topic root, the choice is either **Enabled** or **Disabled**. The default is **Disabled**.

   For a child topic root, the choice can be **Inherit**, **Enabled**, or **Disabled**. The default is **Inherit**.
7. Check the **Automatic Multicast Address** box, or type in the name of the **MC Group Address**.
8. Choose the **Quality of Service** required. The choice is between **Reliable** or **Unreliable**. The default is **Reliable**.
9. Optional: Select the **Encrypted** check box.
10. Click OK.

Handling high-volume publish/subscribe activity on z/OS:

Brokers that handle large numbers of retained subscriptions or publications can use up all the IRLM storage that is allocated by default for DB2 locks. This might cause problems when you try to restart the broker.

The following actions might help stop this happening.
1. Tune the publish/subscribe topology:
   a. Balance execution groups across more brokers; this means that fewer execution groups need to start at the same time and have concurrent locks for the same DB2 subsystem.
   b. Put the brokers in publish/subscribe collectives; this reduces the number of subscriptions in a single broker table and reduces the amount of concurrent access to DB2. See "Publish/subscribe topologies" on page 160 for more information about this.
2. Increase the IRLM storage that is available:
   a. Set the value of MAXCSA so high that the ECSA that is required by the IRLM never reaches this value. Because IRLM gets storage only when it needs it, choose a value that is higher than you expect IRLM to need.
   b. If you are unable to choose a value of MAXCSA sufficiently high that it cannot be exceeded by the ECSA that is required by the IRLM, use the option `PC=YES` on the START irlmproc command. This causes the IRLM to place in its private address space the control block structures that relate to locking. There is more information about this in the DB2 Redbook *DB2 UDB for OS/390 Version 7 Performance Topics, SG24-5351.*

Note: There might be a slight (approximately 1 to 2 percent) performance degradation when you run with `PC=YES`. See *DB2 Universal Database for OS/390 and z/OS Version 7 Administration Guide, SC26-9931* for more information.

Related tasks

Resolving problems when using publish/subscribe

Setting up cloned brokers

Each broker that is to be cloned with other brokers must be told which brokers are to be its clones.

To set up three brokers (broker1 with queue manager QM1, broker2 with queue manager QM2, and broker3 with queue manager QM3) to be clones of each other, as shown in the diagram below, use the mqsichangeproperties command for each of the brokers:

1. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"broker2,QM2,broker3,QM3\"`

2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"broker1,QM1,broker3,QM3\"`

3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"broker1,QM1,broker2,QM2\"`
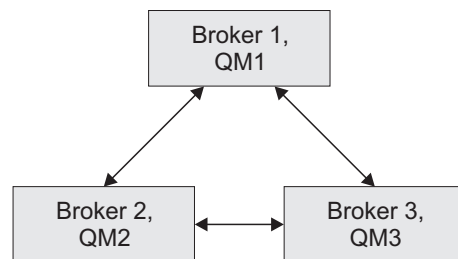
Adding a cloned broker

To add a broker to a set of cloned brokers, use the **mqsichangeproperties** command to define the brokers that are its clones, and to tell each of the other brokers that it has a new clone.

To add broker4 (with queue manager QM4) to a set of three cloned brokers (broker1 with queue manager QM1, broker2 with queue manager QM2, and broker3 with queue manager QM3), use the following **mqsichangeproperties** commands:

1. `mqsichangeproperties broker4 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"broker1,QM1,broker2,QM2,broker3,QM3\"`

2. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"+broker4,QM4\"`

3. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"+broker4,QM4\"`

4. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"+broker4,QM4\"`

Deleting a cloned broker

To delete a broker from a set of cloned brokers, use the **mqsichangeproperties** command to delete the brokers that were its clones, and to tell each of the other brokers that one of its clones has been deleted.

To delete broker3 from a set of three cloned brokers (broker1 with queue manager QM1, broker2 with queue manager QM2, and broker3 with queue manager QM3), as shown in the diagram below, use the following **mqsichangeproperties** commands:

1. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"-broker3\"`

2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"-broker3\"`

3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -nclonedPubSubBrokerList -v \"\"`
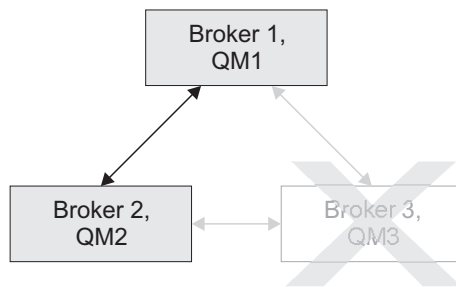
Operating a publish/subscribe domain

After you have set up your publish/subscribe broker domain, you might want to create or delete topics, or view the current status of your subscriptions.

For information about how to do this, refer to the following topics:
- "Adding a new topic"
- "Deleting a topic" on page 175
- "Querying subscriptions" on page 175

Adding a new topic

You can define a new topic explicitly by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.TopicProxy.

You can define a new topic implicitly by sending to the message broker a Publish command that specifies the new topic. However, to define a new topic explicitly:

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double-click on the Topics item to open the Topics Hierarchy editor.
4. Right-click **Topics** in the topics hierarchy displayed by the Topics Hierarchy Editor.

5. From the menu shown, click **Create Topic**; a topic window opens that shows the topic hierarchy.
6. In the topic hierarchy, select the topic that you want to be the parent topic of the topic that you are creating. In the lower pane of the topic window, type the name of your new topic.
7. Click **Next**; the next wizard page opens. The pane on the left of this window shows all the principals (groups and users) that are defined.
8. Select the groups and users that you want to relate to your new topic and click the > icon between the two panes of the window; the pane on the right of the window is updated with the groups and users that you have chosen.
9. For each principal selected in the right-hand pane, you can set **Publish**, **Subscribe**, and **Persistent** attributes by choosing a value from the corresponding list.

   By selecting more than one principal, you can choose values for a set of principals.
10. Click **Finish** to insert the topic into the topic hierarchy and update the access control list (ACL) for the topic. The ACL is in a table with four columns that are entitled Principal, Publish, Subscribe, and Persistent. The rows of the table show the properties of each principal that is relevant to the topic.

    The topic is created locally, but the change is not effective until you have saved or closed the editor.

    When saving or closing the editor, you might be prompted to deploy the new topics hierarchy or the deployment might be automatic, depending on the **Perform topics deploy after change** preference.

Deleting a topic

You can delete a topic by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.TopicProxy.

To delete a topic:
1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double click **Topics** to open the Topics Hierarchy Editor.
4. In the Topics Hierarchy editor, right-click the topic that you want to delete, and select **Delete**; alternatively, select the topic that you want to delete and press the Delete key, or select Delete from the Edit menu.

   The topic is deleted locally, but the delete is not effective until you do a save.

Querying subscriptions

You can query a topic by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.TopicProxy.

To query a subscription:
1. Switch to the Broker Administration perspective.

2. In the **Domains** view, click **Subscriptions** from the list of domain objects shown; the Subscriptions Query Editor opens in the workbench.

   You can also open the editor by double-clicking the **Subscriptions** item in the tree, or by right-clicking the **Subscriptions** item and clicking **Open**, or by clicking the **Subscriptions** item and clicking **Enter**.
3. Fill in the fields that are required to generate your subscriptions query.

   To generate your query, you might not need to fill in all the fields shown.
4. Click **Query**. The results of your query are displayed in the lower part of the edit window.

Changing locales

You can change the locale for the system on which a runtime component is installed.

The way in which you do this depends on the operating system:
- "Changing your locale on UNIX and Linux systems"
- "Changing your locale on Windows" on page 178
- "Changing your locale on z/OS" on page 179

WebSphere Event Broker uses code page converters to support character sets from different environments. "Code page converters" on page 179 describes what a code page converter is, and how to generate new converters.

Changing your locale on UNIX and Linux systems

You can change your system locale on UNIX and Linux systems.

There are many environment variables that control the locale. These can be defined system-wide, in your environment, or on a per-session basis:

LC_ALL
>   Overrides all `LC_*` environment variables with the given value

LC_CTYPE
>   Character classification and case conversion

LC_COLLATE
>   Collation (sort) order

LC_TIME
>   Date and time formats

LC_NUMERIC
>   Non-monetary numeric formats

LC_MONETARY
>   Monetary formats

LC_MESSAGES
>   Formats of informative and diagnostic messages, and of interactive responses

LC_PAPER
>   Paper size

LC_NAME
>   Name formats

LC_ADDRESS
> Address formats and location information

LC_TELEPHONE
> Telephone number formats

LC_MEASUREMENT
> Measurement units (Metric or Other)

LC_IDENTIFICATION
> Metadata about the locale information

LANG
> The default value. This is used when either LC_ALL is not set, or an applicable value for LC_* is not set

NLSPATH
> Delimited list of paths to search for message catalogues

TZ   Time zone

`LC_MESSAGES` and `NLSPATH` are the most important variables to the broker. These variables define the language and location of response messages that the broker uses. The broker profile file, mqsiprofile, sets `NLSPATH`. Either you, or your system must set `LC_MESSAGES`. The value set in `LC_MESSAGES` must be a value that the broker recognizes. `LC_CTYPE` is also important to the broker because it defines the character conversion that the broker performs when interacting with the local environment.

If you use common desktop environment (CDE), use this to set the locale instead of setting `LANG` and `LC_ALL` directly. The `NLSPATH` variable respects either method. Before setting the code page, check that it is one of the Supported code pages.

For example, to set WebSphere Event Broker to run in a UTF-8 environment set the following values in the profile:

```
LANG=en_US.utf-8
LC_ALL=en_US.utf-8
```

Where `en_US` sets the language, and `utf-8` sets the code page.

You can use the executable `locale` to show your current locale. The command `locale -a` displays all the locales currently installed on the machine. Make sure that the locale you select for `LANG` and `LC_ALL` is in the list that `locale -a` returns. The values that `locale` uses and returns are case-sensitive, so copy them exactly when assigning them to an environment variable.

When you start a broker component, the locale of that component is inherited from the shell in which it is started. The broker component uses the `LC_MESSAGES` environment variable as the search path in the `NLSPATH` environment variable (`LC_MESSAGES` is set when variable `LC_ALL` is exported).

Messages are sent to the syslog in the code page set by this locale. If you have multiple brokers that write to this syslog, their messages are in the code page of the locale in which they were started, for example:

locale	syslog code page	ccsid
pt_BR	iso8859-1	819
Pt_BR	ibm-850	850

locale	syslog code page	ccsid
PT_BR	utf-8	1208

Set the locale of the user ID that runs the syslog daemon to one that is compatible with the locales of all brokers that write to the syslog on that system, for example, utf-8. You can do this by setting the default locale. On Solaris, set the LANG and LC_ALL variables in /etc/default/init. On AIX and Linux, these variables are in /etc/environment. This task is not required on HP-UX.

For full time zone support in the broker, set the TZ variable using Continent/City notation. For example set TZ to Europe/London to make London, England the time zone, or set it to America/New_York to make New York, America the time zone.

If you want to add a new locale, refer to the operating system documentation for information on how to complete that task. If the code page of the new locale is not supported by WebSphere Event Broker you must add it by "Generating a new code page converter" on page 179.

Changing your locale on Windows

You can change your system locale on Windows.

The product components are started as services on Windows and are therefore influenced by the system locale. The command line functions are influenced by the user's locale. WebSphere Event Broker on Windows has all its locale information installed by default. However, you might need to install additional locale packages if prompted to do so by Windows.

To change locale, you can do one of the following:
• Install a locale-specific operating system.
• Alter the system or user locale by selecting *Regional Settings* in the Control Panel.

Messages are sent to the Event Log in the code page set by the current locale.

You can use the chcp command to change the active console code page. Enter the command at a command prompt; if you enter chcp without a parameter, it displays the current setting. If you enter it with a code page, it changes to that code page.

For example, to check the current code page setting:
```
C:\>chcp
Active code page: 437
```

The current page is displayed (437 is US-ASCII). If you want to change this to GB18030, enter:
```
C:\>chcp 54936
Active code page: 54936
```

Before using a code page, search for windows-*number* where *number* is the active code page you want to use in the list of Supported code pages. If the code page is not in the list, either use a code page that is in the list, or generate a new code page converter.

Changing your locale on z/OS

You can change your system locale on z/OS.

If you want to change your system locale on z/OS, set the `LANG`, `LC_ALL`, and `NLSPATH` variables. See "Installation information - Broker and User Name Server" on page 94 and "Installation information - Configuration Manager" on page 109 for further information.

The locale is set in the appropriate component profile (`BIPBPROF` for the broker, `BIPCPROF` for the Configuration Manager, `BIPUPROF` for the User Name Server) and you must run `BIPGEN` to create the component `ENVFILE`.

You can use the UNIX System Services (USS) executable `locale` to show your current locale. The command `locale -a` displays all the locales currently installed on the machine. Refer to the operating system documentation for information about adding new locales. If you add a new locale after you have installed WebSphere Event Broker, install that locale's message catalogs from the original install media.

You can set WebSphere Event Broker to operate with a specific code page. Set the code page after a period in the LANG and LC_ALL variable. This example sets the locale to `En_us` and the code page to `IBM-1140` (EBCDIC En_us with euro):

```
LANG=En_us.IBM-1140
LC_ALL=En_us.IBM-1140
```

Make sure that the selected code page is one of the Supported code pages. If the code page is not in the list, either use a code page that is in the list, or generate a new code page converter.

Code page converters

WebSphere Event Broker performs string operations in Universal Character Set coded in 2 octets (UCS-2). If incoming strings are not encoded in UCS-2, they must be converted to UCS-2. The broker uses international components for Unicode (ICU) code page converters to do this. The Unicode Consortium has further information on Unicode.

A code page converter is a mapping from the byte sequence in one code page to a serialized representation of UCS-2, known as UCS Transformation Format 16 bit form (UTF-16). A code page converter allows the broker to create a UCS-2 representation of an incoming string.

An example of the use of code page converter is:
- A message comes in on a queue from z/OS, with the MQ CCSID field set to 1047 (LATIN-1 Open Systems without euro). The broker looks up ibm-1047 and uses the resulting converter to create a UCS-2 representation for internal use.

WebSphere Message Broker currently supports the code pages listed in Supported code pages. If you need support for an additional code page, or if you require a different variant of a code page, you can extend the broker to support this code page.

Generating a new code page converter

Before you start:

- "Code page converters" on page 179 provides information about what a code page converter is, and about the code pages that WebSphere Event Broker supports.

If you need to support a code page that is not in the default set of code pages that WebSphere Event Broker supports, you can generate a new code page converter.

1. Create or find a mapping data file with the file extension .ucm for the converter you require. You can download many .ucm files from the ICU Character set mapping files archive. These mapping data files are available and modifiable without restriction. An example mapping data file is ibm-1284_P100-1996.ucm.

2. Rename the .ucm to a file name with the format ibm-*number*.ucm where *number* is a number you choose to identify the code page. Make sure this number is not already used in one of the Supported code pages. An example of this is renaming ibm-1284_P100-1996.ucm to ibm-1284.ucm.

3. Go to ICU downloads and download the binary distribution for your platform. An exact match is not important as long as the binaries can run. If you have problems building the converter, see the ICU user guide.

4. Extract the files from the binary distribution archive in to a temporary directory.

5. Copy the library and binary files to a directory within the environment path and libpath. Alternatively, copy the library and binary files to directory that is not temporary and modify the environment path and libpath to include this directory.

6. Use the makeconv tool from the extracted files to convert the mapping data file (.ucm files) into a binary converter file (.cnv file). Enter the following command to do this:

   ```
   makeconv -p ICUDATA mapping_file.ucm
   ```

   where

   mapping_file.ucm
   : is the mapping data file that you are using.

   The name of the binary converter file that makeconv produces is:

   ```
   icudt32<platform-suffix>_<mapping_file>.cnv
   ```

   where

   <platform-suffix>
   : is one of the following:
      - *l* for little-endian ASCII platforms
      - *b* for big-endian ASCII platforms
      - *e* for big-endian ASCII platforms

   <mapping_file>
   : is the name of the mapping data file that was converted.

   To make the .cnv file for ibm-1284.ucm, use the following command:

   ```
   makeconv -p ICUDATA ibm-1284.ucm
   ```

7. Copy the file with the file extension .cnv for the code page you need into a directory that WebSphere Event Broker can access. On UNIX an example directory is /var/mqsi/converters.

8. Associate the broker with the code page converter by entering the name of the *directory* where the converter is stored:
   - To create a new broker associated with the converter enter the following at a command prompt: mqsicreatebroker -c *directory*

- To alter an existing broker to recognize the converter enter the following at a command prompt: `mqsichangebroker -c directory`
- To affect all the products and the broker command-line tools using ICU, add the *directory* to the ICU_DATA environment variable. If you have already used either the mqsicreatebroker command or the mqsichangebroker to specify the code page converter to be used, the broker ignores the ICU_DATA value.

Part 3. Administering the broker domain

Administering the broker domain

Before you start:

You must complete the following task:
- "Configuring the broker domain" on page 49

Administering the broker domain includes the tasks that you operate frequently to activate and run your broker domain. These are listed below:
- "Connecting to and disconnecting from the broker domain"
- "Starting and stopping message flows" on page 186
- "Starting and stopping a broker" on page 187
- "Starting and stopping a Configuration Manager" on page 188
- "Starting and stopping the User Name Server" on page 190
- "Starting a WebSphere MQ queue manager as a Windows service" on page 191
- "Stopping a WebSphere MQ queue manager when you stop a component" on page 192
- "Viewing broker domain log information" on page 192
- "Refreshing broker domain log information" on page 193
- "Filtering broker domain log information" on page 194
- "Saving broker domain log information" on page 195
- "Clearing broker domain log information" on page 195
- "Changing Event Log editor preferences" on page 195
- "Backing up resources" on page 196

These tasks can be performed using one, or more, of the administrative techniques available with WebSphere Event Broker:
- The Message Brokers Toolkit
- The WebSphere Event Broker commands
- The Configuration Manager Proxy Java API

For each task, the administrative techniques that can be used are identified.

Connecting to and disconnecting from the broker domain

Before you start:

You must complete the following task:
- "Creating a domain connection" on page 147

You can connect to, and disconnect from, the broker domain by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy, see Connecting to the Configuration Manager using the Configuration Manager Proxy.

Use a domain connection to connect to the broker domain in the workbench.

The following steps show you how to connect to the broker domain and how to disconnect from the broker domain.

1. To connect to the broker domain:
    a. Switch to the Broker Administration perspective.
    b. In the Domains view, right-click the broker domain to which you want to connect, and click **Connect**. This starts the domain connection to the Configuration Manager.

       When connected, the workbench status line is changed (for example *WBRK_QM@localhost:1414* is connected). The Broker Topology and Topics are populated, and the broker domain and broker topology icons change to reflect the connected state.

       On successful connection, the Configuration Manager name is shown in the Domains view in the form `ConfigurationManagerName on QMgrName@Hostname:PortNumber`.

2. To disconnect from the broker domain:
    a. Switch to the Broker Administration perspective.
    b. In the Domains view, right click the broker domain from which you want to disconnect, and click **Disconnect**. The connection to the Configuration Manager is broken.

       When disconnected, the workbench status line is changed (for example *WBRK_QM@localhost:1414* is not connected). All brokers and topics are removed from the domains navigator tree, and the broker domain and broker topology icons change to reflect the disconnected state.

Starting and stopping message flows

You can start and stop a message flow by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy, see Navigating broker domains using the Configuration Manager Proxy.

From the workbench you can start and stop:
- All message flows in all execution groups, assigned to a specific broker.
- All message flows in a specific execution group.
- A single message flow.

1. To start a message flow:
    a. Switch to the Broker Administration perspective.
    b. In the Domains view, expand your broker domain to locate your message flow:
       - To start all message flows in all execution groups for a broker, right-click the broker and click **Start Message Flows**.
       - To start all message flows in a specific execution group, right click the execution group and click **Start Message Flows**.
       - To start a single message flow, right-click the message flow and click **Start**.

    The Configuration Manager sends a message to the broker to start the specified message flows.

A BIP0892I information message is displayed to show that the Configuration Manager has received the request.Verify the results of the deployment by opening the Event Log.

There might be a short delay for the Configuration Manager to respond.

The alert `Message Flow is not running` is removed from the Alert Viewer.

2. To stop a message flow:

   a. Switch to the Broker Administration perspective.

   b. In the Domains view, expand your broker domain to locate your message flow:

      - To stop all message flows in all execution groups for a broker, right-click the broker and click **Stop Message Flows**.

      - To stop all message flows in a specific execution group, right click the execution group and click **Stop Message Flows**.

      - To stop a single message flow, right-click the message flow, and click **Stop**.

   c. Open the Event Log for the broker domain. A BIP0892I information message is displayed to show that the Configuration Manager has received the request.

   There might be a short delay for the Configuration Manager to respond.

   The alert `Message Flow is not running` is added to the Alert Viewer.

Starting and stopping a broker

Before you start:

Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385

To start and stop a broker, use the **mqsistart** and **mqsistop** commands from the command line.

Follow the link for the appropriate platform.
- "Starting and stopping a broker on Linux and UNIX systems"
- "Starting and stopping a broker on Windows" on page 188
- "Starting and stopping a broker on z/OS" on page 188

Starting and stopping a broker on Linux and UNIX systems

The following steps show you how to start and stop a broker.

1. Run '. <install_dir>/bin/mqsiprofile' to source the `mqsiprofile` script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

2. To start a broker:

   a. Enter the following command on the command line:

      `mqsistart WBRK_BROKER`

      Substitute your own broker name for WBRK_BROKER.

      The broker and its associated queue manager are started. Check the syslog to ensure that the broker has initialized successfully.

3. To stop a broker Enter the following command on the command line:

   `mqsistop WBRK_BROKER`

Substitute your own broker name for WBRK_BROKER.

You can also request that the broker's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping a broker on Windows

The following steps show you how to start and stop a broker.

1. Run the `<install_dir>/bin/mqsiprofile` command to set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

2. To start a broker:
   a. Enter the following command on the command line:

      ```
      mqsistart WBRK_BROKER
      ```

      Substitute your own broker name for WBRK_BROKER. The broker and its associated queue manager are started. The command initiates the startup of the broker's Windows service. Check the Application Log of the Windows Event Viewer to ensure that the broker has initialized successfully.

      You can also request that the broker's queue manager is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service" on page 191.

3. To stop a broker enter the following command on the command line:

   ```
   mqsistop WBRK_BROKER
   ```

   Substitute your own broker name for WBRK_BROKER.

   You can also request that the broker's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping a broker on z/OS

The following steps show you how to start and stop a broker.

1. To start a broker:
   a. See "Starting the component" on page 102.

2. To stop a broker:
   a. Issue the following command:

      ```
      P <brokername>
      ```

Starting and stopping a Configuration Manager

Before you start:

Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385

To start and stop a Configuration Manager, use the **mqsistart** and **mqsistop** commands from the command line.

Follow the link for the appropriate platform.

- "Starting and stopping a Configuration Manager on Linux and UNIX systems" on page 189

Starting and stopping a Configuration Manager on Linux and UNIX systems

Use the **mqsistart** and **mqsistop** commands from the command line.

The following steps show you how to start and stop a Configuration Manager.

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

2. To start aConfiguration Manager:

   a. Enter the following command:

      mqsistart CMGR01

      Substitute your own Configuration Manager name for CMGR01. The Configuration Manager and its associated queue manager are started.

3. To stop a Configuration Manager enter the following command on the command line:

   mqsistop CMGR01

   Substitute your own Configuration Manager name for CMGR01.

   You can also request that the Configuration Manager's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping a Configuration Manager on Windows

Use the **mqsistart** and **mqsistop** commands from the command line.

The following steps show you how to start and stop aConfiguration Manager.

1. Run the <install_dir>/bin/mqsiprofile command to set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.

2. To start aConfiguration Manager:

   a. Enter the following command:

      mqsistart CMGR01

      Substitute your own Configuration Manager name for CMGR01. If you do not specify a name, the default of configmgr is used. The Configuration Manager and its associated queue manager are started. The command initiates the startup of the Configuration Manager's Windows service. Check the Application Log of the Windows Event Viewer to ensure that the Configuration Manager has initialized successfully.

      You can also request that the Configuration Manager's queue manager is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service" on page 191.

3. To stop a Configuration Manager enter the following command on the command line:

   mqsistop CMGR01

   Substitute your own Configuration Manager name for CMGR01. If you do not specify a name, the default of configmgr is used.

You can also request that the Configuration Manager's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping a Configuration Manager on z/OS

The following steps show you how to start and stop a Configuration Manager.
1. To start a Configuration Manager:
   a. See "Starting the Configuration Manager component" on page 113.
2. To stop a Configuration Manager:
   a. Issue the following command:
      ```
      P <Configuration Manager Name>
      ```

Starting and stopping the User Name Server

Before you start:

Ensure that your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 385

To start and stop a User Name Server use the **mqsistart** and **mqsistop** commands from the command line.

Follow the link for the appropriate platform.
* "Starting and stopping the User Name Server on Linux and UNIX systems"
* "Starting and stopping the User Name Server on Windows"
* "Starting and stopping the User Name Server on z/OS" on page 191

Starting and stopping the User Name Server on Linux and UNIX systems

1. Run '. <install_dir>/bin/mqsiprofile' to source the mqsiprofile script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
2. To start a User Name Server enter the following command on the command line:mqsistart UserNameServer. The User Name Server and its associated queue manager are started. Check the syslog to ensure that the User Name Server has initialized successfully.
3. To stop a User Name Server enter the following command on the command line: mqsistop UserNameServer . You can also request that the User Name Server's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping the User Name Server on Windows

The following steps show you how to start and stop a User Name Server.
1. Run the <install_dir>/bin/mqsiprofile command to set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Event Broker commands.
2. To start a User Name Server:
   a. Enter the following command on the command line:

```
mqsistart usernameserver
```

The User Name Server and its associated queue manager are started. The command initiates the startup of the User Name Server's Windows service. Check the Application Log of the Windows Event Viewer to ensure that the User Name Server has initialized successfully.

You can also request that the User Name Server's queue manager is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service."

3. To stop a User Name Server enter the following command on the command line:

```
mqsistop usernameserver
```

You can also request that the User Name Server's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 192.

Starting and stopping the User Name Server on z/OS

The following steps show you how to start and stop a User Name Server.

1. To start a User Name Server, see "Starting the component" on page 123.
2. To stop a User Name Server use the command:

```
P <User Name Server name>
```

Starting a WebSphere MQ queue manager as a Windows service

Before you start:

You must complete the following task:

- Stop the queue manager for the WebSphere Event Broker component, using the **endmqm** command or WebSphere MQ Services.

When you start the Configuration Manager, a broker, or User Name Server, the **mqsistart** command starts the associated queue manager if it is not already running. When you start any of these WebSphere Event Broker components on Windows, they start as a service on Windows, but the associated queue manager does not. You can change the properties of the queue manager service to set startup type to automatic to enable the queue manager to run as a Windows service.

This change ensures that the operation of the queue manager is independent of the logged-on status of the user that starts theWebSphere Event Broker component.

To start a WebSphere MQ queue manager as a Windows service.

1. Click **Start** → **Programs** → **IBM** → **IBM WebSphere MQ** → **WebSphere MQ Services**.
2. Right-click the queue manager and select **Properties**, and the **General** tab.
3. Change the **Startup Type** to *Automatic*.
4. This setting ensures that the queue manager is started whenever the WebSphere MQ Service (a Windows service) is started.
5. You are also recommended to change the properties of the WebSphere MQ Services service by updating its **Startup Type** to *Automatic* using the Control

Panel. This setting starts WebSphere MQ Services when Windows itself starts. This isolates the operation of the WebSphere MQ Services service from any logged on user.

6. Restart the queue manager for the WebSphere Event Broker component using the **strmqm** command or WebSphere MQ Services. Do this before you start the component using the **mqsistart** command.

The changes to the queue manager's startup type take effect when you restart Windows.

Stopping a WebSphere MQ queue manager when you stop a component

Before you start:

You must complete the following task:

- If you are using a single queue manager to support more than one WebSphere Event Broker component (a single broker can also be defined on the same queue manager as a Configuration Manager, or the User Name Server, or both), you are recommended to specify the -q flag only on the final stop command, having stopped the other components first. The -q flag initiates a queue manager termination regardless of any other component currently using that queue manager.

If you are preparing to stop a broker, Configuration Manager or User Name Server, you can stop the component's WebSphere MQ queue manager at the same time.

You can specify a -q parameter on the **mqsistop** command to initiate a controlled shutdown of the queue manager for a WebSphere Event Broker component.

To stop a WebSphere MQ queue manager enter the following command on the command line:

```
mqsistop WBRK_BROKER  -q
```

where:

WBRK_BROKER is the name of the component.

-q stops the WebSphere MQ queue manager associated with the component.

The command cannot complete until shutdown of the queue manager has completed.

Viewing broker domain log information

You can view broker domain log information by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.LogProxy.

Broker domain log information is written to the Event Log editor in the workbench.

The Event Log contains information about events that occur within your broker domain. These events can be information, errors, or warnings and relate to your own actions. To view events for a particular broker, look for the name of the broker in the Source column.

Each event contains the following information:

- Message: The event number.
- Source: Where the event has come from (within the broker domain).
- TimeStamp: The date and time that the event occurred. TimeStamps are taken from the computer that is hosting the Configuration Manager.
- Details: What has caused the event and what action is needed to rectify it.

The following steps show how to view broker domain log information.

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the broker domain with which you want to work, to reveal the Event Log.
3. Double-click the Event Log. All broker domain log information specific to the broker domain with which you are working is displayed in the Event Log editor.

   If the broker domain is not connected, you are prompted to connect to the broker domain before the Event Log is opened.

   The Event Log editor has two panes called Logs and Details. The top half of the view lists all the events, in date and time order. The bottom half of the view shows the details of a specific selected event. You can maximize and minimize each pane, and toggle between them.
4. Click the event that you want to view in more detail from the top half of the Event Log view. The details of this event can then be viewed in the bottom half of the view.

   When you filter information, as described in "Filtering broker domain log information" on page 194, a note appears next to the view label to indicate that a filter has been applied.

Refreshing broker domain log information

Messages about events that occur in the broker domain are created in the Event Log editor.

To refresh the Event Log editor with new messages:

1. Switch to the **Broker Administration perspective**.
2. In the **Domains** view, expand the appropriate domain to display its components.
3. Double-click on **Event Log** to launch the Event Log editor in the editor area.
4. Right-click in the **Logs** pane and click **Revert** from the pop-up menu. **Revert** does not remove or overwrite existing messages in the Event Log editor or the configuration repository.

Filtering broker domain log information

Broker domain log information is written to the Event Log editor in the workbench. The Event Log contains information about events that occur in a broker domain. It records the source of the event and whether the event is information, an error or a warning.

You can filter the events by message source, message type, or both. This allows you to view a particular set of events. When you filter information, a note appears next to the view label to indicate that a filter has been applied.

You can also mask and clear entries (for example resolved errors), to reduce the amount information displayed in the Event log editor. This makes the other entries easier to see. When you do this, the mask settings that you use for the current session are retained and used in your next session.

To open the Event Log filter.
1. Switch to the Broker Administration perspective.
2. In the Domains view, open the Event Log for the appropriate broker domain.
3. In the Event Log editor menu click **Event Log editor** → **Filter Log ...**. The Filter Log window opens. It has a check box for each of the three message types, and a drop-down menu containing all possible message sources. The default view for the Log Filter is:
   - All message type check boxes are active, denoting that messages of all three types are displayed.
   - Message source is set to *All*, denoting that messages from all sources are displayed.

Using Log Filter, you can filter broker domain log information in the following ways:
- Filtering by message type. To filter the log so that it displays messages only of a particular type:
  1. Clear the check boxes for all three event types.
  2. Check the check box of the event type that you want to view.

     You can view more than one type by selecting the appropriate check boxes.
  3. Click **OK** to activate the filter.
- Filtering by message source. To filter the log such that it displays messages from only one source:
  1. Open the Message Source drop-down menu by clicking the down arrow. Select the message source by which you want to filter.
  2. Click **OK** to activate the filter.
- Filtering by message type and message source. To filter the log so that it displays messages only of a particular type, from a particular message source:
  1. Clear the check boxes for all three event types.
  2. Check the check box of the event type that you want to view.

     You can view more than one type by selecting the appropriate check boxes.
  3. From the Message Source drop-down menu, select the message source by which you want to filter.
  4. Click **OK** to activate the filter for both message type and message source.

To revert to the default view of the Event Log, where all message types from all message sources are displayed, open the **Filter** dialog box and click **Restore Defaults**.

Saving broker domain log information

Broker domain log information is written to the Event Log Editor in the workbench.

Event Log information is deleted automatically from the Configuration Manager after 72 hours.

To save broker domain log information:
1. Switch to the Broker Administration perspective.
2. Open the Event Log for the appropriate broker domain.
3. Right-click in the Event Log view and click **Save Log As...**.
4. You are prompted to save the log information to an appropriate directory.

   By default the log is saved as log.txt. However, you can change the name of this text file.

   The file can also be saved using XML format, with `.xml` file extension.

Each message recorded in the event log is written to the text file with the same information that is detailed in the event log itself.

To view the log, open the log.txt file in an appropriate text editor.

Clearing broker domain log information

You can clear broker domain log information by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class com.ibm.broker.config.proxy.LogProxy.

To clear all the broker domain log information from the Event Log:
1. Switch to the Broker Administration perspective.
2. Open the Event Log for the appropriate broker domain.
3. In the Event Log editor menu, click **Event Log Editor** → **Clear Log ...**.

   If you have set user preference to warn before deleting events, a prompt asks you to confirm deletion. Click **OK**.

   If you have not set user preferences to warn before deleting events, the event log is cleared automatically.

   When you clear the event log, all recorded events that are in view are deleted from the repository.

Changing Event Log editor preferences

You can change the following preferences for the Event Log editor:
- Choose not to display a warning before deleting log events. The default is to display a warning.

- Change the color for each type of event (Warning, Information, and Error). You can choose from a palette of basic colors or define custom colors. The default color for all events is black.
- Define the style and size of the font used for event details. The default is Tahoma, regular, 8 point.

To change preferences:
1. Switch to the **Broker Administration perspective**.
2. Click **Window>Preferences**.
3. Expand the **Broker Administration** category in the left pane.
4. Click **Event Log Editor** within the expanded **Broker Administration** category to open the Event Log editor preferences page.
5. Make your selections.
6. Click **OK**.

Backing up resources

Brokers rely on a database manager to maintain and control their configuration data. Brokers, the Configuration Manager, and the User Name Server rely on WebSphere MQ to transport and guarantee messages between components. You must establish a backup process that includes these sources of information to preserve the integrity and consistency of your broker domain. In addition to backing up the Configuration Manager domain, you should also back up any development resources, such as message flow files, and broker archive files.

You should refer to the information supplied with the database you are using for details of the relevant backup procedures.

The following topics tell you how to back up and restore brokers and the Configuration Manager:
- "Backing up the Configuration Manager domain on distributed systems"
- "Backing up the Configuration Manager domain on z/OS" on page 197

Backing up the Configuration Manager domain on distributed systems

These steps tell you how to back up a Configuration Manager domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. The backup and restoration of a broker needs to extend to every broker that is deployed to by the Configuration Manager. All the topics that you need to carry out these steps are listed in the 'Related information' section at the end of this topic.

To back up the components:
1. Stop each broker.
2. Stop the Configuration Manager.
3. Back up the Configuration Manager data repository using the **mqsibackupconfigmgr** command.
4. Back up each broker database.

To restore the components:

1. **Stop and remove the existing components in the Configuration Manager domain.**
   a. Stop each broker.
   b. Stop the Configuration Manager.
   c. Exit from the Message Brokers Toolkit.
   d. Delete each broker using the **mqsideletebroker** command, specifying the **-w** parameter, which is an optional parameter on Windows and UNIX platforms that deletes from the workpath all files related to these brokers.
   e. Delete the Configuration Manager using the **mqsideleteconfigmgr** command, specifying the **-w** and **-n** parameters. The **-n** parameter deletes all data in the configuration repository.
   f. Drop each broker database.
2. **Recreate the components.**
   a. Create each broker database.
   b. Create the Configuration Manager.
   c. Create each broker.
   d. Restore the Configuration Manager repository using the mqsirestoreconfigmgr command.
   e. Start each broker.
   f. Start the Configuration Manager.
   g. Start the Message Brokers Toolkit.
   h. Connect to the Configuration Manager. This re-imports the broker topology, excluding execution groups and flows, from the Configuration Manager.
   i. Deploy the full topology. This causes the Configuration Manager to give the UUIDs to the brokers.
3. **Restore the components.**
   a. Restore each broker database.
   b. Start each broker.
   c. Start the Configuration Manager.

Backing up the Configuration Manager domain on z/OS

These steps tell you how to back up a Configuration Manager domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. The backup and restoration of a broker needs to extend to every broker that is deployed to by the Configuration Manager. All the topics that you need to carry out these steps are listed in the 'Related information' section at the end of this topic.

To back up the components:
1. Stop each broker.
2. Note the BrokerUUID value from the following file: <broker directory>/registry/<broker name>/BrokerUUID.
3. Stop the Configuration Manager.
4. Back up the Configuration Manager data repository using the **mqsibackupconfigmgr** command.
5. Back up each broker database.

To restore the components:

1. **Stop and remove the existing components in the Configuration Manager domain.**
   a. Stop each broker.
   b. Stop the Configuration Manager.
   c. Exit from the Message Brokers Toolkit.
   d. Delete each broker using the mqsideletebroker command, specifying the **-w** parameter, that deletes from the work path all files related to these brokers..
   e. Delete the Configuration Manager using the **mqsideleteconfigmgr** command, specifying the **-w** and **-n** parameters. The **-n** parameter deletes all data in the configuration repository.
   f. Drop the Configuration Manager repository.

2. **Recreate the components.**
   a. Create the Configuration Manager repository.
   b. Create the Configuration Manager.
   c. Start the Configuration Manager.
   d. Start each broker.
   e. Run the Message Brokers Toolkit.
   f. Connect to the Configuration Manager.
   g. Add to the Configuration Manager domain new brokers with the same names as the originals.
   h. Deploy the topology configuration.

3. **Restore the components.**
   a. Stop each broker.
   b. Stop the Configuration Manager.
   c. Restore the Configuration Manager repository using the **mqsirestoreconfigmgr**command.
   d. Restore each broker database.
   e. Set the BrokerUUID by editing the following file: <broker directory>/registry/<broker name>/BrokerUUID.
   f. Start the Configuration Manager.
   g. Start each broker.
   h. Connect to the Configuration Manager.
   i. Deploy the topology configuration.

Part 4. Reference

Operations

Follow the links below for more information:
- "Broker properties"
- "Commands" on page 202
- "z/OS specific information" on page 333

Broker properties

The following table is a reference for the broker properties.

Property	Meaning
Queue Manager Name	The name of the WebSphere MQ queue manager being used by the broker. The name must be exactly the same name specified for this broker's queue manager on the **mqsicreatebroker** command.
Interbroker Host Name	Interbroker host name to use.
Interbroker Port Number	Interbroker port number to use.
Authentication Protocol Type	The authentication protocols that the broker supports. There are four authentication protocols: **P** Password in the clear **M** Mutual Challenge and response **S** Asymmetric SSL **R** Symmetric SSL
SSL Key Ring File Name	The filename (including path) to the SSL Key Ring file. This is required for authentication when using the Asymmetric and Symmetric (S and R) authentication protocols. The file name and path refer to the path accessible from the broker and not necessarily the tooling, if on different machines.
SSL Password File Name	The SSL key ring file is encrypted and requires a passphrase to decode it. This field is used to specify the filename containing the passphrase required. The file name and path refer to the path accessible from the broker and not necessarily the tooling, if on different machines.

The following three properties:
- Temporary Topic Quality Of Protection
- Sys Topic Quality Of Protection
- ISys Topic Quality Of Protection

refer to the Message Protection feature (QoP) on the "mqsichangeproperties command" on page 236. See "Implementing quality of protection" on page 28 for details on setting these for temporary topics.

Commands

This topic lists the Message Brokers Toolkit commands, followed by the other commands grouped by function, provided by WebSphere Event Broker on distributed systems.

The "Summary of commands on Windows platforms, Linux, UNIX systems, and z/OS" on page 203 lists the comparable commands on WebSphere Event Broker for z/OS.

WebSphere Event Broker Toolkit commands

WebSphere Event Broker commands

Broker commands

Database commands

Configuration Manager commands

User Name Server commands

Start and stop commands

List and trace commands

Summary of commands on Windows platforms, Linux, UNIX systems, and z/OS

The following table summarizes the commands available on Windows platforms, Linux, and UNIX systems and provides the z/OS equivalent, where it is available.

Command on Windows platforms, Linux, and UNIX systems	z/OS equivalent: type	z/OS equivalent	z/OS References
mqsicbrreport	N/A.	-	-
mqsichangebroker	console command: modify	"mqsichangebroker command" on page 221	"mqsichangebroker command" on page 221
mqsichangeconfigmgr	console command: modify	"mqsichangeconfigmgr command" on page 228	"mqsichangeconfigmgr command" on page 228
mqsichangedbimgr			

mqsichangeflowstats	UNIX System Services command	"mqsichangeflowstats command" on page 232	"mqsichangeflowstats command" on page 232
mqsichangetrace	console command: modify	**changetrace**	"mqsichangetrace command" on page 245
mqsichangeusernameserver	console command: modify	"mqsichangeusernameserver command" on page 250	"mqsichangeusernameserver command" on page 250
mqsiclearmqpubsub	console command: modify	"mqsiclearmqpubsub command" on page 253	"mqsiclearmqpubsub command" on page 253
mqsicreateaclentry	console command: modify	"mqsicreateaclentry command" on page 254	"mqsicreateaclentry command" on page 254
mqsicreatebroker	UNIX System Services command	"mqsicreatebroker command" on page 257	"mqsicreatebroker command" on page 257
mqsicreateconfigmgr	console command: modify.	"mqsicreateconfigmgr command" on page 266	"mqsicreateconfigmgr command" on page 266
mqsicreateexecutiongroup	Utility JCL	BIPCREG	"z/OS utility jobs" on page 382
mqsicreatemsgdefs	N/A	-	-
mqsicreatemsgdefsfromwsdl	N/A	-	-
mqsideleteaclentry	console command: modify	"mqsideleteaclentry command" on page 278	"mqsideleteaclentry command" on page 278
mqsideletebroker	UNIX System Services command	"mqsideletebroker command" on page 281	"mqsideletebroker command" on page 281
mqsideleteconfigmgr	console command: modify	"mqsideleteconfigmgr command" on page 283	"mqsideleteconfigmgr command" on page 283
mqsideleteexecutiongroup	Utility JCL	BIPDLEG	"z/OS utility jobs" on page 382
mqsideleteusernameserver	UNIX System Services command	"mqsideleteusernameserver command" on page 288	"mqsideleteusernameserver command" on page 288
mqsideploy	console command:modify	"mqsideploy command" on page 289	"mqsideploy command" on page 289
mqsiformatlog	Utility JCL	BIPFMLG	"z/OS utility jobs" on page 382
mqsijoinmqpubsub	Utility JCL	BIPJNMP	"z/OS utility jobs" on page 382
mqsilist	console command: modify	"mqsilist (list resources) command" on page 296	"mqsilist (list resources) command" on page 296
mqsilistaclentry	console command: modify	"mqsilistaclentry command" on page 298	"mqsilistaclentry command" on page 298
mqsilistmqpubsub	Utility JCL	BIPLSMP	"z/OS utility jobs" on page 382
mqsimigratecomponents	console command:modify	"mqsimigratecomponents command" on page 304	"mqsimigratecomponents command" on page 304
mqsimigratemsgflows	N/A	-	-
mqsimigratemsgsets	N/A	-	-
mqsireadlog	Utility JCL	BIPRELG	"z/OS utility jobs" on page 382
mqsireload	console command: modify	**reload**	"mqsireload command" on page 313

mqsireportflowstats	UNIX System Services command	**reportflowstats**	"mqsireportflowstats command" on page 314
mqsireportproperties	N/A	-	-
mqsireporttrace	console command: modify	**reporttrace**	"mqsireporttrace command" on page 318
mqsisetdbparms	Utility JCL	BIPSDBP	mqsisetdbparms command
mqsisetsecurity	-	-	-
mqsistart	console command: start	standard MVS start command	-
mqsistart	console command: modify	**startcomponent**	"mqsistart command" on page 323
mqsistartmsgflow	Utility JCL	BIPSTMF	"mqsistartmsgflow command" on page 326
mqsistop	1. console command: stop 2. console command: modify	1. standard MVS stop command 2. 'p'	-
mqsistop	console command: modify	**stopcomponent**	"mqsistop command" on page 328
mqsistopmsgflow	Utility JCL	BIPSPMF	"mqsistopmsgflow command" on page 330

Syntax diagrams: available types

The syntax for commands is presented in the form of a diagram. The diagram tells you what you can do with the command, and indicates relationships between different options and, sometimes, different values of an option. There are two types of syntax diagrams: railroad diagrams and dotted decimal diagrams. Railroad diagrams are a visual format suitable for sighted users. Dotted decimal diagrams are text-based diagrams that are more helpful for blind or partially-sighted users.

To select which type of syntax diagram you use, click the appropriate button above the syntax diagram in the topic that you are viewing.

The following topics describe how to interpret each type of diagram:
- "How to read railroad diagrams"
- "How to read dotted decimal diagrams" on page 207

How to read railroad diagrams

Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

The following examples show other conventions used in railroad diagrams.

This example shows that you must specify values A, B, and C. Required values are shown on the main line of a railroad diagram:

```
►►──A──B──C────────────────────────────────────────────────────────────────►◄
```
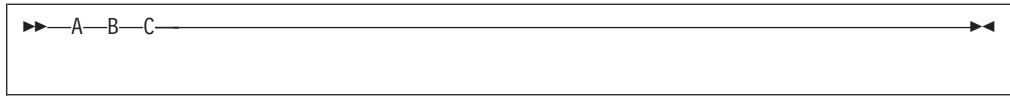
This shows that you can specify value A. Optional values are shown below the main line of a railroad diagram:

```
►►─────────────────────────────────────────────────────────────────────────►◄
   └─A─┘
```
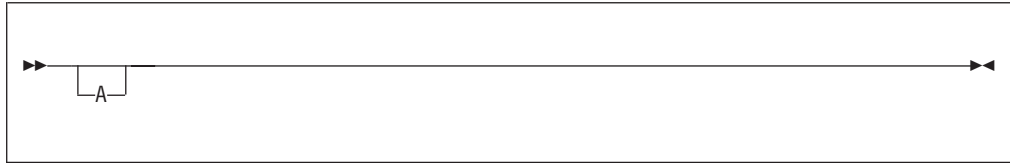
The next example specifies that values A, B, and C are options, one of which you must specify:

```
►►──┬─A─┬──────────────────────────────────────────────────────────────────►◄
    ├─B─┤
    └─C─┘
```
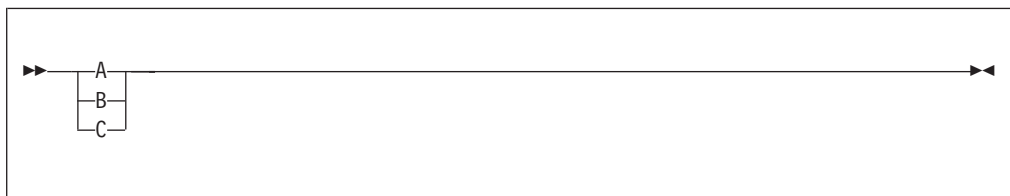
Values A, B, and C are options in this example, one of which you can specify:

```
►►─┬─────┬──────────────────────────────────────────────────────────────────►◄
   ├─A─┤
   ├─B─┤
   └─C─┘
```
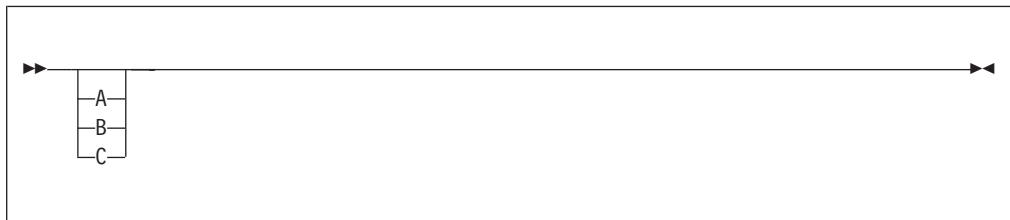
The next example shows that you can specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow:

```
        ┌─,─┐
►►──▼───┴───┬─A─┬───────────────────────────────────────────────────────────►◄
            ├─B─┤
            └─C─┘
```
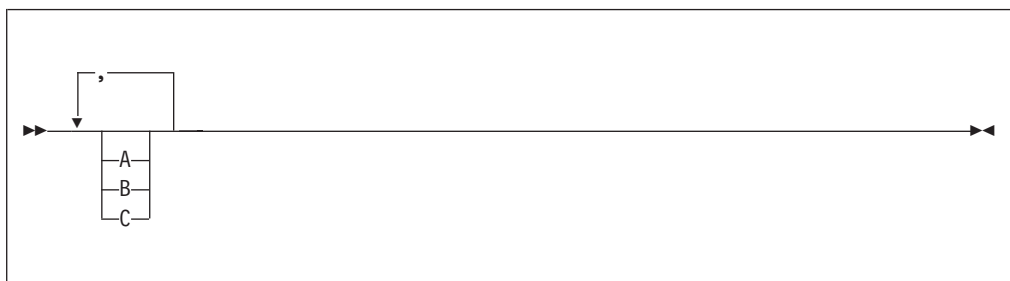
In this example, you can specify value A multiple times. The separator in this example is optional:

Values A, B, and C are alternatives in the next example, one of which you can specify. If you specify none of the values shown, the default A (the value shown above the main line) is used:
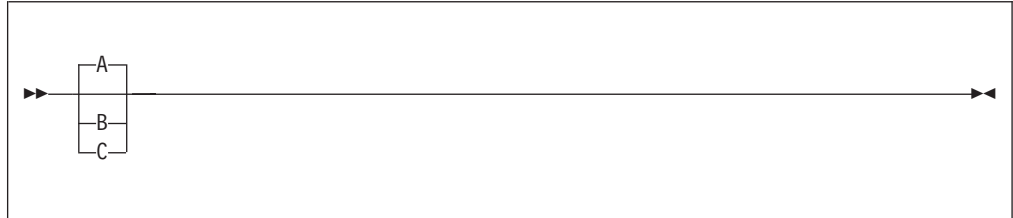
The last example shows the use of a syntax fragment Name, which is shown separately from the main railroad diagram. This technique is used to simplify the diagram, or help fit it into the page of text. The fragment could be used multiple times in the railroad diagram:
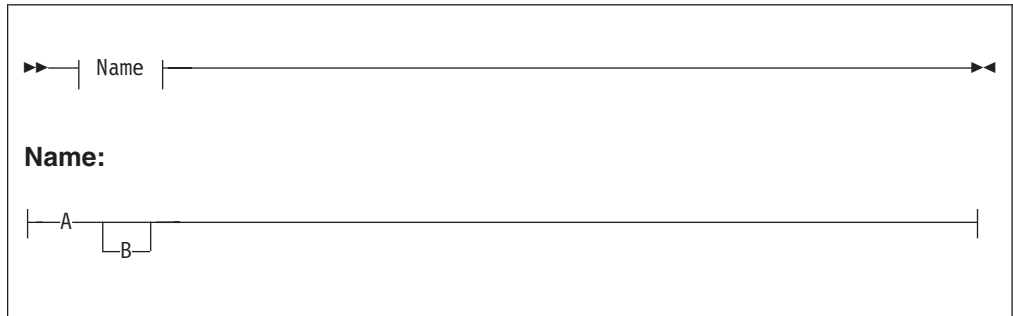
Name:

Punctuation and uppercase values must be specified exactly as shown.

Lowercase values (for example, *name*) indicate where to type your own text in place of the *name* variable.

How to read dotted decimal diagrams

In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line, because they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number, for example 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. For example, if you hear the lines 3.1 USERID, 3.1 SYSTEMID, you know that your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with the dotted decimal number 3 is followed by a series of syntax elements with the dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Characters such as commas that are used to separate a string of syntax elements are shown in the syntax just before the items that they separate. They can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line might also show another symbol giving information about the syntax elements; all these symbols are explained below. For example, the lines 5.1* ,, 5.1 LASTRUN, 5.1 DELETE mean that if you use more than one of the syntax elements LASTRUN and DELETE, they must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, this indicates a reference that is defined elsewhere. The string following the % is the name of a syntax fragment, rather than a literal. For example, the line 2.1 %OP1 means that, at this point, you must refer to the separate syntax fragment OP1. OP1, in the syntax from which this example was taken, gave a list of further options.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the escape character, which is a \ (backslash). For example, the * symbol can be used next to a dotted decimal number to mean that this syntax element can be repeated. If a syntax element actually starts with the * symbol, for example a syntax element * FILE with the dotted decimal number 3, it is given in the format 3 * FILE. If the format is 3* FILE, this means that there is a syntax element FILE, which can be repeated. If the format is 3* * FILE, this means that there is a syntax element * FILE, which can be repeated.

The words and symbols used next to the dotted decimal numbers are as follows:

- **? means an optional syntax element.** If a dotted decimal number is followed by the ? symbol, this means that all the syntax elements with that dotted decimal number, and any subordinate syntax elements that they each have, are optional. If there is only one syntax element with that dotted decimal number, the ? symbol appears on the same line as the syntax element, for example 5? NOTIFY. If there is more than one syntax element with that dotted decimal number, the ? symbol appears on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional; you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

- **! means a default syntax element.** If a dotted decimal number is followed by the ! symbol, appended to the last digit of the dotted decimal number, this means that this syntax element is the default of all the elements with the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a !. For example, if you hear the lines 2? FILE, 2.1! (KEEP), 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. If you include the FILE keyword, but do not state your choice of option, the default option KEEP is applied. As well as the particular syntax element marked with the ! symbol, the default also applies to the next higher dotted decimal number. In the example above, the default applies to 2? FILE as well as to 2.1! (KEEP), meaning that, if you omit the word FILE, the default

FILE(KEEP) is used. However, you might instead hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), 2.1.1 (DELETE). As the default only applies to the next higher dotted decimal number, which in this case is 2.1, it does not apply to 2? FILE. In this case, if you omit the word FILE, nothing is used.

