WebSphere Message Broker

Configuration, Administration, and Security

Version 6 Release 1

WebSphere Message Broker

Configuration, Administration, and Security

Version 6 Release 1

Note
Before you use this information and the product that it supports, read the information in the Notices appendix.

This edition applies to version 6, release 1, modification 0, fix pack 4 of IBM WebSphere Message Broker and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this topic collection

This PDF file has been created from the WebSphere Message Broker Version 6.1 (fix pack 4 update, May 2009) information center topics. Always refer to the WebSphere Message 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 result in a "file not found "error message. 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

Security is an important consideration for both developers of WebSphere® Message Broker applications, and for system administrators configuring WebSphere Message Broker authorities.

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

- "Security overview"
- "Planning for security when you install WebSphere Message Broker" on page 27
- "Setting up message flow security" on page 28
- "Setting up broker domain security" on page 45
- "Setting up z/OS security" on page 73
- Transferring files securely using SFTP
- "Publish/subscribe security" on page 78
- "Securing the publish/subscribe domain" on page 86

Security overview

When you are designing a WebSphere Message Broker application it is important to consider the security measures that are needed to protect the information in the system.

An important aspect of securing an enterprise system is the ability to protect the information that is passed from third parties. This capability includes restricting access to WebSphere MQ and JMS queues. For information about the steps involved, refer to the documentation supplied by your transport provider. If you are using HTTPS, you need to set specific properties in the HTTP nodes. For information on this, see HTTPInput node, HTTPRequest node, and HTTPReply node.

Some security configuration is required to enable WebSphere Message Broker to work correctly and to protect the information in the system. For example, you can secure the messaging transport with SSL connections, restrict access to queues, apply WS-Security to Web services, and secure access to message flows.

In addition, system administrators need WebSphere Message Broker authorities that allow them to perform customization and configuration tasks, run utilities, perform problem determination, and collect diagnostic materials.

Security information is split into several areas:

- "Message flow security" on page 4
- WS-Security
- "Authorization for configuration tasks" on page 17
- "Security for development resources" on page 17
- "Security for runtime resources: Access control lists" on page 17
- "Security exits" on page 20
- "SSL authentication" on page 25 (for the Real-time nodes only)
- "Tunneling" on page 26
- "Quality of protection" on page 26
- "Domain awareness for Message Broker Toolkit users on Windows" on page 27.

Message flow security

WebSphere Message Broker provides a security manager, which enables you to control access to individual messages in a message flow, using the identity of the message.

You can configure the broker to perform end-to-end processing of an identity carried in a message through a message flow. This capability enables you to configure security for a message flow, allowing you to control access based on the identity associated with the message and providing a security mechanism that is independent of both transport type and message format.

If you do not enable message flow security, the default security facilities in WebSphere Message Broker are based on the facilities provided by the transport mechanism. In this case, the broker processes all messages that are delivered to it, using the broker service identity as a proxy identity for all message instances. Any identity that is present in the incoming message is ignored.