- *** means a syntax element that is optional and can be repeated.** If a dotted decimal number is followed by the * symbol, this means that this syntax element is optional, and can be repeated. For example, if you hear the line 5.1* data-area, you know that you can include more than one data area, or you can include none. If you hear the lines 3*, 3 HOST, 3 STATE, you know that you can include HOST, STATE, both, or nothing. If a dotted decimal number has an asterisk next to it, and there is only one item with that dotted decimal number, you can repeat that same item more than once. If a dotted decimal number has an asterisk next to it, and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the example above, you could write HOST STATE, but you could not write HOST HOST. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

- **+ means a syntax element that must be included at least once, and can be repeated.** If a dotted decimal number is followed by the + symbol, this means that this syntax element must be included at least once, and can be repeated. For example, if you hear the line 6.1+ data-area, you know that you must include at least one data area, and you can include more than one. If you hear the lines 2+, 2 HOST, 2 STATE, you know that you must include HOST, STATE, or both. As for the + symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Message Brokers Toolkit commands

This topic is a container for the commands that are part of the WebSphere Event Broker Toolkit.

These commands are available only on a machine that has the Toolkit installed.

See "Commands" on page 202 for a list of all the WebSphere Event Broker commands.

mqsicreatebar command

Supported platforms:
- Windows
- Linux (x86 platform)

Purpose:

The **mqsicreatebar** command provides a command line compiler that creates deployable broker archive files containing message flows and dictionaries.

Together with the command line repository tools, this enables you to write scripts to repeatedly deploy. Although you are recommended not to perform team development without a repository, a repository is not required.

You can share files using the file system, or using the Eclipse **File>Import...** and **File>Export...** capabilities.

Syntax:

```
>>--mqsicreatebar---------------------------------- -b --barName-------------------->
                  └─ -data --workspacePath─┘

>--------------------------------------------------- -o --filePath--------->◄
     └─ -version --versionString─┘  └─ -p --projectName─┘
```
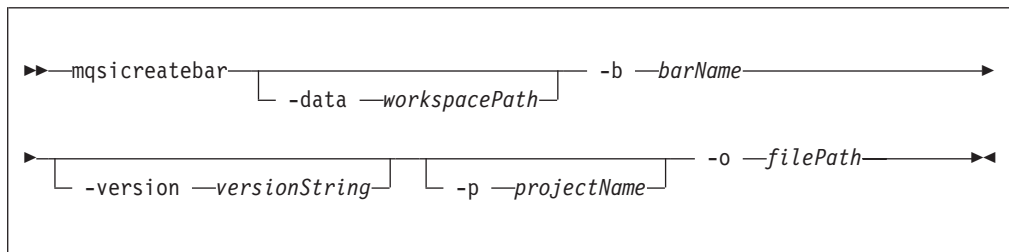
Parameters:

-data *workspacePath*
> (Optional) The location of the workspace, if not the current product installation Eclipse folder. This is an Eclipse provided option

-b *barName*
> (Required) The name of the bar (zip format) archive file where the result is stored. The bar is replaced if it already exists and the META-INF/broker.xml file is created

-version *versionString*
> (Optional) Append the _ (underscore) character and the value of the versionString to the names of the objects added to the bar, before the file extension.

-p *projectName*
> (Optional) Projects containing files to link. You can specify multiple projects that can be a message flow project, a message set project, or a message flow plug-in node project.
>
> This option allows you to use the command line tool to open existing or closed projects. The specified project is fully built and added to the workspace used to compile the bar file.

-o *filePath*
> (Required) The workspace relative path (including the project) of a msgflow or messageSet.mset file to add to the broker archive. The file can be in a message flow project, a message set project, or a message flow plug-in node project.
>
> **Note:** msgflow is the extension of the message flow file that, for example, you can call myFlow.msgflow, whereas the name of the message set file is fixed.

Authorization:

The user ID used to invoke this command must have **Administrator** authority on the local system.

Responses:

This command returns the following responses:
- BIP0956 Unable to start mqsicreatebar
- BIP0957 Incorrect arguments supplied to mqsicreatebar
- BIP0958 Nothing to do in mqsicreatebar
- BIP0959 Incorrect arguments supplied to mqsicreatebar (Project name)
- BIP0960 Incorrect arguments supplied to mqsicreatebar (Project directory)

- BIP0961 Error opening workspace in mqsicreatebar (Project could not be created)
- BIP0962 Error opening workspace in mqsicreatebar (Project could not be opened)
- BIP0963 Error saving file in mqsicreatebar
- BIP0964 Incorrect "-o" argument supplied to mqsicreatebar
- BIP0965 Error compiling files in mqsicreatebar

Examples:

The following example assumes that you are working from the product installation Eclipse folder:

```
mqsicreatebar -b myflow.bar -p TestFlowProject -o TestFlowProject\TestFlow\Test.msgflow
```

The following example specifies a message set:

```
mqsicreatebar -b mySet.bar -o TestSetProject\TestSet\messageSet.mset
```

mqsimigratemfmaps command

Supported platforms:
- Windows
- Linux (x86 platform)

Purpose:

Use the **mqsimigratemfmaps** command to create new .msgmap files in the WebSphere Event Broker Version 6.0 format, based on existing .mfmaps from WebSphere Business Integration Event Broker Version 5.0.

Note:

1. The .mfmap files are left in the workspace so that the migration can be run again.
2. The .mfmap files are not included in builds and are not recognized as valid WebSphere Event Broker development artefacts. They are present in the workspace as simple text files and you can delete them at your own discretion. The Version 5.0 mfmap editor is no longer available in Version 6.0, so the only way to view the contents of the .mfmap files is through a text-editor.
3. Once the .mfmap files are migrated into .msgmap files, these new files are editable through the new Version 6.0 msgmap editor.

The mqsimigratemfmaps.exe file is located in <toolkit install>/eclipse/.

Syntax:

```
►►─mqsimigratemfmaps──┬─all──────────────────┬──┬────────────────────┬──►
                      └─-p ─ProjectName(n)─┘  └─-data ─WorkspacePath─┘

►──┬──────────────────────────┬──┬─────┬──►◄
   └─-log <logFilePath>──────┘  └─-o─┘
```
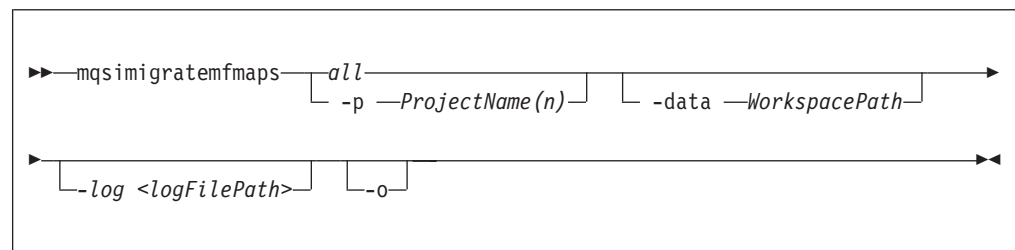
Parameters:

all   (Required) All project names in the work space are checked and migrated. This must be specified unless you specify specific project names to migrate.

-p *ProjectName(n)*
   (Required) The specific project name, or names, in the workspace to be checked and migrated. These must be specified only if you do not specify *all*.

-data *WorkspacePath*
   (Optional) The path of the workspace on which to run the Eclipse platform.

   The workspace location is also the default location for projects. Relative paths are interpreted as being relative to the directory from which Eclipse was started.

-log <filepath>
   (Optional) Alternative report file path.

   If you do not specify an alternative, this command generates a detailed migration log in the `<install_dir>/eclipse/` folder. The log-file has the default name of *"mqsimigratemfmaps.report.txt"*.

-o   (Optional) Overwrite files that already exist.

Authorization:

No specific authority is required to invoke this command.

Examples:

```
mqsimigratemfmaps -p sample -data c:\wmqi\test
```

mqsimigratemsgflows command

Supported platforms:
- Windows
- Linux (x86 platform)

Purpose:

Use the **mqsimigratemsgflows** command to create new flows in the WebSphere Event Broker Version 6.0 format, based on existing exported flows from a WebSphere MQ Event Broker Version 2.1 Configuration Manager.

Note:

1. This command is for one way migration to the WebSphere Event Broker Version 6.0 Message Brokers Toolkit from the Version 2.1 release.
2. When you migrate from Version 5.0 to Version 6.0 you need load only the Version 5.0 `.msgflow` files into the Version 6.0 workspace and when you save the files they are saved in Version 6.0 format.

   In addition, you can use this command to migrate exported user-defined node definitions from Version 2.1 to Version 6.0.
3. Stop your Message Brokers Toolkit session before you invoke the command.
4. Invoke the command from the Eclipse directory where the default workspace is located, in order for the command to operate on the default workspace.
5. If the workspace is in a different location, Specify the location using the **-data** flag.

On Windows operating systems the mqsimigratemsgflows.exe file is located in the
<toolkit install>/eclipse directory. Similarly, on the Linux (x86 platform) operating
system, the mqsimigratemsgflows file is located in the <toolkit install>/eclipse
directory.

Syntax:
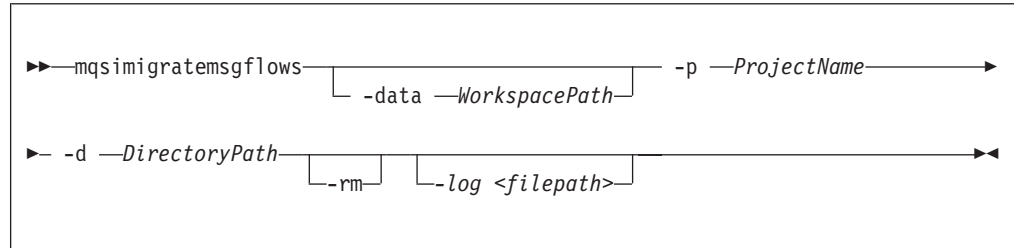
```
►►──mqsimigratemsgflows───────────────────────── -p ──ProjectName──────────►
                        └─ -data ──WorkspacePath─┘

►─ -d ──DirectoryPath──────────────────────────────────────────────────►◄
                     └─-rm─┘ └─-log <filepath>─┘
```

Parameters:

-data *WorkspacePath*
>   (Optional) The path of the workspace on which to run the Eclipse platform.

>   The workspace location is also the default location for projects. Relative paths
>   are interpreted as being relative to the directory from which Eclipse was
>   started.

-p *ProjectName*
>   (Required) The project name where the new flows are created. If the project
>   already exists, the flows are added to this project. If the project does not exist,
>   this command creates it.

>   **Note:** You must specify the **p** parameter before the **d** parameter.

-d *DirectoryPath*
>   (Required) The directory path where the exported flows can be found.

>   A relative path is interpreted from the current working directory. You can
>   specify a full path. All nested directories are ignored.

>   The flow name becomes the name of the .msgflow file. You can change this
>   name to meet the local file naming conventions.

>   Duplicate message flows are ignored; any error of this form is reported in the
>   migration report file.

-rm
>   (Optional) Replace message flow projects if they already exist.

-log <filepath>
>   (Optional) Report file.

>   If you do not specify an alternative, this command generates a detailed
>   migration log in the `<install_dir>/eclipse/` folder. The log-file has the
>   default name of *"mqsimigratemsgflows.report.txt"*.

For further information on flow migration, see Migrating message flows from
Version 2.1.

Authorization:

No specific authority is required to invoke this command.

Examples:

```
mqsimigratemsgflows -p sample -d c:\wmqi\test
```

mqsireadbar command

Supported platforms:
- Windows
- Linux (x86 platform)

Purpose:

The **mqsireadbar** command lists to the console the keywords defined for each deployable file within a deployable broker archive file.

Syntax:

```
►►──mqsicreatebar── -b ──barName──────────────────────────────────►◄
```

Parameters:

-b *barName*
   (Required) The name of the bar (zip format) archive file to read.

Authorization:

The user ID used to invoke this command must have **Administrator** authority on the local system.

Responses:

This command returns the following responses:
- BIP2956 Unable to read broker archive file <file name>

Examples:

The following example reads the file my_bar_file.bar and displays defined keywords within the given file to the console

```
mqsireadbar -b my_bar_file.bar
```

Runtime commands

This topic is a container for the runtime commands of WebSphere Event Broker.

These commands are available on a machine where any of the runtime components are installed

See "Commands" on page 202 for a list of all the WebSphere Event Broker commands.

Setting up a command environment

On distributed systems, you must initialize the environment before you can invoke commands to runtime components. This initialization ensures that all the commands you issue interact with the correct installation of the code.

1. On Windows, open a command console. Click **Start → IBM → WebSphere Message Brokers 6.0 → Command Console** for the appropriate installation.

   This opens a command window with the correct environment. You are not restricted to invoking runtime commands in this window; you can perform typical actions by issuing operating systems commands, invoking your own utilities, and so on.

   See "Sample Windows mqsiprofile.cmd file" for an example Windows profile

2. On Linux and UNIX systems, run the environment profile. The profile is a file called mqsiprofile, and it is in this directory:

   ```
   install_dir/bin
   ```

   To initialize the environment, run this command:

   ```
   . <install_dir>/bin/mqsiprofile
   ```

   You must include the period and space preceding the location for this invocation to work correctly.

   See "Sample Linux mqsiprofile file" on page 216 for an example Linux profile

   You can also initialize the environment in this way on Windows if you prefer. mqsiprofile is located in the same directory.

   You can add this command to your user profile if you choose, so that it is run automatically every time that you log on. The way in which you do this is operating system dependent.

If you intend to create or start a broker, the broker needs access to the broker database and to any user databases that might be accessed from deployed message flows. Access to broker and user databases is through ODBC, therefore you must have created the ODBC connections that are required by these databases, and set the environment to access those connections. For further information, see "Connecting to the databases" on page 74.

If you have installed previous version of this product on the same system, and have defined a broker domain that includes brokers for Version 6 as well as Version 5, or Version 2.1, you are recommended to work with components from different versions on separate user IDs. Run the appropriate profile for that version from that user ID.

The Version 5 and Version 2.1 profiles are provided for each platform, and are named profile.hpux or similar. Check that you run the profile that corresponds to the installation and environment you are working with.

If you use the same user ID, and you run more than one profile, you might get unexpected results. Log off and log on again before you run the second profile.

Each different version (Version 2.1, Version 5, and Version 6) has ODBC definitions that are specific to that version. You cannot use the same ODBC file.

Sample Windows mqsiprofile.cmd file:

This topic lists a sample profile file for Windows platforms.

```
 @echo off
rem Licensed Materials - Property of IBM;
rem 5724-A82
rem (c) Copyright IBM Corp. 2005;
rem All Rights Reserved;
rem US Government Users Restricted Rights - use,
rem duplication or disclosure restricted by GSA
rem ADP Schedule Contract with IBM Corp.;

set MQSI_FILEPATH=C:\Program Files\IBM\MQSI\6.B

set MQSI_WORKPATH=C:\Documents and Settings\All Users\Application Data\IBM\MQSI

set MQSI_VERSION=6.B.0.1

set MQSI_FAD=3


set PATH=%MQSI_FILEPATH%\bin;%MQSI_FILEPATH%\jre\bin;%PATH%;

set LIB=%MQSI_FILEPATH%\lib;%LIB%;

set INCLUDE=%MQSI_FILEPATH%\include;%INCLUDE%;

set MQSI_CONSOLE_NLSPATH=%MQSI_FILEPATH%\messages\
set MQSI_DEVELOPMENT=SOFTWARE\IBM\WebSphere Event Broker\2
set MQSI_REGISTRY=%MQSI_WORKPATH%
set MQSI_JARPATH=%MQSI_FILEPATH%\classes;%MQSI_FILEPATH%\messages
set MQSI_LILPATH=%MQSI_FILEPATH%\bin
set MQSI_EXMLTCONFIGPATH=%MQSI_FILEPATH%\exmltConfig
set DISTHUB_PATH=%MQSI_FILEPATH%
set MQSI_CATALINA_HOME=%MQSI_FILEPATH%\catalina

rem Allow Configuration Manager Proxy applications to be built and run in this environment
set CLASSPATH=%MQSI_FILEPATH%\classes\ConfigManagerProxy.jar;%CLASSPATH%
rem This must go at the end of the CLASSPATH so that the installed version of
WebSphere MQ is read in preference
set CLASSPATH=%CLASSPATH%;%MQSI_FILEPATH%\classes\com.ibm.mq.jar

rem Call the FAD-specific profile to pick up any user-specified settings
call "%MQSI_WORKPATH%\config\mqsiprofile-%MQSI_FAD%.cmd"

echo.
echo MQSI %MQSI_VERSION%
echo %MQSI_FILEPATH%
echo.
```

Sample Linux mqsiprofile file:

This topic lists a sample profile file for Linux.

```
#!/bin/sh

# Licensed Materials - Property of IBM;
# 5724-A82
# (c) Copyright IBM Corp. 2005;
# All Rights Reserved;
# US Government Users Restricted Rights - use,
# duplication or disclosure restricted by GSA
# ADP Schedule Contract with IBM Corp.;

MQSI_FILEPATH=/opt/ibm/mqsi/6.B
export MQSI_FILEPATH

MQSI_WORKPATH=/var/mqsi
export MQSI_WORKPATH

MQSI_VERSION=6.B.0.1
export MQSI_VERSION

MQSI_FAD=3
export MQSI_FAD


export MQSI_REGISTRY=${MQSI_WORKPATH}
export ODBCINI=${MQSI_REGISTRY}/odbc/.odbc.ini
export PATH=${MQSI_FILEPATH}/bin:${PATH}

export MQSI_LILPATH=${MQSI_FILEPATH}/lil:${MQSI_FILEPATH}/jplugin

export MQSI_EXMLTCONFIGPATH=${MQSI_FILEPATH}/exmltConfig

export LD_LIBRARY_PATH=${MQSI_WORKPATH}/lib:${MQSI_FILEPATH}/lib:${MQSI_FILEPATH} \
 /bin:${MQSI_FILEPATH}/merant/lib:${MQSI_FILEPATH}/jre/bin/classic:${MQSI_FILEPATH} \
 /jre/bin:${MQSI_FILEPATH}/xml4c/lib:${LD_LIBRARY_PATH}

export NLSPATH=${MQSI_FILEPATH}/messages/%L/%N:${MQSI_FILEPATH}/messages/En_US/%N:${NLSPATH}
export MQSI_JARPATH=${MQSI_FILEPATH}/classes:${MQSI_FILEPATH}/messages
export MQSI_CATALINA_HOME=${MQSI_FILEPATH}/catalina

export DISTHUB_PATH=${MQSI_FILEPATH}

export MQSI_DEVELOPMENT=${MQSI_REGISTRY}/registry

export CLASSPATH=${MQSI_FILEPATH}/classes/derby.jar:${MQSI_FILEPATH} \
 /classes/ConfigManagerProxy.jar:${MQSI_FILEPATH}/classes/configutil.jar: \
 /opt/mqm/java/lib/com.ibm.mq.jar:/opt/mqm/java/lib/connector.jar:${CLASSPATH}
export LD_LIBRARY_PATH=/opt/mqm/java/lib:${LD_LIBRARY_PATH}
export IBM_JVM_SR_SIGNUM=64
```

Characters allowed in commands

There are a few rules you must adhere to when you provide names or identifiers
for the components and resources in your broker domain.

The components of the broker domain are:

- Brokers
- The Configuration Manager
- The User Name Server (this component is optional)

The broker resources are:
- Execution groups
- Message sets and messages
- Message flows
- Topics

The character set that you can use to name brokers, Configuration Managers, execution groups, and message identifiers is as follows:
- Uppercase A-Z
- Lowercase a-z
- Numerics 0-9
- Any special characters supported by the underlying file system:
  - The following special characters are accepted on Windows platforms:

$	%	' (apostrophe)	' (quote)
- (dash)	_ (underscore)	@	~ (tilde)
!	(	)	{
}	[	]	&
#	&	+	, (comma)
;	=	(space)	

  - The following special characters, with the exception of a space within a broker name, are accepted on UNIX platforms:

. (dot)	%	- (dash)	_ (underscore)
@	~ (tilde)	!	{
}	[	]	&
#	, (comma)	=	(space)

In general, you can use characters **A** through **Z**, **a** through **z**, and **0** through **9**, plus any Unicode character with a decimal value greater than 127 (hexadecimal X'7F'), provided that your operating system can recognize the characters chosen.

If you expect to trace the operation of an execution group, restrict the name of the execution group to include only the valid alphabetic and numeric characters listed. The trace commands do not support the use of special characters for an execution group name.

For all other resources, any characters that are supported by the database configuration are supported.

Broker names, Configuration Manager names, and fixed names (`UserNameServer`) are not case sensitive on the Windows platforms. For example, broker names `Broker1` and `BROKER1` refer to the same broker.

On Linux and UNIX systems, broker names are case sensitive, and the examples above would refer to different brokers. You must use `UserNameServer` as shown.

There are additional rules for naming message service folders within the MQRFH2 header.

Rules for using commands

Observe the following rules when using the WebSphere Event Broker commands on distributed systems. If you are using commands on z/OS, refer to the section on z/OS commands in "Commands" on page 202 and "z/OS utility jobs" on page 382.

- Each command must be issued on the system on which the resource it relates to is defined (or is to be created).
- Each command starts with a primary keyword (the executable command name) followed by one or more blanks.
- Following the primary keyword, flags (parameters) can occur in any order.
- Flags are shown in this book in the form -t, for example. In all cases, the character / can be substituted for the - character.
- If a flag has a corresponding value, its value must follow the flag to which it relates. A flag can be followed by its value directly or can be separated by any number of blanks.
- Flags can be concatenated if they do not have corresponding values, although the last flag in a concatenated group *can* have a value associated with it. For example, the command:

  ```
  mqsireadlog WBRK_BROKER -u -e default -o trace.xml -f
  ```

  could be entered as:

  ```
  mqsireadlog WBRK_BROKER -ufedefault -o trace.xml
  ```

  where the name of the execution group, default, relates to the -e flag. For clarity, all examples given in this documentation are shown with separate flags and with a space before any associated value.
- Repeated flags are not allowed.
- Strings that contain blanks or special characters must be enclosed in double quotation marks. For example:

  ```
  mqsireadlog "My Broker" -u -e default -o trace.xml -f
  ```

  Additionally, you can specify a null, or empty, string with a pair of double quotes with nothing between: "". For example:

  ```
  mqsichangeconfigmgr -s ""
  ```

- The case sensitivity of primary keywords and parameters depends on the underlying operating system. On Windows platforms keywords are not case sensitive; mqsistart, mqsiSTART, and MQSISTART are all acceptable. On UNIX platforms, you must use lower case; only mqsistart is acceptable.

All WebSphere Event Broker commands have dependencies on WebSphere MQ function. You must ensure that WebSphere MQ is available before issuing these commands.

Responses to commands

Responses are issued to the commands as messages. If a command is successful, it returns a return code of zero, and a message with the number BIP8071I (command successful).

Warning and error responses are listed in the command descriptions. If the command is unsuccessful and returns, for example, the message BIP8083, it has an exit code, in this case, of 83.

The following responses are returned by all the commands, and are not listed with each individual command:

- BIP8001 Unknown flag selected
- BIP8002 Selected flags incompatible
- BIP8003 Duplicate flag
- BIP8004 Invalid flags or arguments
- BIP8005 Flag or argument missing
- BIP8006 Mandatory flag missing
- BIP8007 Mandatory argument missing
- BIP8009 Program name not valid
- BIP8083 Invalid component name

mqsibackupconfigmgr command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

This command backs up the Configuration Manager repository online and uses the Configuration Manager Proxy to communicate with the Configuration Manager in the same way that the existing utility mqsideploy does on Windows platforms

Syntax:
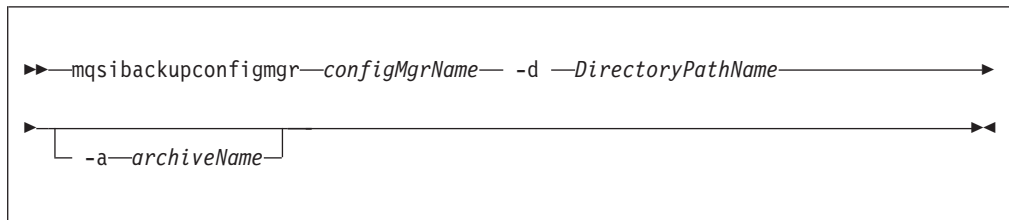
```
►►──mqsibackupconfigmgr──configMgrName── -d ──DirectoryPathName──────────────►

►─┬────────────────────┬──────────────────────────────────────────────────►◄
  └─ -a──archiveName──┘
```

Parameters:

configMgrName
  (Required) The name of the Configuration Manager.

-d *DirectoryPathName*
  (Required) The directory where the back up is placed. An error occurs if the directory does not exist.

-a *archiveName*
  (Optional) Specifies the backup archive name; if you do not specify a name, the name takes the form: `<configMgrName>_<date>_<time>.zip`.

Authorization:

The user ID used to invoke this command must be a member of the Windows **Administrators** group on this local system.

Responses:

This command returns the following responses:

- BIP8056 Unknown queue manager

Examples:

```
mqsibackupconfigmanager ConfigMgr -d c:\mqsitest\db —a archive1
```

mqsicbrreport command

Supported platforms:

- Windows
- Linux and UNIX systems

Purpose:

Use the **mqsicbrreport** command to help identify applications that use a content-based routing filter. The program inspects a broker's subscriptions table, and reports any filters it finds that might cause incompatible behavior.

You must invoke on the system on which the program is running and you must be logged on as the service user ID for the broker.

Syntax:

```
►►──mqsicbrreport──brokername────────────────────────────────►◄
```

Parameters:

brokername

(Required) The name of the broker to inspect. The output is written to stdout and you must review the report generated so that you can make any necessary changes to your programs.

Note: This utility only identifies and reports possible incompatibilities.

Authorization:

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On Linux and UNIX systems, the user ID used to invoke this command must be **root**. It must also be a member of the **mqbrkrs** group.

mqsichangebroker command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsichangebroker** command to change some of the properties of the broker.

You must stop the broker, using **mqsistop** on Windows platforms, Linux, and UNIX systems, or **stopcomponent** on z/OS, (see "mqsistop command" on page 328) before you can issue this command.

When you restart the broker, using **mqsistart** on Windows platforms, Linux, and UNIX systems, or **startcomponent** on z/OS, (see "mqsistart command" on page 323) it uses the changed parameters.

Syntax:

Windows platforms, Linux, and UNIX systems:

z/OS console command:

Synonym: cb

```
                                         ,
                                    ┌─────────┐
>>──┬─changebroker─┬─────┴─ g=ConfigurationTimeOut ──────────┬──────>
    └─cb──────────┘      ├─ k=ConfigurationDelayTimeOut ──────┤
                         ├─ s=UserNameServerQueueManagerName ─┤
                         ├─ l=UserLilPath ────────────────────┤
                         ├─ v=statisticsarchiveinterval ──────┤
                         └─ P=HTTPPort────────────────────────┘

>──┬──────────────────────────────────────────┬──┬─────────────────┬──><
   └─ -y─LdapPrincipal──-z─LdapCredentials ────┘  └─ -c ─ICUDataPath─┘
```
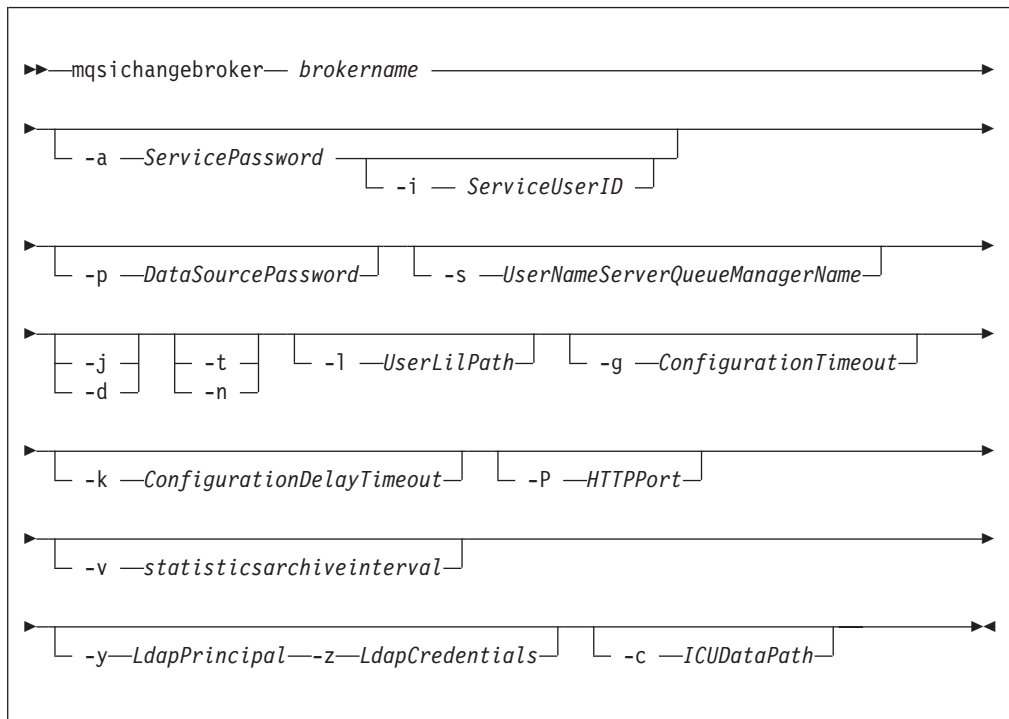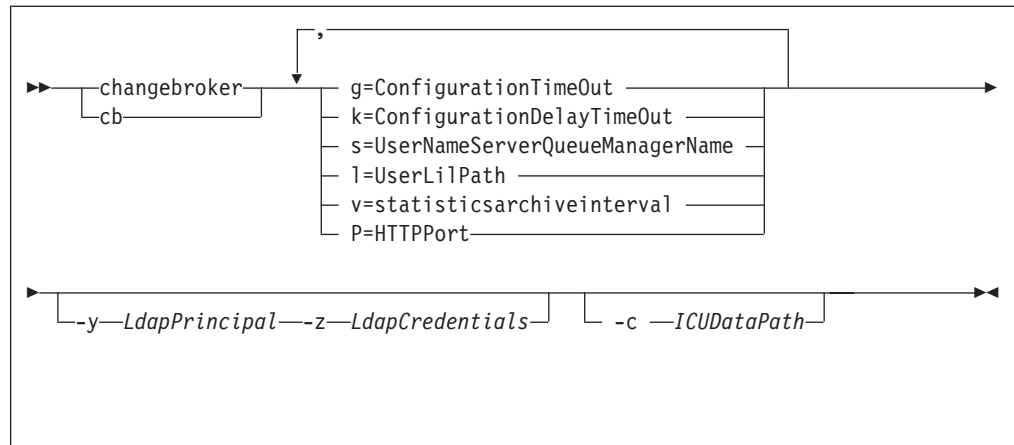
Parameters:

brokername
> (Required - Windows platforms, Linux, and UNIX systems) This must be the first parameter. Specify the name of the broker you want to modify.

-a *ServicePassword*
> (Optional - Windows platforms, Linux, and UNIX systems) The password for the `ServiceUserID`.
>
> For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.
>
> On Linux and UNIX systems, `-a` is required for Windows platforms compatibility but it is not used in relation to `ServiceUserID`; it is used as a default only if `-p` is not specified. (See the description of the `-p` parameter for further details.)
>
> If you have created this broker to also use this user ID and password for database access (that is, either you omitted the `-u DataSourceUserID` and `-p DataSourcePassword` parameters, or you included them but provided the same user ID and password for the service user ID using `-a ServicePassword` and `-i ServiceUserID`), ensure that you update both instances of the password on this command.
>
> To complete a password change successfully, you must:
> * Stop the broker.
> * Change the password using the appropriate operating system facilities.
> * Use this command to update all parameters that reference this same password.
> * Restart the broker.

-i *ServiceUserID*
> (Optional - Windows platforms, Linux, and UNIX systems) The user ID under which the broker runs. You must also change the password (`-a`) if you change this value.
>
> This can be specified in any valid username syntax. On Windows platforms, these are:
> * `domain\username`

- \\server\username
- .\username
- username

On Linux and UNIX systems, only the last format, `username`, is valid.

If you use the unqualified form for this user ID (`username`) in Windows platforms, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The `ServiceUserID` specified must be a member of the local group **mqbrkrs**. On Windows platforms, it can be an indirect or direct member of the group. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the **mqsicreatebroker -w** flag). This ID must also be a member (either direct or indirect) of the local group **mqm**.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 389 for Windows platforms and in "Security requirements for Linux and UNIX platforms" on page 388 for Linux and UNIX platforms.

-p *DataSourcePassword*
 (Optional -Windows platforms, Linux, and UNIX systems) The password of the user ID with which the databases containing broker tables and user data are to be accessed. If not specified, this defaults to the `ServicePassword` specified by `-a`.

 For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

 For DB2 on Linux and UNIX systems, `-p` can be specified as an empty string (two double quotation marks, "). In this case, DB2 grants WebSphere Event Broker the privileges of the `ServiceUserID` which results in a database connection as "already verified". If you specify an empty string for `-a` and `-p`, no passwords are stored by WebSphere Event Broker, creating the most secure configuration.

 You must ensure that you change all instances of the use of this password. If you have created (or changed) the broker to use the same user ID and password for its service user ID as well as its database access, you must update both instances at the same time. See the description of `-a` for further details.

-s *UserNameServerQueueManagerName*
 (Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server. If you want to remove topic-based security, specify an empty string (two double quotation marks, ").

 Note that on z/OS this name is case sensitive and you must include the names in single quotes if they contain mixed case characters.

-j (Optional - Windows platforms, Linux, and UNIX systems) Publish/Subscribe access is enabled for the broker.

-d (Optional - Windows platforms, Linux, and UNIX systems) Publish/Subscribe access is not enabled for the broker.

-t (Optional - Windows platforms, Linux, and UNIX systems) Requests that the broker runs as a WebSphere MQ trusted application.

This option is not available on AIX. If specified, the flag is ignored.

For more details about using WebSphere MQ trusted applications, see *WebSphere MQ Intercommunication*.

-n (Optional - Windows platforms, Linux, and UNIX systems) Requests that the broker ceases to run as a WebSphere MQ trusted application.

-l *UserLilPath*

(Optional) Specifies a list of paths (directories) from which the broker loads LILs (loadable implementation libraries) for user-written plug-in message processing nodes.

You must create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Event Broker install directory.

If you specify more than one additional directory, they must be separated by the default platform-specific path separator (semicolon (`;`) on Windows platforms, colon (`:`) on Linux, UNIX systems, and z/OS).

You cannot include environment variables in this path: if you do so, they are ignored.

Note that on Linux, UNIX systems, and z/OS this name is case sensitive and you should include the names in single quotes if they contain mixed case characters.

-g *ConfigurationTimeout*

(Optional) Defines the length of time (in seconds) that an execution group in the broker is allowed to take to apply a change in configuration (for example, an update that you have deployed from the workbench).

When a message flow is processing an application message, it cannot respond to a configuration change. If any one of the message flows within the execution group that has been requested to change its configuration does not finish processing an application message and apply the configuration change within this timeout, the execution group returns a negative response to the deployed configuration message.

The value that you set for this timeout depends on the system load (including CPU utilization) and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives you an indication of the minimum value that you should set.

The value is specified in seconds and can range from 10 to 3600. The default value is 300.

The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* (described below) represents the maximum length of time that a broker is allowed to process a deployed configuration message before it generates a negative response.

-k *ConfigurationDelayTimeout*

(Optional) Defines the length of time (in seconds) that a broker is allowed to take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).

This represents the time it takes for a minimal deployed configuration message to be processed by the broker and its execution groups, and is dependent on queue manager network delays, the load on the broker's queue manager, and system load.

You can estimate this value by issuing a command to request a simple configuration change, for example:

- On Windows platforms, Linux, and UNIX systems:

```
mqsireporttrace brokerName -e "Execution Group Name" -u
```

- On z/OS

```
F MQP1BRK,reporttrace u=yes,e='exgrp1'
```

Note that on z/OS this name is case sensitive and you should include the names in single quotes if they contain mixed case characters.

The response time of each execution group differs according to system load and the load of its own processes. The value that you set must reflect the longest response time any execution group takes to respond. If the value you set is too low, the broker returns a negative response, and might issue error messages to the local error log.

The value is specified in seconds and can range from 10 to 3600. The default value is 60.

If the broker is on a production system, you are recommended to increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the time-out values decreases the probability of successfully deploying a configuration change.

-P *HTTPPort*
(Optional) Enter the number of the port on which the web services support is listening.

Note that this listener is started by the broker when a message flow that includes web services support is started, and has a default value of 7080.

You must ensure that the port you specify has not been specified for any other purpose.

-v *statisticsarchiveinterval*
(Optional) Specify the timer interval in minutes at which WebSphere Event Broker statistics and accounting is notified that archive records should be output. For internal accounting, the valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the platform has an external method of notification and is not using an internal timer within WebSphere Event Broker .

-y *LdapPrincipal*
(Optional - all platforms, but mandatory when *LdapCredentials* is provided.) The user principal for access to an optional LDAP directory that holds the JNDI administered Initial Context for the JMS provider.

-z *LdapCredentials*
>   (Optional - all platforms, but mandatory when *LdapPrincipal* is provided.) The user password for access to LDAP.

-c *ICUDataPath*
>   (Optional) A delimited set of directories to search for additional code page converters. On Windows systems the delimiter is ";". On UNIX and Linux systems, the delimiter is ":". The code page converters must be either of the form icudt32_*codepagename*.cnv or in an ICU data package called icudt32.dat.

If you want to change other broker properties, you must delete and recreate the broker, then use the workbench to redeploy the broker's configuration. If you want to change the user ID used for database access, see "Administering the broker domain" on page 185.

Authorization:

On Windows systems, the user ID used to invoke this command must have **Administrator** authority on the local system.

On Linux and UNIX systems, the user ID used to invoke this command must be a member of the **mqbrkrs** group.

On z/OS systems, the user ID used to invoke this command must be a member of a group that has READ and WRITE access to the component directory.

Using LDAP: Ensure that the registry is appropriately secured to prevent unauthorized access. The setting of *LdapPrincipal* and *LdapCredentials* options on **mqsichangebroker** is not required for correct operation of the broker. The password is not stored in clear text in the file system.

Responses:

This command returns the following responses:
- BIP2595 Error casting character string '...' to an integer (z/OS only)
- BIP8003 Duplicate flag detected (z/OS only)
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8018 Component running
- BIP8021 User ID/password incorrect
- BIP8022 Invalid user ID/password
- BIP8023 Password required
- BIP8030 Unable to modify user ID privileges
- BIP8068 Argument ... is not a valid integer value for this flag (z/OS only)
- BIP8073 Invalid broker name (Windows platforms and UNIX platforms only)
- BIP8158 Invalid format for command (z/OS only)
- BIP8159 Unknown parameter "..." (z/OS only)

Examples:

Windows platforms, Linux, and UNIX systems:
```
mqsichangebroker WBRK_BROKER -s WBRK_UNS_QM
mqsichangebroker WBRK_BROKER -s ""
```

z/OS:

```
F MQP1BRK,cb g=100,k=200
```

mqsichangeconfigmgr command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsichangeconfigmgr** command to change some of the properties of the Configuration Manager.

You must stop the Configuration Manager, using **mqsistop**, (see "mqsistop command" on page 328).

When you restart the Configuration Manager, using **mqsistart**, (see "mqsistart command" on page 323), it uses the changed parameters.
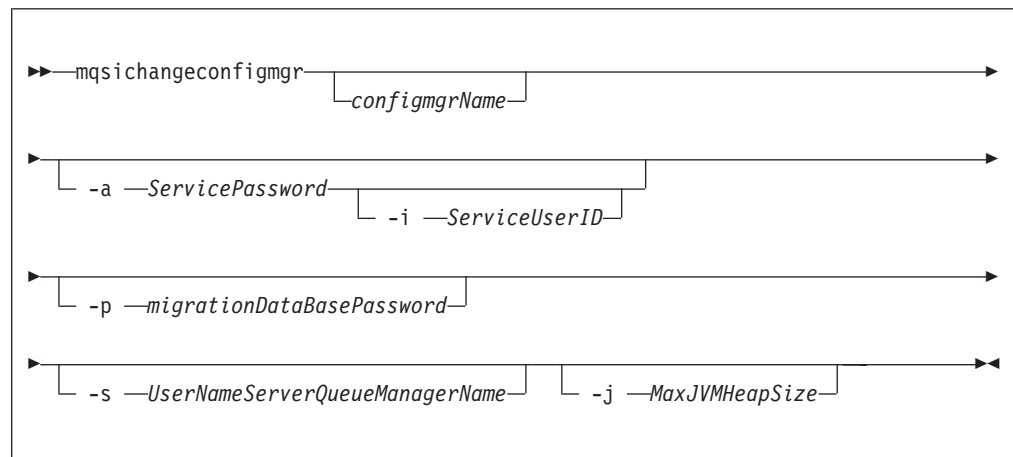
Syntax:

Windows:

Linux and UNIX systems:
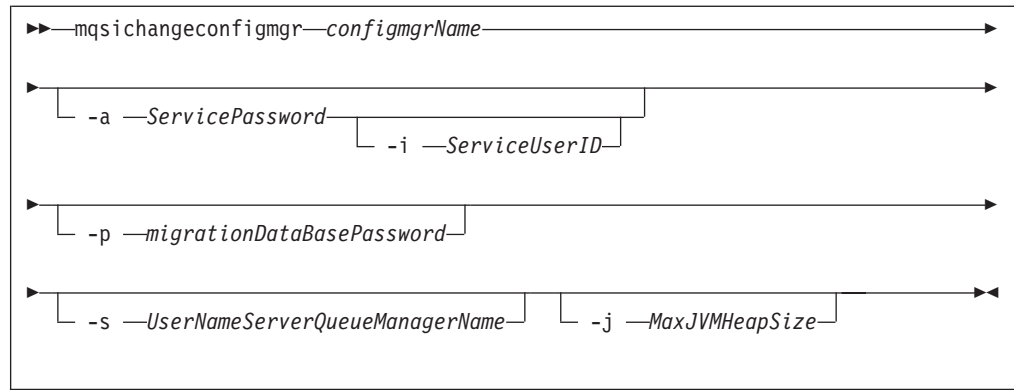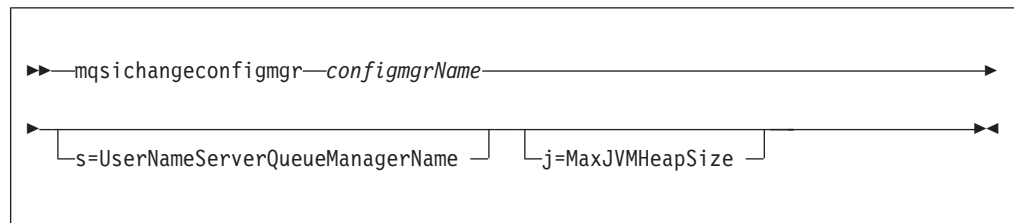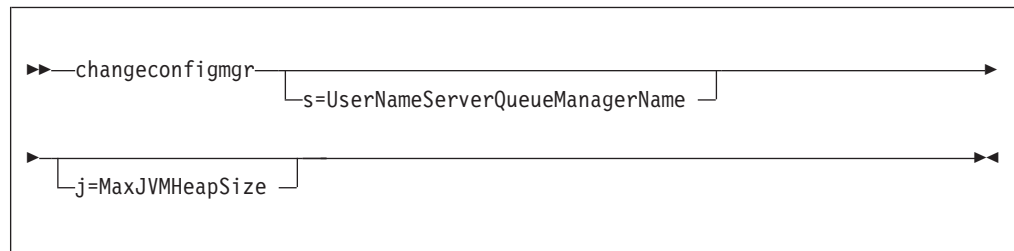
```
►►── mqsichangeconfigmgr ── configmgrName ──────────────────────────────────────►

►──┬──────────────────────────────────────────────┬───────────────────────────►
   └─ -a ── ServicePassword ──┬───────────────────┬┘
                              └─ -i ── ServiceUserID ─┘

►──┬──────────────────────────────────────┬────────────────────────────────────►
   └─ -p ── migrationDataBasePassword ─┘

►──┬─────────────────────────────────────────────────┬──┬─────────────────────┬►◄
   └─ -s ── UserNameServerQueueManagerName ─┘          └─ -j ── MaxJVMHeapSize ─┘
```

z/OS:

```
►►── mqsichangeconfigmgr ── configmgrName ──────────────────────────────────────►

►──┬─────────────────────────────────────────────┬──┬──────────────────────┬───►◄
   └ s=UserNameServerQueueManagerName ─┘            └ j=MaxJVMHeapSize ─┘
```

z/OS console command:

```
►►── changeconfigmgr ─┬──────────────────────────────────────┬──────────────────►
                      └ s=UserNameServerQueueManagerName ─┘

►──┬──────────────────────┬─────────────────────────────────────────────────────►◄
   └ j=MaxJVMHeapSize ─┘
```

Parameters:

configmgrName
>   (Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name of the Configuration Manager that you want to change.
>
>   On Linux, UNIX systems and z/OS, this must be the first parameter specified. It is case-sensitive on Linux and UNIX systems.
>
>   The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-a *ServicePassword*
>   (Optional - Windows platforms, Linux and UNIX systems only) The password for the `ServiceUserID`.
>
>   If you created the Configuration Manager to use this user ID and password for database access as well (that is, either you omitted the configuration repository access parameters -u `DataBaseUserID` and -p `DataBasePassword`, or you included them but provided the same user ID and password for the service user ID using -a `ServicePassword` and -i `ServiceUserID`), ensure that you update all instances of the password on this command.
>
>   To complete a password change successfully, you must:

- Stop the Configuration Manager.
- Change the password using the appropriate operating system facilities.
- Use this command to update all parameters that reference this same password.
- Restart the Configuration Manager.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

-i *ServiceUserID*
(Optional - Windows platforms, Linux and UNIX systems only) The user ID under which the Windows system service runs. You can only change this value if you also change the password.

This can be specified in any valid username syntax for the platform.

If you use the unqualified form for this user ID (`username`) on Windows systems, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The `ServiceUserID` specified must be a member of the local group **mqbrkrs**. It can be an indirect or direct member of the group. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the **mqsicreateconfigmgr -w** flag). This ID must also be a member (either direct or indirect) of the local group **mqm**.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 389 for Windows platforms and "Security requirements for Linux and UNIX platforms" on page 388 for UNIX platforms.

-p *migrationDataBasePassword*
(Optional - Windows platforms, Linux and UNIX systems only) The password for the DB2 database that you are migrating.

Ensure that you change all instances of the use of this password. If you have created (or changed) the Configuration Manager to use the same user ID and password for its service user ID as well as its database access, you must update all instances at the same time. See the description of **-a** for further details.

-s *UserNameServerQueueManagerName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server. If you want to remove topic security, specify an empty string (two double quotation marks, ""）.

-j *MaxJVMHeapSize*
(Optional) The maximum Java virtual machine (JVM) heap size, in megabytes. The smallest value that you can set is 64. If you do not set this parameter, the default size of 128 MB is used.

If you want to change other properties, you must delete and recreate the Configuration Manager. If you want to change the `DataBaseUserID`, see "Modifying component database access" on page 140 for instructions.

Responses:

This command returns the following responses:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8018 Component running
- BIP8021 User ID/password incorrect
- BIP8022 Invalid user ID/password
- BIP8023 Password required
- BIP8030 Unable to modify user ID privileges

Examples:

```
mqsichangeconfigmgr CMGR01 -d WBRK_DOMAIN
mqsichangeconfigmgr CMGR01 -s ""
```

mqsichangedbimgr command

Supported platforms:
- Windows

Purpose:

The **mqsichangedbimgr** command updates the Windows Service that runs the Cloudscape Network Server, with a new user password and/or new user ID.

Syntax:

Windows platforms:

```
►►──mqsichangedbimgr── -a ─ServicePassword───────────────────────────────►◄
                                              └─ -i ─ServiceUserID─┘
```
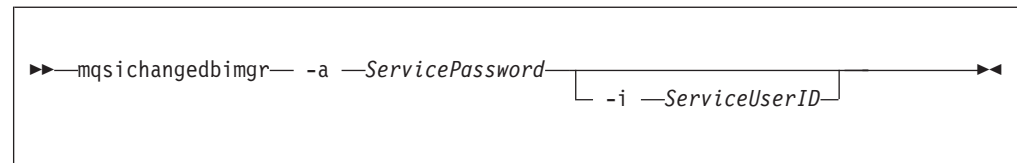
Parameters:

-a *ServicePassword*
  (Required) The new password for the ServiceUserID. To complete a password change successfully, you must:
  - Stop all the broker components.
  - Change the password using the appropriate operating system facilities.
  - Use the mqsichange... commands to update all the broker components that reference this same ServiceUserID and password.
  - Restart all the broker components.

-i *ServiceUserID*
  (Optional) The user ID under which the default database runs. This can be specified in any valid username syntax. On Windows platforms, these are:
  - domain\username
  - \\server\username
  - .\username
  - username

If you use the unqualified form for this user ID (username) on Windows platforms, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete. The ServiceUserID specified must be a member of the local group mqbrkrs. On Windows platforms, it can be a direct or indirect member of the group. The ServiceUserID must also be authorized to access the home directory (where WebSphere Business Integration Message Broker has been installed). The security requirements for the ServiceUserID are detailed in Security requirements for Windows platforms for Windows platforms.

Authorization:

The user ID used to invoke this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Responses:

This command returns the following responses:
- BIP8071 Successful command completion.
- BIP9810 The IBM MQSeries Brokers DatabaseInstanceMgr service was not found.

Examples:

The following example changes the password used for the IBM MQSeries Brokers DatabaseInstanceMgr service:

```
mqsichangedbimgr -a wbrkpw1
```

mqsichangeflowstats command

Supported Platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsichangeflowstats** command to:
- Turn on or off accounting and statistics snapshot publication, or archive record output.
- Specify that the command be applied to a specific flow message flow, or all flows in an execution group, or all execution groups belonging to a broker.
- Modify the granularity of the data collected in addition to the standard message flow accounting and statistics. This extra data can include thread related data, node related data, node terminal related data, or a mixture of this data.

The options set using this command remain active until modified by a subsequent **mqsichangeflowstats** command.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsichangeflowstats── brokername ──┬── -a ──┬──────────────────────────────►
                                        └── -s ──┘

►─┬─ -e ─ExecutionGroupLabel── -f ─MessageFlow─┬───────────────────────────────►
  └─ -g ── -j ──────────────────────────────────┘   └── -c ──control──┘

►─┬──────────────────────────────────────────────────────────────────────────►
  └── -t ─ThreadData─┘  └── -n ─NodeData─┘  └── -r ──┘

►─┬──────────────────────────────────────────────────────────────────────────►◄
  └── -b ─AccountingOrigin─┘  └── -o ─OutputFormat─┘
```

z/OS:

Synonym cs

```
►►─┬─changeflowstats─┬──┬─ a=yes ─┬──────────────────────────────────────────►
   └─ cs ────────────┘  └─ s=yes ─┘

►─┬─ e=ExecutionGroupLabel ── f =MessageFlow─┬────────────────────────────────►
  └─ g=yes ── j=yes ──────────────────────────┘   └─ c=Control ─┘

►─┬──────────────────────────────────────────────────────────────────────────►
  └─ t=ThreadData─┘  └─ n=NodeData─┘  └─ r=yes ─┘  └─ b=AccountingOrigin─┘

►─┬──────────────────────────────────────────────────────────────────────────►◄
  └─ o=OutputFormat─┘
```
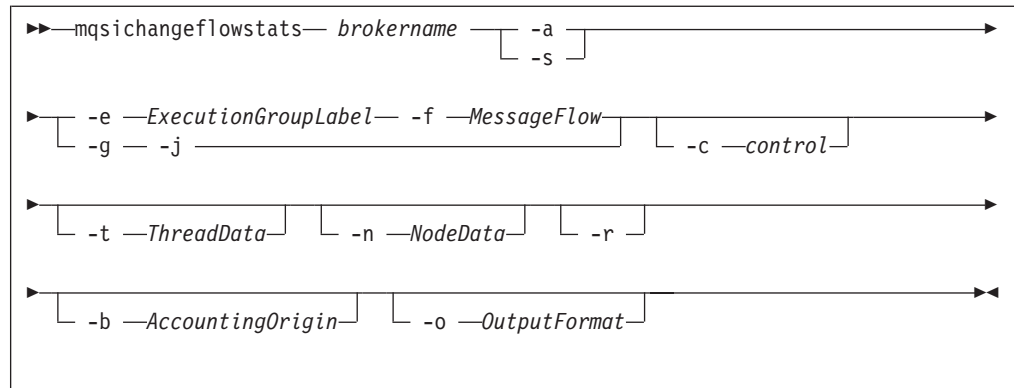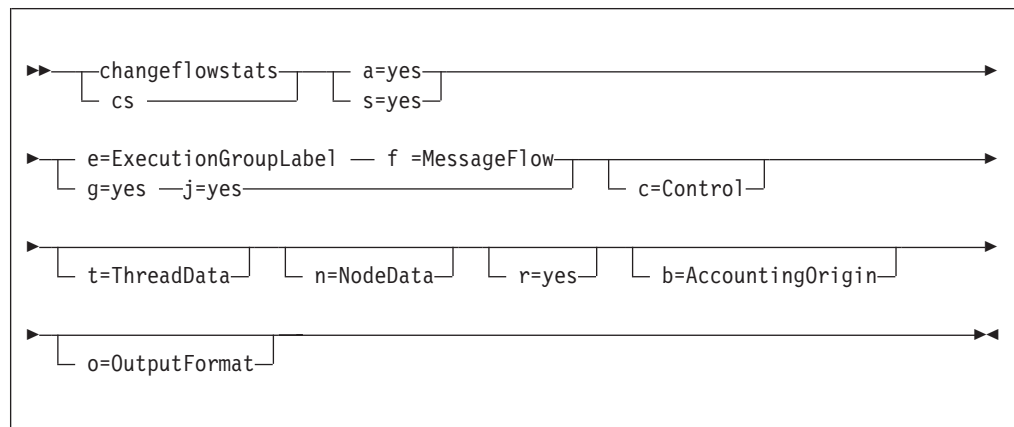
Parameters:

brokername
> (Required -Windows platforms, Linux, and UNIX systems) Specify the label of the broker for which accounting and statistics are to be changed.

-a (Required) Specify that the command modifies archive accounting and statistics collection.

> **Note:** You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

-s (Required) Specify that the command modifies snapshot accounting and statistics collection.

> **Note:** You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

-e *ExecutionGroupLabel*
> (Required) Specify the label for the execution group, for which accounting and statistics options are to be changed.

> **Note:** You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive a warning message.

-f *MessageFlow*

(Required) Specify the label for the message flow, for which accounting and statistics options are to be changed.

Note: You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

-g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

Note: You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive a warning message.

-j (Required) Specifies that the command applies to **all**message flows that belong to the execution group.

Note:

1. You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.
2. If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

-c *control*

(Optional) Specify the string value that controls the level of the action to be applied to accounting and statistics collection for snapshot or archiving. Possible values are:

- active - turn on snapshot or archiving
- inactive - turn off snapshot or archiving.

-t *ThreadData*

(Optional) Specify a string value to modify the collection of thread statistics data for a message flow Possible values are:

- none - exclude thread related data from the statistics
- basic - include thread related data in the statistics

-n *NodeData*

(Optional) Specify a string value to modify the collection of node statistics data for a message flow. Possible values are:

- none - exclude node related data in the statistics
- basic - include node related statistics in the statistics
- advanced - include node related and terminal related data in the statistics

-r (Optional) Specify that a reset of archive data is required.

Note: This action is only valid for archive data.
This results in the clearing out of accounting and statistics data accumulated so far for this interval, and restarts collection from this point. All archive data for all flows in the execution group, or groups, is reset.

The archive interval timer is only reset if the **-v** option (*statistics archive interval*) of **mqsicreatebroker** or **mqsichangebroker** is non zero. That is , the interval timer is only set if the internal interval notification mechanism is being used , and not an external method for example, ENF on z/OS.

-b *AccountingOrigin*

This parameter has no effect.

-o *OutputFormat*

(Optional) Specify the output destination for the statistics reports. Possible values are:

- usertrace - this is the default and writes ″bip″ messages to usertrace, which can be post processed in the normal way using the **mqsireadlog** and **mqsiformatlog** commands

- xml - the statistics reports are generated as XML documents and published by the broker running the message flow.

  The topic on which the data is published has the following structure:

  ```
  $SYS/Broker/<brokerName>/StatisticsAccounting/<recordType>
  /<executionGroupLabel>/<messageFlowLabel>
  ```

  where recordType is set to Snapshot or Archive, and broker, execution group, and message flow names are specified according to the subscriber's requirements.

- smf - (z/OS only). Statistics reports are output as SMF type 117 records.

Authorization:

The user Id used to issue the command must have mqbrkrs authority.

Responses:

This command returns the following responses:

- BIP2226 Request to change attribute in message flow node ′ ′: message flow does not exist
- BIP8004 Invalid flags and arguments selected
- BIP8013 Component does not exist
- BIP8020 Unable to access the database
- BIP8029 Broker not configured
- BIP8033 Unable to send XML message
- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database

Examples:

Turn on snapshot statistics for the message flow ″myFlow1″ in all execution groups of BrokerA and specify that the data is to be gathered by accounting origin:

```
mqsichangeflowstats BrokerA -s -g -j -b none
```

Turn off the collection of archive statistics for message flow ″MyFlow1″ in execution group ″EGRP2″ for BrokerA, and at the same time modify the granularity of data that is to be collected (when next activated) to include thread related data.

```
 mqsichangeflowstats BrokerA -a -e "EGRP2" -f MyFlow1 -c inactive -t basic
```

Turn off snapshot data for all message flows in all execution groups for Broker A.

```
mqsichangeflowstats BrokerA -s -g -j -c inactive
```

mqsichangeproperties command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS through the BIPJCHPR utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsichangeproperties** command to change properties relating to inter-broker communications and the http listener component, which includes the HTTP and HTTPS (SSL) support for the HTTPInput and HTTPReply nodes.

Start the broker using mqsistart, and then issue the command on the machine running the broker.

If you change any value you **must** stop and restart the broker for the change to take effect.

Syntax:

```
►►──mqsichangeproperties──brokername──┬─ -b ─ComponentName──────┬──────────►
                                      └─ -e ─ExecutionGroupLabel─┘

►── -o ─ObjectName── -n ─PropertyName── -v ─PropertyValue──────────►◄
```
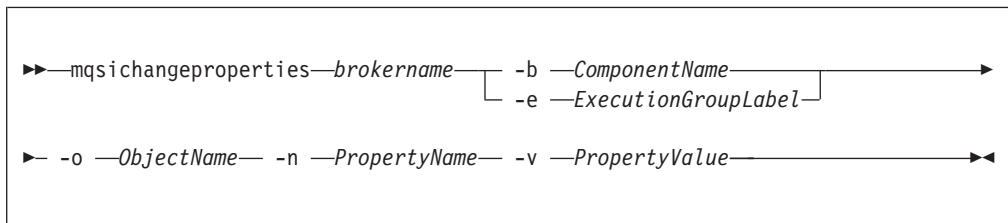
Parameters:

brokername
   (Required) The name of the broker to modify. This must be the first parameter.

-b *ComponentName*
   (Optional) The name of the component whose properties you want to change. For the http listener component use *httplistener*.

-e *ExecutionGroupLabel*
   (Optional) Reserved

-o *ObjectName*
   (Required) The name of the object whose properties you want to change. This must be set to one of the values listed below.

-n *PropertyName*
   (Required) The name of the property being changed.

-v *PropertyValue*
   (Required) The supplied *PropertyValue* is assigned to the property specified by *PropertyName*.

   More than one name and corresponding value can be specified using commas as separators. For example,

   `-n Name1, Name2 -v Value1, Value2`

For Inter-broker communications, the following *ObjectName* values are available:
- *DynamicSubscriptionEngine*

For the *httplistener* component, which includes the HTTPInput and HTTPReply nodes, the following *ObjectName* values are available:

- *HTTPListener* for controlling the HTTPListener process.
- *HTTPConnector* for controlling HTTP communication with the HTTPInput and HTTPReply nodes.
- *HTTPSConnector* for controlling HTTPS communication with the HTTPInput and HTTPReply nodes.

The *httplistener ComponentName* is required when using the HTTP *ObjectName* settings.

Inter-broker communications parameter values

Valid property names for inter-broker communications are listed below:

brokerInputQueues
Specifies the maximum number of dispatch queues which are to be used when processing messages from an interbroker connection. Increasing the value might increase the rate at which messages can be transmitted across an interbroker connection:

- Value type - Integer
- Initial value - 1

brokerInputQueueLength
Defines the maximum number of messages that can be stored in each input queue; the higher the value, the higher the number of input messages that can be stored in each input queue. Note that the higher the value of this property, the larger the amount of memory that the broker requires for each queue:

- Value type - Integer
- Initial value - 99

brokerPingInterval
Time in milliseconds between broker initiated ping messages on broker-broker connections. Ping messages ensure that the communications are still open between both sides of the connection, and are generated internally. If the value is 0, the broker does not initiate pings.

- Value type - Integer
- Initial value - 5000

clientPingInterval
Time in milliseconds between broker initiated ping messages on broker-client connections. Ping messages ensure that the communications are still open between both sides of the connection, and are generated internally. If the value is 0, the broker does not initiate pings.

- Value type - Integer
- Initial value - 30000

enableClientDiscOnQueueOverflow
If true, and if after deleting all possible messages the maxClientQueueSize is still exceeded, the broker disconnects the client.

- Value type - Boolean
- Initial value - False

enableQopSecurity
Enables the level of quality of protection for messages.

By default, Quality of Protection is enabled if either the **isysQopLevel** or **sysQopLevel** value has been changed from the default value of none.

- Value type - string
- Initial value - none

httpDispatchThreads

The value is the number of threads that the broker dedicates to managing HTTP tunneling clients.

- Value type - Integer
- Initial value -32

httpProtocolTimeout

The value is the number of milliseconds in the HTTP protocol timeout interval. You can change this value to update the amount of time a broker is to wait for the next event during any phase of the HTTP tunneling protocol. A value of 0 causes the broker to wait indefinitely

- Value type - Integer
- Initial value -10000

interbrokerHost

Specifies the IP host name of the broker. A single broker configuration can be left to default as null.

```
mqsichangeproperties <broker> -o DynamicSubscriptionEngine -n
interbrokerHost -v <IP host name>
```

- Value type - String
- Initial value - null

If you change the value, the broker must be stopped and restarted. Then you must redeploy the full topology.

interbrokerPort

Port number on which the Broker listens for incoming interbroker connections. If running more than one broker on the same machine, set the **interbrokerPort** property to a different value for each broker. For example:

```
mqsichangeproperties <broker> -o DynamicSubscriptionEngine -n
interbrokerPort -v <port number>
```

If you do not set the **interbrokerPort** value before the topology is deployed, restart the broker.

- Value type - Integer
- Initial value -1507

If you change the value, the broker must be stopped and restarted. Then you must redeploy the full topology.

isysQopLevel

Applies to the system and allows brokers only to publish and subscribe.

- Value type - string
- Initial value - none

maxBrokerQueueSize

Maximum number of bytes that the broker should queue for transmission to another broker. If the maximum is exceeded, the broker deletes all messages queued to that broker, except for the latest message, high-priority messages, and responses. If 0, the broker does not limit the number of bytes queued to another broker:

- Value type - Integer
- Initial value - 1000000

maxClientQueueSize

Maximum number of bytes that the broker should queue for transmission to a client. If the maximum is exceeded, the broker deletes all messages queued to that client, except for the latest message, high-priority messages, and response messages. If 0, the broker does not limit the number of bytes queued to a client.

- Value type - Integer
- Initial value - 100000

The value of this property **must** be greater than, or equal to, the *maxMessageSize* value.

maxConnections

The maximum number of concurrently connected clients which the broker permits. If this limit is reached, the broker denies new connection requests from clients. If this value is less than 0, the number of clients is unlimited.

- Value type - Integer
- Initial value -100

maxHopCount

Specifies that maximum number of multibroker links over which a message is sent, to ensure that messages never loop in a multibroker network. Set this value large enough to ensure that messages can travel the entire multibroker network.

- Value type - Integer
- Initial value - 20

maxMessageSize

Maximum allowed message size in bytes. If a message exceeding this maximum size is received from a client, that client is disconnected.

- Value type - Integer
- Initial value - 100000

The value of this property **must** be less than, or equal to, the *maxClientQueueSize* value.

multicastAddressRangeMax

The highest IP address that the broker can use for its Multicast transmissions.

This should be in the range 224.0.0.2 through 239.255.255.255
- Value type - String
- Initial value - 239.255.255.255

multicastAddressRangeMin

The lowest IP address that the broker can use for its Multicast transmissions.

This should be in the range 224.0.0.2 through 239.255.255.255
- Value type - String
- Initial value - 224.0.0.2

multicastBackoffTimeMillis

The maximum time, in milliseconds, that a client listens for another's NACKs before sending its own NACK. This value can be in the range 0 through 1000.

- Value type -Integer
- Initial value -100

multicastDataPort

The UDP data port through which Multicast packets are sent and received:
- Value type - Integer
- Initial value - 34343

multicastEnabled

Indicates whether the topics that are defined in the multicastTopicsConfigFile are delivered Multicast. If the value is true, the topics in the multicastTopicsConfigFile are delivered Multicast.
- Value type - Boolean
- Initial value - false

multicastHeartbeatTimeoutSec

Indicates the time in seconds between the arrival of control packets at each client. If a control packet does not arrive within the number, defined as twice the value specified by this property, of seconds of the previous control packet's arrival an error can be suspected.
- Value type - Integer
- Initial value -20

multicastLimitTransRate

This property can be used in conjunction with the **multicastTransRateLimitKbps** property to control network congestion. Possible values are:

Disabled

Multicast data is transmitted as fast as possible

Static The transmission rate is limited by the value specified in **multicastTransRateLimitKbps**

Dynamic

The transmission rate can vary throughout the process, but never exceeds the value specified in **multicastTransRateLimitKbps**
- Value type - String
- Initial value - Disabled

multicastMaxKeyAge

Maximum age, in minutes, of a topic encryption key before it must be redefined.
- Value type - String
- Initial value - 360

multicastMaxMemoryAllowedKBytes

Maximum memory consumption by client reception buffers, measured in kilobytes.
- Value type - Integer
- Initial value - 262144

This parameter is available *only* if a Pragmatic General Multicast (PGM) protocol is selected.

multicastMCastSocketTTL

The maximum number of hops that a Multicast packet can make between the client and the broker. This value is one more than the maximum number of routers that there can be between the client and the broker.

A value of 1 indicates that the packet reaches all local nodes, but cannot be relayed by routers. The maximum value is 255.

- Value type - Integer
- Initial value - 1

multicastMinimalHistoryKBytes
>The minimum size, in kilobytes, of a buffer that is allocated as an archive for all transmitted packets. This buffer is shared by all reliable topics, and can be used to recover lost packets. This value can be in the range 1000 through 1,000,000.
>- Value type - Integer
>- Initial value - 60000

multicastMulticastInterface
>Indicates what interface to use for Multicast transmissions. You can specify a host name or an IP address. A value of "none" causes the network interface to be operating system dependent.
>- Value type - String
>- Initial value - "none"
>
>**Note:** If you have only one network card, the default value of none works because the operating system uses the `localhost` value. However, if you have more than one network card you **must** set this parameter to ensure that the correct card is used.

multicastNACKAccumulationTimeMillis
>The time, in milliseconds, that NACKs are aggregated in the broker, before recovered packets are sent. This value can be in the range 50 through 1000.
>- Value type - Integer
>- Initial value - 300

multicastNACKCheckPeriodMillis
>The time, in milliseconds, between periodic checks of reception status and sequence gap detection for NACK building. This value can be in the range 10 thru 1000
>- Value type - Integer
>- Initial value - 500

multicastOverlappingTopicBehavior
>This property is used to control the behavior of the broker when a client requests a Multicast subscription for a topic, that is part of a topic hierarchy containing topics, explicitly excluded for Multicast. Possible values are:
>
>**Accept**
>>A matching Multicast subscription is accepted and all publications matching the topic, except those that are specifically excluded, are multicast.
>
>**Reject** A Multicast subscription to a topic with children that are not enabled for Multicast is rejected by the broker.
>
>**Revert** Subscriptions to a topic, or to children of that topic, that are not enabled for Multicast result in unicast transmission.
>- Value type - String
>- Initial value - Accept

multicastPacketBuffers
>The number of memory buffers that are created at startup for packet reception. Having a high number of buffers available improves the reception performance and minimizes packet loss at high delivery rates, at

the price of increased memory utilization. Each buffer is 33KB and this value can be in the range 1 through 5000.
- Value type - Integer
- Initial value - 500

multicastPacketSizeBytes

Indicates the size, in bytes, of Multicast packets. This can be in the range 500 through 32000.
- Value type - Integer
- Initial value - 7000

multicastProtocolType

Specifies the protocol type. This must be one of:
- PTL
- PGM/IP
- PGM/UDP
- Value type - String
- Initial value - PTL

multicastSocketBufferSizeKbytes

The size, in kilobytes, of the client's socket receiver buffer. Increasing it leads to lower loss rates. This value can be in the range 65 through 10000.
- Value type - Integer
- Initial value - 3000

multicastTransRateLimitKbps

Limits the overall transmission rate in Kb (kilobits) per second.

This property is only effective if the **multicastLimitTransRate** property is not Disabled. The value of this property should not be set higher than the maximum data transmission rate of the system or the network and can be in the range 10 through 1,000,000.
- Value type - Integer
- Initial value - 9500

nonDurableSubscriptionEvents

Indicates whether the user requires event messages when a non-durable subscriber is created or deleted. A true value causes an event publication to be created, false indicates that no event publications are made.
- Value type - Boolean
- Initial value - False

pingTimeoutMultiple

Number of consecutive clientPngIntervals or brokerPngIntervals without a response that the broker waits before disconnecting a client or broker.
- Value type - Integer
- Initial value - 3

statsInterval

The value is the number of milliseconds between statistics publications. If set to 0, statistics publications are not generated. You do not need to restart the broker after changing this property, however it can take the broker up to a minute to start producing statistics after the value is changed.

This should be in the range 0 through 1000
- Value type - Integer
- Initial value - 0

> **Note:** This value refers to the publish/subscribe statistics interval only.

sysQopLevel
Applies to the system and allows brokers only to publish.
- Value type - string
- Initial value - none

Authorization

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On UNIX systems, the user ID used to invoke this command must either be **root** or must be the same as that specified in the `-i` parameter of the mqsicreatebroker command. It must also be a member of the **mqbrkrs** group.

Additionally, the authority required by the broker for the different protocols is as follows:

PGM/IP
The broker requires:
- `Root` authority on Linux and UNIX systems
- `Administrator` authority on Windows platforms
- `root` authority (UNIX System Services only) on z/OS

PGM/UDP
The broker requires `User` authority on all supported platforms

PTL The broker requires `User` authority on all supported platforms

httplistener **component parameter values**

Valid property names for the *HTTPListenerObjectName*:

enableSSLConnector
A boolean value which can be used to enable or disable the HTTPS (SSL) connector. This must be set to "true" before the http listener will listen for inbound SSL connections.
- Value type - Boolean
- Initial value - false

Valid property names for the *HTTPConnectorObjectName*:

address
For servers with more than one IP address, this value specifies which address is used for listening on the specified port. By default, this port is used on all IP addresses associated with the server. If specified, only one address can be used.
- Value type - string
- Initial value - null

port The TCP port number on which this Connector creates a server socket and awaits incoming connections.
- Value type - integer
- Initial value - 7080

Valid property names for the *HTTPSConnectorObjectName*:

address

> For servers with more than one IP address, this value specifies which address is used for listening on the specified port. By default, this port is used on all IP addresses associated with the server. If specified, only one address can be used.
>
> - Value type - string
> - Initial value - null

port   The TCP port number on which this Connector creates a server socket and awaits incoming connections.

> - Value type - integer
> - Initial value - 7083

algorithm

> The certificate encoding algorithm to be used.
>
> - Value type - string
> - Initial value - SunX509 on Solaris and HPUX, and IbmX509 on other platforms (AIX, z/OS, Linux, Windows).

clientAuth

> Set to true if the SSL stack requires a valid certificate chain from the client before accepting a connection. A false value (which is the default) does not require a certificate chain unless the client requests a resource protected by a security constraint that uses CLIENT-CERT authentication.
>
> - Value type - string
> - Initial value - false

keystoreFile

> The path to the keystore file where the server certificate, which is to be loaded, has been stored. By default the http listener will expect a file called .keystore in the home directory of the user who started the broker.
>
> - Value type - string
> - Initial value - *default value* (described above)

keystorePass

> The password used to access the server certificate from the specified keystore file.
>
> - Value type - string
> - Initial value - changeit

keystoreType

> The type of keystore file to be used for the server certificate.
>
> - Value type - string
> - Initial value - JKS

sslProtocol

> The version of the SSL protocol to use.
>
> - Value type - string
> - Initial value - SSLv3

ciphers

> A comma separated list of the encryption ciphers that may be used. If not specified (the default), then any available cipher may be used.
>
> - Value type - string
> - Initial value - null

The possible values for *keystoreType*, *sslProtocol*, and *ciphers* are JSSE-implementation specific, and these values should be found in the JSSE provider documentation.

Authorization

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On UNIX systems, the user ID used to invoke this command must either be **root** or must be the same as that specified in the `-i` parameter of the mqsicreatebroker command. It must also be a member of the **mqbrkrs** group.

Examples:

Change the clientPingInterval to 200

```
mqsichangeproperties TEST -o DynamicSubscriptionEngine -n clientPingInterval -v 200
```

Enable multicast

```
mqsichangeproperties TEST -o DynamicSubScriptionEngine -n multicastEnabled -v true
```

Change the default ssl protocol from SSLv3 to TLS

```
mqsichangeproperties TEST -b httplistener -n sslProtocol -v TLS
```

Enable the HTTPS Connector

```
mqsichangeproperties TEST -b httplistener -o HTTPListener -n enableSSLConnector -v true
```

mqsichangetrace command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsichangetrace** command to set the tracing characteristics for a component. This command is valid for:
- User trace. Specify the `-u` option.
- Service trace. Specify the `-t` option. You are recommended to use this option only if directed to do so by the action described in a BIPxxxx message, or by your IBM Support Center.

You can initiate, modify, or terminate user tracing for a broker, or initiate, modify, or terminate service tracing for a broker, a Configuration Manager, or the User Name Server (identified by component name). You cannot use this command to initiate service tracing for the workbench.

On Windows platforms, Linux, and UNIX systems, you can also start and stop tracing activity for execution groups and message flows using the facilities of the workbench. See User trace for more information.

On z/OS, you can also start, modify, or stop a trace using the console commands or using the facilities of the workbench.

If you specify a broker, or any of its resources (execution group or message flow), you must have deployed them before you can start trace.

The trace output generated by these commands is written to trace files in the `log` subdirectory. When you have completed the work you want to trace, use mqsireadlog to retrieve the log as an XML format file. Use either **mqsiformatlog** (to produce a formatted file) or an XML browser to view the XML records.

When you set tracing on, you cause additional processing to be executed for every activity in the component you are tracing. You must expect to see some impact on performance when trace is active.

If you want to trace the command executables themselves, set the environment variables `MQSI_UTILITY_TRACE` and `MQSI_UTILITY_TRACESIZE` before you initiate trace.

Ensure that you reset these variables when tracing for the selected command is complete. If you do not do so, all subsequent commands are also traced, and their performance degraded.

Syntax:

Windows platforms, Linux, and UNIX systems:
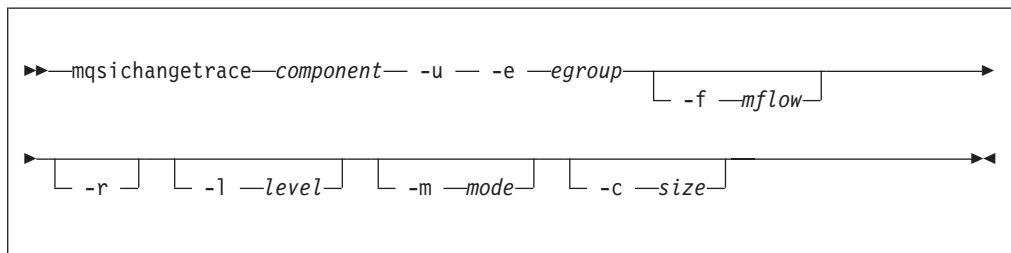
User trace:

Service trace:
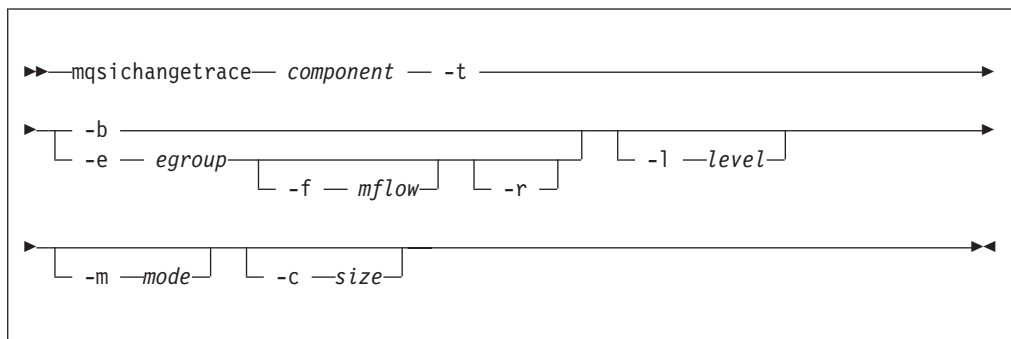
z/OS:

User trace:
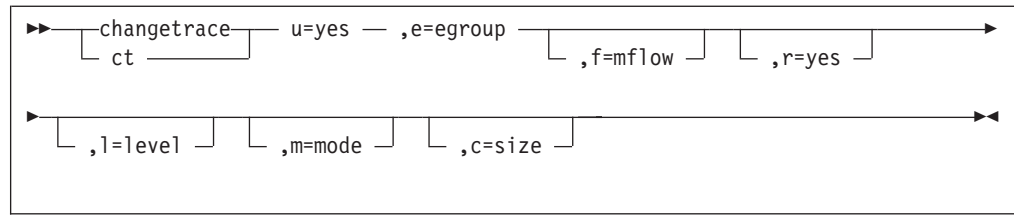
```
►►─┬─changetrace─┬─── u=yes ── ,e=egroup ──────────────────────────────────►
   └─ ct ────────┘                    └─ ,f=mflow ─┘  └─ ,r=yes ─┘

►─────────────────────────────────────────────────────────────────────────►◄
    └─ ,l=level ─┘  └─ ,m=mode ─┘  └─ ,c=size ─┘
```

Service trace:

```
►►─┬─changetrace─┬── t=yes ──┬─ ,b=yes ──────────────────────────────────────►
   └─ ct ────────┘           └─ ,e=egroup ──────────────────────────────
                                          └─ ,f=mflow ─┘  └─ ,r=yes ─┘

►──────────────────────────────────────────────────────────────────────────►◄
    └─ ,l=level ─┘  └─ ,m=mode ─┘  └─ ,c=size ─┘
```
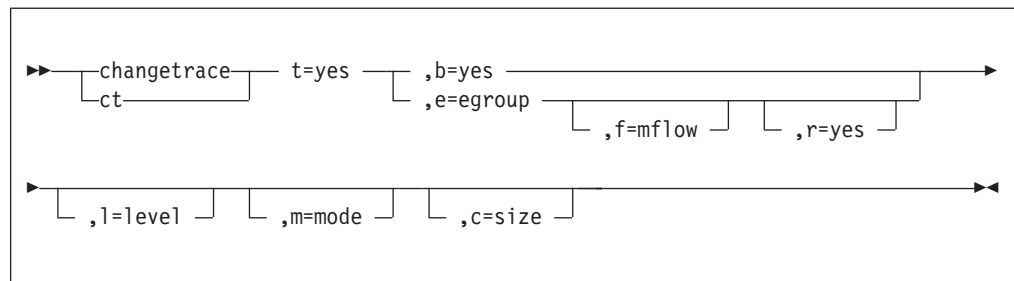
Parameters:

component

(Required - Windows platforms, Linux, and UNIX systems) The name of the
component for which trace parameters are to be changed. This can either be
the name of a broker, or of a Configuration Manager, or the fixed values ,
UserNameServer, workbench, or utility (all are case sensitive on UNIX
systems).

-u (Optional) Specifies that user trace options are to be modified. This option is
only valid if you have:
- Specified a broker name as the component name on Windows platforms,
  Linux, and UNIX systems.
- Issued this command against a broker (that is, not a Configuration Manager
  or User Name Server) on z/OS.

.

-e *egroup*

(Optional) Identifies the execution group for which trace options are to be
modified (for example, started or stopped). This option is only valid for a
broker.

On z/OS, this name is case sensitive and you should include the names in
single quotes if they contain mixed case characters.

-f *mflow*

(Optional) Identifies the message flow for which trace options are to be
modified. This option is only valid if you have specified an execution group
(flag **-e**).

On z/OS, this name is case sensitive and you should include the names in
single quotes if they contain mixed case characters.

-r (Optional) This option requests that the trace log is reset: that is, all current
records are discarded. Use this option when you start a new trace to ensure
that all records in the log are unique to the new trace.

This option is only valid if you have specified an execution group (flag **-e**).

-l *level*

(Optional) Set the level of the trace. This must be one of:

- `normal`. This provides a basic level of trace information.
- `none`. This sets tracing off.
- `debug`. This provides a more comprehensive trace.

Each component is created with a default value of `none`. If you do not specify this parameter, the current value is unchanged. Once you successfully change this value, it is persistent.

This is valid for all components.

-m *mode*

(Optional) Indicate the way trace information is to be buffered:

- `safe`. This mode causes trace entries to be written to file when they are generated.
- `fast`. This mode causes trace entries to be buffered, and only written to file in batches.

Each component starts with a default value of `safe`. If you do not specify this parameter, the current value is unchanged.

This option is only valid if the component you have specified is:

- A broker. If you change this value, it affects tracing for the execution group (if you have specified one), or for the agent component (if you have not specified an execution group).
- The User Name Server. If you change this value, it affects tracing for the entire component. (This is only valid for service trace.) Once you successfully change this value, it is persistent.

-c *size*

(Optional) The size of the trace file in KB (kilobytes). If you do not specify this parameter, the current value is left unchanged. Each component starts with a default value of 4096 KB. Specify this option to reset the value. The maximum value you can specify depends on how you subsequently intend to read the log, using the **mqsireadlog** command.

- If you use **mqsireadlog** with the `-f` option set, the log file is read directly from the file system. In this case, the maximum value that can be specified here is 2097151, which will allow a trace file up to 2 GB (gigabyte) to be created.
- If you use **mqsireadlog** without setting the `-f` option, a WebSphere MQ message is sent to the broker to retrieve the log. In this case, the trace file size should not exceed 70 MB (megabytes). The maximum value that you can set here, should not be appreciably more than 70000.

However you intend to retrieve the trace file, you are recommended to keep its size as small as possible, either by using a low value for this parameter or by using the reset (`-r`) option on this command to clear the trace log. The benefit of adopting this approach is that the formatting process (**mqsiformatlog**) is much faster and requires less resource to carry out its task.

This option is only valid if the component you have specified is:

- A broker. If you change this value, it affects tracing for the execution group (if you have specified one), or for the agent component (if you have not specified an execution group).

- The User Name Server. If you change this value, it affects tracing for the entire component. (This is only valid for service trace.)

If you change the trace size, the new value is persistent over a restart of the broker or User Name Server.

Additional parameters exclusive to service trace:

Use these options only when directed to do so by your IBM Support Center or by a `BIPxxxx` message.

-t   (Optional) Specifies that service trace options are to be modified.

-b   (Optional) Specifies that service trace options for the `agent` subcomponent of the component specified are to be modified (for example, started or stopped). You can only specify this flag if `-t` is also specified.

Authorization:

The user ID used to issue the command must have **mqbrkrs** authority.

Responses:

This command returns the following responses:
- BIP2595 Error casting character string '...' to an integer (z/OS only)
- BIP8002 Selected flags incompatible (z/OS only)
- BIP8003 Duplicate flag detected (z/OS only)
- BIP8013 Component does not exist
- BIP8020 Unable to access database
- BIP8029 Broker not configured
- BIP8031 Invalid flag supplied
- BIP8032 Unable to connect to queue
- BIP8033 Message send failure
- BIP8035 Response not received before time-out
- BIP8036 Negative response received
- BIP8037 Unsupported flag
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database
- BIP8045 Message flow not found
- BIP8068 Integer argument required
- BIP8158 Invalid format for command (z/OS only)
- BIP8159 Unknown parameter "..." (z/OS only)

Examples:

Windows platforms, Linux, and UNIX systems:

To collect and process a user trace for the default execution group use the command:
```
mqsichangetrace WBRK_BROKER -u -e default -l normal -c 5000
```

To collect and process a service trace for flow f1 in the default execution group use the command:

```
mqsichangetrace WBRK_BROKER -u -e default -m fast
```

To collect and process a service trace for an agent use the command:

```
mqsichangetrace WBRK_BROKER -t -b -m -l normal
```

z/OS:

To collect and process a user trace for the default execution group use the command:

```
F MQP1BRK,ct u=yes,e='default',l=normal,c=5000
```

and in the PDSE member BIPJLOG, set the option for **mqsireadlog** to

```
-u -e default
```

To collect and process a service trace for flow f1 in the default execution group use the command:

```
F MQP1BRK,ct u=yes,e='default',f='f1',m=fast
```

and in the PDSE member BIPJLOG, set the option for **mqsireadlog** to

```
-t -e default -f f1-
```

To collect and process a service trace for an agent use the command:

```
F MQP1BRK,ct t=yes,b=yes,m=fast,l=debug
```

and in the PDSE member BIPJLOG, set the option for **mqsireadlog** to

```
-t -b agent
```

mqsichangeusernameserver command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

On Windows platforms, Linux, and UNIX systems, use the **mqsichangeusernameserver** command to change various properties of the User Name Server.

On z/OS, use the **mqsichangeusernameserver** command to change the refresh interval and authentication properties of the User Name Server

You must stop the User Name Server, using **mqsistop** on Windows platforms and UNIX platforms, or **stopcomponent** on z/OS, (see "mqsistop command" on page 328) before you can issue this command.

When you restart the User Name Server, using **mqsistart** on Windows platforms and UNIX platforms, or **startcomponent** on z/OS, (see "mqsistart command" on page 323) it uses the changed parameters.

Syntax:

Windows platforms, Linux and UNIX systems:

```
►►──mqsichangeusernameserver────────────────────────────────────────────────►

►──────────────────────────────────────────────────────────────────────────►
    └─ -a ──ServicePassword─┬──────────────────────┬─┘
                            └─ -i ──ServiceUserID──┘

►──────────────────────────────────────────────────────────────────────────►
    └─ -d ──SecurityDomainName─┘  └─ -r ──RefreshInterval─┘

►─────────────────────────────────────────────────────────────────────────►◄
    └─ -g ──AuthProtocolDataSource─┘  ┌─ -j ─┐
                                      └─ -o ─┘
```

z/OS console command:

Synonym: cu

```
►►─┬─changeusernameserver─┬── r=RefreshInterval ────────────────────────────►
   └─cu─────────────────┘

►─────────────────────────────────────────────────────────────────────────►◄
   └─g=AuthProtocolDataSource─┘  ┌─j ─┐
                                 └─o ─┘
```

Parameters:

-a *ServicePassword*

(Optional - Windows platforms,Linux, and UNIX systems) The password for the `ServiceUserID`.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

On Linux and UNIX systems, **-a** is required for Windows platforms compatibility but it is not used in relation to `ServiceUserID`.

-i *ServiceUserID*

(Optional - Windows platforms, Linux, and UNIXsystems) The user ID under which the broker runs. You can only change this value if you also change the password.

This can be specified in any valid username syntax. On Windows platforms, these are:

- `domain\username`
- `\\server\username`
- `.\username`
- `username`

On Linux and UNIX systems, only the last format, `username`, is valid.

If you use the unqualified form for this user ID (`username`) in Windows, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The `ServiceUserID` specified must be a member of the local group **mqbrkrs**. On Windows platforms, it can be an indirect or direct member of the group. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the **mqsicreateusernameserver -w** flag). This ID must also be a member (either direct or indirect) of the local group **mqm**.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 389 for Windows platforms and in "Security requirements for Linux and UNIX platforms" on page 388 for UNIX platforms.

-d *SecurityDomainName*
    (Optional - Windows platforms, Linux, and UNIX systems) The name of the security domain on the Windows platform.

-r *RefreshInterval*
    (Optional - Windows platforms, Linux, and UNIX systems) (Required - z/OS) The interval, in seconds, at which the User Name Server interrogates the security subsystem for changes to user or group attributes.

-g *AuthProtocolDataSource*
    (Optional) Use this parameter to specify the name of the data source required by the authentication protocol.

-j (Optional) Indicates that groups and group memberships are defined in the data source for the authentication protocol, rather than being drawn from the operating system.

-o (Optional) Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.

Responses:

This command returns the following responses:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8018 Component running
- BIP8021 User ID/password incorrect
- BIP8022 Invalid user ID/password
- BIP8023 Password required
- BIP8030 Unable to modify user ID privileges
- BIP8068 Integer argument required

Examples:

Windows platforms, Linux, and UNIX platforms:
```
mqsichangeusernameserver -r 2000
```

z/OS:
```
F MQP1UNS,cu r=2000
```

mqsiclearmqpubsub command

Supported platforms:
* Windows
* Linux and UNIX systems
* z/OS through the BIPCLMP utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsiclearmqpubsub** command to remove an WebSphere MQ Publish/Subscribe broker as a neighbor of this WebSphere Event Broker broker.

This command removes knowledge of the WebSphere MQ Publish/Subscribe broker from the WebSphere Event Broker broker identified on this command. To complete this action you must also issue the WebSphere MQ Publish/Subscribe command **clrmqbrk** against the WebSphere MQ Publish/Subscribe broker. When both clear commands have completed, all publish/subscribe traffic between the two brokers ceases.

Use this command only if you are integrating this WebSphere Event Broker broker with an WebSphere MQ Publish/Subscribe broker network. Before you issue this command you must ensure that the WebSphere Event Broker broker is ready to receive and process messages on queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS (that is, you must restart the broker after creating this queue.)

Syntax:

```
►►──mqsiclearmqpubsub─brokername── -n ─NeighborQueueManagerName──────────────►◄
```

Parameters:

brokername
>   (Required) The name of the broker from which knowledge of an WebSphere MQ Publish/Subscribe neighbor broker is to be removed.

-n *NeighborQueueManagerName*
>   (Required) The name of the queue manager that hosts the WebSphere MQ Publish/Subscribe broker for which the association as a neighbor is being removed.

Authorization:

The user ID used to invoke this command must have `put` and `inq` authority to the queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Responses:

This command returns the following responses:
* BIP8013 Component does not exist

- BIP8056 Unknown queue manager
- BIP8057 Queue manager error
- BIP8059 Queue manager not available
- BIP8060 Queue error
- BIP8061 No reply received
- BIP8064 Internal broker error
- BIP8066 Invalid broker name
- BIP8070 Database exception
- BIP8072 Database exception

Examples:

```
mqsiclearmqpubsub WBRK_BROKER -n MQBroker1
```

mqsicreateaclentry command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsicreateaclentry** command to create or modify the Configuration Manager data relating to the group or user access control lists that you have defined.

If you create or modify an access control group you must stop and restart the Configuration Manager for the change to take effect.

Note that this command does not check for the existence of the specified component in case you want to set up the access control list first.

Syntax:

Windows:

Linux and UNIX systems:

```
►►──mqsicreateaclentry──configmgrName────────────────────────────────────────►
                              └─-n──ConnectionFileName─┘

►──┬─-f─FileName──────────────────────────────────────────────────┬───────────►
   ├─-g─GroupName──────────────────────┬──-x──AccessLevel──┬─-b──Broker──────┐
   └─-u──UserName──┬─-m──MachineName─┐  │                  │         └─-e──ExeGroup─┘
                   └─-a─────────────┘  │                  ├─-s─────────────┤
                                       │                  ├─-r─────────────┤
                                       │                  ├─-t─────────────┤
                                       │                  └─-p─────────────┘

►──┬────────────────┬─────────────────────────────────────────────────────►◄
   └─-w──waitTime──┘
```

z/OS console command:

```
►►──mqsicreateaclentry──configmgrName────────────────────────────────────────►
                              └─n=ConnectionFileName─┘

►──┬─f =FileName─────────────────────────────────────────────────┬────────────►
   ├─g=GroupName──────────────────────┬──x=AccessLevel──┬─b=Broker────────┐
   └─u=UserName──┬─m=MachineName─┐     │                │       └─e=ExeGroup─┘
                 └─a────────────┘      │                ├─s=─────────────┤
                                       │                ├─r=─────────────┤
                                       │                ├─t=─────────────┤
                                       │                └─p=─────────────┘

►──┬────────────────┬─────────────────────────────────────────────────────►◄
   └─w=waitTime──┘
```
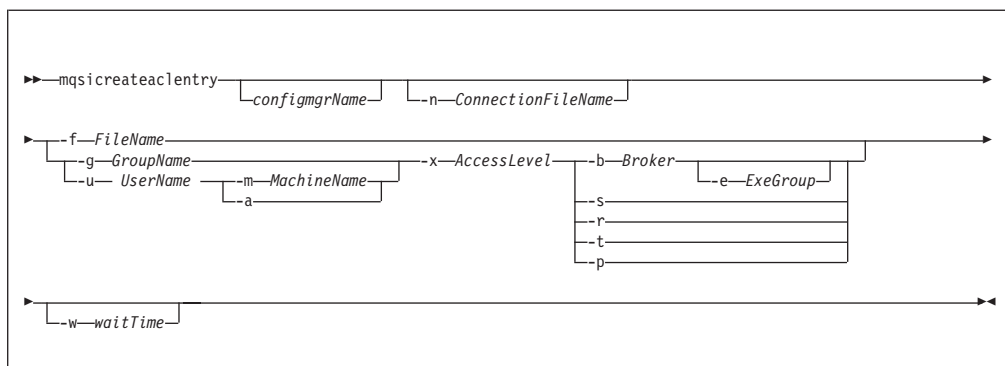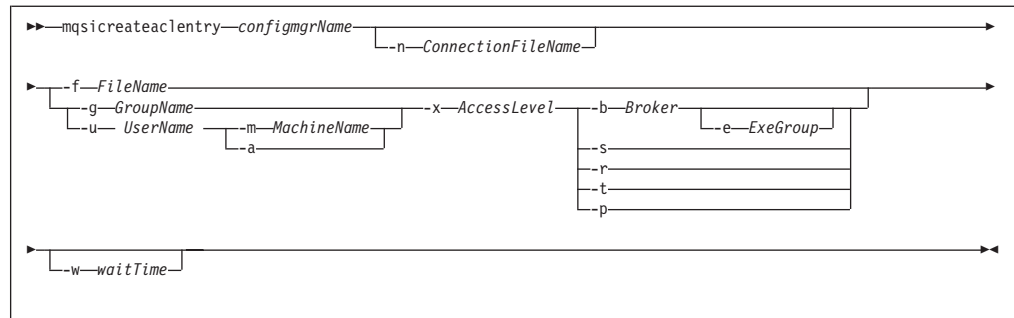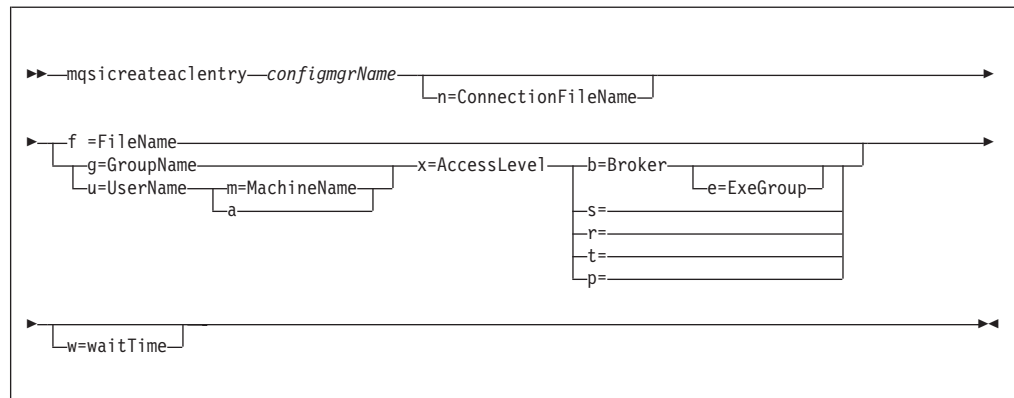
Parameters:

You **must** select either **-f** or

- **-g** or **-u**, and
- **-x**, and
- **-b** and **-e** or **-s** or **-r** or **-t** or **-p**

If you select **-u** you must select either **-m** or **-a**

configmgrName

(Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name
of the Configuration Manager to which the access control lists are to be added.

On Linux, UNIX systems and z/OS, this must be the first parameter specified.
It is case-sensitive on Linux and UNIX systems.

The default name on Windows, if this parameter is not specified, is
'ConfigMgr'.

-n*connectionfileName*

(Optional) The name of a `.configmgr` file that describes the connection
parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For
example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

If you are using this file on z/OS you must remove the statement
`encoding="UTF-8"` from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

-f *FileName*
> (Optional) File from which to import the configuration. The output of the **mqsilistaclentry** command is the correct format.

-g *GroupName*
> (Optional) Local group to which this entry refers. For this reason, the name must adhere to the standard platform convention for group names.
>
> To add a domain group, grant authority to a local group and then add the domain group, or groups, that you want to authorize into that local group. Any members of those domain groups obtain the permissions of the local group indirectly.

-u *UserName*
> (Optional) User name to which this entry refers, for example, TEST\ANOTHER.
>
> **Note:** **-u** and **-g** in this command refer to users and groups within the domain that the Configuration Manager uses for its security. This domain is by default the machine on which the Configuration Manager resides.

-m *MachineName*
> (Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.

-a (Optional) This signifies that the specified user name can be on any machine. This option can not be used with **-m**.

> **Note:** If you select **-u**, you must select either **-a** or **-m**.

-x *AccessLevel*
> (Optional) The required access level given for this group. This can be any combination of the letters FDEV, where:
>
> **F**   Full control
>
> **D**   Deploy
>
> **E**   Edit
>
> **V**   View

-b *Broker*
> (Optional) The object is a broker object, and its name is specified as a parameter.

-e *ExeGroup*
> (Optional) The object is an execution group and its name is specified as a parameter of the form Broker\ExeGroup. You must specify the **b** flag if you specify this flag.

-s *Subscription*
> (Optional) The object is a subscription object, and its name is specified as a parameter.

-r (Optional) The object refers to the root topic.

-t (Optional) The object refers to the main topology.

-p (Optional) The object refers to the "allresources" resource type. The authority that the principal has for this object applies to all objects, including the mqsicreateaclentry, mqsideleteaclentry, and mqsilistaclentry commands themselves.

-w *waitTime*

(Optional) The time in seconds that the command waits for a response from the Configuration Manager. If you do not supply a value the command waits for 30 seconds.

Authorization:

The user ID used to invoke this command must have full control permissions for the object being changed; see "ACL permissions" on page 385 for more information.

Examples:

```
mqsicreateaclentry CMGR01 -f c:\test\mylist
mqsicreateaclentry CMGR01 -g GROUPA -x F -b MYBROKER
```

mqsicreatebroker command

Supported platforms:
* Windows
* Linux and UNIX systems
* z/OS

Purpose:

The **mqsicreatebroker** command:
* Creates a WebSphere MQ queue manager, except on z/OS, if one does not already exist.

> **Note:**
>
> 1. If a WebSphere MQ queue manager is created as a result of using the **mqsicreatebroker** command, the default DLQ provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is automatically enabled. The security settings are the same as those of other broker-specific WebSphere MQ queues.
>
>   If you choose to create the queue manager separately, you must set up a dead letter queue (DLQ). The DLQ is referenced by WebSphere Event Broker when errors occur processing messages in message flows.
>
>   If a message in either a user-defined message flow or in the publish/subscribe model cannot be processed, it is routed to this DLQ as a last resort. If you prefer to back out the message onto the input queue, effectively halting the message flow until the problem is resolved, disable the DLQ.
>
>   The **mqsideletebroker** command does not delete this queue (unless the queue manager is deleted).
>
> 2.
>
>   If you are using a WebSphere MQ queue manager that has been created independently of the **mqsicreatebroker** command, you can define clusters if you choose. This simplifies your configuration in most cases.

- Starts the WebSphere MQ queue manager, except on z/OS, if it is not already running.

  If the queue manager is created by this command, it is not started as a Windows service; it will therefore stop if you log off. To avoid this happening, you must either remain logged on, or change the startup status of the queue manager service. (If you lock your workstation, the WebSphere MQ queue manager does not stop.)
- Creates the broker-specific WebSphere MQ queues, if these do not already exist.
- Creates database tables for the broker, if they do not already exist, or adds rows specific to this broker to existing tables.
- On Windows platforms, installs a service under which the broker will run.
- Creates a record for the component in the broker registry.

Syntax:

Windows platforms, Linux, and UNIX systems:

z/OS console command:

```
>>--mqsicreatebroker--brokername-- -q--QueueManagerName----------------------->

  >--+---------------------------------------------------------------+-->
     '- -s--UserNameServerQueueManagerName-+----+-+----+-+----+-+------+-'
                                           '-1--' '-2--' '-3--' '- -j-'

  >--+----------------------------+--+--------------------------------+-->
     '- -g--ConfigurationTimeout--'  '- -k--ConfigurationDelayTimeout-'

  >--+-----------------------------+--+-------------------------------------+-->
     '- -v--statisticsarchiveinterval-' '- -y--LdapPrincipal---z--LdapCredentials-'

  >--+----------------------+-><
     '- -c --ICUDataPath----'
```
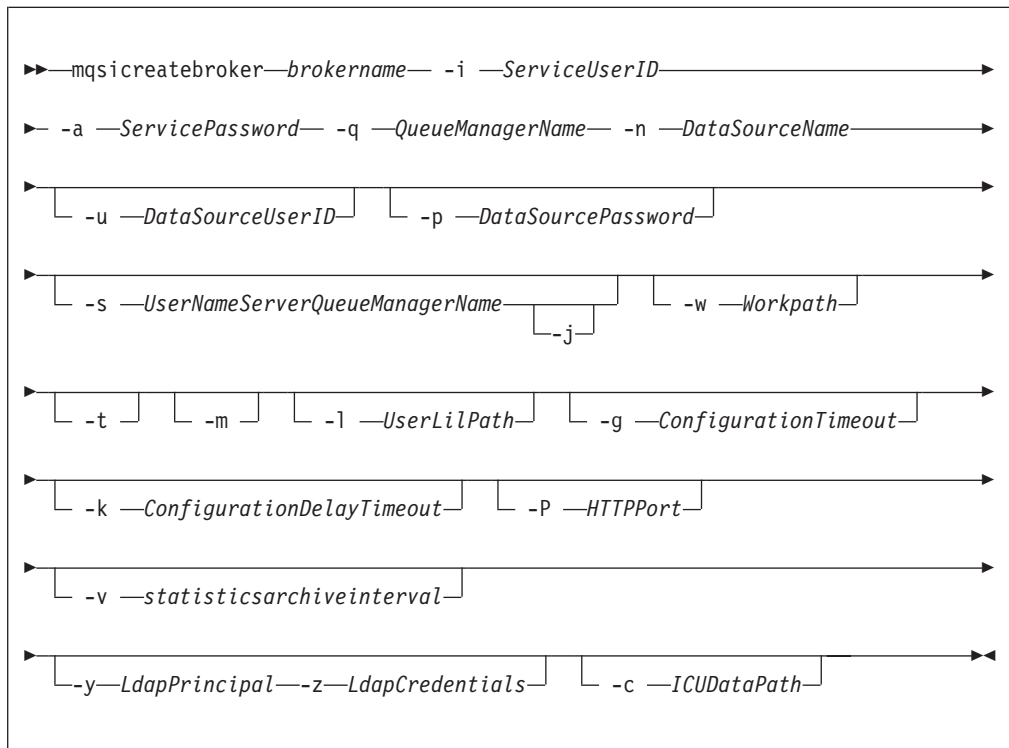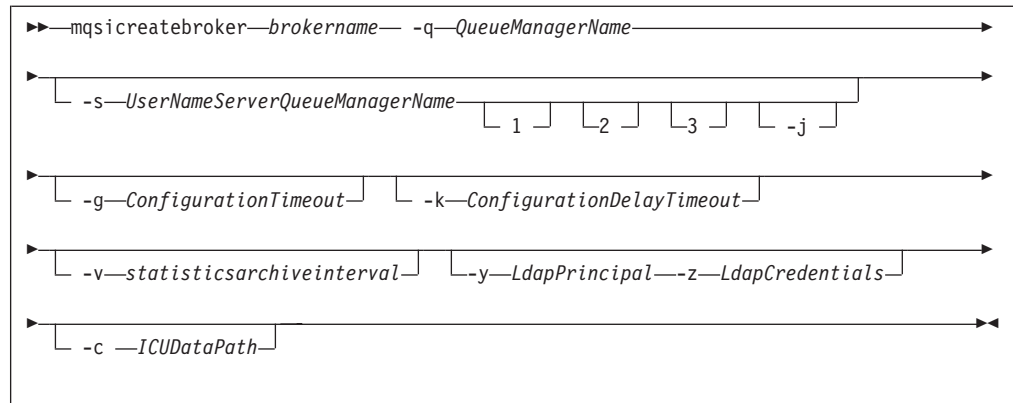
Parameters:

brokername
(Required) The name of the broker that you want to create. This must be the first parameter, and it is case sensitive on Linux and UNIX systems.

On z/OS, if you create an uppercase broker name, you must use this name in uppercase also for your broker in the workbench.For restrictions on the character set that can be used, see "Characters allowed in commands" on page 217.

-i *ServiceUserID*
(Required - Windows platforms, Linux, and UNIX systems) The user ID under which the broker will run.

This can be specified in any valid username syntax. On Windows platforms these are:
- `domain\username`
- `\\server\username`
- `.\username`
- `username`

On Linux and UNIX systems, only the last format, `username`, is valid.

If you use the unqualified form for this user ID (`username`) on Windows platforms, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The `ServiceUserID` specified must be a member of the local group **mqbrkrs**. On Windows platforms, it can be a direct or indirect member of the group. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the **-w** flag).

If you specify, on Windows platforms, that the broker is to run as a WebSphere MQ trusted application (flag **-t**), you must also add this user ID to the group **mqm**. On Linux and UNIX systems, specify the ServiceUserID as `mqm` if you set the **-t** flag.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 389.

If you use this user ID for database access (that is, you do not specify a different user ID with the **-u** flag) and you are using SQL Server for your database, you must create this user ID as an SQL Server login ID and give it

the correct access before you create the broker (see "Considering security for a broker" on page 12 for further details). If your broker database exists in DB2, and this user ID is not known to DB2, DB2 automatically creates if for you.

-a *ServicePassword*

(Required - Windows platforms, Linux, and UNIX systems) The password for the ServiceUserID.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

On Linux andUNIX systems **-a** is required for Windows platforms compatibility, but is not used in relation to ServiceUserID; it is used as a default only if **-p** is not specified. (See notes about the **-p** parameter for further details.)

-q *QueueManagerName*

(Required) The name of the queue manager associated with this broker. Use the same name for your broker and the queue manager to simplify the organization and administration of your network. Queue manager names are limited to 48 characters in length and are case sensitive.

If the queue manager does not already exist, it is created by this command. It is not created as the default queue manager; if you want this queue manager to be the default queue manager on this system, you must either create the queue manager before you issue this command, or use WebSphere MQ Services to change the settings of this queue manager to make it the default.

The queue manager attribute `MAXMSGLN` (maximum length of messages that can be put to queues) is updated to 100 MB. This is done whether or not the queue manager is created by this command.

For restrictions on the character set that can be used, see "Characters allowed in commands" on page 217.

-n *DataSourceName*

(Required - Windows platforms, Linux, and UNIX systems) The ODBC data source name (DSN) of the database in which the broker tables are created. This must be the DSN, not the name of the database, if you have not used the same name for both.

This database must already exist. You must create a System DSN ODBC connection for this DSN, if you have not already done so.

If you have a DB2 database on Linux, enter the appropriate DB database alias name; an ODBC DSN is not required.

-u *DataSourceUserID*

(Optional - Windows platforms, Linux, and UNIX systems) The user ID with which databases containing broker tables and user data are to be accessed. If this is not specified, it defaults to the value specified by **-i**.

This user ID must have the authority to create tables within this database, and read from and write to those tables.

On Windows platforms, if your broker database exists in DB2, and this user ID is not known to DB2, it is created for you within DB2. On Linux and UNIX systems, the service user must have previously been granted the correct privilege. If your database is SQL Server, you must create this user ID as an

SQL Server login ID and give it the correct access before you create the broker (see "Security requirements for Windows platforms" on page 389 for further details).

If you have an application database in DB2 that was created by this user ID, or to which this user ID has appropriate read, write, or create authority, message flows executing in this broker can access and manipulate the application data held within it without having to specify explicit schema names.

-p *DataSourcePassword*
(Optional - Windows platforms, Linux and UNIX systems) The password of the user ID with which databases containing broker tables and user data are to be accessed. If not specified, this defaults to the `ServicePassword` specified by `-a`.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

For DB2 on Linux and UNIX systems, `-u` and `-p` can be specified as empty strings (two double quotation marks "") . In this case, DB2 grants WebSphere Event Broker the privileges of the `ServiceUserID`, which results in a database connection as "already verified". If you specify `-a` as an empty string as well as `-u` and `-p`, no passwords are stored by WebSphere Event Broker, creating the most secure configuration.

-s *UserNameServerQueueManagerName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server. You **must** specify this parameter if you require either authentication services or publish/subscribe access control. If this is not specified, the broker assumes that there is no User Name Server defined.

-j (Optional) You **must** specify this flag in conjunction with the **-s** *UserNameServerQueueManagerName* parameter to enable publish/subscribe access control.

-w *Workpath*
(Optional - Windows platforms, Linux, and UNIX systems) The directory in which working files for this broker are stored. If this is not specified, files are stored in the default workpath, specified when the product was installed. If specified, you must create this directory before you start the broker. On Windows platforms, this cannot be on a networked drive.

This directory is also used for trace records created when tracing is active. These are written to a subdirectory `log`, which you must create before you start the broker.

Error logs written by the broker when a process terminates abnormally are stored in this directory. On Windows platforms, use this option, to specify a directory on a drive other than the one on which the product is installed.

The error log is unbounded and continues to grow. Check this directory periodically and clear out old error information.

You cannot change this option using the **mqsichangebroker** command. If you want to specify, or change, the workpath, delete and recreate the broker.

-t (Optional - Windows platforms, Linux, and UNIX systems) The broker is configured to run as a WebSphere MQ trusted application.

This option is not available on AIX. If specified, the flag is ignored.

If you specify this option on Windows platforms, add the `ServiceUserID` (identified by flag `-i`) to the group **mqm**. If you specify this option on HP-UX and Solaris, specify the `ServiceUserID` as `mqm`. For more details about using WebSphere MQ trusted applications, see *WebSphere MQ Intercommunication*.

-m (Optional -Windows platforms, Linux, and UNIX systems) Migrate an existing WebSphere MQ Publish/Subscribe broker. If you specify this option, the queue manager identified by **-q** must be the queue manager being used by the WebSphere MQ Publish/Subscribe broker.

-l *UserLilPath*

(Optional - Windows platforms, Linux, and UNIX systems) A list of paths (directories) from which the broker loads LILs (loadable implementation libraries) for user-written plug-in message processing nodes.

You must create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Event Broker install directory.

If you specify more than one additional directory, they must be separated by the default platform-specific path separator (semicolon (`;`) on Windows platforms, colon (`:`) on Linux and UNIX systems).

You cannot include environment variables in this path: if you do so, they are ignored.

-g *ConfigurationTimeout*

(Optional) The length of time (in seconds) that an execution group in the broker is allowed to take to apply a change in configuration (for example, an update that you have deployed from the workbench).

When a message flow is processing an application message, it cannot respond to a configuration change. If any one of the message flows within the execution group that has been requested to change its configuration does not finish processing an application message and apply the configuration change within this timeout, the execution group returns a negative response to the deployed configuration message.

The value that you set for this timeout depends on the system load (including CPU utilization) and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives you an indication of the minimum value to set.

The value is specified in seconds and can range from 10 to 3600. The default value is 300.

The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* (described below) represents the maximum length of time that a broker is allowed to process a deployed configuration message before it generates a negative response.

-k *ConfigurationDelayTimeout*

(Optional) The length of time (in seconds) that a broker is allowed to take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).

This represents the time it takes for a minimal deployed configuration message to be processed by the broker and its execution groups, and depends on queue manager network delays, the load on the broker's queue manager, and system load.

You can estimate this value by issuing a command to request a simple configuration change, for example:

- On Windows platforms, Linux, and UNIX systems

  ```
  mqsireporttrace brokerName -e "Execution Group Name" -u
  ```

- On z/OS

  ```
  F MQP1BRK,reporttrace u=yes,e='exgrp1'
  ```

The response time of each execution group differs according to system load and the load of its own processes. The value that you set must reflect the longest response time that any execution group takes to respond. If the value that you set is too low, the broker returns a negative response, and might issue error messages to the local error log.

The value is specified in seconds and can range from 10 to 3600. The default value is 60.

If the broker is on a production system, increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the time-out values decreases the probability of successfully deploying a configuration change.

-P *HTTPPort*
(Optional - Windows platforms, Linux, and UNIX systems) Enter the number of the port on which the Web services support is listening.

This listener is started by the broker when a message flow that includes Web services support is started, and has a default value of 7080.

Ensure that the port that you specify has not been specified for any other purpose.

-v *statisticsarchiveinterval*
(Optional) The timer interval in minutes at which WebSphere Event Broker statistics and accounting is told to output archive records. For internal accounting, the valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the platform has an external method of notification and is not using an internal timer within WebSphere Event Broker.

1 (Optional - z/OS only) The registry pass, which creates only the broker registry.

2 (Optional - z/OS only) The WebSphere MQ pass, which creates only the broker WebSphere MQ queues.

3 (Optional - z/OS only) The DB2 pass, which creates only the broker DB2 tables and indexes.

-y *LdapPrincipal*
(Optional - all platforms, but mandatory when *LdapCredentials* is provided.) The user principal for access to an optional LDAP directory that holds the JNDI administered Initial Context for the JMS provider.

-z *LdapCredentials*
> (Optional - all platforms, but mandatory when *LdapPrincipal* is provided.) The user password for access to LDAP.

-c *ICUDataPath*
> (Optional) A delimited set of directories to search for additional code page converters. On Windows systems the delimiter is ";". On UNIX and Linux systems, the delimiter is ":". The code page converters must be either of the form icudt32_*codepagename*.cnv or in an ICU data package called icudt32.dat.

Authorization:

On Windows systems, the user ID used to invoke this command must have **Administrator** authority on the local system.

On UNIX systems, the user ID used to invoke this command must be a member of the **mqbrkrs** group.

On z/OS systems, the user ID used to invoke this command must be a member of a group that has READ and WRITE access to the component directory.

Using LDAP: Ensure that the registry is appropriately secured to prevent unauthorized access. The setting of *LdapPrincipal* and *LdapCredentials* options on **mqsicreatebroker** is not required for correct operation of the broker. The password is not stored in clear text in the file system.

Websphere MQ queues created:
- SYSTEM.BROKER.ADMIN.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- SYSTEM.BROKER.INTERBROKER.QUEUE
- SYSTEM.BROKER.INTERBROKER.MODEL.QUEUE
- SYSTEM.BROKER.MODEL.QUEUE
- SYSTEM.BROKER.WS.INPUT
- SYSTEM.BROKER.WS.REPLY
- SYSTEM.BROKER.WS.ACK

Access authority is granted for the WebSphere Event Broker group **mqbrkrs** to all these queues. If the DLQ is enabled, it also has the same authority.

Database tables created:

The database tables that this command creates, or adds to, are described in Database contents.

Responses:

This command returns the following responses:
- BIP8011 Unable to create configuration data
- BIP8012 Unable to connect to system components
- BIP8014 Component cannot be created
- BIP8022 Invalid user ID/password

- BIP8030 Unable to modify user ID privileges
- BIP8040 Unable to connect to database
- BIP8048 Unable to start queue manager
- BIP8050 Unable to create queue manager
- BIP8051 Unable to create queue
- BIP8053 Unable to set security for queue manager
- BIP8054 Unable to set security for queue
- BIP8056 Unknown queue manager
- BIP8070 Database exception
- BIP8072 Database exception
- BIP8073 Invalid broker name
- BIP8084 Unable to create directory
- BIP8086 Queue manager in use
- BIP8087 Component already exists
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping

In some circumstances, you might see the following error message issued by DB2:

```
(51002)[IBM][CLI Driver][DB2/NT]SQL0805N
Package "NULLID.SQLLF000" was not found.  SQLSTATE=51002.
```

This error occurs when the bind to the database is not successful.

On Windows platforms, binding is not needed for broker databases, but is required for user databases. If you have created the database using the DB2 Control Center, the bind is completed for you. If you use the command interface, it is not. For the database MYDB, for example, you can create or re-create a bind by entering the following commands at the command prompt:

```
db2 connect to MYDB user db2admin using db2admin
db2 bind X:\sqllib\bnd\@db2cli.lst grant public
db2 connect reset
```

where X: is the drive on which DB2 is installed.

On UNIX platforms, binding is necessary for all databases. For the database WBRKBKDB, for example, you can effect this by entering the following commands at the command prompt (where <user_name> is the user ID under which the database instance was created):

```
db2 connect to WBRKBKDB user db2admin using db2admin
db2 bind ~<user_name>/sqllib/bnd/@db2cli.lst grant public CLIPKG 5
db2 connect reset
```

If you are not using the default DB2 user ID and password (db2admin), you must replace these values in the db2 connect command with the correct values.

If you run the **mqsicreatebroker** command for a second time because of a failure the first time, you receive a series of messages. These indicate any items that cannot be created. There should not be any detrimental effects as a result of this. For example, as long as the reason for the first failure has been resolved, attempting to create a broker which failed the first time should result in a properly created broker.

Examples:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
-q WBRK_QM -s WBRK_UNS_QM -n WBRKBKDB
```

```
mqsicreatebroker BROKERA -i wbrkuid -a wbrkpw -q BROKERA -n BRKA_DB -t
```

mqsicreateconfigmgr command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

This command:
- Creates a WebSphere MQ queue manager, if one does not already exist.

  Note:
  1. If a WebSphere MQ queue manager is created as a result of using the **mqsicreateconfigmgr** command, the default DLQ provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is automatically enabled. The security settings are the same as those of other broker-specific WebSphere MQ queues.

     If you choose to create the queue manager separately, set up a dead letter queue (DLQ). The DLQ is referenced by WebSphere Event Broker when errors occur processing messages in message flows.

     If a message in either a user-defined message flow or in the publish/subscribe model cannot be processed, it is routed to this DLQ as a last resort. If you would prefer the message to be backed out onto the input queue, effectively halting the message flow until the problem is resolved, disable the DLQ.

     The **mqsideleteconfigmgr** command does not delete this queue (unless the queue manager is deleted).
  2. If you are using a WebSphere MQ queue manager that has been created independently of the **mqsicreateconfigmgr** command, you can define clusters if you choose. This simplifies your configuration.
- Starts the WebSphere MQ queue manager, if this is not already running. Note that the Configuration Manager always runs as a WebSphere MQ trusted application.
- Creates the Configuration Manager-specific WebSphere MQ queues and channel, if they do not already exist.
- Creates database tables for the Configuration Manager in its internal repository. If you need to transfer data from the configuration repository of an earlier release, you can use the *db2DatabaseToMigrate*, *migrationDatabaseUserId*, and *migrationDatabasePassword* parameters.
- Installs a Windows service, under which the Configuration Manager runs.
- Creates a record for the component in the broker registry.

If you have installed VisualAge for Java, and selected the WebSphere MQ Connector as part of that installation, ensure that the CLASSPATH entry for VisualAge for Java appears *after* the CLASSPATH entries for WebSphere MQ for the Windows platform that you are using (server or Java client). This is to ensure that the Configuration Manager accesses the correct WebSphere MQ classes (not the VisualAge classes) when it is started by the "mqsistart command" on page 323.

If the Configuration Manager detects an error in this area, it writes message
BIP1004 to the Windows System Event log.

Syntax:

Windows:

```
►►──mqsicreateconfigmgr──────────────────── -i ──ServiceUserID──────────────►
                         └─configmgrName─┘

►──── -a ──ServicePassword── -q ──QueueManagerName───────────────────────────►

►──────────────────────────────────────────────────────────────────────────►
     └─ -n ──db2DataBaseToMigrate─┘  └─ -u ──migrationDataBaseUserID─┘

►──────────────────────────────────────────────────────────────────────────►
     └─ -p ──migrationDataBasePassword─┘

►──────────────────────────────────────────────────────────────────────────►◄
     └─ -s ──UserNameServerQueueManagerName─┘  └─ -w ──Workpath─┘
```

Linux and UNIX systems:

```
►►──mqsicreateconfigmgr──configmgrName── -i ──ServiceUserID──────────────────►

►──── -a ──ServicePassword── -q ──QueueManagerName───────────────────────────►

►──────────────────────────────────────────────────────────────────────────►
     └─ -n ──db2DataBaseToMigrate─┘  └─ -u ──migrationDataBaseUserID─┘

►──────────────────────────────────────────────────────────────────────────►
     └─ -p ──migrationDataBasePassword─┘

►──────────────────────────────────────────────────────────────────────────►◄
     └─ -s ──UserNameServerQueueManagerName─┘  └─ -w ──Workpath─┘
```

z/OS:

```
►►──mqsicreateconfigmgr──configmgrName──q=QueueManagerName──────────────────────►

    ►─┬──────────────────────────┬──┬────────────────────────────┬──────────────►
      └─n=db2DataBaseToMigrate──┘  └─u=migrationDataBaseUserID ─┘

    ►─┬──────────────────────────────┬──┬──────────────────────────────────┬────►
      └─p=migrationDataBasePassword ─┘  └─s=UserNameServerQueueManagerName ─┘

    ►─┬────┬──┬────┬──────────────────────────────────────────────────────────►◄
      └─1─┘   └─2─┘
```

Parameters:

configmgrName
> (Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name of the Configuration Manager that you want to create.

> On Linux, UNIX systems and z/OS, this must be the first parameter specified. It is case-sensitive on Linux and UNIX systems.

> The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-i *ServiceUserID*
> (Required) The user ID under which the service runs.

> This can be specified in any valid username syntax for the platform.

> The ServiceUserID specified must be a member (either direct or indirect) of the local group **mqbrkrs**, and must be authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the **-w** flag). This user ID must also be a member (either direct or indirect) of the local group **mqm** or of the local Windows **Administrators** group, where applicable.

> The security requirements for the ServiceUserID are detailed in "Security requirements for Windows platforms" on page 389 for Windows and "Security requirements for Linux and UNIX platforms" on page 388 for UNIX platforms.

> If you use the unqualified form for this user ID (username), the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

-a *ServicePassword*
> (Required) The password for the ServiceUserID.

> For compatibility with existing systems, you can still specify <password>. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

-q *QueueManagerName*
> (Required) The name of the queue manager associated with the Configuration Manager.

> If the queue manager does not already exist, it is created by this command. It is not created as the default queue manager: if you want it to be the default queue manager on this system, create the queue manager before you issue this command.

The queue manager attribute MAXMSGL (maximum length of messages that can be put to queues) is updated to 100 MB. This is done whether or not the queue manager is created by this command.

-n *db2DatabaseToMigrate*

(Optional) The name of the database that you created at an earlier release to hold the configuration repository tables.

This database must already exist. You do not need to create an ODBC connection for this database, because access is provided by JDBC.

-u *migrationDataBaseUserID*

(Optional) The user ID with which the configuration repository database (created at an earlier release) is to be accessed.

-p *migrationDataBasePassword*

(Optional) The password of the user ID with which the configuration repository database (created at an earlier release) is to be accessed. If not specified, this defaults to the ServicePassword specified by **-a**.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

-s *UserNameServerQueueManagerName*

(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server. If this is not specified, the Configuration Manager assumes that there is no User Name Server defined, and does not attempt to communicate with one.

-w *Workpath*

(Optional) The directory in which working files for the Configuration Manager are stored. If not specified, the default directory specified when the product was installed is used.

1   (Optional - z/OS only) The registry pass that creates only the Configuration Manager registry.

2   (Optional - z/OS only) The WebSphere MQ pass that creates only the Configuration Manager WebSphere MQ queues.

   Note: This can be performed only if the Configuration Manager registry exists.

Authorization:

This command changes security privileges for the ServiceUserID; the user ID used to invoke this command must be a member of the Windows **Administrators** group on this local system.

Websphere MQ queues created:
- SYSTEM.BROKER.CONFIG.QUEUE
- SYSTEM.BROKER.CONFIG.REPLY
- SYSTEM.BROKER.ADMIN.REPLY
- SYSTEM.BROKER.SECURITY.REPLY
- SYSTEM.BROKER.MODEL.QUEUE

Access authority is granted for the WebSphere Event Broker group **mqbrkrs** to all these queues. If the DLQ is enabled, it also has the same authority.

Access authority is granted for the WebSphere Event Broker groups **mqbrdevt**, **mqbrasgn**, **mqbrops**, and **mqbrtpic** to the queues SYSTEM.BROKER.CONFIG.QUEUE and SYSTEM.BROKER.CONFIG.REPLY.

Websphere MQ channels created:
- SYSTEM.BKR.CONFIG

Database tables created:

The database tables that this command creates are administered by the Configuration Manager.

Responses:

This command returns the following responses:
- BIP8011 Unable to create configuration data
- BIP8012 Unable to connect to system components
- BIP8014 Component cannot be created
- BIP8022 Invalid user ID/password
- BIP8030 Unable to modify user ID privileges
- BIP8048 Unable to start queue manager
- BIP8050 Unable to create queue manager
- BIP8051 Unable to create queue
- BIP8053 Unable to set security for queue manager
- BIP8054 Unable to set security for queue
- BIP8055 Unable to load Java class
- BIP8056 Unknown queue manager
- BIP8074 Unable to create JVM
- BIP8075 Java exception
- BIP8076 Unable to set current directory
- BIP8077 Error initializing configuration data
- BIP8078 Error initializing configuration data
- BIP8084 Unable to create directory
- BIP8087 Component already exists
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping
- BIP8097 Unable to create Java object

Examples:
```
mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_CONFIG_QM
```

mqsicreatedb command

Supported platforms:
- Windows

Purpose:

The **mqsicreatedb** command makes a database that has already been installed, accessible to the broker. The command creates the database access, sets up the ODBC data source name, and, depending on the database type, creates and starts a Windows Service. The command creates a single instance of the Windows Service for each installation of a major product version. This command supports only the installed DB2 and Derby databases.

Syntax:

```
►►──mqsicreatedb──DatabaseName──────────────────────────────────────────►

 ►──┬────────────────────────────────────────────┬──┬─────────────────┬──►
    └─ -i──ServiceUserID──┬───────────────────────┘  └─ -p──PortNumber─┘
                          └─ -a──ServicePassword─┘

 ►──┬──────────────┬──┬─────────────────┬──►◄
    └─ -e──dbType──┘  └─ -u──dbUserID──┘
```
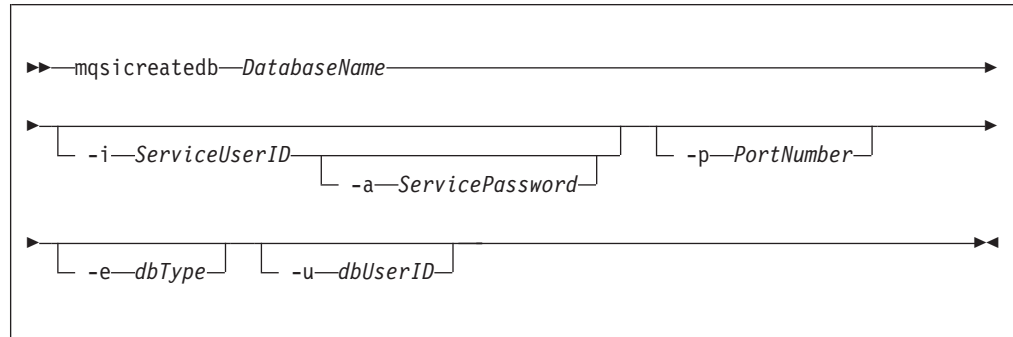
Parameters:

DatabaseName
(Required) The name of the database you want to create. This must be the first parameter, and is case sensitive. Restrictions might be placed on the permissible length of the database name by the database engine. For restrictions on the character set that can be used, see "Characters allowed in commands" on page 217.

-i *ServiceUserID*
(Optional, Derby only) The user ID under which the DatabaseInstanceMgr service runs.

This can be specified in any valid username syntax:
- `domain\username`
- `\\server\username`
- `.\username`
- `username`

If you use the unqualified form for this user ID (username), the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The ServiceUserID specified must be a member of the local group mqbrkrs. The ID can be a direct or indirect member of the group. The ServiceUserID must also be authorized to access the home directory (where WebSphere Event Broker is installed).

This parameter is ignored if the database engine specified or defaulted for the command is DB2. A ServiceUserID is required for Derby, but only for the first invocation of this command. Subsequent invocations are associated with the existing Windows DatabaseInstanceMgr service that runs under the ServiceUserID specified on the earlier command

The security requirements for the ServiceUserID are detailed in "Security requirements for Windows platforms" on page 389.

-a *ServicePassword*
>   (Optional, Derby only) The password for the ServiceUserID. Specify this only if you specify ServiceUserID.

-p *PortNumber*
>   (Optional) The TCP/IP port number that this component will use on the local machine. If not specified, the default value 1527 is used.

-e *dbType*
>   (Optional) The database engine to be used to create and run the database. Currently supported values are DB2 and Derby. If you do not specify this option, and only one database engine is available, that engine is used. If both are available, the default engine is DB2.

-u *dbUserID*
>   (Optional, DB2 only) An additional user name that requires access to the database that is created by this command.

Authorization:

The user ID used to invoke this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Responses:

This command returns the following responses:
- BIP9801 No supported database product found
- BIP9802 A database of name xxxx already exists
- BIP9803 The requested database engine, xxxx, could not be found
- BIP9805 A Cloudscape Network Server already running on port number
- BIP9809 Creating database, this may take a while
- BIP9811 The IBM MQSeries Brokers DatabaseInstanceMgr service already exists
- BIP9812 The IBM Cloudscape Network Server failed to start
- BIP9813 External process time out
- BIP9815 The supplied database name xxxx is too many characters in length
- BIP9816 Multiple database engines are installed. Defaulting to the DB2 database engine
- BIP9817 The DB2 database engine was used to create the xxxx database
- BIP9818 The IBM Cloudscape/Apache Derby database engine was used to create xxxx database

Examples:

The following example sets up a database with the name brokerdb on port 1600:
```
mqsicreatedb brokerdb -p 1600
```

The following example sets up a Derby database with the name derbydb, using port number 1527:
```
mqsicreatedb derbydb -i wbrkuid -a wbrkpw -e Derby -p 1527
```

mqsicreateexecutiongroup command

Supported platforms:
- Windows

- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsicreateexecutiongroup** command to add a new execution group to a broker.

You must start the Configuration Manager, using **mqsistart** on Windows platforms, Linux, and UNIX systems, or **startcomponent** on z/OS, (see "mqsistart command" on page 323) before you can issue this command.

The broker must be defined in the Configuration Manager before this command can be used.

Syntax:

```
►►──mqsicreateexecutiongroup──-b──Brokername──-e──ExecutionGroupName────────►

►┬──- connection cfgParameterFilename─┬──────────┬────-w──timeoutValue─┬──────►
 └─-i──ipAddress──-p──port──-q──qMgr─┘

►┬─────────────────────┬────────────────────────────────────────────────►◄
 └─-v──traceFileName─┘
```
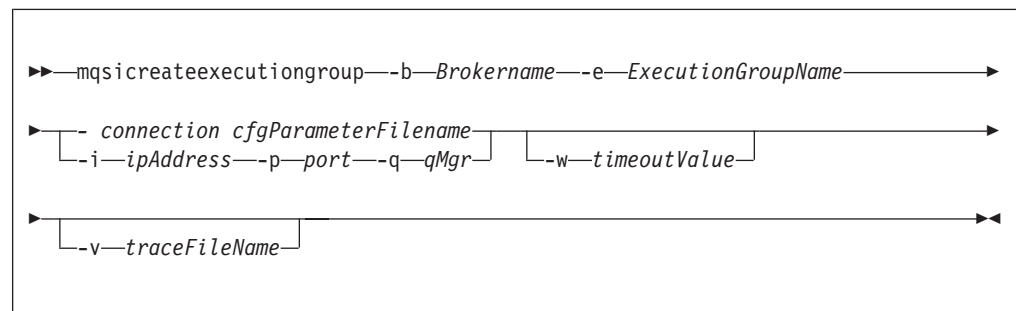
Parameters:

-b *BrokerName*
> (Required) The name of the broker on which to create the execution group.
>
> If you do not specify the **-e** flag, all execution groups on the broker are started.

-e *ExecutionGroupName*
> (Required) The name of the new execution group.

connection cfgParameterFileName
> (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.
>
> The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:
> ```
> <?xml version="1.0" encoding="UTF-8"?>
> <configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
>            securityExit="test.myExit"/>
> ```
>
> If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:
> ```
> <?xml version="1.0"?>
> ```
>
> and remove the value for the host attribute, to leave the statement as:
> ```
> <configmgr host="" listenerPort="1414" queueManager="QNAME"
>            securityExit="test.myExit"/>
> ```
>
> This parameter can also be specified as **-n** (as used in the "mqsideploy command" on page 289)

Note: If you do not supply this parameter, you must supply the **-i**, **-p**, and **-q** parameters.

-i *ipAddress*
(Optional) The host name or IP address of the Configuration Manager.

If you are using this file on z/OS and want to connect to the local host you must set the value to \"\".

-p *Port*
(Optional) The port number of the Configuration Manager.

-q *qMgr*
(Optional) The name of the WebSphere MQ queue manager that the Configuration Manager is using.

If you do not supply the **-i**, **-p**, and **-q** parameters, you must specify the *-connection* parameter.

-w *timeoutValue*
(Optional) The time in seconds to wait before the execution group is created; the default value is 60.

-v *TraceFileName*
(Optional) Sends the verbose internal trace to the specified file.

Authorization:

You need to be a member of the 'mqm' group and the command only succeeds if the user Id running the command has the correct authority defined in the Configuration Manager's access control list.

In order to create an execution group, full control authority is required over the broker object; see "ACL permissions" on page 385 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

0   (Success) States that the request completed successfully and the execution group has been created successfully in the Configuration Manager's repository. The next time the broker is deployed, the new execution group list is initialized on the broker.

2   (Failure) States that the execution group could not be created for any reason.

98   States that the Configuration Manager cannot be reached.

99   States that the supplied arguments to the utility are not valid.

Examples:

On the domain controlled by theConfiguration Manager whose queue manager is called QMGR and is listening on `fred.abc.com:1414`, create an execution group called 'EG1' on broker 'BROKER'.

```
mqsicreateexecutiongroup -i fred.abc.com -p 1414 -q QMGR -b BROKER -e EG1
```

On the domain specified by the file `domain1.configmgr`, create an execution group called 'EG2' on broker 'BROKER'.

```
mqsicreateexecutiongroup –n domain1.configmgr –b BROKER –e EG2
```

On the domain specified by the file `domain2.configmgr`, create an execution group
'EG3' on broker 'FRED'. Wait five minutes for the Configuration Manager to respond
and send output to `trace.txt`.

```
mqsicreateexecutiongroup –n domain2.configmgr –b FRED –e EG3 –w 300 –v trace.txt
```

mqsicreateusernameserver command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

On Windows platforms, Linux, and UNIX systems, this command:
- Creates a WebSphere MQ queue manager, if one does not already exist.

    Note:
    1. (Not z/OS) If a WebSphere MQ queue manager is created as a result
       of using the **mqsicreateusernameserver** command, the default DLQ
       provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is
       automatically enabled. The security settings are the same as those of
       other broker-specific WebSphere MQ queues.

       If you choose to create the queue manager separately, set up a dead
       letter queue (DLQ). The DLQ is referenced by WebSphere Event
       Broker when errors occur processing messages in message flows.

       If a message in either a user-defined message flow or in the
       publish/subscribe model cannot be processed, it is routed to this DLQ
       as a last resort. If you would prefer the message to be backed out onto
       the input queue, effectively halting the message flow until the problem
       is resolved, disable the DLQ.

       The **mqsideleteusernameserver** command does not delete this queue
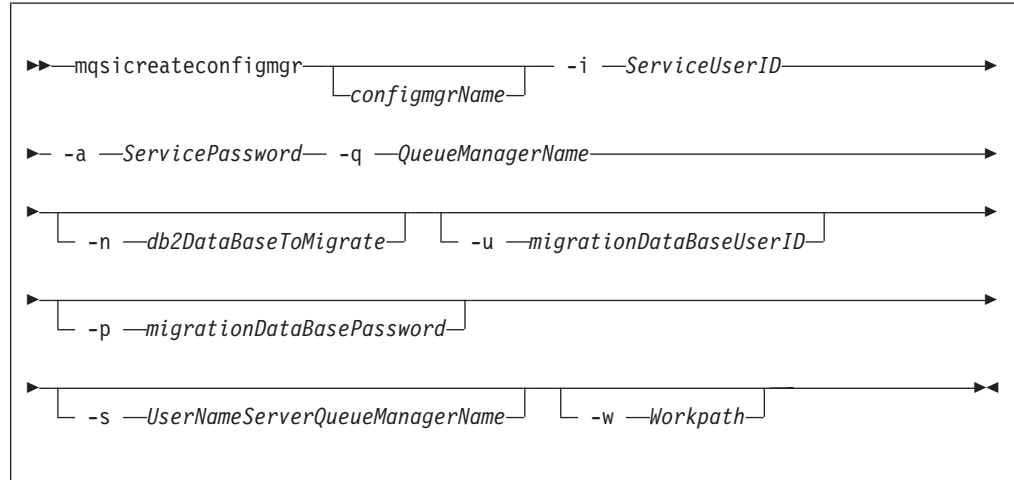   (unless the queue manager is deleted).
    2.
       If you are using a WebSphere MQ queue manager that has been
       created independently of the **mqsicreateusernameserver** command,
       you can define clusters. This simplifies your configuration.
- (Not z/OS) Starts the WebSphere MQ queue manager, if this is not already
  running.

  If the queue manager is created by this command, it is not started as a Windows
  service; it stops if you log off. To avoid this happening, either remain logged on,
  or change the start up status of the queue manager service. (If you lock your
  workstation, the WebSphere MQ queue manager does not stop.)
- Creates the User Name Server-specific WebSphere MQ queues, if these do not
  already exist.
- On Windows, installs a service under which the User Name Server runs.
- Creates a record for the component in the broker registry.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsicreateusernameserver── -i ─ServiceUserID── -a ─ServicePassword──────────────►

►── -q ─QueueManagerName──────────────────────────────────────────────────────────►
                    └─ -d ─SecurityDomainName─┘

►──────┬──────────────────────┬───┬──────────────┬────────────────────────────────►
       └─ -r ─RefreshInterval─┘   └─ -w ─Workpath─┘

►──────┬───────────────────────────┬───┬────┬─────────────────────────────────────►◄
       └─ -g ─AuthProtocolDataSource─┘   └─ -j ─┘
```

z/OS console command:

```
►►──mqsicreateusernameserver── -q ─QueueManagerName────────────────────────────────►

►──────┬─────────────────────┬───┬───┬───┬───┬────────────────────────────────────►◄
       └─ -r─RefreshInterval─┘   └─1─┘   └─2─┘
```
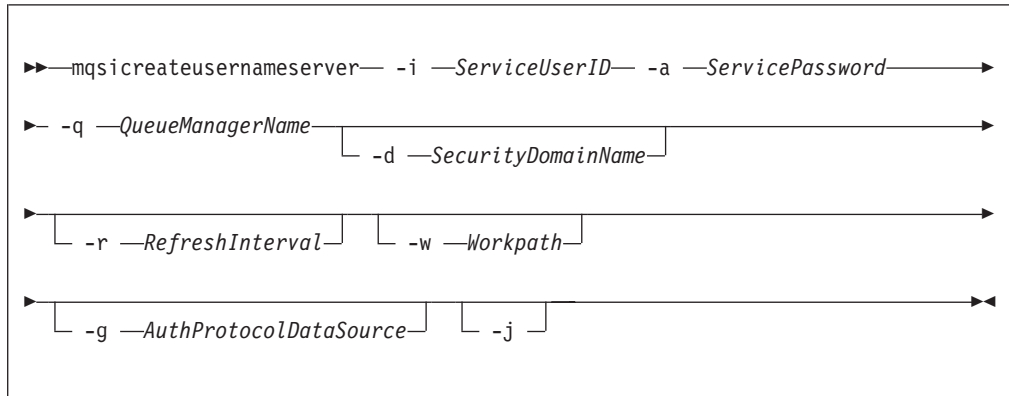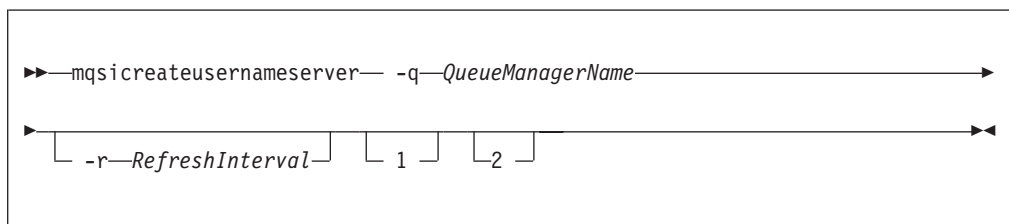
Parameters:

-i *ServiceUserID*

(Required - Windows platforms, Linux, and UNIX systems) The user ID under which the broker runs.

This can be specified in any valid username syntax. On Windows platforms, these are:

- `domain\username`
- `\\server\username`
- `.\username`
- `username`

On Linux and UNIX systems, only the last format, `username`, is valid.

If you use the unqualified form for this user ID (`username`) on Windows platforms, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The `ServiceUserID` specified must be a member of the local group **mqbrkrs**. On Windows platforms, it can be a direct or indirect member of the group. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Event Broker has been installed).

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 389 for Windows platforms and in "Security requirements for Linux and UNIX platforms" on page 388 for Linux and UNIX systems.

-a *ServicePassword*
(Required - Windows platforms, Linux, and UNIX systems) The password for the `ServiceUserID`.

For compatibility with existing systems, you can still specify `<password>`. However, if you do not specify a password with this parameter when you run the command you are prompted to enter a password during its invocation, and to enter the password a second time to verify that you have entered it correctly.

-q *QueueManagerName*
(Required) The name of the queue manager associated with the User Name Server.

On Windows platforms, Linux, and UNIX systems, if the queue manager does not already exist, it is created by this command. It is not created as the default queue manager: if you want it to be the default queue manager on this system, you must create the queue manager before you issue this command.

The queue manager attribute MAXMSGL (maximum length of messages that can be put to queues) is updated to 100 MB. This is done whether or not the queue manager is created by this command.

-d *SecurityDomainName*
(Optional - Windows platforms) The name of the Windows system security domain. If this is not specified, it defaults to the system's local Windows system security domain. For more details about the implementation of security in WebSphere Event Broker, see "Setting up broker domain security" on page 8.

-r *RefreshInterval*
(Optional) The interval, specified in seconds, at which the User Name Server interrogates the security subsystem for changes to user or group attributes. If it is not specified, the User Name Server's default interval of 60 seconds is used.

-w *Workpath*
(Optional - Windows platforms, Linux, and UNIX systems) The directory in which working files for the User Name Server are stored. If not specified, the default value specified when the product was installed is used.

-g *AuthProtocolDataSource*
(Optional - Windows platforms, Linux, and UNIX systems) Use this parameter to specify the name and location of the password file used to source any protocol related information. By default, the file is expected to be found in the home directory. If you store the file in a different location, specify the full path location with file name.

Two samples, `password.dat` and `pwgroup.dat`, are provided in the `examples/auth` directory under the product home directory.

-j (Optional - Windows platforms, Linux, and UNIX systems) Indicates that groups and group memberships are defined in the data source for the authentication protocol, rather than drawn from the operating system.

1 (Optional - z/OS only) The registry pass, which creates only the User Name Server registry.

2 (Optional - z/OS only) The WebSphere MQ pass, which creates only the User Name Server WebSphere MQ queues.

WebSphere MQ queues created:
- SYSTEM.BROKER.SECURITY.QUEUE

- SYSTEM.BROKER.MODEL.QUEUE

Access authority is granted for the WebSphere Event Broker group **mqbrkrs** to all these queues.

Responses:

This command returns the following responses:
- BIP8011 Unable to create configuration data
- BIP8012 Unable to connect to system components
- BIP8014 Component cannot be created
- BIP8022 Invalid user ID/password
- BIP8030 Unable to modify user ID privileges
- BIP8048 Unable to start queue manager
- BIP8050 Unable to create queue manager
- BIP8051 Unable to create queue
- BIP8053 Unable to set security for queue manager
- BIP8054 Unable to set security for queue
- BIP8056 Unknown queue manager
- BIP8068 Integer argument required
- BIP8084 Unable to create directory
- BIP8087 Component already exists
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping

Examples:

```
mqsicreateusernameserver -i wbrkuid -a wbrkpw
   -q WBRK_QM -r 1000
```

mqsideleteaclentry command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsideleteaclentry** command to delete the Configuration Manager database table relating to the group or user access control lists that you have defined.

Syntax:

Windows:

```
►►──mqsideleteaclentry─────────────────────────────────────────────────────►
                       └─configmgrName─┘  └─ -n ──ConnectionFileName─┘

►──┬─ -f ──FileName────────────────────────────────────────────────┬────────►
   ├─ -g ──GroupName─────────────────────┐                          │
   └─ -u ── UserName ─┬─ -m ──MachineName─┴─┬─ -b ──Broker──────────┤
                      └─ -a ───────────────┘ │           └─ -e ──ExeGroup─┘
                                             ├─ -s ──────┤
                                             ├─ -r ──────┤
                                             ├─ -t ──────┤
                                             └─ -p ──────┘

►──┬──────────────┬─────────────────────────────────────────────────────────►◄
   └─ -w──waitTime─┘
```

Linux and UNIX systems:

```
►►──mqsideleteaclentry──configmgrName────────────────────────────────────────►
                                      └─ -n ──ConnectionFileName─┘

►──┬─ -f ──FileName──────────────────────────────────────────────┬───────────►
   ├─ -g──GroupName─────────────────────┐                         │
   └─ -u ── UserName ─┬─ -m──MachineName─┴─┬─ -b ──Broker─────────┤
                      └─ -a──────────────┘ │          └─ -e ──ExeGroup─┘
                                           ├─ -s ─────┤
                                           ├─ -r ─────┤
                                           ├─ -t ─────┤
                                           └─ -p ─────┘

►──┬───────────────┬─────────────────────────────────────────────────────────►◄
   └─ -w──waitTime─┘
```

z/OS console command:

```
►►──mqsideleteaclentry──configmgrName─────────────────────────────────────────►
                                     └─n=ConnectionFileName─┘

►──┬─f =FileName─────────────────────────────────────────────────┬────────────►
   ├─g=GroupName──────────────────────┐                          │
   └─ u=UserName ─┬─m=MachineName──────┴─┬─b=Broker──────────────┤
                  └─a─────────────────┘  │         └─e=ExeGroup─┘
                                         ├─s=──────┤
                                         ├─r=──────┤
                                         ├─ t=─────┤
                                         └─ p=─────┘

►──┬──────────────┬────────────────────────────────────────────────────────────►◄
   └─w=waitTime─┘
```
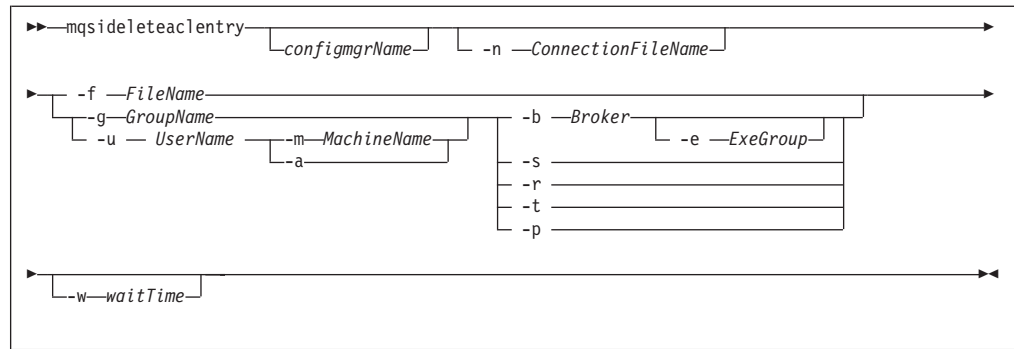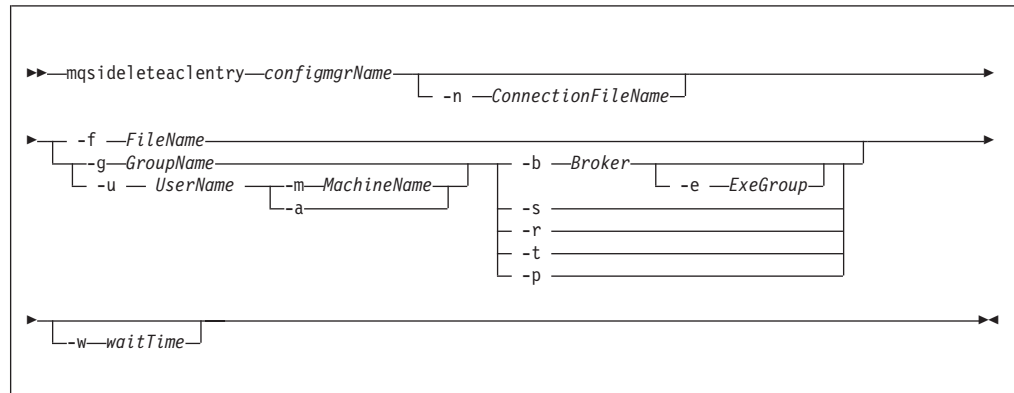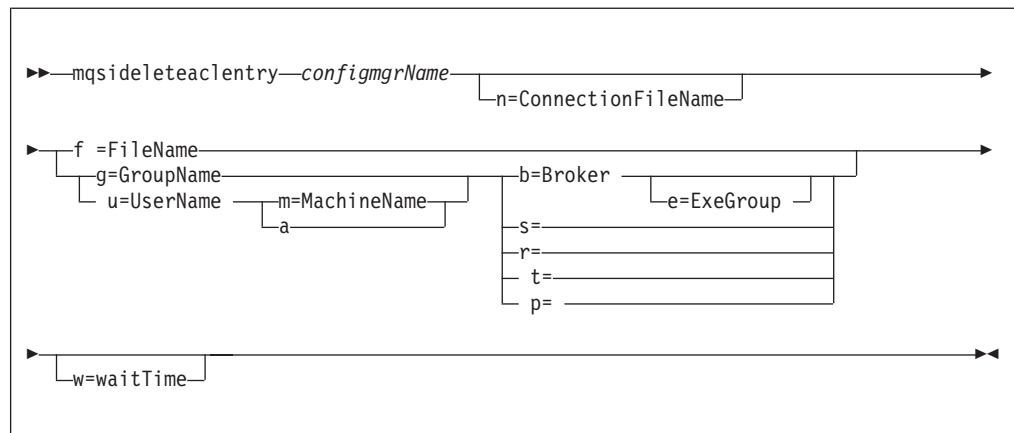
Parameters:

You **must** select either **-f** or

- **-g** or **-u**, and
- **-b** and **-e** or **-s** or **-r** or **-t** or **-p**

If you select **-u** you must select either **-m** or **-a**

configmgrName
> (Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name
> of the Configuration Manager from which access control lists are to be deleted.

On Linux, UNIX systems and z/OS, this must be the first parameter specified. It is case-sensitive on Linux and UNIX systems.

The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-n*connectionfileName*
(Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

-f *FileName*
(Optional) File from which to import the configuration. The output of the **mqsilistaclentry** command is the correct format.

-g *GroupName*
(Optional) Group to which this entry refers. For this reason, the name must adhere to the standard platform convention for group names.

-u *UserName*
(Optional) User name to which this entry refers, for example, TEST\ANOTHER.

-m *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.

-a (Optional) This signifies that the specified user name can be on any machine. This option can not be used with **-m**.

Note: If you select **-u**, you must select either **-a** or **-m**.

-b *Broker*
(Optional) The object is a broker object, and its name is specified as a parameter.

-e *ExeGroup*
(Optional) The object is an execution group and its name is specified as a parameter of the form Broker\ExeGroup. You must specify the **b** flag if you specify this flag.

-s (Optional) The object is a subscription object, and its name is specified as a parameter.

-r (Optional) The object refers to the root topic.

-t (Optional) The object refers to the main topology.

-p (Optional) The object refers to the "allresources" resource type. The authority

that the principal has for this object applies to all objects, including the
mqsicreateaclentry, mqsideleteaclentry, and mqsilistaclentry commands
themselves.

-w *waitTime*
> (Optional) The time in seconds that the command waits for a response from
> the Configuration Manager. If you do not supply a value the command waits
> for 30 seconds.

Authorization:

The user ID used to invoke this command must have full control permissions for
the object being changed; see "ACL permissions" on page 385 for more
information.

Examples:
```
mqsideleteaclentry CMGR01 -f c:\test\mylist
mqsideleteaclentry CMGR01 -g GROUPA F -b MYBROKER
```

mqsideletebroker command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsideletebroker** command to delete a named broker. The command also
deletes the queues on the broker's local queue manager (created when the broker
was created), and its data in the broker database. You can also specify that the
queue manager is to be deleted.

Even though this command deletes all the data related to this broker from the
broker database tables, it does not check if the tables are empty, nor does it delete
the tables.

This command:
- On Windows platforms, stops the service that runs the broker.
- Stops and deletes the WebSphere MQ queue manager for the broker, if
  requested.
- Removes the broker's data from the database.
- Removes the record for the component in the broker registry.

If you delete a broker that has WebSphere MQ publish/subscribe broker neighbors,
also invoke the command **clrmqbrk** at each of these neighbors, specifying the
WebSphere Event Broker broker that you are deleting with this command.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsideletebroker──brokername──────────────────────────────────────────►◄
                                  └─ -q ─┘   └─ -w ─┘
```

z/OS:

```
►►──mqsideletebroker──brokername──────────────────────────────────►◄
                                 └─ 1 ─┘  └─ 2 ─┘  └─ 3 ─┘
```
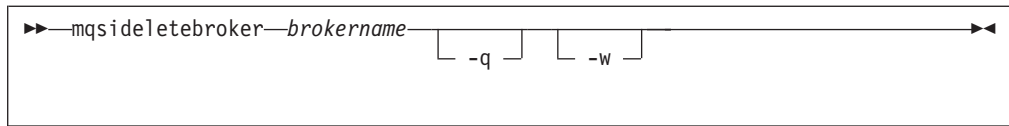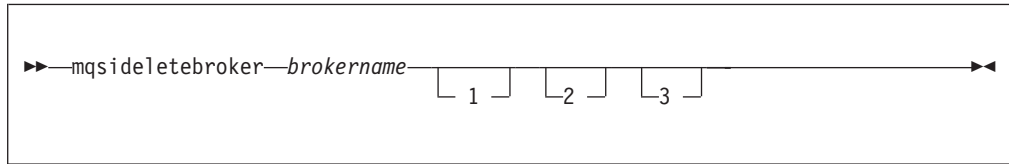
Parameters:

brokername
   (Required) The name of the broker that you want to delete. This must be the first parameter.

-q (Optional - Windows platforms, Linux, and UNIX systems) Specifies that the broker's queue manager is deleted. (If this option is not specified, only the WebSphere Event Broker queues and broker's data are deleted.)

   If the queue manager hosts another component (the Configuration Manager, or the User Name Server, or both in addition to this broker) that still exists, this command fails.

-w (Optional - Windows platforms, Linux, and UNIX systems) Deletes all files related to this broker from the workpath.

1 (Optional - z/OS only) Deletes only the broker registry.

2 (Optional - z/OS only) Deletes only the broker WebSphere MQ queues.

3 (Optional - z/OS only) Deletes only the broker DB2 tables and indexes.

Responses:
*   BIP8012 Unable to connect to system components
*   BIP8013 Component does not exist
*   BIP8017 Component cannot be deleted
*   BIP8018 Component running
*   BIP8040 Unable to connect to database
*   BIP8048 Unable to start queue manager
*   BIP8049 Unable to stop queue manager
*   BIP8052 Unable to delete queue
*   BIP8073 Invalid broker name
*   BIP8082 Unable to delete queue manager
*   BIP8093 Queue manager being created
*   BIP8094 Queue manager stopping
*   BIP8095 Queue manager reserved

Examples:
```
mqsideletebroker WBRK_BROKER -q
```

mqsideleteconfigmgr command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

This command:

- On the Windows platform, stops the service that runs the Configuration Manager.
- Stops and deletes the WebSphere MQ queue manager for the Configuration Manager, if requested.
- Removes the tables from the configuration repository, if requested.
- Removes the record for the component in the broker registry.
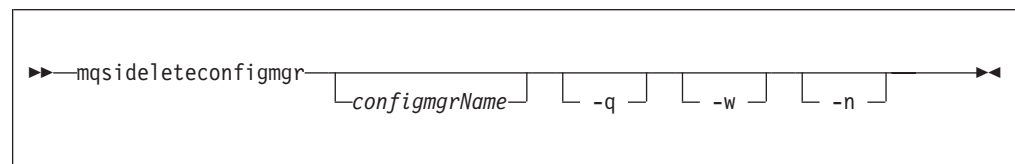
Syntax:

Windows:

```
>>-- mqsideleteconfigmgr --+------------+--+------+--+------+--+------+--><
                           |            |  |      |  |      |  |      |
                           '-configmgrName-'  '- -q -'  '- -w -'  '- -n -'
```

Linux and UNIX systems:

```
>>-- mqsideleteconfigmgr -- configmgrName --+------+--+------+--+------+--><
                                            |      |  |      |  |      |
                                            '- -q -'  '- -w -'  '- -n -'
```

z/OS:

```
>>-- mqsideleteconfigmgr -- configmgrName --+-----+--+-----+--+------+--><
                                            |     |  |     |  |      |
                                            '- 1 -'  '- 2 -'  '- n= -'
```
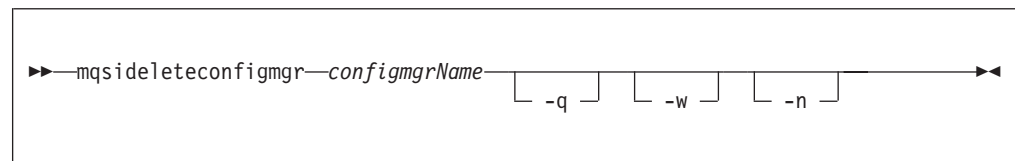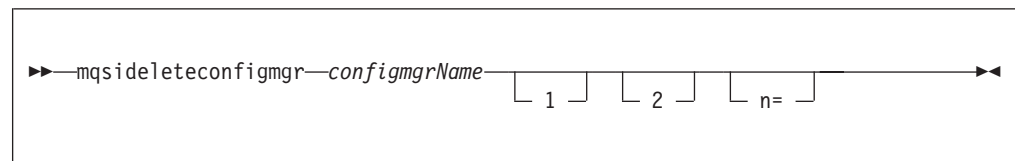
Parameters:

configmgrName
> (Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name of the Configuration Manager that you want to delete.

> On Linux, UNIX systems and z/OS, this must be the first parameter specified. It is case-sensitive on Linux and UNIX systems.

> The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-q (Optional - Windows platforms, Linux, and UNIX systems) Deletes the

Configuration Manager's queue manager. (If this option is not specified, only the WebSphere Event Broker queues are deleted.)

If the queue manager hosts another component (a broker, or the User Name Server, or both) that still exists, this command fails.

-w (Optional - Windows platforms, Linux, and UNIX systems) Deletes all files related to the Configuration Manager from the workpath.

-n

(Optional) Deletes the configuration repository.

Be very careful using this option. The configuration repository contains the configuration data for the whole broker domain, not just data internal to the Configuration Manager. Deleting this repository destroys all information pertinent to the broker domain, and requires you to recreate every resource within it to recover the broker domain.

If the configuration repository is not deleted as part of deleting the Configuration Manager, and the Configuration Manager is later recreated with the same name, it will continue to use the existing configuration repository.

If you do not specify –n to delete the configuration repository data, you can delete it manually later by locating the directory in which the configuration repository is stored and deleting the entire directory. This should be done with extreme caution as all domain information for the Configuration Manager is deleted. To delete the configuration repository manually:

1. Locate the Configuration Manager's working directory. The default location is:
   - For Windows, C:\Documents and Settings\All Users\Application Data\IBM\MQSI
   - For UNIX, /var/mqsi
   - For z/OS, ++COMPONENTDIRECTORY++
2. Locate the components directory. Delete the directory with the same name as the Configuration Manager.

1 (Optional - z/OS only) The registry pass that deletes only the Configuration Manager registry.

2 (Optional - z/OS only) The WebSphere MQ pass that deletes only the Configuration Manager WebSphere MQ queues.

Note: This can be performed only if the Configuration Manager registry exists.

Responses:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8017 Component cannot be deleted
- BIP8018 Component running
- BIP8038 Unsupported command option
- BIP8048 Unable to start queue manager
- BIP8049 Unable to stop queue manager
- BIP8052 Unable to delete queue
- BIP8055 Unable to load Java class
- BIP8074 Unable to create JVM
- BIP8075 Java exception

- BIP8076 Unable to set current directory
- BIP8077 Error initializing configuration data
- BIP8078 Error initializing configuration data
- BIP8082 Unable to delete queue manager
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping
- BIP8095 Queue manager reserved
- BIP8097 Unable to create Java object

Examples:

```
mqsideleteconfigmgr CMGR01 -q
```

mqsideletedb command

Supported platforms:
- Windows

Purpose:

The **mqsideletedb** removes a database that has been created by mqsicreatedb. This includes deleting the database and removing the ODBC data source name. If this is the last database managed by DatabaseInstanceMgr, the Cloudscape Network Server and DatabaseInstanceMgr service are also stopped and removed.

Syntax:

```
>>─mqsideletedb──┬─DatabaseName─────────┬──────────────────><
                 └─ -d─DeleteAllDatabases─┘
```

Parameters:

DatabaseName
> This must be the only parameter. Specify the name of the database that you want to delete. **Warning:** all the data stored in this database is permanently deleted.

-d DeleteAllDatabases
> This must be the only parameter. Specify this option to delete all the databases that have been created by mqsicreatedb. **Warning:** all the data stored in **ALL** databases is permanently deleted.

Authorization:

The user ID used to invoke this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Responses:

This command returns the following responses:
- BIP9804 No database of name xxx exists, no action taken
- BIP9807 The IBM Cloudscape Network Server on Port number xxxx, has been shutdown

- BIP9810 The IBM MQSeries Brokers DatabaseInstanceMgr service could not be found
- BIP9813 External process time out

Examples:

The following example deletes the database brokerdb:

```
mqsideletedb brokerdb
```

mqsideleteexecutiongroup command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsideleteexecutiongroup** command to delete an execution group from a broker.

You must start the Configuration Manager, using **mqsistart** on Windows platforms, Linux, and UNIX systems, or **startcomponent** on z/OS, (see "mqsistart command" on page 323) before you can issue this command.

If you are deleting an execution group to which a deployment has previously been made, you must also start the broker before issuing this command.

Syntax:

```
►►──mqsideleteexecutiongroup──-b─Brokername──-e─ExecutionGroupName──────────►

►──┬──- connection cfgParameterFilename───────┬──┬─────────────────────┬────►
   └─-i─ipAddress──-p─port──-q─qMgr─┘          └─-w─timeoutValue─┘

►──┬──────────────────────┬──────────────────────────────────────────────►◄
   └─-v─traceFileName─┘
```
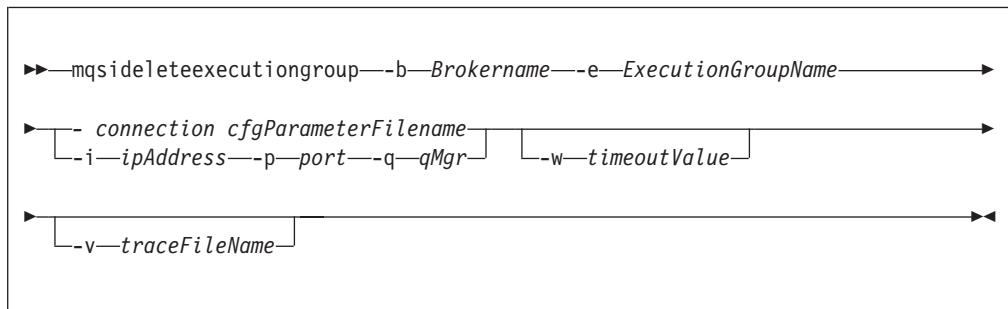
Parameters:

-b *BrokerName*
    (Required) The name of the broker on which the execution group resides.

-e *ExecutionGroupName*
    (Required) The name of the execution group to delete.

connection cfgParameterFileName
    (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

    The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

If you are using this file on z/OS you must remove the statement
encoding="UTF-8" from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

This parameter can also be specified as **-n** (as used in the "mqsideploy command" on page 289)

Note: If you do not supply this parameter, you must supply the **-i**, **-p**, and **-q** parameters.

-i *ipAddress*
    (Optional) The host name or IP address of the Configuration Manager.

    If you are using this file on z/OS and want to connect to the local host you must set the value to \"\".

-p *Port*
    (Optional) The port number of the Configuration Manager.

-q *qMgr*
    (Optional) The name of the WebSphere MQ queue manager that the Configuration Manager is using.

    If you do not supply the **-i**, **-p**, and **-q** parameters, you must specify the *-connection* parameter.

-w *timeoutValue*
    (Optional) The time in seconds to wait before the execution group is deleted; the default value is 60.

-v *TraceFileName*
    (Optional) Sends the verbose internal trace to the specified file.

Authorization:

You need to be a member of the 'mqm' group and the command only succeeds if the user Id running the command has the correct authority defined in the Configuration Manager's access control list.

In order to delete an execution group, full control authority is required over the broker object; see "ACL permissions" on page 385 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

0   (Success) States that the request completed successfully and the execution group has been deleted successfully. If the command was to delete an execution group to which a deployment has previously been made, this return code means that the broker has stopped and freed all resources associated with that execution group, for example, message flows.

2   (Failure) States that the execution group could not be deleted for any reason.

98   States that the Configuration Manager cannot be reached.

99   States that the supplied arguments to the utility are not valid.

Examples:

On the domain controlled by theConfiguration Manager whose queue manager is called QMGR and is listening on `fred.abc.com:1414`, delete an execution group called 'EG1' on broker 'BROKER'.

```
mqsideleteexecutiongroup -i fred.abc.com -p 1414 -q QMGR -b BROKER —e EG1
```

On the domain specified by the file `domain1.configmgr`, delete an execution group called 'EG2' on broker 'BROKER'.

```
mqsideleteexecutiongroup —n domain1.configmgr —b BROKER —e EG2
```

On the domain specified by the file `domain2.configmgr`, delete an execution group 'EG3' on broker 'FRED'. Wait five minutes for the Configuration Manager to tidy up any resources and send output to `trace.txt`.

```
mqsideleteexecutiongroup —n domain2.configmgr —b FRED —e EG3 —w 300 —v trace.txt
```

mqsideleteusernameserver command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

This command:
- Stops the service that runs the User Name Server.
- Stops and deletes the WebSphere MQ queue manager for the User Name Server, if requested.
- Removes the record for the component in the broker registry.

Syntax:

Windows platforms, Linux and UNIX systems:

```
►►──mqsideleteusernameserver──────────────────────────────►◄
                           └─ -q ─┘  └─ -w ─┘
```

z/OS:

```
►►──deleteusernameserver── c=ComponentDirectory────────────►◄
                                             └─ 1 ─┘  └─ 2 ─┘
```
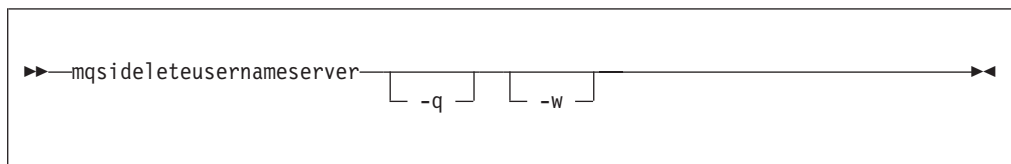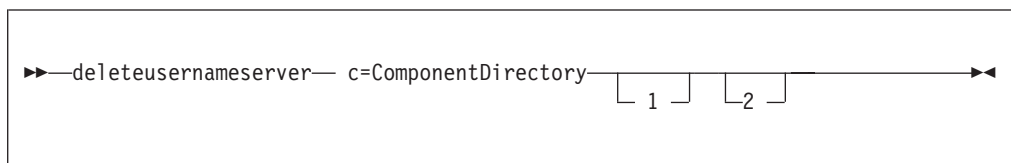
Parameters:

-q (Optional - Windows platforms, Linux, and UNIX systems) Deletes the User

Name Server's queue manager when the User Name Server has been deleted. (If this option is not specified, only the WebSphere Event Broker queues are deleted.)

If the queue manager hosts another component (a broker, or the Configuration Manager, or both) that still exists, this command fails.

-w (Optional - Windows platforms, Linux, and UNIX systems) Deletes all files related to the User Name Server from the workpath.

-c *ComponentDirectory*
(Required - z/OS) The name of the component directory. For example `/var/wmqi/MQP1UNS`

-1 (Optional - z/OS only) Deletes only the User Name Server registry.

-2 (Optional - z/OS only) Deletes only the User Name Server WebSphere MQ queues.

Responses:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8017 Component cannot be deleted
- BIP8018 Component running
- BIP8048 Unable to start queue manager
- BIP8049 Unable to stop queue manager
- BIP8052 Unable to delete queue
- BIP8082 Unable to delete queue manager
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping
- BIP8095 Queue manager reserved

Examples:
`mqsideleteusernameserver`

mqsideploy command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsideploy** command to make a deployment request to the Configuration Manager. This allows you to make the various flavors of deployment requests from a batch command script, without the need for manual interaction.

The default situation is a delta or incremental deployment. Select **-m** to override the default operation.

mqsideploy does not require a Configuration Manager name parameter because all of the required connection details can be obtained from the **-n**, **-q**, **-i**, and **-p** parameters.

Syntax:

```
>>-mqsideploy----n---cfgParameterFilename---------------------->
                 |-i--ipAddress---p--port---q--qMgr-|

>------b--Brokername--| ** |-----w--timeoutValue---------------->
      |-l--+----+---------|   |-_-|
      |    |-m--|         |
      |    |-_--|         |
      |-t--+----+---------|
      |    |-m--|
      |    |-_--|
      |-c----------------|

>------v---TraceFileName--------------------------------------><
     |-_-|


** :

      |---e--EGName----a BARFileName----m----------------|
      |             |              |-_-|          |
      |             |-d DeployedObjects-------|
      |-c----------------------------------------|
      |-_-|
```
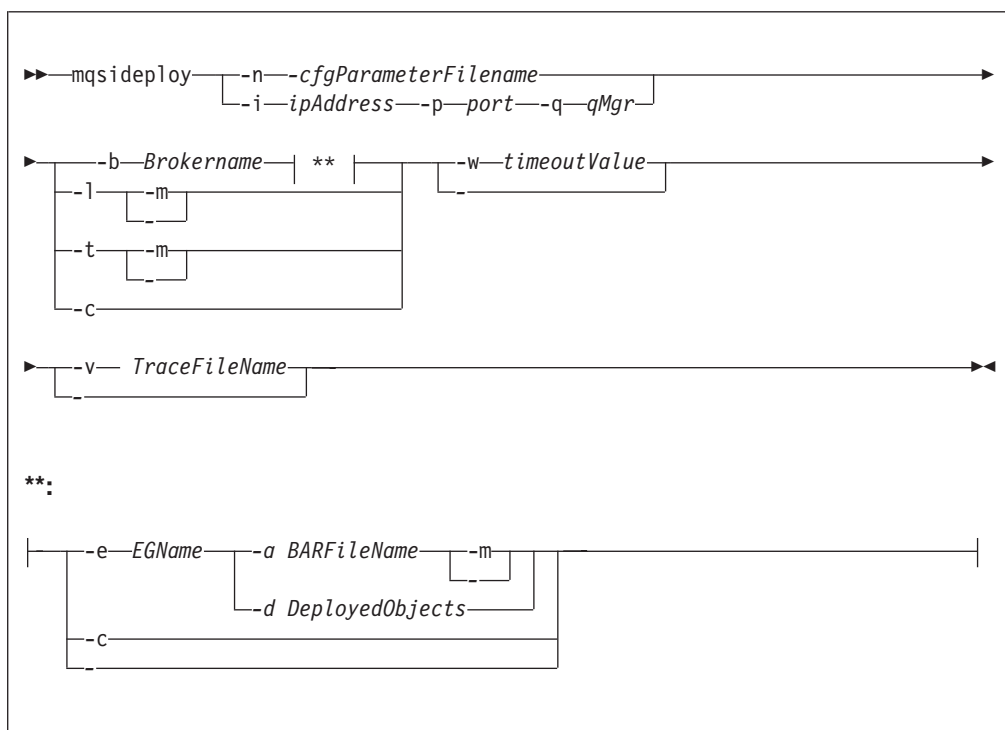
Note: The command flags are different from those used in earlier versions of **mqsideploy**. On Windows platforms, to use the same command flags as in the previous versions, use the command *mqsideploy.bat*.

Parameters:

-n *cfgParameterFileName*
> (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.
>
> The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:
>
> ```
> <?xml version="1.0" encoding="UTF-8"?>
> <configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
>            securityExit="test.myExit"/>
> ```
>
> If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:
>
> ```
> <?xml version="1.0"?>
> ```
>
> and remove the value for the host attribute, to leave the statement as:
>
> ```
> <configmgr host="" listenerPort="1414" queueManager="QNAME"
>            securityExit="test.myExit"/>
> ```
>
> **Note:** If you do not supply this parameter, you must supply the **-i**, **-p**, and **-q** parameters.

-i *ipAddress*
> (Optional) The host name or IP address of the Configuration Manager.
>
> If you are using this file on z/OS and want to connect to the local host you must set the value to \"\".

-p *Port*

(Optional) The port number of the Configuration Manager.

-q *qMgr*

(Optional) The name of the WebSphere MQ queue manager that the Configuration Manager is using.

If you do not supply the **-i**, **-p**, and **-q** parameters, you must specify the **-n** parameter.

-b *Brokername*

(Required) The name of the broker to which to deploy. If either of the **-t** or **-l**, flags are specified, the **-b** option is ignored. This is because, when deploying topics or topology, all brokers in the domain are affected. Without the **-e** and **-a** flags, a broker configuration deploy is initiated.

With the **-c** option deployment is cancelled to the specific broker.

-e *EGName*

(Optional) The name of the execution group to which to deploy. The **-b** and **-a** options must also be specified with this option.

-a*BARFileName*

(Optional) The name of the BAR (Broker Archive) file that is to be used for a message flow or message set deploy. The **-b** and **-e** options must also be specified with this option.

-t-*topics*

(Optional) Deploy all the topics configuration information. Information is deployed to all brokers in the domain.

-l-*topology*

(Optional) This flag specifies that the Topology configuration should be deployed. Information is deployed to all brokers in the domain if the **-m** flag is also set, otherwise the information is deployed only to brokers with a changed topology configuration.

-c-*cancel*

(Optional) Tells the Configuration Manager to stop waiting for responses to previously submitted deployment requests. If used with the **-b** option, the Configuration Manager stops waiting for outstanding deployment responses from the specified broker; without the **-b** option, the Configuration Manager stops waiting for responses to all outstanding deployment requests in the domain.

The **-c** option must be specified with caution. Only use it if there is no possibility that the affected brokers will respond to the deployment request; if a broker subsequently processes a deployment request that had been cancelled, the Configuration Manager ignores the response, and therefore become out of synchronization with the broker.

-w *timeoutValue*

(Optional) The time in seconds that the utility waits for the broker to reply before returning control to the command line. The **mqsideploy** command polls the Configuration Manager log records looking for the results of the deploy request that has just been sent. The relevant log records contain information indicating whether the deploy has been successful. The *timeoutValue* is the number of seconds to wait before timing out, in the range 1 - 999 999. If no *timeoutValue* is provided, or a value less than 1 or greater than 999 999 is specified, an error is returned

-d *DeployedObjects*

    (Optional) Describes the set of objects to be removed from the execution group. Multiple deployed files can be specified, separated by the "**:**" character.

    Objects of all types can be specified, with the following limitations:

1. Message flows can be specified only if the Configuration Manager to which the command is connected is not Version 6.
2. An attempt to remove, for example, an XSL file when connected to a Version 6.0 Configuration Manager results in an "object not found" type error being reported.
3. If an ambiguous object name is specified (for example "top", when both "top.dictionary" and "top.cmf" are deployed to the same execution group) the entire command fails with the message BIP1089. In these circumstances, the fully qualified name of the objects to remove must be specified, for example "top.dictionary:top.cmf".

-v *TraceFileName*

    (Optional) Sends the internal Configuration Manager Proxy debugging tracing to the specified file.

-m *-complete*

    (Optional) Deploy complete information:

- For a **BAR file deploy**, **-m** removes all currently-deployed message flows and message sets from the execution group as part of the deploy. If **-m** is not set, the contents of the BAR file are deployed in addition to what is already deployed on the execution group. In this case, any deployed objects with the same name as an item inside the BAR are replaced by the version inside the BAR.
- For a **topology configuration deploy**, **-m** deploys complete interbroker configuration information to all brokers. If **-m** is not set, only changed interbroker configuration is deployed to brokers whose interbroker configuration has changed.
- For a **broker deploy**, **-m** deploys all the broker's available configuration attributes to the specified broker, for example, multicast parameters. If **-m** is not set, only changed configuration attributes are deployed.
- For a **topic tree deploy**, **-m** deploys the entire topic tree to all brokers. If **-m** is not set, only changes to the topic tree are deployed to all brokers.
- For a **remove message flow operation**, the flag is ignored.

Authorization:

In order to successfully deploy, the user ID issuing the command needs to have sufficient authority defined in the Configuration Manager. The permissions required are the same as the permission required to do the equivalent function in the Message Brokers Toolkit. See "ACL permissions" on page 385 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

0   (Success) States that the Configuration Manager issued the deployment request and all relevant brokers responded successfully before the timeout period expired.

2   (Failure) States that the Configuration Manager issued the deployment

request and at least one broker responded negatively. See the messages output from the utility (or the Configuration Manager's event log) for more information.

3   (Initiated) States that the Configuration Manager has replied, stating that deployment has started, but that no broker responses were received before the timeout occurred.

5   (Submitted) States that the deploy message was sent to the Configuration Manager , but that no response was received before the timeout occurred.

6   (SuccessSoFar) States that the Configuration Manager issued the deployment request and some, but not all of, the relevant brokers responded successfully before the timeout period expired; no brokers responded negatively.

98   States that the Configuration Manager cannot be reached.

99   States that the supplied arguments to the utility are not valid.

Examples:

Connect to the Configuration Manager whose connection parameters are described in the file `cm1.configmgr` and attempt to remove the message flow `top` and the dictionary `bar` from the execution group `default` on broker `b1`.

Note that, if there are no other objects called `top` and `bar` deployed to the execution group, the value of the `-d` option can be shortened to `top:bar`.

```
mqsideploy –n cm1.configmgr –b B1 –e default –d top.cmf:bar.dictionary
```

mqsiformatlog command

Supported platforms:
*   Windows
*   Linux and UNIX systems
*   z/OS through the BIPFMLG utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsiformatlog** command to process the XML log created by **mqsireadlog**. The command retrieves and formats any messages that the XML log contains into a form suitable for the locale of the user invoking the command.

This command interprets an input log file that has been created on any system in a platform-independent code page, `utf-8`. Use it to produce formatted output from input log files transferred from other systems to the system on which you issue the command. If you use this facility, ensure that you use a file transfer program that does not convert the data (for example, by specifying a binary transfer option).

You can direct the output to a file, or to the command shell.

Syntax:

```
▶▶──mqsiformatlog── -i ──inputfilename──────────────────────────────────▶◀
                                      └─ -o ──outputfilename─┘
```

Parameters:

-i *inputfilename*
> (Required) The filename of the XML log file that is to be formatted. This file is created by the **mqsireadlog** command; it is encoded in `utf-8`.

-o *outputfilename*
> (Optional) The filename of the file into which the formatted log output is to be written. If this is not specified, the formatted log data is written to stdout.
>
> Output written by this command (to file or stdout) is written in a code page suitable for the current user locale.

Authorization:

The user ID used to invoke this command must have read access to the input file, and write access to the output file.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Responses:
- BIP8041 Unable to open file
- BIP8042 Insufficient memory
- BIP8043 Invalid trace file
- BIP8046 Unable to initialize XML
- BIP8047 Unable to parse data
- BIP8069 Unable to find message
- BIP8079 Unable to locate XML function
- BIP8080 Unable to load resource
- BIP8081 Error processing command

Examples:

```
mqsiformatlog -i trace.xml -o formattrace.log
```

The following extract illustrates the output generated by this command:

```
Timestamps are formatted in local time, local time is GMT.
.
.
.
2003-02-12 12:57:21.895999    388  UserTrace   BIP2638E:
MQPUT to queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager
'WBRK_QM': MQCC=0, MQRC=0; node ConfigurationMessageFlow.outputNode'.
The node 'ConfigurationMessageFlow.outputNode' attempted
to write a message to the specified queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY'
connected to queue manager 'WBRK_QM'.
The MQCC was 0 and the MQRC was 0.
No user action required.

2003-02-12 12:57:21.895999    388  UserTrace   BIP2622I:
Message successfully output by output node 'ConfigurationMessageFlow.outputNode'
to queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager
'WBRK_QM'. The WebSphere MQ output node ConfigurationMessageFlow.outputNode'
```

```
successfully wrote an output message to the specified queue
SYSTEM.BROKER.EXECUTIONGROUP.REPLY connected to queue manager WBRK_QM.
No user action required.
.
.
.
Threads encountered in this trace:  335  388
```

mqsijoinmqpubsub command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS through the BIPJNMP utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsijoinmqpubsub** command to join this WebSphere Event Broker broker to an WebSphere MQ Publish/Subscribe broker network. The command identifies a specific WebSphere MQ Publish/Subscribe broker to be the parent of the WebSphere Event Broker broker.

This is an asynchronous command. Successful completion of this command indicates that the WebSphere Event Broker broker has accepted the request, not that the required action has completed.

Use the **mqsilistmqpubsub** command to monitor the status of the asynchronous actions that result from this command.

Use this command only if you are integrating this WebSphere Event Broker broker with an WebSphere MQ Publish/Subscribe broker network. Before you issue this command, ensure that the WebSphere Event Broker broker is ready to receive and process messages on queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS (that is, you must have restarted the broker after creating this queue).

Syntax:

```
►►──mqsijoinmqpubsub──brokername── -p ──ParentQueueManagerName─────────────────►◄
```

Parameters:

brokername
(Required) The name of the broker that is to be joined to an WebSphere MQ Publish/Subscribe broker.

-p *ParentQueueManagerName*
(Required) The name of the queue manager that hosts the WebSphere MQ Publish/Subscribe broker to which this WebSphere Event Broker broker is to be joined.

Authorization:

The user ID used to invoke this command must have `put` and `inq` authority to the queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Responses:
- BIP8013 Component does not exist
- BIP8056 Unknown queue manager
- BIP8057 Queue manager error
- BIP8059 Queue manager not available
- BIP8060 Queue error
- BIP8061 No reply received
- BIP8064 Internal broker error
- BIP8066 Invalid broker name

Examples:

mqsijoinmqpubsub WBRK_BROKER -p MQBroker1

mqsilist (list resources) command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

On Windows platforms, Linux, and UNIX systems, use the **mqsilist** command to list all the components installed on the system, all the execution groups defined to a specific broker, or all the message flows contained in a named execution group on a specific broker.

On Windows platforms, Linux, and UNIX systems, the output is directed to stdout.

From a z/OS console, use the **list** command to list all the execution groups defined to a specific broker, or all the message flows contained in a named execution group on a specific broker.

Syntax:

Windows platforms, Linux, and UNIX systems:
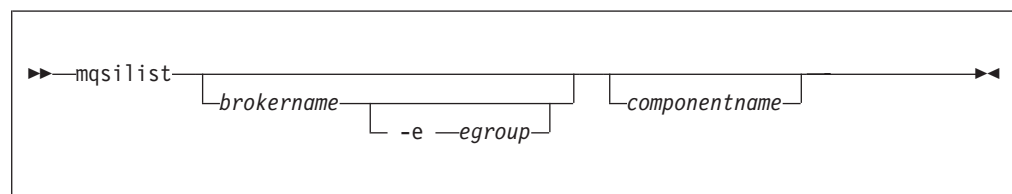
```
►►──mqsilist──┬─────────────────────────────┬──┬───────────────────┬──►◄
              └─brokername──┬─────────────┬──┘  └─componentname─────┘
                           └─ -e ──egroup─┘
```

If you do not specify any parameters when you issue this command, a list of components and queue manager names is displayed for each component created on this system, in the form:

```
BIP8099I: Broker: brokername - queuemanagername
BIP8099I: ConfigMgr: configmgrname - queuemanagername
BIP8099I: UserNameServer: UserNameServer - queuemanagername
BIP8071I: Successful command completion
```

z/OS:

Synonym: l
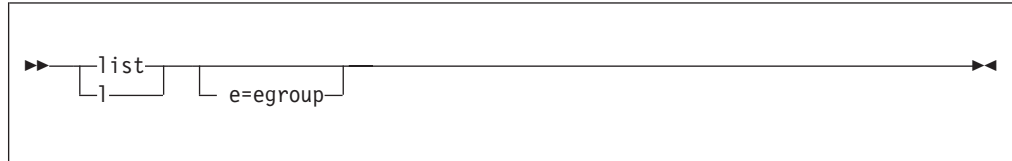
If you do not specify any parameters when you issue this command, a list of the execution groups is displayed.

You issue the **list** command only against a broker task name. For example:
```
F MQP1BRK,list
```

The output is the list of execution groups and process IDs in the form:
```
BIP8130I: Execution Group: <name> -<process ID>
BIP8071I: Successful command completion
```

If you specify an execution group, for example:
```
F MQP1BRK, list e='exgrp1'
```

the output is a list of message flows in the form:
```
BIP8131I: Messageflow: <MessageFlowName>
BIP8071I: Successful command completion
```

Parameters:

Windows platforms, Linux, and UNIX systems:

brokername
    (Optional) The name of the broker for which you want to list resources. This
    must be a deployed broker. A list of execution groups configured on this
    broker and the process ID (pid) of each is displayed.

componentname
    (Optional) (Optional) The name of a component for which you want to list
    resources.

All platforms:

-e *egroup*
    (Optional) Selects an execution group within a broker. Specify the label of the
    execution group for which you want to list message flows. The command
    returns a list of message flows assigned to the specified execution group within
    the broker.

    The broker specified must be active for any message flow information to be
    returned.

Authorization:

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

On Windows platforms, if you have specified a broker name and the **-e** flag, the user ID used to invoke this command must have **mqbrkrs** group membership.

Responses:
- BIP8013 Component does not exist
- BIP8020 Unable to access database
- BIP8029 Broker not configured
- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database
- BIP8158 Invalid format for command (z/OS only)
- BIP8159 Unknown parameter "..." (z/OS only)

Examples:

Windows platforms, Linux, and UNIX systems:
```
mqsilist WBRK_BROKER -e DefaultEG
mqsilist DatabaseInstanceMgr
```

z/OS
```
F MQP1BRK,list e='exgrp1'
```

mqsilistaclentry command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsilistaclentry** command to view or list the currently defined:
- User groups
- Users
- Objects
- Access control lists

If you do not specify any parameters, all the groups, users, and objects are listed.

If you specify *GroupName*, only those access control lists relating to that group are listed.

If you specify *UserName*, only those access control lists relating to that specific user are listed, including any access control lists to which they belong.

If you specify *Broker*, only those groups, users, or access control lists relating to that broker are listed.

The output from this command is a description of the access rights that match the criteria specified in the command line arguments; each line takes the following form:

```
<principal> - <principaltype> - <accesstype> - <objectname> - <objecttype>
```

where
- `<principal>` is the name of the user or group for which a policy has been defined.
- `<principaltype>` is USER if the principal refers to a user, or GROUP if the principal refers to a group.
- `<accesstype>` describes the type of authority that has been granted, and can be one of:

  V   View access

  F   Full control

  D   Deploy access

  E   Editor access

- `<objectname>` applies only to execution groups and brokers, and describes the name of the object that has had a policy defined.
- `<objecttype>` describes the type of object that has had a policy defined, and can be one of:

  Broker
  > A broker

  ConfigManagerProxy
  > Configuration Manager Proxy

  ExecutionGroup
  > An execution group

  PubSubTopology
  > The topology

  Subscription
  > The list of active subscriptions

  TopicRoot
  > The root topic

For example:

```
wrkgrp\ali - USER  - F - EXE - BROKER\default
```

means that user "ali" in domain "wrkgrp" has been granted full control over the execution group default in broker "BROKER".

Syntax:

Windows:
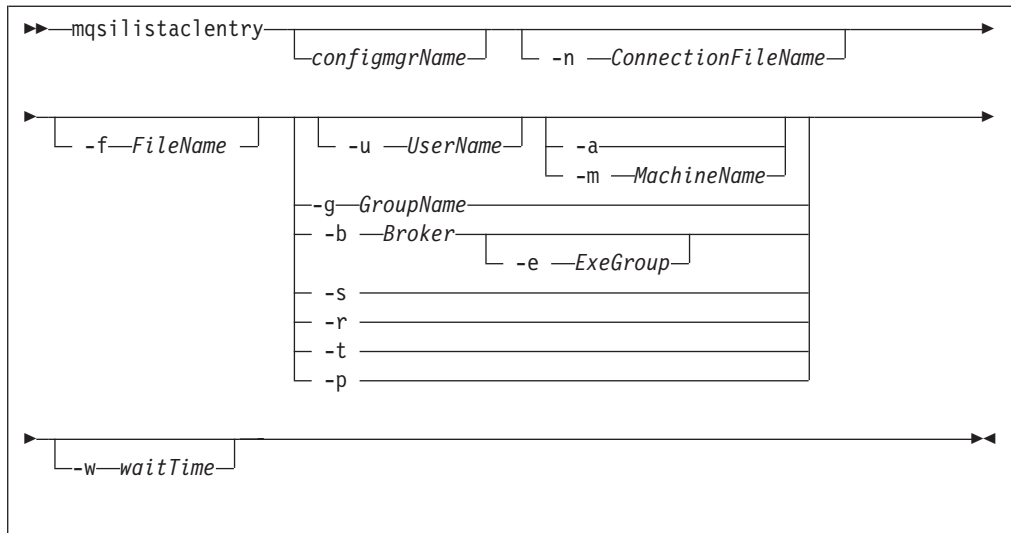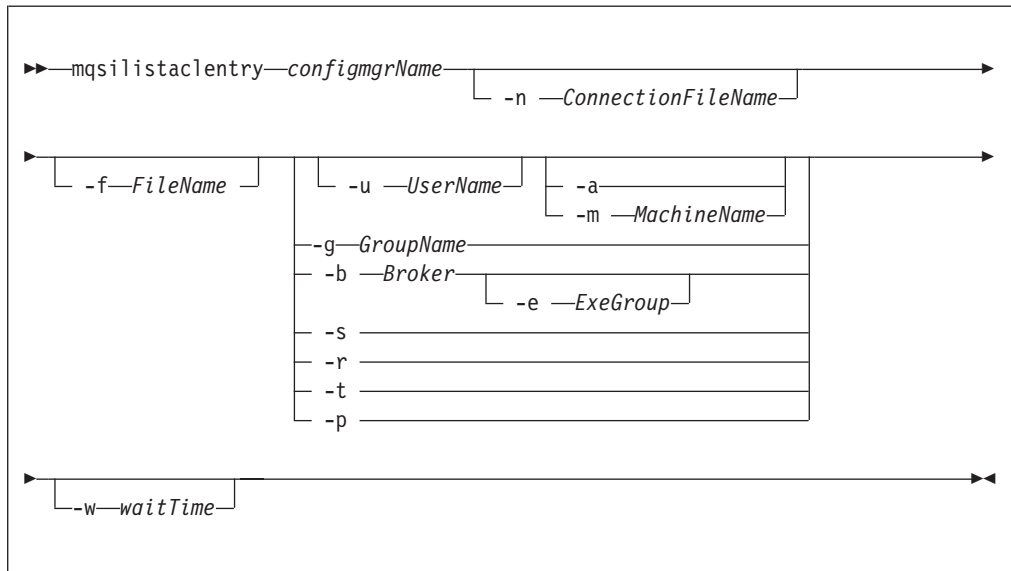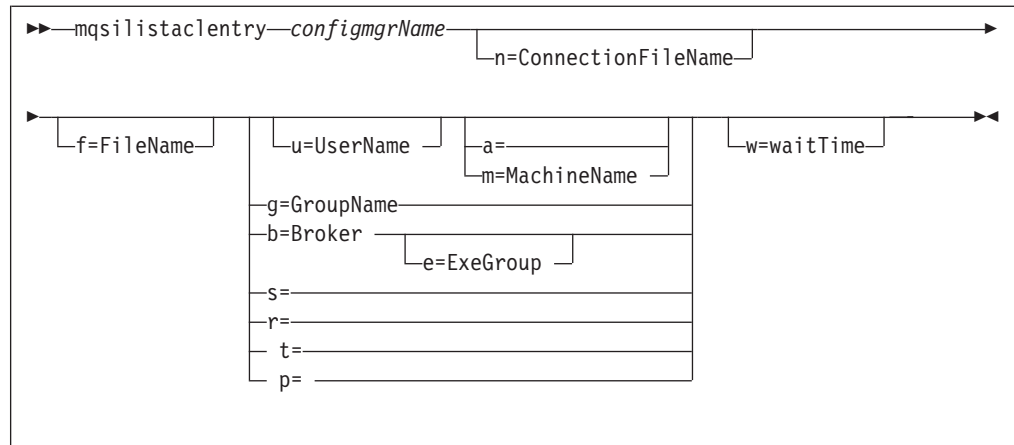
```
►►──mqsilistaclentry─────────────────────────────────────────────────────────►
                    └─configmgrName─┘   └─ -n ─ConnectionFileName─┘

►────────────────────────────────────────────────────────────────────────────►
    └─ -f─FileName─┘   ┌─ -u ─UserName─┐   ┌─ -a──────────────────┐
                       │                │   └─ -m ─MachineName─┘
                       ├─ -g─GroupName────────────────────────────┤
                       ├─ -b ─Broker──────────────────────────────┤
                       │            └─ -e ─ExeGroup─┘
                       ├─ -s──────────────────────────────────────┤
                       ├─ -r──────────────────────────────────────┤
                       ├─ -t──────────────────────────────────────┤
                       └─ -p──────────────────────────────────────┘

►────────────────────────────────────────────────────────────────────────────►◄
    └─ -w─waitTime─┘
```

Linux and UNIX systems:

```
►►──mqsilistaclentry─configmgrName────────────────────────────────────────────►
                                     └─ -n ─ConnectionFileName─┘

►────────────────────────────────────────────────────────────────────────────►
    └─ -f─FileName─┘   ┌─ -u ─UserName─┐   ┌─ -a──────────────────┐
                       │                │   └─ -m ─MachineName─┘
                       ├─ -g─GroupName────────────────────────────┤
                       ├─ -b ─Broker──────────────────────────────┤
                       │            └─ -e ─ExeGroup─┘
                       ├─ -s──────────────────────────────────────┤
                       ├─ -r──────────────────────────────────────┤
                       ├─ -t──────────────────────────────────────┤
                       └─ -p──────────────────────────────────────┘

►────────────────────────────────────────────────────────────────────────────►◄
    └─ -w─waitTime─┘
```

z/OS console command:

```
►►──mqsilistaclentry──configmgrName──────────────────────────────────►
                                      └─n=ConnectionFileName─┘

►─────────────────────────────────────────────────────────────────────►◄
   └─f=FileName─┘  └─u=UserName─┘  ┌─a=──────────────┐  └─w=waitTime─┘
                                   └─m=MachineName ─┘
                  ─g=GroupName────────────────────────
                  ─b=Broker ──────────────────────────
                        └─e=ExeGroup ─┘
                  ─s=─────────────────────────────────
                  ─r=─────────────────────────────────
                   ─t=────────────────────────────────
                   ─p= ───────────────────────────────
```

Parameters:

configmgrName

> (Optional - Windows. Required - Linux, UNIX systems, and z/OS) The name of the Configuration Manager for which the access control lists are to be displayed.

> On Linux, UNIX systems and z/OS, this must be the first parameter specified. It is case-sensitive on Linux and UNIX systems.

> The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-n*connectionfileName*

> (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

> The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

> If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

> and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

-f *FileName*

> (Optional) Place the results of this command into an XML file.

-u *UserName*

> (Optional) User name to which this entry refers, for example, TEST\ANOTHER.

-a (Optional) Allows a specified user to connect to all machines.

> **Note:** You can select -a or -m.

-m *MachineName*

> (Optional) The name of the machine from which a specified user can connect.

-g_GroupName_
> (Optional) Group to which this entry refers. For this reason, the name must adhere to the standard platform convention for group names.

-b _Broker_
> (Optional) The object is a broker object, and its name is specified as a parameter.

-e _ExeGroup_
> (Optional) The object is an execution group and its name is specified as a parameter of the form 'Broker\ExeGroup'. You must specify the **b** flag if you specify this flag.

-s _Subscription_
> (Optional) The object is a subscription object, and its name is specified as a parameter.

-r (Optional) The object is referring to the root topic.

-t (Optional) The object is referring to the main topology.

-p (Optional) The object refers to the "allresources" resource type. The authority that the principal has for this object applies to all objects, including the mqsicreateaclentry, mqsideleteaclentry, and mqsilistaclentry commands themselves.

-w _waitTime_
> (Optional) The time in seconds that the command waits for a response from the Configuration Manager. If you do not supply a value the command waits for 30 seconds.

Authorization:

The user ID used to invoke this command must have full control permissions for the object being changed; see "ACL permissions" on page 385 for more information.

Examples:
```
mqsilistaclentry CMGR01 -g GROUPA
mqsilistaclentry CMGR01 -b BROKER01
mqsilistaclentry CMGR01 -e BROKER01\ExeGrp01
```

mqsilistmqpubsub command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS through the BIPLSMP utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsilistmqpubsub** command to display the status of the WebSphere MQ Publish/Subscribe neighbor brokers to the specified WebSphere Event Broker broker.

This command indicates the status of the activity started by a previous join request (see "mqsijoinmqpubsub command" on page 295). The command reports on the status of each neighbor broker, which can be:

Active Broker status is active if the join request has completed successfully.

Inactive

> Broker status is inactive if the join has been initiated, but has not completed.

This command also shows the streams that are recognized by both the WebSphere Event Broker broker and its neighbor (on which messages can be published and distributed between the brokers). Stream information is provided only for neighbors with *active* status.

Use this command only if you are integrating with, or migrating from, an WebSphere MQ Publish/Subscribe broker network.

The output generated by this command is directed to stdout.

Syntax:

```
►►──mqsilistmqpubsub──brokername────────────────────────────────►◄
```

Parameters:

brokername

> (Required) The name of the broker for which you want a list of neighbors.

Authorization:

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

On Windows platforms, no authorization is required.

Responses:

- BIP8013 Component does not exist
- BIP8020 Unable to access database
- BIP8029 Broker not configured
- BIP8040 Unable to connect to database
- BIP8064 Internal broker error
- BIP8070 Database exception
- BIP8072 Database exception

Examples:

If there are no WebSphere MQ Publish/Subscribe brokers, and no **mqsijoinmqpubsub** command has been issued, this command returns the following message:

```
BIP8088I: There are no WebSphere MQ Publish/Subscribe neighbors
```

If an **mqsijoinmqpubsub** command has been issued, one of two response messages is displayed:

- For every broker that is an inactive neighbor of WBRK_BROKER (that is, a request has been made, using the mqsijoinmqpubsub or strmqbrk command, to

add the broker to the network, but negotiations for common streams are still in progress), the following message is displayed:

```
BIP8089I: WebSphere MQ Publish/Subscribe neighbor <brokername> is inactive.
```

- For every broker that is an active neighbor of WBRK_BROKER (that is, the two brokers are exchanging publications and subscriptions for each of the common streams), the following message is displayed:

```
BIP8090I: WebSphere MQ Publish/Subscribe neighbor <brokername> is active.
```

  Additional messages are displayed for active brokers to indicate the common streams for which publications and subscriptions are exchanged, in the following form:

```
BIP8091I:    Common stream  streamname
```

For example,

```
mqsilistmqpubsub WBRK_BROKER
```

might return the following responses:

```
BIP8090I: MQSeries Publish/Subscribe neighbor MQPS_BROKER_1 is active.
BIP8091I:    Common stream SYSTEM.BROKER.DEFAULT.STREAM.
BIP8091I:    Common stream STREAM0.
BIP8090I: MQSeries Publish/Subscribe neighbor MQPS_BROKER_2 is active.
BIP8091I:    Common stream SYSTEM.BROKER.DEFAULT.STREAM.
BIP8091I:    Common stream STREAM150.
BIP8089I: MQSeries Publish/Subscribe neighbor MQPS_BROKER_3 is inactive.
```

In this example, the WebSphere Event Broker broker has three WebSphere MQ Publish/Subscribe neighbors. Two of these neighbors are active and have been successfully joined to the WebSphere Event Broker broker. The third is inactive and is in the process of being joined.

The list of streams that are common to the WebSphere Event Broker broker and the two active WebSphere MQ Publish/Subscribe brokers are included in the response. For MQPS_BROKER_1, the streams SYSTEM.BROKER.DEFAULT.STREAM and STREAM0 are common. For MQPS_BROKER_2, the streams SYSTEM.BROKER.DEFAULT.STREAM and STREAM150 are common.

If a neighbor is inactive for a long period of time, it is likely that the communication link between the two brokers has been broken. Ensure that the WebSphere MQ connections between the two brokers (channels and transmission queues) are running, and that the WebSphere Event Broker and WebSphere MQ Publish/Subscribe brokers are both active.

mqsimigratecomponents command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

The **mqsimigratecomponents** command moves one or more components from one previously installed version (either Version 2.1 or Version 5 only) of the product to another. This command *must* be run from whichever version of the installed product is the later; whether it is the source or the destination.

Note:

1. For Version 2.1 of the product, Version 2.1.0.8 is the earliest release of the product supported.
2. For Version 5 of the product, Version 5.0.0.4 is the earliest release of the product supported .
3. The **-t** and **-s** parameters can accept version numbers in the form `Version.Release.Modification.Fixpack`, for example 5.0.0.4, and can also accept "5" and "2.1" as a shortened form for the previous versions that are supported.

You *must* have a Version 6.0 installation of the product with the required component code installed, that is, the broker component is installed if it is needed, and so on.

Before starting migration, stop any debugging sessions in the Control Center. It is not possible to migrate message flows that are being debugged.

You can invoke the command with various options to perform one of the following actions:

- Check on one or more components, without making any changes, to ensure that the components are suitable for the required migration.
- Move one or more components to a different version, in full or part.
- Undo, that is reverse, a successful move from one version to another, in full or part.
- Verify that a move has been successful.

Syntax:

```
>>--mqsimigratecomponents--+-check--+-+--------------------Component name--+-----+-><
                           | move   | |                                    | -q |
                           | undo   | |<---------|                         +-----+
                           +-verify-+ '----------'
```

check:

```
|-- -c ----------------------------------------------------------|
        '-s--source version-'  '-t--target version-'
```

move:

```
   .----.
   v    |
|--+-----+--+------------------+--+------------------+--|
   +--1--+  '-s--source version-'  '-t--target version-'
   +--2--+
   '--3--'
```

undo:

```
        .----.
        v    |
|-- -u --+----+--s--source version--t--target version--|
         +--1-+
         +--2-+
         '--3-'
```

verify:

```
|-- -v-----------------------------------------------|
      '-t--target version-'
```
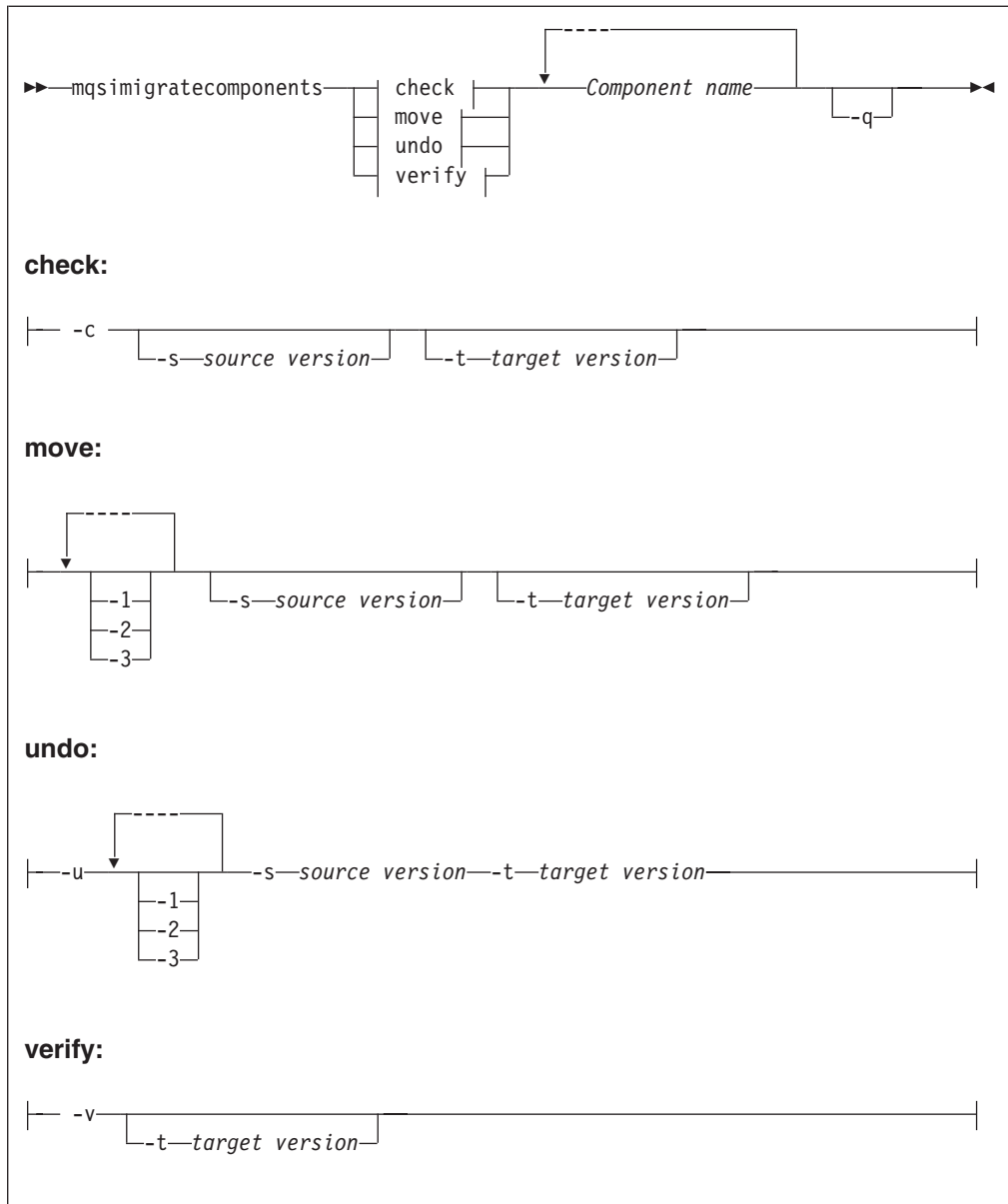
Parameters:

-c (Optional) Do a pre migration check of specified components to ensure the following:

- If more than one broker component is specified, the brokers share a database schema
- In all cases, a migrating broker database schema cannot be shared with a broker that is not being migrated at the same time.
- The auto-detected version of the broker matches any version specified on the command line.
- There are no 64 bit execution groups, if migrating from Version 6.0 to a previous release.
- The database tables to be copied from a previous release do not contain any rows that violate Version 6.0 index requirements:
  - Scanning through all the rows is the easiest way

– Global lock is taken if the broker is Version 2.1

The migration check can be run against a running component, or set of components. This does not impact the components, except for imposing a slight performance penalty. Note that on UNIX systems, the odbc.ini file needs to be migrated (that is, a new-format odbc.ini file needs to be created with the same set of data sources as the old one) before the check can be run, because the checking command needs to be able to access the broker database.

The check command either succeeds or fails, and prints a message about whether or not the migration should succeed, but no modifications are made during the process.

-v (Optional) Do a post-migration check of specified components to ensure the following:
- The correct database tables and queues exist for the specified version.
- The registry is in the correct format for the specified version.

-q (Optional) Print fewer status messages during the operation.

-1 (Optional) Do only registry and file system work. Use the **-1** parameter before the **-2** or **-3** parameters.

-2 (Optional) Do only WebSphere MQ work.

-3 (Optional) Do only database work.

-u (Optional) Undo a failed migration step; you must also specify at least one of -1, -2, or -3. You should only use this option when migration has failed, and also failed to auto-recover (a failure during split migration being one example).

-s_Source Version_
    (Optional) The previous version of the component. This value is detected automatically if not specified. See "Purpose" on page 304 for the restrictions to the version numbers of the product that are supported.

-t_Target Version_
    (Optional) The destination version of the component. This value is assumed to be the current version if not specified. See "Purpose" on page 304 for the restrictions to the version numbers of the product that are supported.

Component Name
    (Required) The name of the component to migrate; at least one must be specified.

Authorization:

When running single-step migration, the user ID used to invoke this command must have the ability to:
- Write to the registry for the product
- Modify databases associated with the components
- Modify queue definitions

For split migration, the user ID used to invoke this command must always have the ability to read from the registry for the product, and also have specific authorization for each step to succeed:
- -1 requires the ability to modify queue definitions
- -2 requires the ability to write to the registry for the product
- -3 requires the ability to modify databases associated with the components

Responses:

This command can produce a large number of possible responses, depending on the results of the various operations. Note that this command differs from other commands in the way it produces messages – they are displayed as needed, rather than being produced in a batch at the end of the program.

Examples:

The following example checks for migration of BROKER1 from V2.1 to Version 6.0:

```
mqsimigratecomponents -c BROKER1
BIP 0001I: Starting migration check for component 'BROKER1' to FAD level '3'
BIP 0002I: 'BROKER1' is version 2.1 (auto-detected)
BIP 0003I: Broker database 'BKRDB' and schema 'WMQIUSER' are not shared with other components.
BIP 0004I: No invalid rows found in broker database.
BIP 0005I: Migration check passed.
BIP 8071I: Successful command completion.
```

The following example does automatic migration of BROKER1 from V2.1 to Version 6.0:

```
mqsimigratecomponents BROKER1
BIP 0001I: Starting migration check for component 'BROKER1' to FAD level '3'
BIP 0002I: 'BROKER1' is version 2.1 (auto-detected)
BIP 0003I: Broker database 'BKRDB' and schema 'WMQIUSER' are not shared with other components.
BIP 0004I: No invalid rows found in broker database.
BIP 0005I: Migration check passed.
BIP 0020I: Starting registry migration for component 'BROKER1'.
BIP 0021I: Created top-level 'CurrentVersion' key
BIP 0021I: Created 'DSN' subkey
BIP 0022I: Created 'HTTPListener' subkey
BIP 0023I: Created 'HTTPListener.HTTPConnector' subkey
BIP 0024I: Created 'FADLevel' value; set to '3'
BIP 0025I: Created 'converters' value; set to ''
BIP 0028I: Moving registry data into 'CurrentVersion'.
BIP 0029I: Moved value 'AdminAgentPID'.
[repeat for each value]
BIP 0030I: Completed moving registry data into 'CurrentVersion'.
BIP 0025I: Finished registry migration for component 'BROKER1'.
BIP 0011I: Starting database table migration for 'BKRDB'.'WMQIUSER'.
BIP 0012I: Moving table 'BROKERRESOURCES' to 'somethingBROKERRESOURCES'.
BIP 0013I: Creating new table 'BROKERRESOURCES'.
BIP 0014I: Copying data from 'somethingBROKERRESOURCES' to 'BROKERRESOURCES'.
BIP 0015I: Successfully copied all data (322 rows).
 [repeat for each table]
BIP 0016I: Finished database table migration for 'BKRDB'.'WMQIUSER'.
BIP 0017I: Starting queue migration for 'BKRQM'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.AGGR.CONTROL'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.AGGR.REPLY'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.AGGR.REQUEST'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.AGGR.TIMEOUT'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.AGGR.UNKNOWN'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.INTERBROKER.MODEL.QUEUE'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.TIMEOUT.QUEUE'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.WS.ACK'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.WS.INPUT'.
BIP 0018I: Creating new queue 'SYSTEM.BROKER.WS.REPLY'.
BIP 0026I: Clearing queue 'SYSTEM.BROKER.ADMIN.QUEUE'.
BIP 0026I: Clearing queue 'SYSTEM.BROKER.EXECUTIONGROUP.QUEUE'.
BIP 0026I: Clearing queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY'.
BIP 0026I: Clearing queue 'SYSTEM.BROKER.IPC.QUEUE'.
BIP 0019I: Finished queue migration for 'BKRQM'.
BIP 0005I: Migration succeeded for component 'BROKER1'.
BIP 8071I: Successful command completion.
```

The following example is a repeat of the preceding example, but with the -q flag specified:

```
mqsimigratecomponents -q BROKER1
BIP 0001I: Starting migration check for component 'BROKER1' to FAD level '3'
BIP 0002I: 'BROKER1' is version 2.1 (auto-detected)
BIP 0005I: Migration check passed.
BIP 0020I: Starting registry migration for component 'BROKER1'.
BIP 0025I: Finished registry migration for component 'BROKER1'.
BIP 0011I: Starting database table migration for 'BKRDB'.'WMQIUSER'.
BIP 0016I: Finished database table migration for 'BKRDB'.'WMQIUSER'.
BIP 0017I: Starting queue migration for 'BKRQM'.
BIP 0019I: Finished queue migration for 'BKRQM'.
BIP 0005I: Migration succeeded for component 'BROKER1'.
BIP 8071I: Successful command completion.
```

mqsireadlog command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS through the BIPRELG utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsireadlog** command to retrieve the trace log for the specified component. This command is valid for:

User trace
> Specify the **-u** option.

Service trace
> Specify the **-t** option. You are recommended to use this option only if directed to do so by the action described in a BIPxxxx message, or by your IBM Support Center.

You can specify the output to be directed to file, or to stdout. The trace records returned by this command are in XML format and can be browsed with an XML browser. If you direct output to file, the data is written in code page utf-8. The file is therefore platform-independent, and can be transferred to other systems for browsing or formatting using the **mqsiformatlog** command.

If you transfer this file to another system, ensure that you use a file transfer program that does not convert the data (for example, by specifying a binary transfer option).

If you specify a broker, or any of its resources (execution group or message flow) you must have deployed them before you can start trace and read the log files.

Syntax:

User trace:

```
►►──mqsireadlog──component─────────────────────────────────────────────►
                        └─ -u ─┘  └─ -e ──egroup─┘  └─ -f ─┘

►────────────────────────────────────────────────────────────────────►◄
  └─ -o ──outputfilename─┘
```

Service trace:

```
►►──mqsireadlog──component── -t ──┬─ -b ──qualifier── -f ─┬────────────►
                                  └─ -e ──egroup──────────┘
                                                  └─ -f ─┘

►────────────────────────────────────────────────────────────────────►◄
  └─ -o ──outputfilename─┘
```
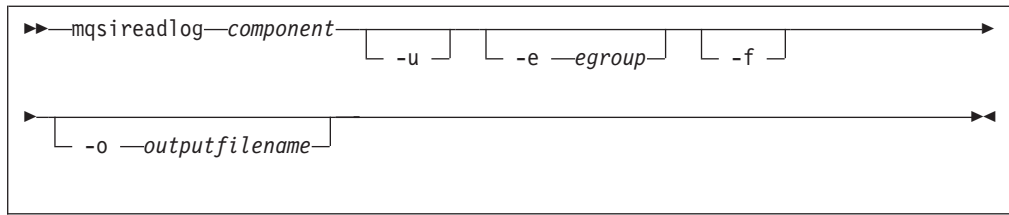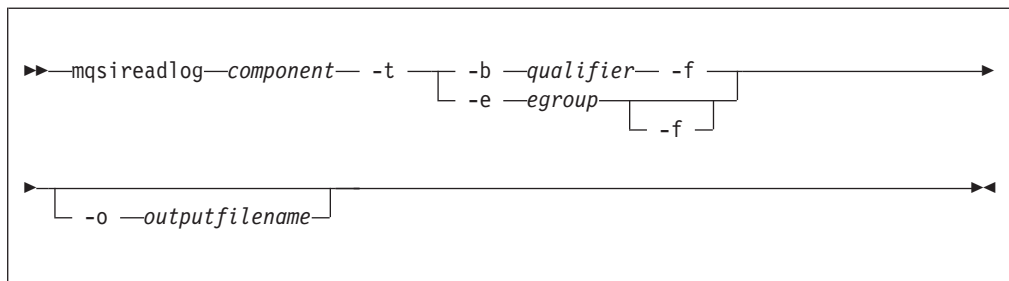
Parameters:

component
> (Required) The name of the component for which the log is to be read. This can be either a broker name or Configuration Manager name, or the fixed values , `UserNameServer`, `workbench`, or `utility` (all are case sensitive on Linux and UNIX systems).

-u (Optional) Read the log contents from the user trace log. This is valid **only** if you select the `broker` component.

-e *egroup*
> (Optional) The label of the execution group for which log information is to be read.

-o *outputfilename*
> (Optional) The name of the file into which to write the log data. If you specify a full pathname, the file is created in the directory specified. If you specify just the filename, the file is created in the current working directory. The contents of the file are written in code page `utf-8`, which is platform-independent and preserves data such as DBCS characters.
>
> You must specify a file name if you want to format the log using the **mqsiformatlog** command. If you do not specify a filename, the contents of the log are written to stdout. You are recommended to use a file extension of `.xml`.

-f (Optional) Read the log file directly from the file system. If you do not specify this option, the command sends an XML message to the component to request the log contents. If you have specified **-t** (service trace), you must specify this flag as well. Further details are given in "Additional parameters exclusive to service trace" on page 311.

> If you specify this option, stop tracing (using **mqsichangetrace**) before you use the **mqsireadlog** command. If the log file is in use when you issue this command with this flag specified, partial XML records might be returned. You can reduce the risk of this happening by specifying `-m safe` on the **mqsichangetrace** command. If the component being traced has itself stopped, you do not then need to issue a **mqsichangetrace** command.

If you do not stop tracing before you issue this command, check the contents of the log file created and remove any partial records from the end using a text editor before using the **mqsiformatlog** command, as partial records cannot be read by the format command.

Additional parameters exclusive to service trace:

Use these options only when directed to do so by your IBM Support Center or by a BIPxxxx message.

-t (Required) Read the log contents from the service trace log.

-b *qualifier*
(Required) Read the contents of the log for the broker agent, Configuration Manager agent, or User Name Server agent, or for the specified command utility program. This option is valid only if you have specified -t (service trace).

The following table shows the valid combinations of qualifier and component for service trace.

This option is generally used to trace the commands themselves. If you want to trace a particular command, run that command with environment variable MQSI_UTILITY_TRACE set to debug or normal before you issue this command to read the trace output generated.

Enter these values exactly as shown.

The agent trace is initiated when you specify the -b flag on the **mqsichangetrace** command. Do this only when directed to do so by a WebSphere Event Broker error message or when instructed to do so by your IBM Support Center.

The service trace is initiated when you specify the -b flag on the **mqsichangetrace** command. The format of the command is:

```
mqsireadlog <brokername> -t -b service -f -o service.xml
```

Do this only when directed to do so by a WebSphere Event Broker error message or when instructed to do so by your IBM Support Center.

-f (Required) Read the log file directly from the file system. When used with service trace, this flag has the same characteristics as when used with user trace. It remains optional if the -e flag is specified. You must specify this option if you specify the -b flag.

Qualifier	Component= <broker_name>	Component= ConfigMgr_name	Component= UserNameServer	Component= workbench	Component= utility
mqsichangebroker	x				
mqsichangeconfigmgr		x			
mqsichangetrace	x	x	x		
mqsichangeusernameserver			x		
mqsiclearmqpubsub	x				
mqsicreateaclentry					x
mqsicreatebroker	x				
mqsicreateconfigmgr		x			
mqsicreateusernameserver			x		

Qualifier	Component=<br_broker_name>	Component=ConfigMgr_name	Component=UserNameServer	Component=workbench	Component=utility
mqsideleteaclentry					x
mqsideletebroker	x				
mqsideleteconfigmgr		x			
mqsideleteusernameserver			x		
mqsiformatlog[1]					x
mqsijoinmqpubsub		x			
mqsilist[2]		x			x
mqsilistmqpubsub	x				
mqsilistaclentry					x
mqsireadlog	x	x	x		x
mqsireporttrace		x		x	
mqsistart	x	x	x		
mqsistop	x	x	x		
agent	x	x	x		
service	x	x	x		
workbench				x	

Notes:

1. Because this command does not have a component parameter, trace information is recorded in, and retrieved from, the *utility* component trace files. For further details see "mqsichangetrace command" on page 245.

2. If this command is invoked without a component, trace information is recorded in, and retrieved from, the *utility* trace files in addition to component specific files. For further details see "mqsichangetrace command" on page 245.

Authorization:

On UNIX platforms, the user ID must be a member of the **mqbrkrs** group. If the -f flag is specified, the user ID used to invoke this command must also have access to the trace file.

On Windows platforms, if the -f flag is specified, the user ID used to invoke this command must have access to the trace file. If the -f flag is not specified, the user ID used to issue the command must have **mqbrkrs** authority.

Responses:
- BIP8020 Unable to access database
- BIP8029 Broker not configured
- BIP8032 Unable to connect to queue
- BIP8033 Unable to send XML message
- BIP8035 Response not received within time-out
- BIP8036 Negative response received
- BIP8037 Unsupported flag selected

- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database
- BIP8132 Invalid qualifier

Examples:

User trace for broker `WBRK_BROKER`:

`mqsireadlog WBRK_BROKER -u -e default -o trace.xml`

Service trace for component `ConfigMgr`:

`mqsireadlog ConfigMgr -t -b agent -f mqsicreateconfigmgr -o trace.xml`

Service trace for utility **mqsiformatlog**:

`mqsireadlog utility -t -b agent -f mqsiformatlog -o trace.xml`

You can format the log file (`trace.xml` in the above examples) using the command **mqsiformatlog**, or view it using an XML editor or viewer.

mqsireload command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsireload** command to request the broker to stop and restart execution groups.

When you issue this command, a message is sent to the broker, which stops and restarts all its execution groups. You can specify a single execution group to be reloaded, but you are recommended to use the default form of this command to reload all execution groups.

Because an execution group does not stop until all message flows within it terminates, the ability of the broker to reload quickly depends on the processing time for the longest running message flow. This affects the performance of this command, and you are recommended to review any long-running message flows.

If you have included a plug-in node or parser within a message flow on the broker, these are deleted by this command, and the relevant termination functions called. When message flows are restarted, the resources used by these nodes and parsers are re-accessed and reacquired. However, you are recommended to ensure that user-written plug-in nodes and parsers provide their own mechanism to reload persistent state and data dynamically, and do not rely on the use of this command.

Syntax:
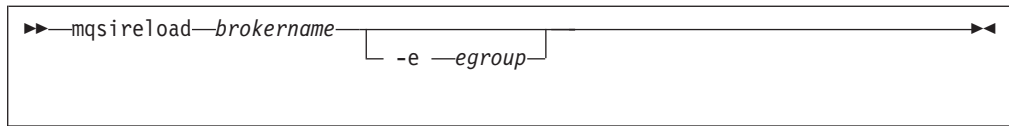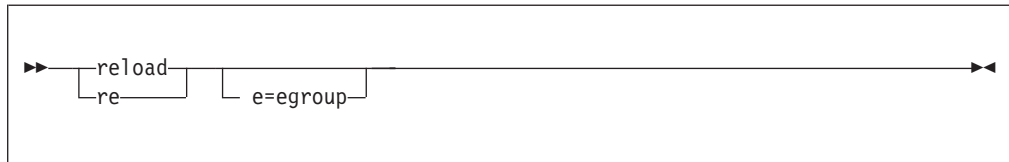
Windows platforms, Linux and UNIX systems:

```
►►──mqsireload──brokername──────────────────────────────────────────►◄
                          └─ -e ──egroup─┘
```

z/OS:

Synonym: re

```
►►──┬─reload─┬──┬──────────┬──────────────────────────────────────────►◄
    └─re─────┘  └─ e=egroup ─┘
```

Parameters:

brokername
> (Required - Windows platforms, Linux, and UNIX systems) The name of the broker to which the reload request is sent.

-e *egroup*
> (Optional) The name of the execution group that is to be reloaded. If this parameter is not specified, all execution groups on the specified broker are stopped and restarted.

Authorization:

The user ID used to issue the command must be a member of the group **mqbrkrs**.

Responses:

No additional responses are returned.

Examples:

Windows platforms and UNIX platforms:
```
mqsireload broker1
```

z/OS:
```
F MQP1BRK,re
```

mqsireportflowstats command

Supported Platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsireportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsichangeflowstats** command.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►── mqsireportflowstats── brokername ──┬──-a──┬──────────────────────────────────►
                                        └──-s──┘

►──┬── -e ──ExecutionGroupLabel── -f ──MessageFlow──┬──────────────────────────►◄
   └── -g ── -j ───────────────────────────────────┘
```

z/OS:

Synonym rs
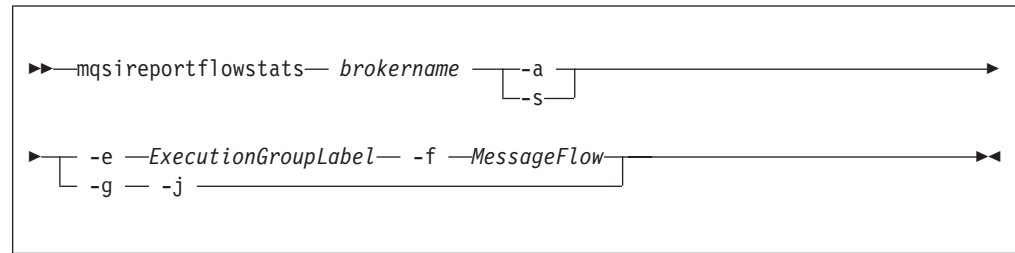
```
►►──┬──reportflowstats──┬──┬──a=yes──┬───────────────────────────────────────────►
    └──rs───────────────┘  └──s=yes──┘

►──┬── e=ExecutionGroupLabel ── f =MessageFlow──┬──────────────────────────────►◄
   └── g=yes ──j=yes──────────────────────────┘
```
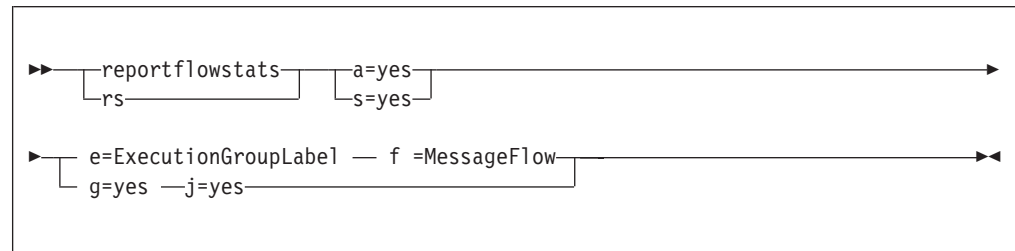
Parameters:

brokername
> (Required - Windows platforms, Linux, and UNIX systems) Specify the label of the broker for which the previously stored accounting and statistics options are to be reported.

-a (Required) Specify that the command reports the stored settings for the archive accounting and statistics collection.

> **Note:** You must specify either **-a** or **-s**. If you do not specify at least one of these arguments you receive a warning message.

-s (Required) Specify that the command reports the stored settings for the snapshot accounting and statistics collection.

> **Note:** You must specify either **-a** or **-s**. If you do not specify at least one of these arguments you receive a warning message.

-e *ExecutionGroupLabel*
> (Required) Specify the label for the execution group, for which accounting and statistics options are to be changed.

> **Note:** You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive a warning message.

-f *MessageFlow*
> (Required) Specify the label for the message flow, for which accounting and statistics options are to be changed.

Note: You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

-g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

> **Note:** You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive a warning message.

-j (Required) Specifies that the command applies to **all**message flows that belong to the execution group.

> **Note:**
> 1. You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.
> 2. If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

Authorization:

The user Id used to issue the command must have mqbrkrs authority.

Responses:

This command returns the following responses:
- BIP2268 Request to report on message flow node ' ': message flow does not exist
- BIP8001 Unknown flag selected
- BIP8004 Invalid flags or arguments missing
- BIP8007 Mandatory argument missing
- BIP8013 Component does not exist
- BIP8020 Unable to access the database
- BIP8029 Broker not configured
- BIP8033 Unable to send XML message
- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database
- BIP8071: Successful command completion

Examples:

Request a report for message flow "MyFlow1" in the execution group "default" for broker "BrokerA" for both archive and snapshot statistics collection:
```
mqsireportflowstats BrokerA -s -a -e default -f MyFlow1
```

Request a report of the snapshot options that are currently stored for all message flows in all execution groups for broker "BrokerA" :
```
  mqsireportflowstats BrokerA -s -g -j
```

mqsireportproperties command

Supported platforms:
- Windows
- Linux and UNIX systems

- z/OS through the BIPRPPT utility; see "z/OS utility jobs" on page 382

Purpose:

Use the **mqsireportproperties** command to examine the values of properties set using the **mqsichangeproperties** command.

Start the broker using mqsistart, and then issue the command on the machine running the broker.

If you change any value (with **mqsichangeproperties**) you **must** stop and restart the broker for the change to take effect.

Syntax:

```
>>--mqsireportproperties--brokername--+--b --ComponentName----------->
                                      |                          |
                                      +--e --ExecutionGroupLabel-+

>--o --ObjectName--+--n --name--+------------------------------------><
                   |            |
                   +--a---------+
                   |            |
                   +--r---------+
```
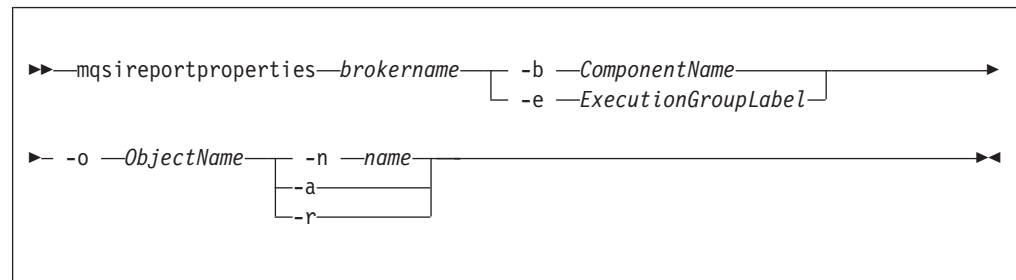
Parameters:

brokername
   (Required) The name of the broker.

-e *ExecutionGroupLabel*
   (Optional) The label of the execution group for which a report is required.

-b *ComponentName*
   (Optional) The name of the component for which a report is required. For the http listener component use *httplistener*.

-o *ObjectName*
   (Required) The name of the object whose properties you want to change. This must be set to one of the values listed below.

-n *name*
   (Optional) Display only the named property

-a   (Optional) Indicates that all property values of the object are displayed, and does not recurse into properties that have child values.

-r   (Optional) Indicates that all property values of the object are displayed and, additionally, displays the child values for all properties that have child values.

- You **must** select one, and only one, option from **-n, -a**, and **-r**.
- You **must** select one, and only one, option from **-e** and **-b**.

The *ObjectName* parameter can have a value for inter-broker communications or for the *httplistener* component. For inter-broker communications, use the following value:

- *DynamicSubscriptionEngine*

For the *httplistener* component, which includes the HTTPInput and HTTPReply nodes, the following *ObjectName* values are available:

- *HTTPListener* for controlling the HTTPListener process.
- *HTTPConnector* for controlling HTTP communication with the HTTPInput and HTTPReply nodes.
- *HTTPSConnector* for controlling HTTPS communication with the HTTPInput and HTTPReply nodes.

The *httplistener ComponentName* is required when using the HTTP *ObjectName* settings.

For the *httplistener* component, **mqsireportproperties** does not report those properties that have not been explicitly set with **mqsichangeproperties**, even if those properties have a default setting. For example, the default HTTPSConnector port that is used (unless it has been changed) is 7083. However this value is not reported by **mqsireportproperties** unless it has been changed from this default with **mqsichangeproperties**. To see the default values for all properties that **mqsireportproperties** can report on, see the documentation for **mqsichangeproperties**.

Authorization:

No specific authority is required to invoke this command.

Examples:

Display all the current HTTPListener settings (that have been changed from their default value)

```
mqsireportproperties TEST -b httplistener -o HTTPListener -a
```

Display the HTTPSConnector port setting (if it has been changed from the default value)

```
mqsireportproperties TEST -b httplistener -o HTTPSConnector -n port
```

Display recursively all the current settings for the inter-broker communication.

```
mqsireportproperties TEST -e default -o DynamicSubscriptionEngine -r
```

mqsireporttrace command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsireporttrace** command to display the trace options currently in effect. This command is valid for:
- User trace. Specify the **-u** option.

  On z/OS, user trace is run against a specific execution group for a broker. You can narrow the command down further to trace a specific message flow.
- Service trace. Specify the **-t** option. Use this option only if directed to do so by the action described in a BIPxxxx message, or by your IBM Support Center.

  On z/OS, service trace can be run for a specific execution group (like user trace). Unlike user trace, service trace can also be run against an active agent (that is, a broker or User Name Server).

If you specify a broker, or any of its resources (execution group or message flow), you must have deployed them before you can query trace settings.

Syntax:

Windows platforms, Linux, and UNIX systems:
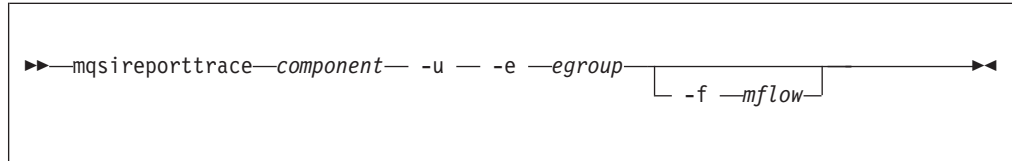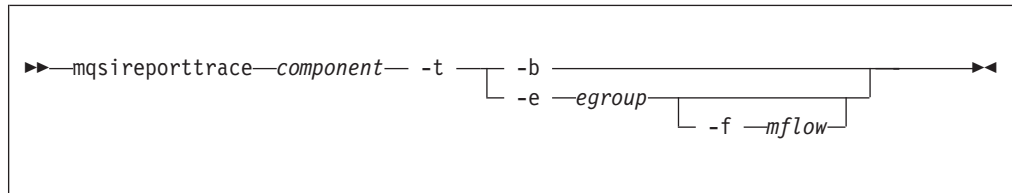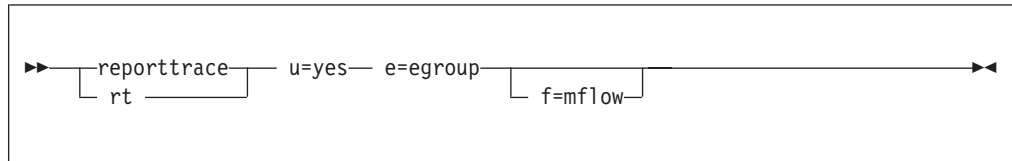
User trace:

```
►►──mqsireporttrace─component── -u ── -e ─egroup──────────────────────►◄
                                               └─ -f ──mflow─┘
```

Service trace:

```
►►──mqsireporttrace─component── -t ──┬─ -b ──────────────────────┬──►◄
                                     └─ -e ──egroup──────────────┘
                                                   └─ -f ──mflow─┘
```

z/OS:

User trace:

```
►►──┬─reporttrace─┬── u=yes── e=egroup──────────────────────►◄
    └─ rt ────────┘                   └─f=mflow─┘
```

Service trace:

```
►►──┬─reporttrace─┬── t=yes──┬─ b=yes ─────────────────────►◄
    └─ rt ────────┘          └─ e=egroup───────────────┘
                                         └─ f=mflow─┘
```
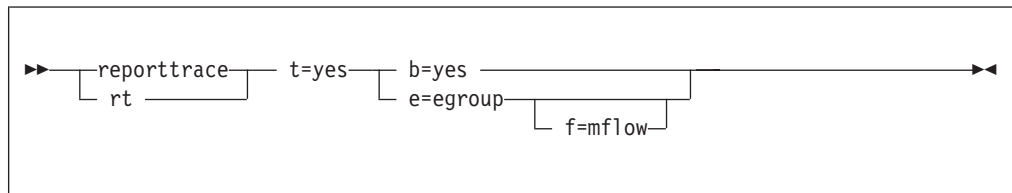
Parameters:

component
> (Required - Windows platforms, Linux, and UNIX systems) The name of the broker for which options are reported, or the fixed value `UserNameServer`. Both values are case sensitive on Linux and UNIX systems.

-u (Required for user trace) Derive report information from the user trace.

-e *egroup*
> (Required for user trace, otherwise optional) The label of the execution group for which a report is required. This is valid only if you have specified a broker as the component.

On z/OS this name is case sensitive. Include the name in single quotes if it contains mixed-case characters.

-f *mflow*

(Optional) The label of the message flow for which a report is required. This is valid only if you have specified both a broker as the component and an execution group.

On z/OS this name is case sensitive. Include the name in single quotes if it contains mixed-case characters.

Additional parameters exclusive to service trace:

Use these options only when directed to do so by your IBM Support Center or by a BIPxxxx message.

-t   (Required for service trace) Derive report information from the service trace.

-b   (Alternative to **-e** on all platforms) Request a report for `agent` function.

Authorization:

The user ID used to issue the command must have **mqbrkrs** authority.

Responses:
- BIP8002 Selected flags incompatible (z/OS only)
- BIP8003 Duplicate flag detected (z/OS only)
- BIP8013 Component does not exist
- BIP8020 Unable to access database
- BIP8029 Broker not configured
- BIP8032 Unable to connect to queue (Windows platforms, Linux, and UNIX systems only)
- BIP8033 Unable to send XML message
- BIP8035 Response not received within time-out
- BIP8036 Negative response received
- BIP8037 Unsupported flag selected
- BIP8038 Unsupported command option
- BIP8039 Execution group not available
- BIP8040 Unable to connect to database
- BIP8045 Message flow not available
- BIP8158 Invalid format for command (z/OS only)
- BIP8159 Unknown parameter "..." (z/OS only)

Examples:

Windows platforms, Linux, and UNIX systems:
```
mqsireporttrace BrokerA -t -e "exgrp1"
```

z/OS:
```
F MQP1BRK,reporttrace t=yes,e='exgrp1'
```

mqsirestoreconfigmgr command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

This command restores a previously backed up Configuration Manager Archive.

Note:

1. It must be run on the same machine as the Configuration Manager and the Configuration Manager must not be running.
2. The command tries to stop the Configuration Manager if it is running before it attempts a backup restore.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsirestoreconfigmgr──configMgrName── -d ──DirectoryPathName──────────────►

►── -a──archiveName──────────────────────────────────────────────────────►◄
```

z/OS:

```
►►──mqsirestoreconfigmgr──configMgrName──d=DirectoryPathName──a=archiveName──►◄
```

Parameters:

configMgrName
    (Required) The name of the Configuration Manager.

-d *DirectoryPathName*
    (Required) The directory where the archive is placed.

-a *archiveName*
    (Required) Specifies the backup archive name.

Authorization:

This command changes security privileges for the ServiceUserID; the user ID used to invoke this command must be a member of the Windows **Administrators** group on this local system.

Responses:

This command returns the following responses:

- BIP8056 Unknown queue manager

Examples:

mqsisetsecurity command
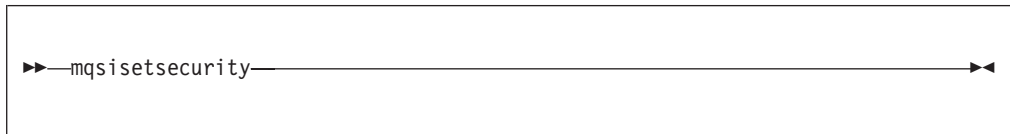
Supported platforms:
- Windows

Purpose:

Use the **mqsisetsecurity** command to create the Windows groups that WebSphere Event Broker requires for secure access to its runtime libraries and data.

This command runs automatically as part of the installation process of WebSphere Event Broker. If WebSphere MQ is installed after WebSphere Event Broker, you can issue this command to add your account to the MQM group provided you have **Administrator** authority.

Syntax:

```
►►──mqsisetsecurity──────────────────────────────────────────►◄
```

Parameters:

None

Authorization:

The user ID used to invoke this command must belong to the **Administrators** group on the local system because creating groups is an administrative task.

Responses:
- BIP8610 User does not have **Administrator** authority
- BIP8611 Unable to create **mqbrkrs** group because the user does not have access to the requested information
- BIP8612 Unable to create **mqbrkrs** group because the computer name is invalid
- BIP8613 Unable to create **mqbrkrs** group because an error occurs when accessing the alias
- BIP8614 The **mqbrkrs** group already exists
- BIP8615 The **mqbrkrs** group has been created successfully
- BIP8616 The current user is already a member of the **mqbrkrs** group
- BIP8617 The **mqbrkrs** group does not exist
- BIP8614 The user could not be added because the account type is invalid

mqsi_setupdatabase command

Supported platforms:
- Linux and UNIX systems

Purpose:

Use the **mqsi_setupdatabase** command if you are setting up a supported database on Linux or UNIX systems.

This command sets up the required database links and should be run for each database vendor type (DB2, Oracle, or Sybase) used within your WebSphere Event Broker environment. This applies to both broker and user (application) databases.

You must complete this procedure before you create the broker database. You are recommended to do this after you have installed your database, but you can complete this process before database installation if you specify the intended database installation directory correctly during this setup.

Syntax:

```
►►──mqsi_setupdatabase── database──database_home_directory────────────────►◄
```

Parameters:

database
    (Required) The database that you are installing. Possible values are :
    • db2
    • oracle9
    • oracle10
    • sybase12_5.1

database_home_directory
    (Required) The name of the directory in which the database is (or will be) installed (for example, `/usr/lpp/oracle9`).

Authorization:

The user ID used to invoke this command must be **root**. It must also be a member of the **mqbrkrs** group.

mqsistart command

Supported platforms:
• Windows
• Linux and UNIX systems
• z/OS

Purpose:

Windows platforms, Linux, and UNIX systems:

Use **mqsistart** to start a WebSphere Event Broker component. If the queue manager associated with this component (defined in the corresponding create command) is not already running, it is also started by this command. However, no additional WebSphere MQ services (listeners, channels, or channel initiators) associated with the started component are started. Use WebSphere MQ Services (or MQSC) to start any additional services required.

Successful completion of this command indicates that the Windows service, or Linux or UNIX daemon has started successfully, and that the component startup has been initiated. Check the Windows system event log or the Linux orUNIX syslog to determine if the component and all related software have started successfully, are initially active, and remain in an active state. Any errors that have prevented successful startup, detected by the component are recorded in the log. Continue to monitor the Windows system event log or Linux or UNIX syslog.

If the queue manager supporting the component specified on this command is not already running, it is started by this command, but on Windows platforms , is not started as a service. On this system, it stops if you log off. To avoid this happening, either remain logged on, or change the startup status of the queue manager service (described in "Creating a Configuration Manager on Windows" on page 107). (If you lock your workstation, the queue manager does not stop.)

z/OS:

Use the **startcomponent** command to start a broker, Configuration Manager, or User Name Server when its controller (control process) is already running.

The broker consists of a controller and the component, either a broker, Configuration Manager, or User Name Server.

When the controller address space is started, this causes the component to start automatically. This behavior can be changed by an optional start parameter in the started task. If the parameter is set to MAN, the component does not start automatically; the default is AUTO.

Issuing commands against the controller means issuing start, stop, or modify commands from the console to the controller address space. There are two scenarios using this command. The first is that the controller is started with the parameter MAN instead of AUTO. The second is that, after a **stopcomponent** command, the component has to be restarted.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsistart──component──────────────────────────────►◄
```

z/OS:
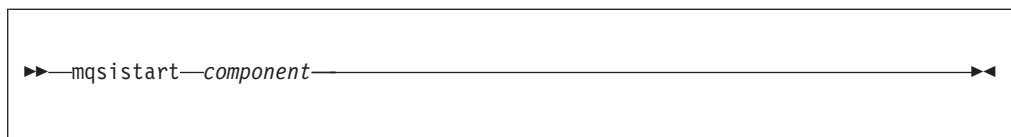
Synonym: sc

```
►►──┬─startcomponent─┬──────────────────────────────►◄
     └─sc────────────┘
```
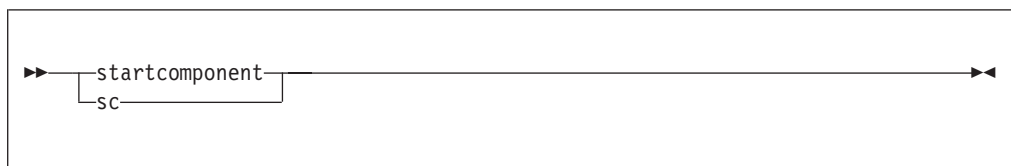
Parameters:

component
> (Required - Windows platforms, Linux, and UNIX systems) This must be either a broker name, Configuration Manager name, or the fixed value `UserNameServer/DatabaseInstanceMgr`. All the names are case sensitive on Linux and UNIX systems.

Authorization:

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On Linux and UNIX systems, the user ID used to invoke this command must either be **root** or must be the same as that specified in the `-i` parameter when the component is created. It must also be a member of the **mqbrkrs** and **mqm** groups.

When the Windows service for your platform or Linux or UNIX daemon is started, it runs under the user ID specified by the `-i` flag on the appropriate`mqsicreatexxxx` command. The component only starts if the ServiceUserID specified is authorized to access the home directory (where WebSphere Event Broker has been installed), and the working directory (if specified by the `-w` flag on the mqsicreatexxxx command).

The security requirements for using this command are summarized in "Security requirements for Windows platforms" on page 389 and "Security requirements for Linux and UNIX platforms" on page 388.

Responses:

Windows platforms, Linux, and UNIX systems only:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8015 Component cannot be started
- BIP8018 Component running
- BIP8024 Unable to locate executable
- BIP8025 Component disabled
- BIP8026 Unable to start component
- BIP8027 Unable to start WebSphere MQ
- BIP8028 WebSphere MQ unavailable
- BIP8030 Unable to modify user privileges
- BIP8048 Unable to start queue manager
- BIP8056 Unknown queue manager
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping

Examples:

Windows platforms, Linux, and UNIX systems:
```
mqsistart WBRK_BROKER
mqsistart DatabaseInstanceMgr
```

z/OS
```
 F MQ00BRK,sc
```

mqsistartmsgflow command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsistartmsgflow** command to start message flows.

Syntax:

```
>>--mqsistartmsgflow---b--Brokername------------------------------->

>------------------------------------------------------------------->
        -e--ExecutionGroupName-------------------
                                -m--MsgFlowName-

>---- connection cfgParameterFilename------------------------------->
      -i--ipAddress--p--port--q--qMgr-        -w--timeoutValue-

>-----------------------------------------><
        -v--traceFileName-
```

Parameters:

-b *BrokerName*
:   (Required) The name of the broker on which to start message flows.

    If you do not specify the **-e** and **-m** flags, all message flows on the broker are started.

-e *ExecutionGroupName*
:   (Optional) The name of the execution group on which message flows are started.

-m *msgFlowName*
:   (Optional) The name of the message flow being started.

    You can specify only one message flow in a single command. However, if you do not specify this parameter, all message flows on the execution group or broker are started.

    If you specify this flag you must also specify the **-e** flag.

connection cfgParameterFileName
:   (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

    The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

This parameter can also be specified as **-n** (as used in the "mqsideploy command" on page 289)

Note: If you do not supply this parameter, you must supply the **-i**, **-p**, and **-q** parameters.

-i *ipAddress*
> (Optional) The host name or IP address of the Configuration Manager.
>
> If you are using this file on z/OS and want to connect to the local host you must set the value to \"\".

-p *Port*
> (Optional) The port number of the Configuration Manager.

-q *qMgr*
> (Optional) The name of the WebSphere MQ queue manager that the Configuration Manager is using.
>
> If you do not supply the **-i**, **-p**, and **-q** parameters, you must specify the *-connection* parameter.

-w *timeoutValue*
> (Optional) The time in seconds that the utility waits to ensure that the start command completed; the default value is 60.

-v *TraceFileName*
> (Optional) Sends internal debug trace information to the specified file.

Authorization:

To start message flows you need to have sufficient authority defined in the Configuration Manager's access control list.

The permissions required are the same as the permission required to do the equivalent function in the Message Brokers Toolkit; see "ACL permissions" on page 385 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

0   (Success) States that the request completed successfully and the state of all message flows has been updated.

2   (Failure) States that at least one message flow can not be put into the correct state for any reason.

98   States that the Configuration Manager cannot be reached.

99   States that the supplied arguments to the utility are not valid.

Examples:

Starts all message flows on broker B1, which is controlled by the Configuration
Manager whose connection details are described in cm1.configmgr. Control is
returned to the caller when all message flows in the broker are reported as started,
or the default time of one minute elapses, whichever is sooner.

```
mqsistartmsgflow -n cm1.configmgr -b B1
```

Starts all message flows on broker B1, which is controlled by the Configuration
Manager whose connection details are described in cm1.configmgr. Control is
returned to the caller when all message flows in the broker are reported as started,
or two minutes elapses, whichever is sooner.

```
mqsistartmsgflow -connection cm1.configmgr -b B1 -w 120
```

Start all message flows on Broker B1, which is controlled by the Configuration
Manager listening on queue manager QM1 on local host:1414.

```
 mqsistartmsgflow -q QM1 -i localhost -p 1414 -b B1
```

Enter **mqsistartmsgflow** to display usage information:

```
> mqsistartmsgflow
BIP1024I: Starts message flows.

> Syntax:
mqsistartmsgflow (-n cfgParameterFileName | (-i ipAddress -p port -q qMgr))
 -b brokerName [-e executionGroupName [-m flowName]] [-w timeoutValue]
[-v traceFileName]
Command Options:
'-n cfgParameterFileName' File containing Configuration Manager connection parameters (.configmgr)
'-i ipAddress' IP address or host name of the Configuration Manager
'-p port' port number of the Configuration Manager
'-q qMgr' queue manager of the Configuration Manager
'-b brokerName' name of the broker on which to start message flows
'-e executionGroupName' name of the execution group on which to start message flows.
If this is not specified, all message flows on the broker will be started.
'-m flowName' name of the message flow to start.
If this is not specified, all message flows on the execution group will be started.
'-w timeoutValue' time to wait (in seconds) for message flows to start (Default=60)
'-v traceFileName' send verbose internal trace to the specified file.
```

mqsistop command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsistop** command to stop a WebSphere Event Broker component.

Use the **stopcomponent** command to stop a broker, Configuration Manager, or
User Name Server when its controller (control process) is running.

The broker consists of a controller and the component, either a broker,
Configuration Manager, or User Name Server.

Issuing commands against the controller means issuing start, stop, or modify
commands from the console to the controller address space. After a
stopcomponent command, the component has to be restarted.

Syntax:

Windows platforms, Linux, and UNIX systems:

```
►►──mqsistop──component──────────────────────────────────────►◄
                        └─ -q ─┘  └─ -i ─┘
```

z/OS:

Synonym: pc

```
►►──┬─stopcomponent─┬──────────────►◄
    └─pc────────────┘  └─ i=yes ─┘
```
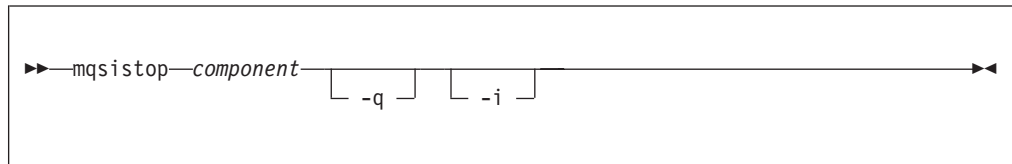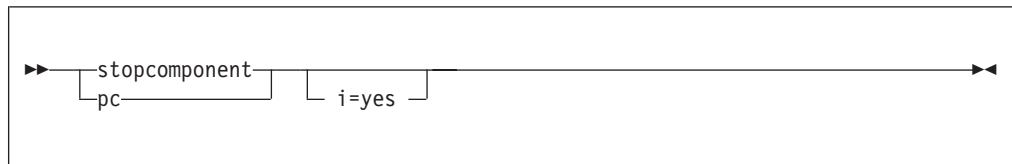
Parameters:

component

(Required - Windows platforms, Linux, and UNIX systems) This must be either a broker name, Configuration Manager name, or the fixed value `UserNameServer`. All the names are case sensitive on Linux and UNIX systems.

-q (Optional - Windows platforms, Linux, and UNIX systems only) Stop the WebSphere MQ queue manager associated with this resource.

Specify this flag only if the WebSphere Event Broker component is the last (or only) WebSphere Event Broker component active on this queue manager. This command initiates a controlled shutdown of the queue manager, and informs other users of the queue manager that it is closing. If you do not stop other WebSphere Event Broker components that use this queue manager before you issue this command with this option, stop them afterwards or restart the queue manager.

If you use this option, be aware that any listeners associated with this queue manager are not stopped with the queue manager. Stop these manually after issuing this command.

-i (Optional) Immediately stop the broker.

Specify this flag only if invoking the command without it, has failed to stop the broker.

On z/OS this flag fails if a previous stop command has failed to complete.

Authorization:

On Windows platforms, the user ID used to invoke this command must belong to the **Administrators** group.

On Linux and UNIX systems, the user ID used to invoke this command must either be **root** or must be the same as that specified in the **-i** parameter when the

component was created. It must also be a member of the **mqbrkrs** group. If you specify the **-q** parameter to stop the queue manager, the user ID must also be a member of the **mqm** group.

The security requirements for using this command are summarized in "Security requirements for Windows platforms" on page 389 and "Security requirements for Linux and UNIX platforms" on page 388.

Responses:

Windows platforms, Linux, and UNIX systems only:
- BIP8012 Unable to connect to system components
- BIP8013 Component does not exist
- BIP8016 Component cannot be stopped
- BIP8019 Component stopped
- BIP8030 Unable to modify user privileges
- BIP8049 Unable to stop queue manager
- BIP8093 Queue manager being created
- BIP8094 Queue manager stopping

Examples:

Windows platforms, Linux, and UNIX systems:
```
mqsistop ConfigMgr -q
mqsistop DatabaseInstanceMgr
```

z/OS:
```
F MQ00BRK,pc
```

mqsistopmsgflow command

Supported platforms:
- Windows
- Linux and UNIX systems
- z/OS

Purpose:

Use the **mqsistopmsgflow** command to stop message flows.

Syntax:

```
>>--mqsistopmsgflow---b--Brokername-----------------------------------------------------------------------------------------------------►

  ►----------------------------------------------------------------------------------------------------------------►
        └-e--ExecutionGroupName-----------------------------------┐
                                  └-m--MsgFlowName-┘

  ►--┬-- connection cfgParameterFilename-----------┬---┬---------------------┬---►
     └-i--ipAddress---p--port---q--qMgr-┘          └-w--timeoutValue-┘

  ►---------------------------------------------------------►◄
     └-v--traceFileName-┘
```
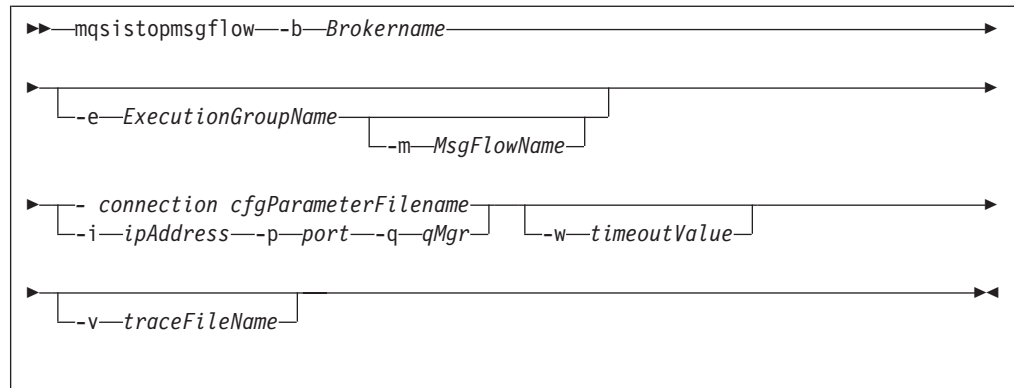
Parameters:

-b *BrokerName*
:   (Required) The name of the broker on which to stop message flows.

    If you do not specify the **-e** and **-m** flags, all message flows on the broker are stopped.

-e *ExecutionGroupName*
:   (Optional) The name of the execution group on which message flows are stopped.

-m *msgFlowName*
:   (Optional) The name of the message flow being stopped.

    You can specify only one message flow in a single command. However, if you do not specify this parameter, all message flows on the execution group or broker are stopped.

    If you specify this flag you must also specify the **-e** flag.

connection cfgParameterFileName
:   (Optional) The name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

    The file is in XML, using the `.configmgr` format saved by the Eclipse GUI. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr host="localhost" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

    If you are using this file on z/OS you must remove the statement `encoding="UTF-8"` from the first line, to leave the statement as:

```
<?xml version="1.0"?>
```

    and remove the value for the host attribute, to leave the statement as:

```
<configmgr host="" listenerPort="1414" queueManager="QNAME"
           securityExit="test.myExit"/>
```

    This parameter can also be specified as **-n** (as used in the "mqsideploy command" on page 289)

    Note: If you do not supply this parameter, you must supply the **-i**, **-p**, and **-q** parameters.

-i *ipAddress*
:   (Optional) The host name or IP address of the Configuration Manager.

If you are using this file on z/OS and want to connect to the local host you must set the value to \"\".

-p *Port*
(Optional) The port number of the Configuration Manager.

-q *qMgr*
(Optional) The name of the WebSphere MQ queue manager that the Configuration Manager is using.

If you do not supply the **-i**, **-p**, and **-q** parameters, you must specify the *-connection* parameter.

-w *timeoutValue*
(Optional) The time in seconds that the utility waits to ensure that the stop command completed; the default value is 60.

-v *TraceFileName*
(Optional) Sends internal debug trace information to the specified file.

Authorization:

To stop message flows you need to have sufficient authority defined in the Configuration Manager's access control list.

The permissions required are the same as the permission required to do the equivalent function in the Message Brokers Toolkit; see "ACL permissions" on page 385 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

0   (Success) States that the request completed successfully and the state of all message flows has been updated.

2   (Failure) States that at least one message flow can not be put into the correct state for any reason.

98   States that the Configuration Manager cannot be reached.

99   States that the supplied arguments to the utility are not valid.

Examples:

Stops all message flows on execution group `default` on broker B1, which is controlled by the Configuration Manager whose connection details are described in `cm1.configmgr`. Control is returned to the caller when all message flows in the execution group are reported as stopped, or the default time of one minute elapses, whichever is sooner.

```
mqsistopmsgflow –n cm1.configmgr –b B1 -e default
```

Stops the message flow `flow1` on execution group `default` on broker B1, which is controlled by the Configuration Manager whose connection details are described in `cm1.configmgr`. Control is returned to the caller when the message flow is reported as stopped, or the default time of one minute elapses, whichever is sooner.

```
mqsistopmsgflow –connection cm1.configmgr –b B1 -e default -m flow1
```

Enter **mqsistopmsgflow** to display usage information:

```
> mqsistopmsgflow
BIP1025I: Stops message flows.

> Syntax:
mqsistopmsgflow (-n cfgParameterFileName | (-i ipAddress -p port -q qMgr))
 -b brokerName [-e executionGroupName [-m flowName]] [-w timeoutValue]
[-v traceFileName]
Command Options:
'-n cfgParameterFileName' Configuration Manager connection file (.configmgr)
'-i ipAddress' IP address or host name of the Configuration Manager
'-p port' port number of the Configuration Manager
'-q qMgr' queue manager of the Configuration Manager
'-b brokerName' name of the broker on which to stop message flows
'-e executionGroupName' name of the execution group on which to stop message flows.
If this is not specified, all message flows on the broker will be stopped.
'-m flowName' name of the message flow to start.
If this is not specified, all message flows on the execution group will be stopped.
'-w timeoutValue' time to wait (in seconds) for message flows to stop (Default=60)
'-v traceFileName' send verbose internal trace to the specified file.
```

z/OS specific information

Follow the links below for more information:
- "Administration in z/OS"
- "z/OS customization" on page 335
- "z/OS JCL variables" on page 345
- "z/OS utility jobs" on page 382

Administration in z/OS

In the z/OS environment, commands are issued through the console and others in batch jobs.
- <hlq>.SBIPSAMP has all the JCL samples to customize.
- <hlq>.SBIPPROC has all the JCL procedures to customize.

See the following topics for more information:

-
- "Issuing commands to the z/OS console"
- "Guidance for issuing console commands in z/OS" on page 334

Issuing commands to the z/OS console

You operate the broker or User Name Server using the z/OS START, STOP, and MODIFY commands. You can issue commands, and get responses back from:
- The z/OS operator console
- The TSO CONSOLE facility
- The CONSOLE interface from REXX
- Products such as SDSF
- z/OS automation products, such as NetView®

However, you are likely to need to issue commands with mixed case, because execution group names are often in mixed case. You can issue commands with mixed case on the z/OS console, using the REXX CONSOLE interface, with products like SDSF V2.10 and higher, and NetView. Releases of SDSF before V2.10, and the TSO CONSOLE facility, do not support passing mixed case data.

If you do not have support for mixed case, you can submit the commands through a batch job. For example:

```
//MI01CMD JOB MSGCLASS=H
//  COMMAND 'f MQP1BRK,ct t=yes,e='default',l=debug,f='lowmfl''
//STEP1 EXEC PGM=IEFBR14
```

If your product for issuing console commands supports mixed case input, you might need to take special actions to use mixed case. For example in SDSF on OS/390® V2.10 and above, you can enter mixed case keyword values by typing /, pressing ENTER, and entering the command in the popup window. If you enter / followed by the command, the command is translated to uppercase. In NetView, prefix the command with NETVASIS to get lowercase support.

Guidance for issuing console commands in z/OS

These examples use a broker called MQP1BRK. Start and stop the broker or User Name Server using the MVS START (S) and STOP (P) commands. If you want to pass information to the broker or User Name Server while it is running, use the MVS MODIFY command (F) to issue commands. For example:

```
F MQP1BKR,rt
```

This list summarizes the rules you must follow when issuing console commands:
- Each command starts with a verb followed by zero or more keyword=value pairs.
- There must be one or more blanks between the verb and the first keyword.
- All characters are converted to lowercase, unless they are within quotation marks.
- Multiple keywords are separated by, with no blanks.
- The keyword is always case insensitive.
- The value is case sensitive, unless specified otherwise (for example Yes/No, trace-modes).
- Each keyword must be followed by the equals sign = and parameter value. Enter parameters in any order in the form:

  ```
  flag=value
  ```

  and separate them with commas. Repeated parameters are not allowed.
- Strings that contain blanks or special characters must be enclosed in single quotation marks (’). If the string itself contains a quotation mark, the quotation mark is represented by two single quotation marks.
- The verb and keywords are not case sensitive.
- On mixed case consoles the case of values in single quotation marks (’) is not changed.
- For YES/NO flags, all case combinations are allowed.
- The maximum length of the MODIFY command is 126 characters, including

  ```
  F taskname,
  ```

An example console command:

```
F MQP1BRK,changetrace u=Yes,l=normal,e='myExecutionGroup'
```

z/OS customization

This is an introduction topic for a number of reference topics in the area of z/OS customization. See the links under Related reference information.

Naming conventions for WebSphere Event Broker for z/OS

Decide upon a naming convention for your WebSphere Event Broker components to make customizing, operating, and administering easier.

Each broker requires its own queue manager. Base your broker name on the queue manager name. For example, append BRK to the queue manager name of MQP1, to give MQP1BRK. This naming convention has the following advantages:

- It is easy to associate the broker with the queue manager, because they both begin with the same characters.
- The started task name has the same name as the broker.
- You can have the broker name as part of the data set name, which makes it easier to administer.

Similarly, base the Configuration Manager and User Name Server name on the queue manager name. For example, append CMGR orUNS to the queue manager name.

Using this convention, you might have the following component names for a queue manager MQP1.

- A broker called MQP1BRK with:
  - A started task name of MQP1BRK.
  - A started task user ID of MQP1BRK.
  - A PDSE containing definitions called hlq.MQP1BRK.xxx.
  - A UNIX System Services directory structure like /xxx/yyy/MQP1BRK.
  - A DB2 group called MQP1GRP.
- A User Name Server called MQP1UNS with:
  - A started task name of MQP1UNS.
  - A started task user ID of MQP1UNS.
  - A PDSE containing definitions called hlq.MQP1UNS.xxx.
  - A UNIX System Services directory structure like /xxx/yyy/MQP1UNS.
- A Configuration Manager called MQP1CMGR with:
  - A started task name of MQP1CMGR.
  - A started task user ID of MQP1CMGR.
  - A PDSE containing definitions called hlq.MQP1CMGR.xxx.
  - A UNIX System Services directory structure like /xxx/yyy/MQP1CMGR.

Plan for expansion by selecting names that allow growth.

If your broker name is in uppercase, ensure the broker name is also in uppercase in the workbench.

Customization tasks and roles (z/OS)

Systems programmers do most of the customization of WebSphere Event Broker. Tasks that need to be performed by other people in your organization are

identified in the table below:

Role	Task
z/OS systems programmer	"Customizing the z/OS environment" on page 57
	"Setting up z/OS security" on page 29
	"Summary of required access (z/OS)" on page 337
	"Customizing UNIX System Services" on page 61
	"Creating the broker component" on page 101
	"Starting the component" on page 102
	"Checking APF attributes of bipimain on z/OS" on page 66
DB2 administrator	"Setting up DB2" on page 31
	"Summary of required access (z/OS)" on page 337
	"DB2 planning (z/OS)" on page 62
	"Prime DB2" on page 100
WebSphere MQ administrator	"Setting up WebSphere MQ" on page 32
	"Summary of required access (z/OS)" on page 337
	"WebSphere MQ planning (z/OS)" on page 63
	"Creating the broker component" on page 101
	"Defining the started tasks to z/OS Workload Manager (WLM)" on page 64
WebSphere Event Broker administrator	"Setting up workbench access on z/OS" on page 33
	"Summary of required access (z/OS)" on page 337
	"Creating a broker on z/OS" on page 94
	"Creating the broker component" on page 101
	"Checking APF attributes of bipimain on z/OS" on page 66
Performance specialist	"Defining the started tasks to z/OS Workload Manager (WLM)" on page 64
Security administrator	"Setting up z/OS security" on page 29
	"Setting up DB2" on page 31
	"Summary of required access (z/OS)" on page 337
	"Creating Publish/Subscribe user IDs" on page 33
Data administrator	"Setting up z/OS security" on page 29
	"Summary of required access (z/OS)" on page 337
	"Disk space requirements (z/OS)" on page 340
	"Starting the component" on page 102
	"Using the file system on z/OS" on page 65

Some tasks, for example defining queue security, overlap two different roles.

Summary of required access (z/OS):

The following information summarizes the access that the professionals in your organization require.

Authorizations required for the WebSphere Event Broker started-task user ID:

No access is required to the component PDSE.

The directory authorizations required are:
- READ/EXECUTE access to the executables in `<INSTPATH>/bin`, where `<INSTPATH>` is the directory where WebSphere Event Broker for z/OS is installed by SMP/E.
- READ/WRITE/EXECUTE access to the component directory.
- READ/WRITE access to the home directory.
- In UNIX System Services, the started task user ID and the WebSphere Event Broker administrator user ID must both be members of the groups that have access to the installation and component directories, because they both need privileges over these. The owner of these directories needs to give the appropriate permissions to this group.

DB2 authorizations for the started task user ID and the table owner ID are required:
- If there is a profile for db2subsystem.RRSAF in the DSNR class the started task user ID needs access to the profile. For example, the following RACF command shows whether the profile exists:
  ```
  RLIST  DSNR (DB2P.RRSAF)
  ```

  and the following command gives the required access:
  ```
  PERMIT  DB2P.RRSAF  CLASS(DSNR) ID(WQMITASK)  ACCESS(READ)
  ```
- SELECT privilege on the tables SYSIBM.SYSTABLES, SYSIBM.SYSSYNONYMS, and SYSIBM.SYSDATABASE.
- SELECT, UPDATE, INSERT, and DELETE privileges on all broker system tables.
- `DB2_TABLE_OWNER` must be a valid authorization ID of the started task user ID.
- EXECUTE authority on the `DSNACLI` plan, or equivalent for the started task user ID.

WebSphere MQ authorizations:

Enable WebSphere MQ security to protect your WebSphere MQ resources. If all WebSphere MQ security switches are enabled, define the following profiles and give the started task user ID the listed access to each profile. For each profile access listed, `<MQ_QMNAME>` represents the WebSphere MQ queue manager that the WebSphere Event Broker component is connected to, and `TASKID` represents the WebSphere Event Broker started-task user ID.
- Connection security: READ access to profile `<MQ_QMNAME>.BATCH` of class `MQCONN`. For example, for queue manager MQP1 and started task ID `TASKID`, use the RACF commands:

```
RDEFINE MQCONN MQP1.BATCH UACC(NONE)
PERMIT MQP1.BATCH CLASS(MQCONN) ID(TASKID) ACCESS(READ)
```
- Queue security: UPDATE access to profile <MQ_QMNAME>.queue of class MQQUEUE for all queues. Consider creating profiles for the following queues:
  - All component queues using the generic profile SYSTEM.BROKER.**
  - Any transmissions queues defined between component queue managers.
  - Any queues defined in message flows.
  - Dead-letter queues.

  For example, for queue manager MQP1 and started task ID TASKID, use the following RACF commands to restrict access to the component queues:
  ```
  RDEFINE MQQUEUE MQP1.SYSTEM.BROKER.** UACC(NONE)
  PERMIT MQP1.SYSTEM.BROKER.** CLASS(MQQUEUE) ID(TASKID) ACCESS(UPDATE)
  ```
- Context security: CONTROL access to profile <MQ_QMNAME>.CONTEXT of class MQADMIN. For example, for queue manager MQP1 and started task ID TASKID, use the following RACF commands:
  ```
  RDEFINE MQADMIN MQP1.CONTEXT UACC(NONE)
  PERMIT MQP1.CONTEXT CLASS(MQADMIN) ID(TASKID) ACCESS(CONTROL)
  ```
- Alternate user security: Define the alternate user authority as: UPDATE access to profile <MQ_QMNAME>.ALTERNATE.USER.id of class MQADMIN, where id represents the service ID of the Windows Configuration Manager component. For example, for queue manager MQP1, started task ID TASKID, and configuration service ID CFGID, use the following RACF commands:
  ```
  RDEFINE MQADMIN MQP1.ALTERNATE.USER.CFGID UACC(NONE)
  PERMIT MQP1.ALTERNATE.USER.CFGID  CLASS(MQADMIN) ID(TASKID) ACCESS(UPDATE)
  ```

  UPDATE access to profile <MQ_QMNAME>.ALTERNATE.USER.id of class MQADMIN, where id represents the user ID of, for example, a Publish/Subscribe request.
- Process and namelist security: If you have WebSphere MQ security switches enabled in your system for process and namelist security, you do not need to define any access profiles in a WebSphere Event Broker default configuration.

Authorizations required for the WebSphere Event Broker administrator:
- ALTER access to the component PDSE.
- READ, WRITE, and EXECUTE access to the component directory.
- READ/EXECUTE access to <INSTPATH>, where <INSTPATH> is the directory where WebSphere Event Broker for z/OS is installed by SMP/E.
- EXECUTE access to the shell scripts in <INSTPATH>/bin.
- In UNIX System Services, the started task user ID and the WebSphere Event Broker administrator user ID must both be members of the groups that have access to the installation and component directories, because they both need privileges over these. The owner of these directories needs to give the appropriate permissions to this group.

Authorizations required for the DB2 administrator:

If you use security, the DB2 coordinator needs to have the following authorizations to run the DB2 configuration jobs (BIPxDBxx):
- PDSE authorizations: ALTER access to the component PDSE.
- Directory authorizations: EXECUTE access to the component directory.
- DB2 authorizations: SYSCTRL or SYSADM authority.

Authorizations required for the WebSphere MQ administrator:

To run the WebSphere MQ configuration jobs (BIPxMQxx) you require the following authorizations:

- PDSE authorizations: READ access to the component PDSE.
- Directory authorizations:
  - READ/EXECUTE access to the executables in `<INSTPATH>/bin`, where `<INSTPATH>` is the directory where WebSphere Event Broker for z/OS is installed by SMP/E.
  - EXECUTE access to the component directory.

Enable WebSphere MQ security to protect your WebSphere MQ resources. If all WebSphere MQ security switches are enabled, define the following profiles and give the WebSphere MQ administrator the listed access to each profile in order to run the WebSphere MQ configurations jobs. For each profile access listed, `MQ_QMNAME` represents the WebSphere MQ queue manager that the WebSphere Event Broker component is connected to, and `MQADMIN` represents the WebSphere MQ administrator ID:

- Connection security: READ access to profile `<MQ_QMNAME>.BATCH` of class `MQCONN`. For example, for queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the following RACF commands:

```
RDEFINE MQCONN MQP1.BATCH UACC(NONE)
PERMIT MQP1.BATCH CLASS(MQCONN) ID(MQADMIN) ACCESS(READ)
```

- Queue security: UPDATE access to profile `<MQ_QMNAME>.queue` of class `MQQUEUE` for component queues created or deleted. You can create a generic profile SYSTEM.BROKER.** For example, for queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the following RACF commands to restrict access to the component queues:

```
RDEFINE MQQUEUE MQP1.SYSTEM.BROKER.** UACC(NONE)
PERMIT MQP1.SYSTEM.BROKER.** CLASS(MQQUEUE) ID(MQADMIN) ACCESS(UPDATE)
```

  UPDATE access to profile `<MQ_QMNAME>.queue` of class `MQQUEUE` for some system queues used during the create/delete job. You can create a generic profile <MQ_QMNAME>.**

- Command security:
  - To run the `BIP$MQ01` job you need:
    - ALTER access to `<MQ_QMNAME>.DEFINE.QLOCAL` of class `MQCMDS`.
    - ALTER access to `<MQ_QMNAME>.DEFINE.QMODEL` of class `MQCMDS`.
  - To run the `BIP#MQ01` job you need:
    - ALTER access to `<MQ_QMNAME>.DELETE.QLOCAL` of class `MQCMDS`.
    - ALTER access to `<MQ_QMNAME>.DELETE.QMODEL` of class `MQCMDS`.

  For queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the following RACF commands:

```
RDEFINE MQADMIN MQP1.DELETE.QLOCAL UACC(NONE)
PERMIT MQP1.DELETE.QLOCAL CLASS(MQADMIN) ID(MQADMIN) ACCESS(ALTER)
```

- Resource command security: ALTER access to `MQP1.QUEUE.queue` of class `MQADMIN` for each queue created or deleted. You can create a generic profile SYSTEM.BROKER.**. For example, for queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the RACF commands:

```
RDEFINE MQADMIN MQP1.QUEUE.SYSTEM.BROKER.** UACC(NONE)
PERMIT MQP1.QUEUE.SYSTEM.BROKER.** CLASS(MQADMIN) ID(MQADMIN) ACCESS(ALTER)
```

- Process and namelist security: If you have WebSphere MQ security switches enabled in your system for process and namelist security, you do not need to define any access profiles in a WebSphere Event Broker default configuration.

For a description of how to implement WebSphere MQ security using RACF, see "Security requirements for z/OS" on page 391.

Authorizations required for the DB2 subsystem started-task user ID:

DB2 needs ALTER access to the catalog value specified in DB2_STOR_GROUP_VCAT because it creates data sets with this high-level qualifier.

Disk space requirements (z/OS)

The WebSphere Event Broker for z/OS installation uses approximately 400 MB of disk space; you should plan on using 500 MB which will also allow for new service to be applied and also for the component directories.

When you apply service, if you do not replace your existing install (for example you apply the new fix pack level alongside your existing install) then you need to plan the same amount of disk space for the higher service level libraries.

If you are transferring the files using tar to package them, you need approximately 200 MB of space for the .tar file.

You can check how much space is used and how much is free in an HFS using the OMVS command:

```
df -P /pathname
```

100 MB is 3 276 800 512 byte sectors.

The following table gives guidance on the space required for a minimum installation (base installation and verification test) of WebSphere Event Broker for z/OS components.

Component	Space required	Purpose
DB2 database (broker only)	6 MB	Holds the broker system tables.
Component directory	20 MB	Holds the runtime information and output directories for the component. This includes trace files and other user problem determination data, which can become large.
Component PDSE	1 MB	Holds the customization and administration jobs, procedures, and data for the component. The data set must be allocated with a fixed record length of 80 (LRECL=80) and a format of FB 80. Reserve directory space for 50 members, or use a PDSE if possible.
Started task user ID home directory	8 GB	Collect diagnostic materials, for example dumps. Dumps are usually more than 500MB in size. 8 GB of space needs to be available in the HFS, but many user IDs can have their home directory in the HFS.

> The Component directory and the Started task user ID home directory must be different to ensure that, when dumps are taken in the Started task user ID home directory, they do not cause problems with the runtime broker that still needs to write to the Component directory.

Binding a DB2 plan to use data-sharing groups

During customization, you can specify which plan name to use, or use the default DSNACLI. If you are using XPLINK, the default plan is called DSNACLX. If you want your broker to access DB2 data-sharing groups other than its own, the DSNACLI plan must be bound in a special way. If the broker uses one data sharing group, but might want to access tables on DSNONE and DSNTWO, which are in different data-sharing groups, amend the DB2 supplied job DSNTIJCL to do the following:

```
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNAOCLI)MEMBER(DSNCLIQR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNONE.DSNAOCLI)MEMBER(DSNCLIQR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLICS)ISOLATION(CS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLINC)ISOLATION(NC)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIRR)ISOLATION(RR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIRS)ISOLATION(RS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIUR)ISOLATION(UR)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIC1)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIC2)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIF4)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIMS)
BIND PACKAGE (DSNTWO.DSNAOCLI)MEMBER(DSNCLIQR)
BIND PLAN(DSNACLI)-
PKLIST(*.DSNAOCLI.DSNCLICS -
*.DSNAOCLI.DSNCLINC -
*.DSNAOCLI.DSNCLIRR -
*.DSNAOCLI.DSNCLIRS -
*.DSNAOCLI.DSNCLIUR -
*.DSNAOCLI.DSNCLIC1 -
*.DSNAOCLI.DSNCLIC2 -
*.DSNAOCLI.DSNCLIF4 -
*.DSNAOCLI.DSNCLIMS -
*.DSNAOCLI.DSNCLIQR )
```

Customization planning checklist for z/OS

Use the information contained in the following tables to make a note of the values to use when you customize your system variables. For further information, see:

- "Installation information - Broker and User Name Server" on page 94
- "Installation information - Configuration Manager" on page 109

- "DB2 information" on page 94
- "Component information - broker" on page 95
- "Component information - Configuration Manager" on page 109
- "Component information - User Name Server" on page 119

Contents of the broker PDSE

After successfully customizing the broker, the members in your broker PDSE are as follows:

Description	Type	Name
Alter DB2 table spaces and tables	JCL	BIPALDB
Alter WebSphere MQ table spaces and tables	JCL	BIPALMQ
Broker profile	profile	BIPBPROF
Run mqsibrowse command	JCL	BIPBRWS
Run the mqsichangebroker command	JCL	BIPCHBK
Run the mqsichangeflowstats command	JCL	BIPCHMS
Run the mqsichangeproperties command	JCL	BIPCHPR
Run the mqsiclearmqpubsub command	JCL	BIPCLMP
Job to: • Create the DB2 tables • Create and defineWebSphere MQ resources • Create database resources • Create the broker registry "Sample BIPCBRK file" on page 352 lists the shipped BIPCRBK file.	JCL	BIPCRBK
Create the DB2 storage group, database and table spaces. "Sample BIPCRDB file" on page 360 lists the shipped BIPCRDB file.	JCL	BIPCRDB
Run the mqsideletebroker command	JCL	BIPDLBK
Delete DB2 table spaces and tables	JCL	BIPDLDB
Broker DB2 dsnaoini file	dsnaoini	BIPDSNAO
Broker started task JCL, explicitly customized. Rename BIPEDIT to a name that identifies it with the current component	JCL	BIPEDIT (MQP1BRK)
Format log	JCL	BIPFMLG
Job to convert BIPBPROF profile to a valid ENVFILE. "Sample BIPGEN file" on page 370 lists the shipped BIPGEN file.	JCL	BIPGEN
Run the mqsijoinmqpubsub command	JCL	BIPJNMP
Run the mqsilist command	JCL	BIPLIST
Run the mqsilistmqpubsub command	JCL	BIPLSMP
Change DB2 definitions when migrating from Version 2.1 to Version 6.0	JCL	BIPMGTB
Read logs	JCL	BIPRELG
Run the mqsireportflowstats command	JCL	BIPRPMS

Run the `mqsireportproperties` command	JCL	BIPRPPR
Define a data source, user ID, and password for user data sources.	JCL	BIPSDBP
(started task). Rename BIPBRKP to **the same as the** **++STARTEDTASKNAME++**. "Sample BIPBRKP file" on page 375 lists the shipped BIPBRKP file.	JCL	BIPBRKP

Use the standard z/OS IPCS facilities to take a dump. You require access to the following:

- `COMPONENT_PDSE`
- `ComponentDirectory`
- `SYS1.MIGLIB`
- `SYS1.SBLSCLI0`
- `SYS1.PARMLIB`
- `IPCS`

Contents of the User Name Server PDSE

After successfully customizing the User Name Server, the members in your User Name Server PDSE are as follows:

Description	Type	Name
Alter WebSphere MQ table spaces and tables	JCL	BIPALMQ
Issue the mqsichangeusernameserver command.	JCL	BIPCHUN
Create User Name Server. "Sample BIPCRUN file" on page 368 lists the shipped BIPCRUN file.	JCL	BIPCRUN
Issue the mqsideleteusernameserver command.	JCL	BIPDLUN
JCL customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDUN	JCL	BIPEDIT
Format log	JCL	BIPFMLG
Generate ENVFILE. "Sample BIPGEN file" on page 370 lists the shipped BIPGEN file.	JCL	BIPGEN
Issue the mqsilist command.	JCL	BIPLIST
Read logs	JCL	BIPRELG
started task. Rename BIPUNSP to **the same as the** **++STARTEDTASKNAME++**. "Sample BIPUNSP file" on page 380 lists the shipped BIPUNSP file.	JCL	BIPUNSP

Use the standard z/OS IPCS facilities to take a dump. You require access to the following:

- `COMPONENT_PDSE`

- ComponentDirectory
- SYS1.MIGLIB
- SYS1.SBLSCLI0
- SYS1.PARMLIB
- IPCS

Contents of the Configuration Manager PDSE

After successfully customizing the Configuration Manager, the members in your Configuration Manager PDSE are as follows:

Description	Type	Name
Alter WebSphere MQ table spaces and tables	JCL	BIPALMQ
Issue the mqsibackupconfigmgr command	JCL	BIPBUCM
Issue the mqsichangeconfigmgr command	JCL	BIPCHCM
Issue the mqsicreateaclentry command	JCL	BIPCRACL
Create a Configuration Manager. "Sample BIPCRCM file" on page 357 lists the shipped BIPCRCM file.	JCL	BIPCRCM
Issue the mqsicreateexecutiongroup command	JCL	BIPCREG
Issue the mqsideleteaclentry command	JCL	BIPDLACL
Issue the mqsideleteconfigmgr command	JCL	BIPDLCM
Issue the mqsideleteexecutiongroup command	JCL	BIPDLEG
Issue the mqsideploy command	JCL	BIPDPLY
Edit macro for customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDCM	JCL	BIPEDIT
Format logs	JCL	BIPFMLG
Job to convert BIPCPROF profile to a valid ENVFILE. "Sample BIPGEN file" on page 370 lists the shipped BIPGEN file.	JCL	BIPGEN
Issue the mqsilistaclentry command	JCL	BIPLIACL
Issue the mqsilist command	JCL	BIPLIST
Read logs	JCL	BIPRELG
Issue the mqsirestoreconfigmgr command	JCL	BIPRSCM
Issue the mqsistopmsgflow command	JCL	BIPSPMF
Issue the mqsistartmsgflow command	JCL	BIPSTMF
Started task. Rename BIPCMGRP to **the same as the ++STARTEDTASKNAME++**. "Sample BIPCMGRP file" on page 378 lists the shipped BIPCMGRP file.	JCL	BIPCMGRP

Use the standard z/OS IPCS facilities to take a dump. You require access to the following:

- COMPONENT_PDSE
- ComponentDirectory

- SYS1.MIGLIB
- SYS1.SBLSCLI0
- SYS1.PARMLIB
- IPCS

z/OS JCL variables

The following table lists all the JCL variables that you can customize in alphabetical order, together with a description and an example value.

JCL variable	Description	Example value
++ARM++	Specifies if a component should use Automatic Restart Management (ARM).	YES
++ARMNAME++	ARM name. Required if ++ARM++ is set to YES.	Q482BRK
++ARMTYPE++	ARM type. Required if ++ARM++ is set to YES.	SYSWMQI
++COMPONENTDATASET++	The dataset where all JCL relevant to a particular component is saved.	BIP.V6R0M0.BROKER.MQP1BRK
++COMPONENTDIRECTORY++	The HFS directory where the component exists. This directory includes subdirectories, for example, /log and /registry	/mqsi/V6R0M0/brokers/MQP1BRK
++COMPONENTNAME++	The name you give the component when you create it.	MQP1BRK
++COMPONENTPROFILE++	Profile name.	BIPBPROF, BIPCPROF, or BIPUPROF
++COMPONENTRESOURCE++	Required specifically by BIPBRWS. It represents the resource to be displayed.	BNBRCONNECTIONS
++DB2BUFFERPOOL++	The name of the DB2 buffer pool associated with the component for which the JCL is submitted.	BP0
++DB2CONVERSION++	Specifies the DB2 Converter.	SINGLE
++DB2CURRENTSQLID++	The DB2 user ID for the component and commands.	MQP1BRK
++DB2DATABASE++	The name of the DB2 database associated with the component for which the JCL is submitted.	DMQP1BRK
++DB2HLQ++	DB2 high-level-qualifier	SYS2.DB2.V710
++DB2LOCATION++	The DB2 location value of the DB2 subsystem to which the component connects.	DSN710PK
++DB2PLAN++	The DB2 plan value.	DSNTEP71
++DB2PLANNAME++	The DB2 plan name.	DSNACLI

JCL variable	Description	Example value
++DB2PROGRAM++	The DB2 program value.	DSNTEP2
++DB2RUNLIB++	The DB2 run library value.	DSN710PK.RUNLIB.LOAD
++DB2STORAGEGROUP++	The name of the DB2 storage group associated with the component for which the JCL is submitted.	MQP1STOR
++DB2SUBSYSTEM++	The DB2 subsystem ID which the component connects.	DFK4
++DB2TABLEOWNER++	Broker tables schema name.	MQP1BRK
++EXECUTIONGROUPNAME++	The name of an execution group.	Default
++HOME++	The file system home directory for the component's user ID. This is required to dynamically generate the ENVFILE from BIPBPROF, and is also used for STDOUT and STDERR output. You must have the appropriate RACF authorities to write to this file system directory when submitting JCL to run a command.	/u/home
++INSTALL++	The directory where you install the product.	/usr/lpp/mqsi/V6R0M0
++JAVA++	Location of Java installation.	/usr/lpp/java/IBM/J1.4
++LANGLETTER++	The letter for the language in which you want messages shown.	E (English)
++LEHLQ++	LE high-level-qualifier	PP.ADLE370.ZOS140
++LOCALE++	Locale of environment where commands are run by submitting JCL.	C
++MESSAGECASE++	Determines if messages should appear in mixed case.	YES
++NEIGHBOURQUEUEMANAGER++	Required specifically by BIPCLMP for the mqsiclearmqpubsub command. It represents the name of the neighboring QueueManager	MQP1
++OBJECTNAME++	Required specifically by BIPRPPR for the mqsireportproperties command. It represents the name of the object.	DynamicSubscriptionEngine

JCL variable	Description	Example value
++OPTIONS++	Many commands submitted by JCL require additional options. See each command's reference material for additional information on options specific to that command.	N/A
++PARENTQUEUEMANAGERNAME++	Required specifically by BIPJNMP for the mqsijoinmqpubsub command. It represents the name of the parent QueueManager.	MQP1
++PROPERTYNAME++	Required specifically by BIPCHPR for the mqsichangeproperties command. It represents the name of the property to be changed.	multicastEnabled
++PROPERTYVALUE++	Required specifically by BIPCHPR for the mqsichangeproperties command. It represents the new value of the property to be changed.	false
++QUEUEMANAGER++	The name of the QueueManager associated with the component for which you submit the JCL.	MQP1
++STARTEDTASKNAME++	Name of the Started Task JCL. This can be a maximum of 8 characters.	MQP1BRK
++TIMEZONE++	Time zome of environment where commands are run by submitting JCL.	GMT0BST
++WMQHLQ++	WebSphere MQ high-level-qualifier	MQM.V530

Related tasks
"Information required to create a broker" on page 94

z/OS sample files supplied

The following files are provided as sample source that you can customize:
- "Sample BIPBPROF file" on page 348
- "Sample BIPCBRK file" on page 352
- "Sample BIPCPROF file" on page 354
- "Sample BIPCRCM file" on page 357
- "Sample BIPCRDB file" on page 360
- "Sample BIPCRUN file" on page 368
- "Sample BIPDSNAO file" on page 369
- "Sample BIPGEN file" on page 370

Sample BIPBPROF file

```
#*********************************************************************
#*                                                                 *
#* @START_COPYRIGHT@                                                *
#*                                                                 *
#* Licensed Materials - Property of IBM;                           *
#* 5655-G97 (c) Copyright IBM Corp. 2004;                          *
#* All Rights Reserved;                                            *
#* US Government Users Restricted Rights - use,                    *
#* duplication or disclosure restricted by GSA                     *
#* ADP Schedule Contract with IBM Corp.;                           *
#* See Copyright Instructions                                      *
#*                                                                 *
#* @END_COPYRIGHT@                                                  *
#*                                                                 *
#*********************************************************************
#*           IBM WebSphere Event/Message Brokers                   *
#*                                                                 *
#* Sample profile for a broker.                                    *
#*                                                                 *
#* IMPORTANT:                                                       *
#*                                                                 *
#*     You must submit BIPGEN each time you update this profile!   *
#*                                                                 *
#*********************************************************************
#* MORE INFORMATION - See:                                         *
#*                                                                 *
#*      WebSphere Event/Message Brokers Information Centre.         *
#*                                                                 *
#*********************************************************************
#* CUSTOMIZE HERE FOR YOUR INSTALLATION
#* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
#*                                                                 *
#*     Replace    ++INSTALL++
#*                     WBI Brokers installation directory.
#*                     e.g. '/usr/lpp/mqsi'
#*
#*     Replace    ++COMPONENTDIRECTORY++
#*                     Broker directory.
#*                     e.g. '/mqsi/brokers/MQ01BRK'
#*
#*     Replace    ++COMPONENTDATASET++
#*                     Component dataset.
#*                     e.g. 'BIP.BROKER.MQ01BRK'
#*
#*     Replace    ++COMPONENTNAME++
#*                     Broker name.
#*                     e.g. 'MQ01BRK'
#*
#*     Replace    ++LOCALE++
#*                     Locale.
#*                     e.g. 'C'
#*
#*     Replace    ++TIMEZONE++
#*                     Time zone.
#*                     e.g. 'GMT0BST'
#*
#*     Replace    ++JAVA++
#*                     Java location.
#*                     e.g. '/usr/lpp/java/IBM/J1.4'
```

```
#*
#*    Replace    ++XMLTOOLKIT++
#*                        IBM XML Toolkit location.
#*                        e.g. '/usr/lpp/ixm/IBM/xml4c-5_5'
#*
#*    Replace    ++DB2CONVERSION++
#*                        Specifies the DB2 Converter.
#*                        e.g. 'SINGLE'
#*
#*    Replace    ++DB2DATABASE++
#*                        Needed by the mqsicreatebroker command to
#*                        determine the DB2 database.
#*                        e.g. 'DMQ01BRK'
#*
#*    Replace    ++DB2STORAGEGROUP++
#*                        Needed by the mqsicreatebroker command to
#*                        determine the DB2 storagegroup.
#*                        e.g. 'MQ01STOR'
#*
#********************************************************************
# 1. Component Settings
#********************************************************************
#
# 1.1 MQSI_REGISTRY references the component path. Also needed by
#     commands.
#     e.g. MQSI_REGISTRY=/mqsi/brokers/MQ01BRK
#
# 1.2 MQSI_COMPONENT_NAME is set to the broker name.
#     e.g. MQSI_COMPONENT_NAME=MQ01BRK
#
# 1.3 MQSI_FILEPATH needed by commands to reference the component
#     version install path.
#     e.g. MQSI_FILEPATH=usr/lpp/mqsi/V6R0M0
#
# 1.4 MQSI_LILPATH needed by components to find LILs
#     e.g. MQSI_FILEPATH=usr/lpp/mqsi/V6R0M0/lil
#
export MQSI_REGISTRY=++COMPONENTDIRECTORY++
export MQSI_COMPONENT_NAME=++COMPONENTNAME++
export MQSI_FILEPATH=++INSTALL++
export MQSI_LILPATH=$MQSI_FILEPATH/lil


#********************************************************************
# 2. NLS Settings
#********************************************************************
#
# 2.1 LANG and LC_ALL determine in which locale the component
#     will run.
#     e.g. LANG=Ja_JP.IBM-939 and LC_ALL=Ja_JP.IBM-939 for
#         japanese locale.
#         LANG=C, LC_ALL=C for US English locale.
#
# 2.2 TZ has the timezone setting in which you are located.
#     e.g. TZ=EST5                    for USA Eastern Standard Time
#         TZ=MEZ-1MES,M3.5.0,M10.5.0 for Central Europe
#         TZ=GMT0BST                  for the UK
#         Please refer to the IBM Manual
#         "Unix System Services Command Reference SC28-1892.
#
# 2.3 NLSPATH contains the location of the message catalog(s).
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 2.4 MQSI_CONSOLE_NLSPATH is used to locate the messages for
#     the console.
#     For Japanese or S-Chinese messages, change En_US to
#     Ja_JP or Zh_CN below. For English messages these can be
```

```
#      displayed in mixed or upper case only. (see MC_MESSAGES)
#      Note that MQSI_CONSOLE_NLSPATH does not use %L or %N
#
export LANG=++LOCALE++
export LC_ALL=++LOCALE++
export TZ=++TIMEZONE++
export NLSPATH=$MQSI_FILEPATH/messages/%L/%N
export NLSPATH=$NLSPATH:$MQSI_FILEPATH/nnsy/MIF/messages/%N
export MQSI_CONSOLE_NLSPATH=$MQSI_FILEPATH/messages/En_US


#********************************************************************
# 3. Automatic Restart Management (ARM) Settings
#********************************************************************
#
# 3.1 MQSI_USE_ARM specifies whether to use ARM.
#     e.g. MQSI_USE_ARM=YES for ARM enabled.
#          MQSI_USE_ARM=NO  for ARM not enabled.
#
# 3.2 MQSI_ARM_ELEMENTNAME required if ARM enabled.
#
# 3.3 MQSI_ARM_ELEMENTTYPE required if ARM enabled.
#
export MQSI_USE_ARM=NO
export MQSI_ARM_ELEMENTNAME=++COMPONENTNAME++
export MQSI_ARM_ELEMENTTYPE=


#********************************************************************
# 4. DB2 Settings
#********************************************************************
#
# 4.1 MQSI_DB2_ALWAYS_PREPARE
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 4.2 MQSI_DB2_CONVERSION specifies the DB2 Converter.
#     e.g. MQSI_DB2_CONVERSION=SINGLE
#                    WBIMB uses a SQL_EBCDIC_SCCSID to determine
#                    the DB2 converter.
#                    Note that this setting reflects the current
#                    WBIMB behavior.
#          MQSI_DB2_CONVERSION=MIXED
#                    WBIMB uses SQL_EBCDIC_MCCSID to determine
#                    the DB2 converter.
#                    Note that this setting requires that DB2 is
#                    configured to accept mixed byte data. This
#                    setting should be used when the customer
#                    wants data to be stored in the configured
#                    DB2 EBCDIC mixed byte code page.
#          MQSI_DB2_CONVERSION=LOCAL
#                    WBIMB uses localConverter identified by
#                    LC_ALL/LANG settings. This complies to what
#                    is done on distributed. This setting
#                    requires that WBIMB and DB2 are using the same
#                    codepage, otherwise only WBIMB can read DB2
#                    data correctly, it gets unreadable for other
#                    ( non-WBIMB ) applications that want to read
#                    the data.
#
# 4.3 MQSI_DB2_DATABASE needed by the mqsicreatebroker command to
#     determine the DB2 database.
#     e.g. MQSI_DB2_DATABASE=DMQ01BRK
#
# 4.4 MQSI_DB2_STORAGEGROUP needed by the mqsicreatebroker command
#     to determine the DB2 storagegroup.
#     e.g. MQSI_DB2_STORAGEGROUP=MQ01STOR
#
```

```
# 4.5 DSNAOINI references the component dsnaoini file.
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
export MQSI_DB2_ALWAYS_PREPARE=NO
export MQSI_DB2_CONVERSION=++DB2CONVERSION++
export MQSI_DB2_DATABASE=++DB2DATABASE++
export MQSI_DB2_STORAGEGROUP=++DB2STORAGEGROUP++
export DSNAOINI=//\'++COMPONENTDATASET++\(BIPDSNAO\)\'


#*******************************************************************
# 5. Java Settings
#*******************************************************************
#
# 5.1 JAVAHOME contains the root directory of the JAVA install.
#     e.g. JAVAHOME=/usr/lpp/java/IBM/J1.4
#     Note that the Java version must be at least 1.4.1
#
export JAVAHOME=++JAVA++



#*******************************************************************
# 6. DistHub Settings
#*******************************************************************
#
# 6.1 DISTHUB_PATH is the path to DistHub product executables
#     (NO NEED TO CHANGE DEFAULT UNLESS USING A NON-STANDARD
#      INSTALLATION PATH FOR DISTHUB!)
#
export DISTHUB_PATH=$MQSI_FILEPATH



#*******************************************************************
# 7. NNSY Settings
#*******************************************************************
#
# 7.1 NNSY_ROOT references NNSY executables.
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 7.2 NNSY_CATALOGUES references NNSY catalogues.
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 7.3 NN_CONFIG_FILE_PATH default location of nnsyreg.dat
#     configuration file.
#     Regardless of where NN_CONFIG_FILE_PATH points, the first
#     place that will be checked for the nnsyreg.dat files, is
#     $MQSI_FILEPATH/bin.  Any nnsyreg.dat file in this directory
#     will be used, even if another exists in the location pointed
#     to by the NN_CONFIG_FILE_PATH environment variable. It is
#     recommended that any instances of the nnsyreg.dat in the
#     $MQSI_FILEPATH/bin directory be removed, and placed in
#     environment specific locations, to preserve multi-user
#     capability.
#     e.g. NN_CONFIG_FILE_PATH=/usr/lpp/NNSY/bin
#
export NNSY_ROOT=$MQSI_FILEPATH/nnsy
export NNSY_CATALOGUES=$MQSI_FILEPATH/nnsy/NNSYCatalogues/en_US
export NN_CONFIG_FILE_PATH=



#*******************************************************************
# 8. WBI Broker Settings
#*******************************************************************
#
# 8.1 _BPX_BATCH_SPAWN
#     (MUST NOT CHANGE FROM DEFAULT!)
#
# 8.2 MQSI_MC_MESSAGES determines if messages should appear in
```

```
#      mixed case or upper case.
#      e.g. MQSI_MC_MESSAGES=YES for mixed case
#           MQSI_MC_MESSAGES=NO for upper case
#
# 8.3 MQSI_COMMAND_DATABASE_ECHO if defined, mqsi commands
#      display information when creating DB2 tables/indexes.
#
# 8.4 MQSI_COMMAND_ZOS_MQ_ECHO if defined, mqsi commands display
#      information returned from the MQ command server when
#      creating/deleting queues.
#
# 8.5 MQSI_COMMAND_ZOS_MQ_ECHO_RC if defined, mqsi commands display
#      reason and return codes from the MQ command server when
#      creating/deleting queues.
#
# 8.6 MQSI_DEPLOY_PROGRESS if defined, shows deployment progress
#      by the execution group
#
# 8.7 STEPLIB
#      (MUST NOT CHANGE FROM DEFAULT!)
#
export _BPX_BATCH_SPAWN=NO
export MQSI_MC_MESSAGES=NO
export MQSI_COMMAND_DATABASE_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO_RC=1
export MQSI_DEPLOY_PROGRESS=1
export STEPLIB=CURRENT


#******************************************************************
# 9. Other Settings
#******************************************************************
#
# NO NEED TO CHANGE FROM DEFAULT!
#
CP=$MQSI_FILEPATH/classes
CP=$CP:$MQSI_FILEPATH/classes/config.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxy.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxySamples.jar
CP=$CP:$MQSI_FILEPATH/classes/configutil.jar
CP=$CP:$JAVAHOME/lib
export CLASSPATH=$CP
export ICU_DATA=$MQSI_FILEPATH/nnsy/lib
XMLTOOLKIT=++XMLTOOLKIT++
LP=$MQSI_FILEPATH/lib/wbirf
LP=$LP:$MQSI_FILEPATH/lib/wbimb
LP=$LP:$MQSI_FILEPATH/lib/wbieb
LP=$LP:$MQSI_FILEPATH/lib
LP=$LP:$MQSI_FILEPATH/nnsy/lib
LP=$LP:$MQSI_FILEPATH/nnsy/MIF/lib
LP=$LP:$JAVAHOME/lib
LP=$LP:$JAVAHOME/bin
LP=$LP:$JAVAHOME/bin/classic
LP=$LP:$XMLTOOLKIT/lib
export LIBPATH=$LP
export PATH=$MQSI_FILEPATH/bin
export PATH=$PATH:$MQSI_FILEPATH/nnsy/bin
export PATH=$PATH:/bin
export PATH=$PATH:$JAVAHOME/bin
```

Sample BIPCBRK file

```
//BIPCRBK JOB
//******************************************************************
//*                                                                *
//* @START_COPYRIGHT@                                              *
//*                                                                *
```

```
//* Licensed Materials - Property of IBM;                          *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                         *
//* All Rights Reserved;                                           *
//* US Government Users Restricted Rights - use,                   *
//* duplication or disclosure restricted by GSA                   *
//* ADP Schedule Contract with IBM Corp.;                         *
//* See Copyright Instructions                                     *
//*                                                                *
//* @END_COPYRIGHT@                                                *
//*                                                                *
//********************************************************************
//*                IBM WebSphere Event/Message Brokers            *
//*                                                                *
//* Sample job to create a broker (mqsicreatebroker).             *
//*                                                                *
//********************************************************************
//* MORE INFORMATION - See:                                        *
//*                                                                *
//*      WebSphere Event/Message Brokers Information Centre.       *
//* Topic "an07080"                                                *
//*                                                                *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                *
//*    Replace    ++HOME++
//*                      Home directory where ENVFILE and STDERR
//*                      and STDOUT files will be created.
//*                      e.g. '/u/home'
//*
//*    Replace    ++INSTALL++
//*                      WBI Brokers installation directory.
//*                      e.g. '/usr/lpp/mqsi'
//*
//*    Replace    ++COMPONENTNAME++
//*                      Broker name.
//*                      e.g. 'MQ01BRK'
//*
//*    Replace    ++QUEUEMANAGER++
//*                      Queue manager name.
//*                      e.g. 'MQ01'
//*
//*    Replace    ++DB2TABLEOWNER++
//*                      DB2 table owner userid.
//*                      e.g. 'MQ01BRK'
//*
//*    Replace    ++DB2LOCATION++
//*                      DB2 location.
//*                      e.g. 'DSN810PK'
//*
//*    Replace    ++OPTIONS++
//*                      Options for mqsicreatebroker command.
//*                      e.g. '-1'
//*
//*                      z/OS specific options are
//*                          -1 Registry pass only
//*                             This creates the broker directory.
//*                          -2 MQ pass only
//*                             This creates the broker MQ queues.
//*                          -3 DB2 pass only
//*                             This creates the broker tables/indexes.
//*
//*                      Please see documentation for other options.
//*
//*    Replace    ++DB2HLQ++
//*                      DB2 high-level-qualifier.
//*                      e.g. 'SYS2.DB2.V810'
```

```
//*
//*     Replace    ++WMQHLQ++
//*                       WebSphere MQ high-level-qualifier.
//*                       e.g. 'MQM.V530'
//*
//******************************************************************
//*
//******************************************************************
//* Copy ENVFILE to SYSOUT
//******************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//******************************************************************
//* Run mqsicreatebroker command
//******************************************************************
//*
//BIPCRBK  EXEC PGM=IKJEFT01,REGION=0M
//*        DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//*        MQSeries Runtime Libraries
//         DD DISP=SHR,DSN=++WMQHLQ++.SCSQANLE
//         DD DISP=SHR,DSN=++WMQHLQ++.SCSQAUTH
//         DD DISP=SHR,DSN=++WMQHLQ++.SCSQLOAD
//STDENV   DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
BPXBATCH PGM -
++INSTALL++/bin/-
mqsicreatebroker -
++COMPONENTNAME++ -
-q ++QUEUEMANAGER++ -
-u ++DB2TABLEOWNER++ -
-n ++DB2LOCATION++ -
++OPTIONS++
/*
//
```

Sample BIPCPROF file

```
#******************************************************************
#*                                                               *
#* @START_COPYRIGHT@                                              *
#*                                                               *
#* Licensed Materials - Property of IBM;                         *
#* 5655-G97 (c) Copyright IBM Corp. 2004;                        *
#* All Rights Reserved;                                          *
#* US Government Users Restricted Rights - use,                  *
#* duplication or disclosure restricted by GSA                   *
#* ADP Schedule Contract with IBM Corp.;                         *
#* See Copyright Instructions                                    *
#*                                                               *
#* @END_COPYRIGHT@                                               *
#*                                                               *
#******************************************************************
#*              IBM WebSphere Event/Message Brokers              *
#*                                                               *
```

```
#* Sample profile for a configmgr.                                  *
#*                                                                  *
#******************************************************************
#* MORE INFORMATION - See:                                         *
#*                                                                  *
#*      WebSphere Event/Message Brokers Information Centre.         *
#*                                                                  *
#* IMPORTANT:                                                       *
#*                                                                  *
#*      You must submit BIPGEN each time you update this profile!   *
#*                                                                  *
#******************************************************************
#* CUSTOMIZE HERE FOR YOUR INSTALLATION
#* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
#*                                                                  *
#*     Replace    ++INSTALL++
#*                     WBI Brokers installation directory.
#*                     e.g. '/usr/lpp/mqsi'
#*
#*     Replace    ++COMPONENTDIRECTORY++
#*                     Configmgr directory.
#*                     e.g. '/mqsi/configmgr/MQ01CMGR'
#*
#*     Replace    ++COMPONENTNAME++
#*                     ConfigMgr Name.
#*                     e.g. 'MQ01CMGR'
#*
#*     Replace    ++LOCALE++
#*                     Locale.
#*                     e.g. 'C'
#*
#*     Replace    ++TIMEZONE++
#*                     Time zone.
#*                     e.g. 'GMT0BST'
#*
#*     Replace    ++JAVA++
#*                     Java location.
#*                     e.g. '/usr/lpp/java/IBM/J1.4'
#*
#*     Replace    ++XMLTOOLKIT++
#*                     IBM XML Toolkit location.
#*                     e.g. '/usr/lpp/ixm/IBM/xml4c-5_5'
#*
#*     Replace    ++MQPATH++
#*                     MQ location.
#*                     e.g. '/usr/lpp/mqm'
#*
#******************************************************************
# 1. Component Settings
#******************************************************************
#
# 1.1 MQSI_REGISTRY references the component path. Also needed by
#     commands.
#     e.g. MQSI_REGISTRY=/mqsi/configmgr/MQ01CMGR
#
# 1.2 MQSI_COMPONENT_NAME is set to the ConfigMgr name.
#     e.g. MQSI_COMPONENT_NAME=MQ01CMGR
#
# 1.3 MQSI_FILEPATH needed by commands to reference the component
#     version install path.
#     e.g. MQSI_FILEPATH=usr/lpp/mqsi/V6R0M0
#
export MQSI_REGISTRY=++COMPONENTDIRECTORY++
export MQSI_COMPONENT_NAME=++COMPONENTNAME++
export MQSI_FILEPATH=++INSTALL++
```

```
#********************************************************************
# 2. NLS Settings
#********************************************************************
#
# 2.1 LANG and LC_ALL determine in which locale the component
#     will run.
#     e.g. LANG=Ja_JP.IBM-939 and LC_ALL=Ja_JP.IBM-939 for
#          japanese locale.
#          LANG=C, LC_ALL=C for US English locale.
#
# 2.2 TZ has the timezone setting in which you are located.
#     e.g. TZ=EST5                     for USA Eastern Standard Time
#          TZ=MEZ-1MES,M3.5.0,M10.5.0 for Central Europe
#          TZ=GMT0BST                  for the UK
#          Please refer to the IBM Manual
#          "Unix System Services Command Reference SC28-1892.
#
# 2.3 NLSPATH contains the location of the message catalog(s).
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 2.4 MQSI_CONSOLE_NLSPATH is used to locate the messages for
#     the console.
#     For Japanese or S-Chinese messages, change En_US to
#     Ja_JP or Zh_CN below. For English messages these can be
#     displayed in mixed or upper case only. (see MC_MESSAGES)
#     Note that MQSI_CONSOLE_NLSPATH does not use %L or %N
#
export LANG=++LOCALE++
export LC_ALL=++LOCALE++
export TZ=++TIMEZONE++
export NLSPATH=$MQSI_FILEPATH/messages/%L/%N
export MQSI_CONSOLE_NLSPATH=$MQSI_FILEPATH/messages/En_US


#********************************************************************
# 3. Automatic Restart Management (ARM) Settings
#********************************************************************
#
# 3.1 MQSI_USE_ARM specifies whether to use ARM.
#     e.g. MQSI_USE_ARM=YES for ARM enabled.
#          MQSI_USE_ARM=NO  for ARM not enabled.
#
# 3.2 MQSI_ARM_ELEMENTNAME required if ARM enabled.
#
# 3.3 MQSI_ARM_ELEMENTTYPE required if ARM enabled.
#
export MQSI_USE_ARM=NO
export MQSI_ARM_ELEMENTNAME=++COMPONENTNAME++
export MQSI_ARM_ELEMENTTYPE=


#********************************************************************
# 4. Java Settings
#********************************************************************
#
# 4.1 JAVAHOME contains the root directory of the JAVA install.
#     e.g. JAVAHOME=/usr/lpp/java/IBM/J1.4
#     Note that the Java version must be at least 1.4.1
#
export JAVAHOME=++JAVA++


#********************************************************************
# 5. WBI Configmgr Settings
#********************************************************************
#
# 5.1 _BPX_BATCH_SPAWN
```

```
#     (MUST NOT CHANGE FROM DEFAULT!)
#
# 5.2 MQSI_MC_MESSAGES determines if messages should appear in
#     mixed case or upper case.
#     e.g. MQSI_MC_MESSAGES=YES for mixed case
#          MQSI_MC_MESSAGES=NO for upper case
#
# 5.3 MQSI_COMMAND_DATABASE_ECHO if defined, mqsi commands
#     display information when creating DB2 tables/indexes.
#
# 5.4 MQSI_COMMAND_ZOS_MQ_ECHO if defined, mqsi commands display
#     information returned from the MQ command server when
#     creating/deleting queues.
#
# 5.5 MQSI_COMMAND_ZOS_MQ_ECHO_RC if defined, mqsi commands display
#     reason and return codes from the MQ command server when
#     creating/deleting queues.
#
# 5.6 STEPLIB
#     (MUST NOT CHANGE FROM DEFAULT!)
#
export _BPX_BATCH_SPAWN=NO
export MQSI_MC_MESSAGES=NO
export MQSI_COMMAND_DATABASE_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO_RC=1
export STEPLIB=CURRENT


#*******************************************************************
# 6. Other Settings
#*******************************************************************
#
# NO NEED TO CHANGE FROM DEFAULT!
#
CP=++MQPATH++/java/lib/com.ibm.mq.jar
CP=$CP:++MQPATH++/java/lib/connector.jar
CP=$CP:$MQSI_FILEPATH/classes
CP=$CP:$JAVAHOME/lib
CP=$CP:$MQSI_FILEPATH/classes/config.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxy.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxySamples.jar
CP=$CP:$MQSI_FILEPATH/classes/configutil.jar
export CLASSPATH=$CP
XMLTOOLKIT=++XMLTOOLKIT++
LP=++MQPATH++/java/lib
LP=$LP:$MQSI_FILEPATH/lib/wbirf
LP=$LP:$MQSI_FILEPATH/lib/wbimb
LP=$LP:$MQSI_FILEPATH/lib/wbieb
LP=$LP:$MQSI_FILEPATH/lib
LP=$LP:$JAVAHOME/lib
LP=$LP:$JAVAHOME/bin
LP=$LP:$JAVAHOME/bin/classic
LP=$LP:$XMLTOOLKIT/lib
export LIBPATH=$LP
export PATH=$MQSI_FILEPATH/bin
export PATH=$PATH:$JAVAHOME/bin
```

Sample BIPCRCM file

```
//BIPCRCM JOB
//*****************************************************************
//*                                                              *
//* @START_COPYRIGHT@                                            *
//*                                                              *
//* Licensed Materials - Property of IBM;                       *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                      *
//* All Rights Reserved;                                        *
```

```
//* US Government Users Restricted Rights - use,                   *
//* duplication or disclosure restricted by GSA                    *
//* ADP Schedule Contract with IBM Corp.;                          *
//* See Copyright Instructions                                     *
//*                                                                *
//* @END_COPYRIGHT@                                                *
//*                                                                *
//********************************************************************
//*              IBM WebSphere Event/Message Brokers               *
//*                                                                *
//* Sample job to create a configmgr (mqsicreateconfigmgr).        *
//*                                                                *
//********************************************************************
//* MORE INFORMATION - See:                                        *
//*                                                                *
//*       WebSphere Event/Message Brokers Information Centre.       *
//* Topic "an23000"                                                *
//*                                                                *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                *
//*    Replace    ++HOME++
//*                      Home directory where ENVFILE and STDERR
//*                      and STDOUT files will be created.
//*                      e.g. '/u/home'
//*
//*    Replace    ++INSTALL++
//*                      WBI Brokers installation directory.
//*                      e.g. '/usr/lpp/mqsi'
//*
//*    Replace    ++COMPONENTNAME++
//*                      ConfigMgr Name.
//*                      e.g. 'MQ01CMGR'
//*
//*    Replace    ++QUEUEMANAGER++
//*                      Queue manager name.
//*                      e.g. 'MQ01'
//*
//*    Replace    ++OPTIONS++
//*                      Options for mqsicreateconfigmgr command.
//*                      e.g. '-1'
//*
//*                      z/OS specific options are
//*                          -1 Registry pass only
//*                             This creates the configmgr directory.
//*                          -2 MQ pass only
//*                             This creates the configmgr MQ queues.
//*
//*                      Please see documentation for other options.
//*
//*    Replace    ++LEHLQ++
//*                      LE high-level-qualifier.
//*                      e.g. 'PP.ADLE370.ZOS150'
//*
//*    Replace    ++WMQHLQ++
//*                      WebSphere MQ high-level-qualifier.
//*                      e.g. 'MQM.V530'
//*
//*    Replace    ++SYSHLQ++
//*                      System high-level-qualifier.
//*                      e.g. 'SYS1'
//*
//********************************************************************
//*
//********************************************************************
//* Copy ENVFILE to SYSOUT
```

```
//********************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//            PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* Run mqsicreateconfigmgr command
//********************************************************************
//*
//BIPCRCM  EXEC PGM=IKJEFT01,REGION=0M
//*          LE Runtime Library
//STEPLIB  DD DISP=SHR,DSN=++LEHLQ++.SCEERUN
//            DD DISP=SHR,DSN=++LEHLQ++.SCEERUN2
//*          MQSeries Runtime Libraries
//            DD DISP=SHR,DSN=++WMQHLQ++.SCSQANLE
//            DD DISP=SHR,DSN=++WMQHLQ++.SCSQAUTH
//            DD DISP=SHR,DSN=++WMQHLQ++.SCSQLOAD
//*          ISPF and concatenated USS Runtime Libraries
//ISPPLIB  DD DISP=SHR,DSN=++SYSHLQ++.SBPXPENU
//ISPSLIB  DD DISP=SHR,DSN=++SYSHLQ++.SISPSENU
//            DD DISP=SHR,DSN=++SYSHLQ++.SISPSLIB
//ISPTLIB  DD DISP=SHR,DSN=++SYSHLQ++.SISPTENU
//ISPMLIB  DD DISP=SHR,DSN=++SYSHLQ++.SBPXMENU
//            DD DISP=SHR,DSN=++SYSHLQ++.SISPMENU
//STDENV   DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//STDOUT   DD PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='++HOME++/bipcrcm.out'
//STDERR   DD PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='++HOME++/bipcrcm.err'
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
BPXBATCH PGM -
++INSTALL++/bin/-
mqsicreateconfigmgr -
++COMPONENTNAME++ -
-q ++QUEUEMANAGER++ -
++OPTIONS++
/*
//********************************************************************
//* Copy stdout to SYSOUT
//********************************************************************
//*
//COPYOUT  EXEC PGM=IKJEFT01,COND=EVEN,
//            PARM='OCOPY INDD(BIPFROM) OUTDD(STDOUT)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/bipcrcm.out'
//STDOUT   DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* Copy stderr to SYSOUT
//********************************************************************
//*
//COPYERR  EXEC PGM=IKJEFT01,COND=EVEN,
//            PARM='OCOPY INDD(BIPFROM) OUTDD(STDERR)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/bipcrcm.err'
```

```
//STDERR   DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//
```

Sample BIPCRDB file

```
//BIPCRDB JOB
//********************************************************************
//*                                                                *
//* @START_COPYRIGHT@                                              *
//*                                                                *
//* Licensed Materials - Property of IBM;                         *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                        *
//* All Rights Reserved;                                          *
//* US Government Users Restricted Rights - use,                  *
//* duplication or disclosure restricted by GSA                  *
//* ADP Schedule Contract with IBM Corp.;                        *
//* See Copyright Instructions                                   *
//*                                                                *
//* @END_COPYRIGHT@                                               *
//*                                                                *
//********************************************************************
//*            IBM WebSphere Event/Message Brokers                *
//*                                                                *
//* Create broker storagegroup / database and tablespaces.        *
//*                                                                *
//********************************************************************
//* MORE INFORMATION - See:                                        *
//*                                                                *
//*      WebSphere Event/Message Brokers Information Centre.       *
//*                                                                *
//*                                                                *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                *
//*    Replace    ++DB2HLQ++
//*                    DB2 high-level-qualifier.
//*                    e.g. 'SYS2.DB2.V810'
//*
//*    Replace    ++DB2RUNLIB++
//*                    DB2.
//*                    e.g. 'DSN810PK.RUNLIB.LOAD'
//*
//*    Replace    ++DB2SUBSYSTEM++
//*                    DB2 SSID.
//*                    e.g. 'DFK4'
//*
//*    Replace    ++DB2LOCATION++
//*                    DB2 location.
//*                    e.g. 'DSN810PK'
//*
//*    Replace    ++DB2CURRENTSQLID++
//*                    SET CURRENT SQLID user ID.
//*                    e.g. 'MQ01GRP'
//*
//*    Replace    ++DB2DATABASE++
//*                    DB2 database.
//*                    e.g. 'DMQ01BRK'
//*
//*    Replace    ++DB2STORAGEGROUP++
//*                    DB2 storagegroup.
//*                    e.g. 'MQ01STOR'
//*
//*    Replace    ++DB2BUFFERPOOL++
//*                    DB2 base table bufferpool.
//*                    e.g. 'BP0'
```

```
//*
//*     Replace    ++DB2INDEXBP++
//*                    DB2 index bufferpool.
//*                    e.g. 'BP0'
//*
//*     Replace    ++DB2LOBBP++
//*                    DB2 LOB table bufferpool.
//*                    e.g. 'BP0'
//*
//*     Replace    ++DB2SAMPLEPROGRAM++
//*                    DB2 sample program name.
//*                    e.g. 'DSNTEP2'
//*
//*     Replace    ++DB2SAMPLEPROGRAMPLAN++
//*                    DB2 sample program plan name.
//*                    e.g. 'DSNTEP81'
//*
//*******************************************************************
//*
//*******************************************************************
//* Create broker storagegroup
//*******************************************************************
//*
//STORGRP  EXEC PGM=IKJEFT01,REGION=0M
//*         DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//         DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(++DB2SUBSYSTEM++)
   RUN -
     PROGRAM(++DB2SAMPLEPROGRAM++) -
     PLAN(++DB2SAMPLEPROGRAMPLAN++)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
SET CURRENT SQLID='++DB2CURRENTSQLID++';
CREATE STOGROUP ++DB2STORAGEGROUP++
   VOLUMES('*')
   VCAT ++DB2LOCATION++;
/*
//*
//*******************************************************************
//* Create broker database
//*******************************************************************
//*
//DATABASE EXEC PGM=IKJEFT01,REGION=0M
//*         DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//         DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(++DB2SUBSYSTEM++)
   RUN -
     PROGRAM(++DB2SAMPLEPROGRAM++) -
     PLAN(++DB2SAMPLEPROGRAMPLAN++)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
SET CURRENT SQLID='++DB2CURRENTSQLID++';
CREATE DATABASE ++DB2DATABASE++
   STOGROUP ++DB2STORAGEGROUP++
   INDEXBP  ++DB2INDEXBP++;
```

```
/*
//*
//*****************************************************************
//* Create broker tablespaces
//*****************************************************************
//*
//TABLESPC EXEC PGM=IKJEFT01,REGION=0M
//*        DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//         DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//         DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(++DB2SUBSYSTEM++)
   RUN -
     PROGRAM(++DB2SAMPLEPROGRAM++) -
     PLAN(++DB2SAMPLEPROGRAMPLAN++)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN    DD *
SET CURRENT SQLID='++DB2CURRENTSQLID++';
CREATE TABLESPACE BSUBSCRI
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ROW
  CLOSE NO;

CREATE TABLESPACE BPUBLISH
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ANY
  CLOSE NO;

CREATE TABLESPACE BCLIENTU
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ANY
  CLOSE NO;

CREATE TABLESPACE BUSERCON
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
```

```
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BTOPOLOG
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY      7200
        SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BNBRCONN
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY      7200
        SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BRETAINE
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY      7200
        SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ROW
   CLOSE NO;

CREATE TABLESPACE BACLENTR
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY      7200
        SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BMQPSTOP
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY      7200
        SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
```

```
     SEGSIZE     32
     BUFFERPOOL ++DB2BUFFERPOOL++
     LOCKSIZE ANY
     CLOSE NO;

CREATE TABLESPACE BUSERNAM
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
          PRIQTY      7200
          SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BGROUPNA
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
          PRIQTY      7200
          SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BUSERMEM
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
          PRIQTY      7200
          SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BROKERAA
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
          PRIQTY      7200
          SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BROKERAE
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
          PRIQTY      7200
          SECQTY       720
   ERASE NO
   FREEPAGE      5
   PCTFREE      20
   SEGSIZE      32
   BUFFERPOOL ++DB2BUFFERPOOL++
```

```
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BROKERRE
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
         PRIQTY     7200
         SECQTY      720
   ERASE NO
   FREEPAGE     5
   PCTFREE     20
   SEGSIZE     32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BRMINFO
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
         PRIQTY     7200
         SECQTY      720
   ERASE NO
   FREEPAGE     5
   PCTFREE     20
   SEGSIZE     32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BRMRTDIN
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
         PRIQTY     7200
         SECQTY      720
   ERASE NO
   FREEPAGE     5
   PCTFREE     20
   SEGSIZE     32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BRMRTDDE
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
         PRIQTY     7200
         SECQTY      720
   ERASE NO
   FREEPAGE     5
   PCTFREE     20
   SEGSIZE     32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;

CREATE TABLESPACE BRMWFDIN
   IN ++DB2DATABASE++
   USING STOGROUP ++DB2STORAGEGROUP++
         PRIQTY     7200
         SECQTY      720
   ERASE NO
   FREEPAGE     5
   PCTFREE     20
   SEGSIZE     32
   BUFFERPOOL ++DB2BUFFERPOOL++
   LOCKSIZE ANY
   CLOSE NO;
```

```
CREATE TABLESPACE BRMPHYSI
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ANY
  CLOSE NO;

CREATE TABLESPACE BAGGREGA
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ROW
  CLOSE NO;

CREATE TABLESPACE BMULTICA
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     7200
        SECQTY      720
  ERASE NO
  FREEPAGE     5
  PCTFREE     20
  SEGSIZE     32
  BUFFERPOOL ++DB2BUFFERPOOL++
  LOCKSIZE ANY
  CLOSE NO;

CREATE LOB TABLESPACE BSUBLOB1
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     72000
        SECQTY      7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BSUBLOB2
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     72000
        SECQTY      7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BSUBLOB3
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY     72000
        SECQTY      7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BPUBLLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
```

```
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BCLIELOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BRETLOB1
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BRETLOB2
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BACLELOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BMQPSLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BMULTLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BROKALOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BROKRLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
```

```
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BRMRTLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
       PRIQTY    72000
       SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BRMPHLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
       PRIQTY    72000
       SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;

CREATE LOB TABLESPACE BAGGRLOB
  IN ++DB2DATABASE++
  USING STOGROUP ++DB2STORAGEGROUP++
        PRIQTY    72000
        SECQTY     7200
  BUFFERPOOL ++DB2LOBBP++
  LOCKSIZE   LOB;
/*
//
```

Sample BIPCRUN file

```
//BIPCRUN JOB
//********************************************************************
//*                                                                 *
//* @START_COPYRIGHT@                                               *
//*                                                                 *
//* Licensed Materials - Property of IBM;                          *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                         *
//* All Rights Reserved;                                            *
//* US Government Users Restricted Rights - use,                   *
//* duplication or disclosure restricted by GSA                    *
//* ADP Schedule Contract with IBM Corp.;                          *
//* See Copyright Instructions                                      *
//*                                                                 *
//* @END_COPYRIGHT@                                                 *
//*                                                                 *
//********************************************************************
//*           IBM WebSphere Event/Message Brokers                  *
//*                                                                 *
//* Sample job to create a usernameserver                          *
//*                                  (mqsicreateusernameserver).  *
//*                                                                 *
//********************************************************************
//* MORE INFORMATION - See:                                        *
//*                                                                 *
//*       WebSphere Event/Message Brokers Information Centre.      *
//* Topic "an07200"                                                *
//*                                                                 *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                 *
//*    Replace   ++HOME++
//*                    Home directory where ENVFILE and STDERR
//*                    and STDOUT files will be created.
//*                    e.g. '/u/home'
//*
//*    Replace   ++INSTALL++
//*                    WBI Brokers installation directory.
```

```
//*                       e.g. '/usr/lpp/mqsi '
//*
//*    Replace    ++QUEUEMANAGER++
//*                       Queue manager name.
//*                       e.g. 'MQ01'
//*
//*    Replace    ++OPTIONS++
//*                       Options for mqsicreateusernameserver.
//*                       e.g. '-1'
//*
//*                       z/OS specific options are
//*                           -1 Registry pass only
//*                              This creates the uns directory.
//*                           -2 MQ pass only
//*                              This creates the uns MQ queues.
//*
//*                       Please see documentation for other options.
//*
//*    Replace    ++WMQHLQ++
//*                       WebSphere MQ high-level-qualifier.
//*                       e.g. 'MQM.V530'
//*
//*******************************************************************
//*
//*******************************************************************
//* Copy ENVFILE to SYSOUT
//*******************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//*******************************************************************
//* Run mqsicreateusernameserver command
//*******************************************************************
//*
//BIPCRUN  EXEC PGM=IKJEFT01,REGION=0M
//*        MQSeries Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=++WMQHLQ++.SCSQANLE
//         DD DISP=SHR,DSN=++WMQHLQ++.SCSQAUTH
//         DD DISP=SHR,DSN=++WMQHLQ++.SCSQLOAD
//STDENV   DD PATHOPTS=(ORDONLY),
//            PATH='++HOME++/ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
BPXBATCH PGM -
++INSTALL++/bin/-
mqsicreateusernameserver -
-q ++QUEUEMANAGER++ -
++OPTIONS++
/*
//
```

Sample BIPDSNAO file

```
;*******************************************************************
;*                                                                *
;* @START_COPYRIGHT@                                              *
;*                                                                *
;* Licensed Materials - Property of IBM;                          *
;* 5655-G97 (c) Copyright IBM Corp. 2004;                         *
;* All Rights Reserved;                                           *
```

```
;* US Government Users Restricted Rights - use,            *
;* duplication or disclosure restricted by GSA            *
;* ADP Schedule Contract with IBM Corp.;                  *
;* See Copyright Instructions                             *
;*                                                        *
;* @END_COPYRIGHT@                                        *
;*                                                        *
;*********************************************************
;*              IBM WebSphere Event/Message Brokers       *
;*                                                        *
;* Sample dsnaoini for a broker, usernamerserver or configmgr.  *
;*                                                        *
;*********************************************************
;* MORE INFORMATION - See:                                *
;*                                                        *
;*      WebSphere Event/Message Brokers Information Centre.    *
;*                                                        *
;*********************************************************
;* CUSTOMIZE HERE FOR YOUR INSTALLATION
;* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
;*                                                        *
;*    Replace    ++COMPONENTDIRECTORY++
;*                    Component directory.
;*                    e.g. '/mqsi/brokers/MQ01BRK'
;*
;*    Replace    ++DB2SUBSYSTEM++
;*                    DB2 SSID.
;*                    e.g. 'DFK4'
;*
;*    Replace    ++DB2LOCATION++
;*                    DB2 location.
;*                    e.g. 'DSN810PK'
;*
;*    Replace    ++DB2CURRENTSQLID++
;*                    CURRENT SQLID user ID.
;*                    e.g. 'MQ01GRP'
;*
;*    Replace    ++DB2DSNACLIPLAN++
;*                    DB2 plan name for dsnacli.
;*                    e.g. 'DSNACLI'
;*
;*********************************************************
[COMMON]
APPLTRACE=0
APPLTRACEfilename=++COMPONENTDIRECTORY++/output/traceodbc
CONNECTTYPE=2
DIAGTRACE=0
DIAGTRACE_NO_WRAP=0
MAXCONN=0
MULTICONTEXT=0
MVSDEFAULTSSID=++DB2SUBSYSTEM++

; SUBSYSTEM
[++DB2SUBSYSTEM++]
MVSATTACHTYPE=RRSAF
PLANNAME=++DB2DSNACLIPLAN++

; DATASOURCES
[++DB2LOCATION++]
CURRENTSQLID=++DB2CURRENTSQLID++
```

Sample BIPGEN file

```
//BIPGEN JOB
//*********************************************************
//*                                                        *
//* @START_COPYRIGHT@                                       *
//*                                                        *
```

```
//* Licensed Materials - Property of IBM;                          *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                         *
//* All Rights Reserved;                                           *
//* US Government Users Restricted Rights - use,                   *
//* duplication or disclosure restricted by GSA                    *
//* ADP Schedule Contract with IBM Corp.;                          *
//* See Copyright Instructions                                     *
//*                                                                *
//* @END_COPYRIGHT@                                                *
//*                                                                *
//*********************************************************************
//*             IBM WebSphere Event/Message Brokers                 *
//*                                                                *
//* Copy component profile to the file system and generate an       *
//* ENVFILE.                                                        *
//*                                                                *
//* IMPORTANT:                                                      *
//*                                                                *
//*     You must submit BIPGEN each time you update a profile!      *
//*                                                                *
//*********************************************************************
//* MORE INFORMATION - See:                                         *
//*                                                                *
//*      WebSphere Event/Message Brokers Information Centre.         *
//*                                                                *
//*********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                *
//*     Replace    ++HOME++
//*                     Home directory where ENVFILE and STDERR
//*                     and STDOUT files will be created.
//*                     e.g. '/u/home'
//*
//*     Replace    ++COMPONENTDATASET++
//*                     Component dataset.
//*                     e.g. 'BIP.BROKER.MQ01BRK'
//*
//*     Replace    ++COMPONENTPROFILE++
//*                     Component profile name.
//*                     e.g. Broker        = 'BIPBPROF'
//*                          Usernameserver = 'BIPUPROF'
//*                          Configmgr      = 'BIPCPROF'
//*
//*********************************************************************
//*
//*********************************************************************
//* Copy BIPBPROF/BIPUPROF/BIPCPROF to file system
//*********************************************************************
//*
//COPYPROF EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(BIPPROF)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD DISP=SHR,DSN=++COMPONENTDATASET++(++COMPONENTPROFILE++)
//BIPPROF  DD PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='++HOME++/bipprof'
//SYSTSIN  DD DUMMY
//*
//*********************************************************************
//* Copy BIPPROF to SYSOUT
//*********************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(BIPPROF)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
```

```
//              PATH='++HOME++/bipprof'
//BIPPROF  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* Create ENVFILE from BIPPROF
//********************************************************************
//*
//CREATENV EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
BPXBATCH SH -
. ++HOME++/bipprof; -
/bin/printenv > -
++HOME++/ENVFILE
//*
//********************************************************************
//* Copy ENVFILE to SYSOUT
//********************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//              PATH='++HOME++/ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//
```

Sample BIPUPROF file

```
#********************************************************************
#*                                                                 *
#* @START_COPYRIGHT@                                               *
#*                                                                 *
#* Licensed Materials - Property of IBM;                           *
#* 5655-G97 (c) Copyright IBM Corp. 2004;                          *
#* All Rights Reserved;                                            *
#* US Government Users Restricted Rights - use,                    *
#* duplication or disclosure restricted by GSA                    *
#* ADP Schedule Contract with IBM Corp.;                          *
#* See Copyright Instructions                                      *
#*                                                                 *
#* @END_COPYRIGHT@                                                 *
#*                                                                 *
#********************************************************************
#*           IBM WebSphere Event/Message Brokers                   *
#*                                                                 *
#* Sample profile for a usernameserver.                           *
#*                                                                 *
#********************************************************************
#* MORE INFORMATION - See:                                         *
#*                                                                 *
#*     WebSphere Event/Message Brokers Information Centre.         *
#*                                                                 *
#* IMPORTANT:                                                      *
#*                                                                 *
#*     You must submit BIPGEN each time you update this profile!  *
#*                                                                 *
#********************************************************************
#* CUSTOMIZE HERE FOR YOUR INSTALLATION
#* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
#*                                                                 *
#*    Replace   ++INSTALL++
#*                  WBI Brokers installation directory.
#*                  e.g. '/usr/lpp/mqsi'
#*
```

```
#*     Replace   ++COMPONENTDIRECTORY++
#*                    Usernameserver directory.
#*                    e.g. '/mqsi/uns/MQ01BRK'
#*
#*     Replace   ++LOCALE++
#*                    Locale.
#*                    e.g. 'C'
#*
#*     Replace   ++TIMEZONE++
#*                    Time zone.
#*                    e.g. 'GMT0BST'
#*
#*     Replace   ++JAVA++
#*                    Java location.
#*                    e.g. '/usr/lpp/java/IBM/J1.4'
#*
#*     Replace   ++XMLTOOLKIT++
#*                    IBM XML Toolkit location.
#*                    e.g. '/usr/lpp/ixm/IBM/xml4c-5_5'
#*
#********************************************************************
# 1. Component Settings
#********************************************************************
#
# 1.1 MQSI_REGISTRY references the component path. Also needed by
#     commands.
#     e.g. MQSI_REGISTRY=/mqsi/uns/MQ01BRK
#
# 1.2 MQSI_COMPONENT_NAME is set to the UserNameServer name.
#     (DO NOT CHANGE FROM DEFAULT!)
#
# 1.3 MQSI_FILEPATH needed by commands to reference the component
#     version install path.
#     e.g. MQSI_FILEPATH=usr/lpp/mqsi/V6R0M0
#
export MQSI_REGISTRY=++COMPONENTDIRECTORY++
export MQSI_COMPONENT_NAME=UserNameServer
export MQSI_FILEPATH=++INSTALL++


#********************************************************************
# 2. NLS Settings
#********************************************************************
#
# 2.1 LANG and LC_ALL determine in which locale the component
#     will run.
#     e.g. LANG=Ja_JP.IBM-939 and LC_ALL=Ja_JP.IBM-939 for
#          japanese locale.
#          LANG=C, LC_ALL=C for US English locale.
#
# 2.2 TZ has the timezone setting in which you are located.
#     e.g. TZ=EST5                    for USA Eastern Standard Time
#          TZ=MEZ-1MES,M3.5.0,M10.5.0 for Central Europe
#          TZ=GMT0BST                 for the UK
#          Please refer to the IBM Manual
#          "Unix System Services Command Reference SC28-1892.
#
# 2.3 NLSPATH contains the location of the message catalog(s).
#     (NO NEED TO CHANGE FROM DEFAULT!)
#
# 2.4 MQSI_CONSOLE_NLSPATH is used to locate the messages for
#     the console.
#     For Japanese or S-Chinese messages, change En_US to
#     Ja_JP or Zh_CN below. For English messages these can be
#     displayed in mixed or upper case only. (see MC_MESSAGES)
#     Note that MQSI_CONSOLE_NLSPATH does not use %L or %N
#
```

```
export LANG=++LOCALE++
export LC_ALL=++LOCALE++
export TZ=++TIMEZONE++
export NLSPATH=$MQSI_FILEPATH/messages/%L/%N
export MQSI_CONSOLE_NLSPATH=$MQSI_FILEPATH/messages/En_US


#*********************************************************************
# 3. Automatic Restart Management (ARM) Settings
#*********************************************************************
#
# 3.1 MQSI_USE_ARM specifies whether to use ARM.
#     e.g. MQSI_USE_ARM=YES for ARM enabled.
#          MQSI_USE_ARM=NO  for ARM not enabled.
#
# 3.2 MQSI_ARM_ELEMENTNAME required if ARM enabled.
#
# 3.3 MQSI_ARM_ELEMENTTYPE required if ARM enabled.
#
export MQSI_USE_ARM=NO
export MQSI_ARM_ELEMENTNAME=
export MQSI_ARM_ELEMENTTYPE=


#*********************************************************************
# 4. Java Settings
#*********************************************************************
#
# 4.1 JAVAHOME contains the root directory of the JAVA install.
#     e.g. JAVAHOME=/usr/lpp/java/IBM/J1.4
#     Note that the Java version must be at least 1.4.1
#
export JAVAHOME=++JAVA++


#*********************************************************************
# 5. WBI Usernameserver Settings
#*********************************************************************
#
# 5.1 _BPX_BATCH_SPAWN
#     (MUST NOT CHANGE FROM DEFAULT!)
#
# 5.2 MQSI_MC_MESSAGES determines if messages should appear in
#     mixed case or upper case.
#     e.g. MQSI_MC_MESSAGES=YES for mixed case
#          MQSI_MC_MESSAGES=NO for upper case
#
# 5.3 MQSI_COMMAND_DATABASE_ECHO if defined, mqsi commands
#     display information when creating DB2 tables/indexes.
#
# 5.4 MQSI_COMMAND_ZOS_MQ_ECHO if defined, mqsi commands display
#     information returned from the MQ command server when
#     creating/deleting queues.
#
# 5.5 MQSI_COMMAND_ZOS_MQ_ECHO_RC if defined, mqsi commands display
#     reason and return codes from the MQ command server when
#     creating/deleting queues.
#
# 5.6 STEPLIB
#     (MUST NOT CHANGE FROM DEFAULT!)
#
export _BPX_BATCH_SPAWN=NO
export MQSI_MC_MESSAGES=NO
export MQSI_COMMAND_DATABASE_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO=1
export MQSI_COMMAND_ZOS_MQ_ECHO_RC=1
export STEPLIB=CURRENT
```

```
#*******************************************************************
# 6. Other Settings
#*******************************************************************
#
# NO NEED TO CHANGE FROM DEFAULT!
#
CP=$MQSI_FILEPATH/classes
CP=$CP:$MQSI_FILEPATH/classes/config.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxy.jar
CP=$CP:$MQSI_FILEPATH/classes/ConfigManagerProxySamples.jar
CP=$CP:$MQSI_FILEPATH/classes/configutil.jar
CP=$CP:$JAVAHOME/lib
export CLASSPATH=$CP
XMLTOOLKIT=++XMLTOOLKIT++
LP=$MQSI_FILEPATH/lib/wbirf
LP=$LP:$MQSI_FILEPATH/lib/wbimb
LP=$LP:$MQSI_FILEPATH/lib/wbieb
LP=$LP:$MQSI_FILEPATH/lib
LP=$LP:$JAVAHOME/lib
LP=$LP:$JAVAHOME/bin
LP=$LP:$JAVAHOME/bin/classic
LP=$LP:$XMLTOOLKIT/lib
export LIBPATH=$LP
export PATH=$MQSI_FILEPATH/bin
export PATH=$PATH:$JAVAHOME/bin
```

Sample BIPBRKP file

```
//*******************************************************************
//*                                                                 *
//* @START_COPYRIGHT@                                               *
//*                                                                 *
//* Licensed Materials - Property of IBM;                           *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                          *
//* All Rights Reserved;                                            *
//* US Government Users Restricted Rights - use,                    *
//* duplication or disclosure restricted by GSA                     *
//* ADP Schedule Contract with IBM Corp.;                           *
//* See Copyright Instructions                                      *
//*                                                                 *
//* @END_COPYRIGHT@                                                 *
//*                                                                 *
//*******************************************************************
//*             IBM WebSphere Event/Message Brokers                 *
//*                                                                 *
//* Sample job to start a broker.                                   *
//*                                                                 *
//*******************************************************************
//* MORE INFORMATION - See:                                         *
//*                                                                 *
//*      WebSphere Event/Message Brokers Information Centre.         *
//*                                                                 *
//*******************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                 *
//*    Replace    ++HOME++
//*                    Home directory where ENVFILE and STDERR
//*                    and STDOUT files will be created.
//*                    e.g. '/u/home'
//*
//*    Replace    ++INSTALL++
//*                    WBI Brokers installation directory.
//*                    e.g. '/usr/lpp/mqsi'
//*
//*    Replace    ++QUEUEMANAGER++
```

```
//*                    Queue manager name.
//*                    e.g. 'MQ01'
//*
//*    Replace    ++COMPONENTDIRECTORY++
//*                    Broker directory.
//*                    e.g. '/mqsi/brokers/MQ01BRK'
//*
//*    Replace    ++STARTEDTASKNAME++
//*                    Started Task Name (max 8 chars uppercase).
//*                    e.g. 'MQ01BRK'
//*
//*    Replace    ++COMPONENTDATASET++
//*                    Broker dataset.
//*                    e.g. 'BIP.BROKER.MQ01BRK'
//*
//*    Replace    ++DB2HLQ++
//*                    DB2 high-level-qualifier.
//*                    e.g. 'SYS2.DB2.V810'
//*
//*    Replace    ++DB2RUNLIB++
//*                    DB2.
//*                    e.g. 'DSN810PK.RUNLIB.LOAD'
//*
//*    Replace    ++WMQHLQ++
//*                    WebSphere MQ high-level-qualifier.
//*                    e.g. 'MQM.V530'
//*
//******************************************************************
//*
//* Following variables are changed when starting a DataFlowEngine:
//*     STDOUT and STDERR files (append for DataFlowEngine)
//*        STDD=OAPPEND (default is OTRUNC)
//*     Semaphore ID
//*        SE=<Semaphore ID> (default is '')
//*     Shared Memory Segment ID
//*        SH=<Shared Memory Segment ID> (default is '')
//*     Component Unique Name
//*        COMPK='' (default is ++STARTEDTASKNAME++)
//*     Start Parameter
//*        STRTP='' (default is 'AUTO')
//*
//******************************************************************
//*
//++STARTEDTASKNAME++ PROC INSTP='++INSTALL++',
//         MAINP='bipimain',
//         SRVMP='bipservice',
//         COMPK='++STARTEDTASKNAME++',
//         COMDS='++COMPONENTDATASET++',
//         STRTP='AUTO',
//         COMPDIR='++COMPONENTDIRECTORY++',
//         STDD='OTRUNC',
//         SE='',
//         SH='',
//         HOME='++HOME++',
//         DB2HLQ='++DB2HLQ++',
//         WMQHLQ='++WMQHLQ++',
//         SYSHLQ='++SYSHLQ++'
//*
//******************************************************************
//* Copy ENVFILE to SYSOUT
//******************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//             PATH='&HOME./ENVFILE'
```

```
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* Copy DSNAOINI to SYSOUT
//********************************************************************
//*
//COPYDSN  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(DSNAOINI)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD DISP=SHR,DSN=&COMDS.(BIPDSNAO)
//DSNAOINI DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* Test to see if starting a DataFlowEngine address space.
//* Should return RC=0 if starting a Control address space or
//* UserNameServer address space, RC=12 if starting a DataFlowEngine
//* address space.
//********************************************************************
//*
//CHECKDFE EXEC PGM=IKJEFT01,
//         PARM='LISTDS ''&COMDS.&SE.'''
//SYSTSIN  DD DUMMY
//SYSTSPRT DD DUMMY
//SYSMDUMP DD SYSOUT=*
//*
//         IF (RC=0) THEN
//*
//********************************************************************
//* Broker DB2 and MQ verification
//********************************************************************
//*
//VFYDB2MQ EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/bipcvp b ++QUEUEMANAGER++'
//*        DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&DB2HLQ..SDSNEXIT
//         DD DISP=SHR,DSN=&DB2HLQ..SDSNLOAD
//         DD DISP=SHR,DSN=&DB2HLQ..SDSNLOD2
//         DD DISP=SHR,DSN=++DB2RUNLIB++
//*        MQSeries Runtime Libraries
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//         ENDIF
//*
//********************************************************************
//* Check RCs from previous steps
//********************************************************************
//*
//         IF (CHECKDFE.RC NE 0) OR
//            (VFYDB2MQ.RC EQ 0) THEN
//*
//********************************************************************
//* Step to delete residual locks
//* (this is only needed if the broker is ARM enabled)
//********************************************************************
//*
//*RMLOCKS  EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//*         PARM='SH rm -f &COMPDIR./common/locks/*'
//*
```

```
//********************************************************************
//* Start either :
//* Control Address Space (bipimain, bipservice and bipbroker)
//* DataFlowEngine Address Space (bipimain and DataFlowEngine)
//********************************************************************
//*
//BROKER   EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/&MAINP. &SRVMP. &SE. &SH. &COMPK.
//              &STRTP.'
//*       DB2 Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&DB2HLQ..SDSNEXIT
//         DD DISP=SHR,DSN=&DB2HLQ..SDSNLOAD
//         DD DISP=SHR,DSN=&DB2HLQ..SDSNLOD2
//         DD DISP=SHR,DSN=++DB2RUNLIB++
//*       MQSeries Runtime Libraries
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//         ENDIF
//*
//*----------------------------------------------------------------
//         PEND
//*----------------------------------------------------------------
//*
//
```

Sample BIPCMGRP file

```
//********************************************************************
//*                                                                  *
//* @START_COPYRIGHT@                                                *
//*                                                                  *
//* Licensed Materials - Property of IBM;                            *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                           *
//* All Rights Reserved;                                             *
//* US Government Users Restricted Rights - use,                     *
//* duplication or disclosure restricted by GSA                      *
//* ADP Schedule Contract with IBM Corp.;                            *
//* See Copyright Instructions                                       *
//*                                                                  *
//* @END_COPYRIGHT@                                                  *
//*                                                                  *
//********************************************************************
//*            IBM WebSphere Event/Message Brokers                   *
//*                                                                  *
//* Sample job to start a configmgr.                                 *
//*                                                                  *
//********************************************************************
//* MORE INFORMATION - See:                                          *
//*                                                                  *
//*     WebSphere Event/Message Brokers Information Centre.           *
//*                                                                  *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                  *
//*    Replace   ++HOME++
//*                    Home directory where ENVFILE and STDERR
//*                    and STDOUT files will be created.
//*                    e.g. '/u/home'
//*
//*    Replace   ++INSTALL++
```

```
//*                  WBI Brokers installation directory.
//*                  e.g. '/usr/lpp/mqsi'
//*
//*    Replace    ++QUEUEMANAGER++
//*                  Queue manager name.
//*                  e.g. 'MQ01'
//*
//*    Replace    ++COMPONENTDIRECTORY++
//*                  ConfigMgr directory.
//*                  e.g. '/mqsi/configmgr/MQ01BRK'
//*
//*    Replace    ++STARTEDTASKNAME++
//*                  Started Task Name (max 8 chars uppercase).
//*                  e.g. 'MQ01CMGR'
//*
//*    Replace    ++WMQHLQ++
//*                  WebSphere MQ high-level-qualifier.
//*                  e.g. 'MQM.V530'
//*
//********************************************************************
//*
//++STARTEDTASKNAME++ PROC INSTP='++INSTALL++',
//         MAINP='bipimain',
//         SRVMP='bipservice',
//         COMPK='++STARTEDTASKNAME++',
//         STRTP='AUTO',
//         COMPDIR='++COMPONENTDIRECTORY++',
//         STDD='OTRUNC',
//         HOME='++HOME++',
//         WMQHLQ='++WMQHLQ++',
//         SYSHLQ='++SYSHLQ++'
//*
//********************************************************************
//* Copy ENVFILE to SYSOUT
//********************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='&HOME./ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* ConfigMgr MQ verification
//********************************************************************
//*
//VFYMQ    EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/bipcvp c ++QUEUEMANAGER++'
//*       MQSeries Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//********************************************************************
//* Check RC from previous steps
//********************************************************************
//*
//         IF (RC=0) THEN
//*
//********************************************************************
```

```
//* Step to delete residual locks
//* (this is only needed if the configmgr is ARM enabled)
//********************************************************************
//*
//*RMLOCKS  EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//*         PARM='SH rm -f &COMPDIR./common/locks/*'
//*
//********************************************************************
//* Start :
//* ConfigMgr Address Space (bipimain, bipservice and bipconfigmgr)
//********************************************************************
//*
//CMGR     EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/&MAINP. &SRVMP. &COMPK. &STRTP.'
//*        MQSeries Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//         ENDIF
//*
//*----------------------------------------------------------------
//         PEND
//*----------------------------------------------------------------
//*
//
```

Sample BIPUNSP file

```
//********************************************************************
//*                                                                 *
//* @START_COPYRIGHT@                                                *
//*                                                                 *
//* Licensed Materials - Property of IBM;                           *
//* 5655-G97 (c) Copyright IBM Corp. 2004;                          *
//* All Rights Reserved;                                            *
//* US Government Users Restricted Rights - use,                    *
//* duplication or disclosure restricted by GSA                     *
//* ADP Schedule Contract with IBM Corp.;                           *
//* See Copyright Instructions                                      *
//*                                                                 *
//* @END_COPYRIGHT@                                                  *
//*                                                                 *
//********************************************************************
//*          IBM WebSphere Event/Message Brokers                    *
//*                                                                 *
//* Sample job to start a usernameserver.                           *
//*                                                                 *
//********************************************************************
//* MORE INFORMATION - See:                                         *
//*                                                                 *
//*      WebSphere Event/Message Brokers Information Centre.         *
//*                                                                 *
//********************************************************************
//* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
//* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
//*                                                                 *
//*    Replace   ++HOME++
//*                   Home directory where ENVFILE and STDERR
//*                   and STDOUT files will be created.
//*                   e.g. '/u/home'
//*
//*    Replace   ++INSTALL++
```

```
//*                    WBI Brokers installation directory.
//*                    e.g. '/usr/lpp/mqsi'
//*
//*    Replace   ++QUEUEMANAGER++
//*                    Queue manager name.
//*                    e.g. 'MQ01'
//*
//*    Replace   ++COMPONENTDIRECTORY++
//*                    Usernameserver directory.
//*                    e.g. '/mqsi/brokers/MQ01BRK'
//*
//*    Replace   ++STARTEDTASKNAME++
//*                    Started Task Name (max 8 chars uppercase).
//*                    e.g. 'MQ01BRK'
//*
//*    Replace   ++WMQHLQ++
//*                    WebSphere MQ high-level-qualifier.
//*                    e.g. 'MQM.V530'
//*
//********************************************************************
//*
//++STARTEDTASKNAME++ PROC INSTP='++INSTALL++',
//         MAINP='bipimain',
//         SRVMP='bipservice',
//         COMPK='++STARTEDTASKNAME++',
//         STRTP='AUTO',
//         COMPDIR='++COMPONENTDIRECTORY++',
//         STDD='OTRUNC',
//         HOME='++HOME++',
//         WMQHLQ='++WMQHLQ++',
//         SYSHLQ='++SYSHLQ++'
//*
//********************************************************************
//* Copy ENVFILE to SYSOUT
//********************************************************************
//*
//COPYENV  EXEC PGM=IKJEFT01,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(ENVFILE)'
//SYSTSPRT DD DUMMY
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH=' &HOME./ENVFILE'
//ENVFILE  DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//********************************************************************
//* UserNameServer MQ verification
//********************************************************************
//*
//VFYDB2MQ EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/bipcvp u ++QUEUEMANAGER++'
//*         MQSeries Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//         DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//********************************************************************
//* Check RC from previous steps
//********************************************************************
//*
//         IF (RC=0) THEN
//*
//********************************************************************
```

```
//* Step to delete residual locks
//* (this is only needed if the broker is ARM enabled)
//*****************************************************************
//*
//*RMLOCKS  EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//*          PARM='SH rm -f &COMPDIR./common/locks/*'
//*
//*****************************************************************
//* Start :
//* UserNameServer Address Space (bipimain, bipservice and bipuns)
//*****************************************************************
//*
//UNS     EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
//          PARM='PGM &INSTP./bin/&MAINP. &SRVMP. &COMPK. &STRTP.'
//*        MQSeries Runtime Libraries
//STEPLIB  DD DISP=SHR,DSN=&WMQHLQ..SCSQANLE
//          DD DISP=SHR,DSN=&WMQHLQ..SCSQAUTH
//          DD DISP=SHR,DSN=&WMQHLQ..SCSQLOAD
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//*
//SYSMDUMP DD SYSOUT=*
//*
//        ENDIF
//*
//*-------------------------------------------------------------
//       PEND
//*-------------------------------------------------------------
//*
//
```

z/OS utility jobs

Purpose

Invoke these utility jobs on WebSphere Event Broker for z/OS, by running JCL jobs, which are in the component PDSE.

The following table summarizes the descriptions and JCL names of these utilities, together with the names of the equivalent command on Windows platforms, Linux, and UNIX systems where appropriate.

JCL name	Description	Equivalent command on distributed platforms
BIPALDB	Commented sample job to identify database resources, including tables and table spaces for performance tuning purposes	-
BIPALMQ	Commented sample job to identify database WebSphere MQ resources, including queues for performance tuning purposes	-
BIPBUCM	Existing command ported to z/OS.	"mqsibackupconfigmgr command" on page 220
BIPCHBK	Existing command ported to z/OS.	"mqsichangebroker command" on page 221
BIPCHMS	New JCL for existing z/OS command.	"mqsichangeflowstats command" on page 232
BIPCHPR	Changes broker property settings	"mqsichangeproperties command" on page 236

BIPCHUN	Existing command ported to z/OS.	"mqsichangeusernameserver command" on page 250
BIPCLMP	Removes anMQSeries Publish/Subscribe broker as a neighbor.	"mqsiclearmqpubsub command" on page 253
BIPCRBK	Existing command ported to z/OS. Command modified to support multiple passes.	"mqsicreatebroker command" on page 257
BIPCREG	Adds an execution group to an existing broker definition.	"mqsicreateexecutiongroup command" on page 272
BIPCRUN	Existing command ported to z/OS. Command modified to support multiple passes.	"mqsicreateusernameserver command" on page 275
BIPDLBK	Existing command ported to z/OS. Command modified to support multiple passes.	"mqsideletebroker command" on page 281
BIPDLUN	Existing command ported to z/OS. Command modified to support multiple passes.	"mqsideleteusernameserver command" on page 288
BIPDREG	Deletes an execution group.	"mqsideleteexecutiongroup command" on page 286
BIPEDIT	REXX program used for customizing JCL files.	
BIPFMLG	Reads the WebSphere Event Broker trace and formats it to a text file.	"mqsiformatlog command" on page 293
BIPJNMP	Lists Publish/Subscribe neighbor broker status.	"mqsijoinmqpubsub command" on page 295
BIPLIST	New JCL for existing z/OS command.	"mqsilist (list resources) command" on page 296
BIPLSMP	Lists Publish/Subscribe neighbor broker status.	"mqsilistmqpubsub command" on page 302
BIPMGTB	Migrates broker database tables	
BIPRELG	Reads the WebSphere Event Broker trace and formats it to a text file.	"mqsireadlog command" on page 309 "mqsiformatlog command" on page 293
BIPRPMS	Existing command ported to z/OS.	"mqsireportflowstats command" on page 314
BIPRPPR	Reports broker property settings	"mqsireportproperties command" on page 316
BIPRSCM	Existing command ported to z/OS.	"mqsirestoreconfigmgr command" on page 321
BIPSPMF	Stop message flows	"mqsistopmsgflow command" on page 330
BIPSTMF	Start message flows	"mqsistartmsgflow command" on page 326

Security requirements for administrative tasks

This section contains the following topics:
- "ACL permissions"
- "Security requirements for Linux and UNIX platforms" on page 388
- "Security requirements for Windows platforms" on page 389
- "Security requirements for z/OS" on page 391

ACL permissions

WebSphere Event Broker uses Access Control Lists (ACLs) to govern which users and groups can manipulate objects within the Configuration Manager and Message Brokers Toolkit. There are four different access levels that can be granted for a user or group: Full, View, Deploy and Edit. Not all access levels are valid for all object types; the table below describes which permissions can be applied to each object type and summarizes the actions that the user or group can perform.

In order to reduce the number of access control entries that a broker administrator must create, the ACL permissions behave in a hierarchical manner. The root of the tree is the ConfigManangerProxy object, which has three children: RootTopic, Subscriptions and PubSubTopology. The PubSubTopology object has zero or more brokers as children, and each broker can have zero or more execution groups as children. When an ACL entry is added to a given object, that permission is granted to that object and all objects beneath it in the hierarchy unless it is overridden by another ACL entry. The following diagram shows an example hierarchy:

```
                            CMP
              ┌──────────────┼──────────────┐
         RootTopic      Subscriptions   PubSubTopology
                                     ┌────────┴────────┐
                                  Broker1           Broker2
                                ┌────┴────┐       ┌────┴────┐
                              Eg1A      Eg1B    Eg2A      Eg2B
```
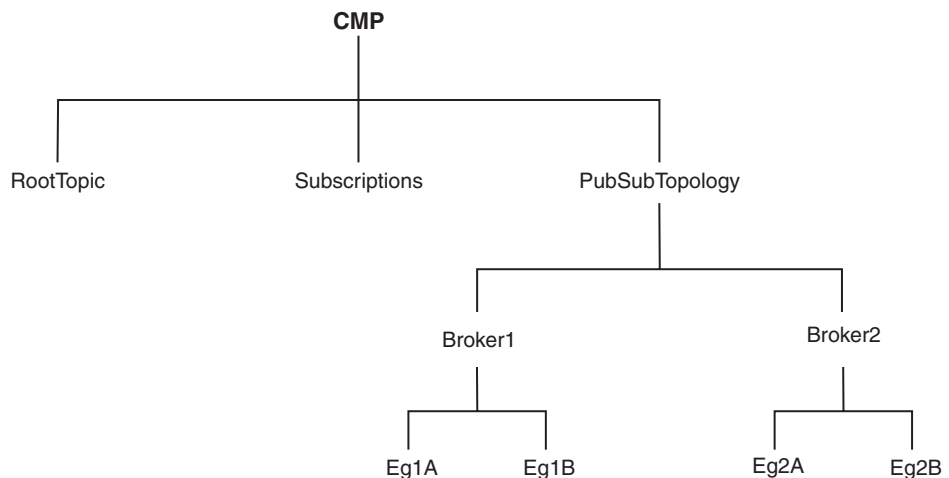
The following scenarios show how this hierarchical behavior works in practice.

Scenario 1

UserA has no access control entries. Therefore, UserA cannot manipulate any objects in the hierarchy, or see any of the objects defined in the hierarchy.

Scenario 2

UserB has a Deploy entry for Eg1A. Therefore, UserB has the following permissions:

CMP   None
RootTopic None
Subs   None
Topology View
Broker1   View
Eg1A   Deploy
Eg1B   None
Broker2   None
Eg2A   None
Eg2B   None

Note that because an access control entry has been given to the execution group, it is given an implied View permission to the parent broker and the topology, otherwise it would not be visible in the tooling. As there are no actual ACL entries for this user for the broker or topology, however, there is no inherited access to the other broker or execution groups. In practice, this means that UserB can see that there is another execution group defined on that broker, but cannot see any details (including the name).

Similarly, UserB can see that another broker exists within the topology, but cannot see any details. UserB has no access to the RootTopic or to the subscriptions table.

Scenario 3

UserC has a View entry for the Configuration Manager Proxy (CMP), and a Full control entry for Broker1. Therefore, UserC has the following permissions:

CMP   View
RootTopic View
Subs   View
Topology View
Broker1   Full
Eg1A   Full
Eg1B   Full
Broker2   View
Eg2A   View
Eg2B   View

Scenario 4

UserD has Full control entry for the CMP and a View entry for Broker1. Therefore, UserD has the following permissions:

CMP   Full
RootTopic Full
Subs   Full
Topology Full
Broker1   View
Eg1A   View
Eg1B   View
Broker2   Full
Eg2A   Full
Eg2B   Full

Note that in this example, without the View entry for Broker1, the user would have full control. This use of View entries is useful because it allows users who usually have full control over a given object to reduce their access temporarily in order to prevent accidental deletion or deployment. If the user needs full control of the object, removing the View entry restores full control, so that they can perform the operations that they need, then restore the View entry.

In order to change the access control entries for an object, a user must have full control over that object or any parent in the hierarchy. This means that the permission to change the ACLs themselves works the same as described above, with the exception that access to the ACLs cannot be removed by granting a lower permission further down the tree; this is necessary because otherwise a user would be able to give themselves a View entry and would not then be able to remove it.

ACL entries can be manipulated using the Java Configuration Manager Proxy API or using the **mqsicreateaclentry**, **mqsideleteaclentry**, and **mqsilistaclentry** commands.

The following table explains the actions which can be performed by a user with a given permission:

Object	Permission	Rights
Topology	Full control	• Create and delete brokers. • Create and delete collectives. • Add and remove brokers from collectives. • Create and delete connections. • Deploy topology. • All topology View permission rights
	View	• View topology configuration and managed subcomponents.
Broker	Full control	• Create and delete execution groups. • Edit all broker properties. • All broker Deploy permission rights. • All execution groups Full control permission rights for contained execution groups. • All broker View permission rights.
	Deploy	• Deploy broker configuration. • All broker View permission rights.
	View	• View broker configuration and managed subcomponents. • Implicit view access to Topology.
Execution group	Full control	• Edit all execution group properties. • Start and stop execution groups. • All execution group Deploy permission rights. • All execution group View permission rights.
	Deploy	• Deploy execution group configuration. • Start and stop assigned message flows. • Start and stop trace. • All execution group View permission rights.
	View	• View execution group configuration and managed subcomponents. • Implicit View access to parent broker and topology.

Object	Permission	Rights
Root topic	Full control	• Edit "Topic Access Control List". • All root topic Deploy permissions. • All root topic Edit permissions. • All root topic View permissions.
	Deploy	• Deploy entire topic configuration. • All root topic View permissions.
	Edit	• Create and delete child topics. • All root topic View permissions.
	View	• View all topics (including child topics), and any managed subcomponents.
Subscription	Full control	• Delete any subscription. • All subscription "View" permissions.
	View	• View or query all subscriptions and any managed subcomponents.

Security requirements for Linux and UNIX platforms

This table is a summary of the authorizations in a Linux or UNIX environment.

User is...	Linux or UNIX domain
Creating a component	• Member of **mqbrkrs** and **mqm**. • In most situations, the component runs under the login ID used to issue the create command. When root is used to issue the create command, it can nominate any user to run the component.
Installing	• Superuser.
Uninstalling	• Superuser.
Changing a component	• Member of **mqbrkrs**.
Deleting a component	• Member of **mqbrkrs** and **mqm**.
Starting a component	• Member of **mqbrkrs**. • Member of **mqm**. • Service user ID.
Stopping a component	• Member of **mqbrkrs**. • Member of **mqm** if -q is specified. • Service user ID.
Listing a component	• Member of **mqbrkrs**.
Changing, displaying, retrieving trace information.	• Member of **mqbrkrs**.
Running User Name Server (login ID).	• Member of **mqbrkrs**. The User Name Server runs under the login ID specified in the create command.
Running broker (WebSphere MQ non-trusted application) (login ID).	• Member of **mqbrkrs**. The broker runs under the login ID specified in the create command.
Running broker (WebSphere MQ trusted application) (login ID).	• Login ID must be **mqm**. • **mqm** must be a member of **mqbrkrs**.
Clearing, joining, listing WebSphere MQ publish/subscribe brokers.	• Member of **mqbrkrs**.

User is...	Linux or UNIX domain
Running publish/subscribe applications.	• Any user, subject to topic and WebSphere MQ queue access control.

> **Note:** When the service user ID is root, all libraries loaded by the broker, including all user-written plug-in libraries and all shared libraries that they might access, also have root access to all system resources (for example, file sets). Review and assess the risk involved in granting this level of authorization.

Security requirements for Windows platforms

The following table summarizes the security requirements for the WebSphere Event Broker administrative tasks. It illustrates what group membership is required if you are using a local security domain defined on your local system SALONE, or a primary domain named PRIMARY, or a trusted domain named TRUSTED. The contents of this table assume that you have created both the Configuration Manager and the User Name Server with the same security domain.

User is...	Local domain (SALONE)	Primary domain (PRIMARY) / Windows Single domain (PRIMARY)	Trusted domain (TRUSTED) / Windows Parent/Child domain in domain tree (TRUSTED)
Creating broker, Configuration Manager, User Name Server, or databases withmqsicreatedb	• Must be a user ID defined in SALONE • Member of **Administrators**	• Must be a user ID defined in PRIMARY • Member of SALONE**Administrators**	• Must be a user ID defined in TRUSTED • Member of SALONE**Administrators**
Changing broker, Configuration Manager, User Name Server, DatabaseInstanceMgr	• Must be a user ID defined in SALONE • Member of **Administrators**	• Must be a user ID defined in PRIMARY • Member of SALONE**Administrators**	• Must be a user ID defined in TRUSTED • Member of SALONE**Administrators**
Deleting broker, Configuration Manager, User Name Server, or databases withmqsideletedb	• Member of **Administrators**	• Member of SALONE**Administrators**	• Member of SALONE**Administrators**
Starting broker, Configuration Manager, User Name Server, orDatabaseInstanceMgr	• Member of **Administrators**	• Member of SALONE**Administrators**	• Member of SALONE**Administrators**
Listing broker, Configuration Manager, User Name Server, or DatabaseInstanceMgr	• Must be a user ID defined in SALONE • User ID must have the authority to query the registry values under *WebSphereMQIntegrator* entry in the registry. • Member of **mqbrkrs** if issuing the command: **mqsilist** <broker name><execution group name>.	• Must be a user ID defined in PRIMARY • User ID must have the authority to query the registry values under *WebSphereMQIntegrator* entry in the registry. • Member of PRIMARY**Domain mqbrkrs** if issuing the command: **mqsilist** <broker name><execution group name>.	• Must be a user ID defined in TRUSTED • User ID must have the authority to query the registry values under *WebSphereMQIntegrator* entry in the registry. • Member of TRUSTED**Domain mqbrkrs** if issuing the command: **mqsilist** <broker name><execution group name>.
Changing, displaying, retrieving trace information	• Must be a user ID defined in SALONE • Member of **mqbrkrs**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs**

User is...	Local domain (SALONE)	Primary domain (PRIMARY) / Windows Single domain (PRIMARY)	Trusted domain (TRUSTED) / Windows Parent/Child domain in domain tree (TRUSTED)
Running User Name Server(service user ID)	• Must be a user ID defined in SALONE • Member of **mqbrkrs**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs**
Running DatabaseInstanceMgr (service user ID)	• Must be a user ID defined in SALONE • Member of mqbrkrs	• Must be a user ID defined in PRIMARY • Member of PRIMARY\\Domain mqbrkrs	• Must be a user ID defined in TRUSTED • Member of TRUSTED\\Domain mqbrkrs
Running Configuration Manager (service user ID)	• Must be a user ID defined in SALONE • Member of **mqbrkrs** • Member of **mqm** • Member of **Adminstrators**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs** • Member of SALONE**mqm** (see note 1) • Member of **SALONE/Adminstrators**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs** • Member of SALONE**mqm** (see note 2) • Member of **SALONE/Adminstrators**
Running broker (WebSphere MQ fastpath off) (service user ID)	• Must be a user ID defined in SALONE • Member of **mqbrkrs**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs**
Running broker (WebSphere MQ fastpath on) (service user ID)	• Must be a user ID defined in SALONE • Member of **mqbrkrs** • Member of **mqm**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs** • Member of SALONE**mqm**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs** • Member of SALONE**mqm**
Clearing, joining, listing WebSphere MQ Publish/Subscribe brokers	• Must be a user ID defined in SALONE • Member of **mqbrkrs**	• Must be a user ID defined in PRIMARY • Member of PRIMARY**Domain mqbrkrs**	• Must be a user ID defined in TRUSTED • Member of TRUSTED**Domain mqbrkrs**
Running Message Brokers Toolkit (see note 3)	• Must be a user ID defined in SALONE (see note 4). For example, SALONE\\User1 is valid, PRIMARY\\User2 and TRUSTED\\User3 are not.	• For both domain awareness enabled and domain awareness disabled, when using Message Brokers Toolkit ACLs, user IDs must be members of any local ACL groups created on SALONE.	• For both domain awareness enabled and domain awareness disabled, when using Message Brokers Toolkit ACLs, user IDs must be members of any local ACL groups created on SALONE.
Running publish/subscribe applications	• Must be a user ID defined in SALONE. For example, SALONE\\User1 is valid, PRIMARY\\User2 and TRUSTED\\User3 are not.	• Must be a user ID defined in PRIMARY. For example, PRIMARY\\User2 is valid, SALONE\\User1 and TRUSTED\\User3 are not.	• Must be a user ID defined in TRUSTED. For example, TRUSTED\\User3 is valid, SALONE\\User1 and PRIMARY\\User2 are not.

Notes:

1. If you are running in a primary domain, you can also:
   - Define the user ID in the domain PRIMARY.
   - Add this ID to the group **PRIMARY\\Domain mqm**.
   - Add the **PRIMARY\\Domain mqm** group to the group **SALONE\\mqm** .
2. If you are running in a trusted domain, you can also:

- Define the user ID in the domain TRUSTED.
- Add this ID to the group **TRUSTED\Domain mqm**.
- Add the **TRUSTED\Domain mqm** group to the group **SALONE\mqm** .

3. All Message Brokers Toolkit users need read access to the WebSphere MQ java \lib subdirectory of the WebSphere MQ home directory (the default is X:\Program Files \WebSphere MQ , where X: is the operating system disk). This access is restricted to users in the local group **mqm** by WebSphere MQ. WebSphere Event Broker installation overrides this restriction and gives read access for this subdirectory to all users.

4. If a valid user ID is defined in the domain used by the Configuration Manager (for example, PRIMARY\User4), an identical user defined in a different domain (for example, DOMAIN2\User4) can access the Message Brokers Toolkit with the authorities of PRIMARY\User4.

5. Ensure that the service user ID has the required access to relevant directories of the product directory tree; for example, write access to the logs directory. If a workpath other than the default has been set for any component, ensure that the services user ID has appropriate access to this location.

6. If you are running a Configuration Manager with one user ID and a broker with a different user ID on another computer, you might see an error message when trying to deploy message flows to the broker. To avoid this, do the following:

   - Ensure that the broker's user ID is a member of the **mqm** and **mqbrkrs** groups.
   - Define the broker's user ID on the computer where the Configuration Manager is running.
   - Define the Configuration Manager's user ID on the computer where the broker is running.
   - Ensure that all IDs are in lowercase so that they are compatible between computers.

Security requirements for z/OS

This table is a summary of the UNIX System Services file access authorizations in a z/OS environment.

User is...	File access
Creating broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS user ID issuing the command • The broker, User Name Server and Configuration Manager run under their z/OS assigned started task ID
Installing	• READ and WRITE access to the installation directory by the z/OS user ID installing the product
Uninstalling	• READ and WRITE access to the installation directory by the z/OS user ID uninstalling the product
Changing broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS user ID issuing the command
Deleting broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS user ID issuing the command

User is...	File access
Starting broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS assigned started task user ID
Stopping broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS assigned started task user ID
Listing broker, User Name Server, Configuration Manager	• READ and WRITE access to the component directory by the z/OS user ID issuing the command
Changing, displaying, retrieving trace information.	• READ and WRITE access to the component directory by the z/OS user ID issuing the command
Clearing, joining, listing WebSphere MQ publish/subscribe brokers.	• READ and WRITE access to the component directory by the z/OS assigned started task user ID • Member of **mqbrkrs** group
Running publish/subscribe applications.	• Any user, subject to topic and WebSphere MQ queue access control.

Related concepts

"Security for runtime resources" on page 4

Related tasks

"Setting up z/OS security" on page 29

"Setting up broker domain security" on page 8

"Enabling topic-based security" on page 25

Related reference

"Summary of required access (z/OS)" on page 337

"Customization tasks and roles (z/OS)" on page 335

Part 5. Appendixes

Appendix. Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032,
Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England
SO21 2JN

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

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

(C) (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

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

AIX	CICS	Cloudscape
DB2	DB2 Connect	DB2 Universal Database
developerWorks	Domino	
Everyplace	FFST	First Failure Support Technology
IBM	IBMLink	IMS
IMS/ESA	iSeries	Language Environment
Lotus	MQSeries	MVS
NetView	OS/400	OS/390
pSeries	RACF	Rational
Redbooks	RETAIN	RS/6000
SupportPac	Tivoli	VisualAge
WebSphere	xSeries	z/OS
zSeries		

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

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

Intel and Pentium are trademarks or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

A

access control lists 33
  permissions 385
administration
  broker domain 185
  security requirements 385
  z/OS 333
authentication 38
  implementing 16
  SSL 5
authorization for configuration tasks 3

B

backup
  resources 196
    distributed systems 196
    z/OS 197
broker administration preferences,
  changing 157
broker database
  changing access 140
  failure 140
broker domain
  adding a broker 150
  administration 185
  configuring 49, 157
    components 66
    in the workbench 146
  configuring locales 176
    code page converters 179
    generating code page
     converters 179
    UNIX 176
    Windows 178
    z/OS 179
  connecting and disconnecting 185
  default configuration 128
  designing 49
  log information
    clearing 195
    filtering 194
    refreshing 193
    saving 195
    viewing 192
  removing a broker 153
broker domains
  security 8
broker networks 160
  heterogeneous 164
  migrated 164
Broker Topology editor
  changing properties 164
brokers
  adding 150
  changing 131
    Linux and UNIX systems 132
    Windows 133
    z/OS 134
  cloned 163

brokers *(continued)*
  connecting
    in a collective 165
  connecting a user name server 124
  copying 151
  creating 86
    AIX 88
    HP-UX 89
    Linux 90
    Solaris 91
    Windows 92
  customizing a new broker on
    z/OS 94
    component dataset operations 97
    Component information 95
    copying the started task 101
    creating the broker
     component 101
    creating the broker directory 96
    creating the broker PDSE 96
    creating the broker runtime
     environment 97
    creating the environment file 100
    customizing the JCL 98
    DB2 information 94
    installation information 94, 118
    JCL variables 345
    priming DB2 100
    required information 94
    sample files 347
    starting components 102
  deleting 141
    Linux and UNIX systems 141
    Windows 142
    z/OS 142
  execution groups, adding 155
  properties 201
    changing 152
  removing 153
  removing deployed children 154
  renaming 153
  sample files, z/OS
    BIPBPROF 348
    BIPBRKP 375
    BIPCBRK 352
    BIPCRDB 360
    BIPDSNAO 369
    BIPGEN 370
  security 12
  service ID 12
  starting and stopping 187
    UNIX 187
    Windows 188
    z/OS 188
  WebSphere MQ resources 53
  z/OS
    connecting a user name
     server 125

C

c security on z/OS 127
channels, starting WebSphere MQ 126
characters allowed in commands 217
cloned brokers 163
  adding 173
  defining 173
  deleting 174
code page converters 179
  new 179
collectives 161
  adding a broker 166
  creating 165
  deleting 165
  removing a broker 166
command console
  command environment, setting
   up 215
commands 202, 214
  characters allowed in 217
  command environment,
   mqsiprofile 215
  command environment, setting
   up 215
  Linux mqsiprofile file 216
  mqsi_setupdatabase 322
  mqsibackupconfigmgr 220
  mqsicbrreport 221
  mqsichangebroker 221
  mqsichangeconfigmgr 228
  mqsichangedbimgr 231
  mqsichangeflowstats 232
  mqsichangeproperties 236
  mqsichangetrace 245
  mqsichangeusernameserver 250
  mqsiclearmqpubsub 253
  mqsicreateaclentry 254
  mqsicreatebroker 257
  mqsicreateconfigmgr 266
  mqsicreatedb 270
  mqsicreateexecutiongroup 272
  mqsicreateusernameserver 275
  mqsideleteaclentry 278
  mqsideletebroker 281
  mqsideleteconfigmgr 283
  mqsideletedb 285
  mqsideleteexecutiongroup 286
  mqsideleteusernameserver 288
  mqsideploy 289
  mqsiformatlog 293
  mqsijoinmqpubsub 295
  mqsilist 296
  mqsilistaclentry 298
  mqsilistmqpubsub 302
  mqsimigratecomponents 304
  mqsireadlog 309
  mqsireload 313
  mqsireportflowstats 314
  mqsireportproperties 316
  mqsireporttrace 318
  mqsirestoreconfigmgr 321

IBM ®

Printed in USA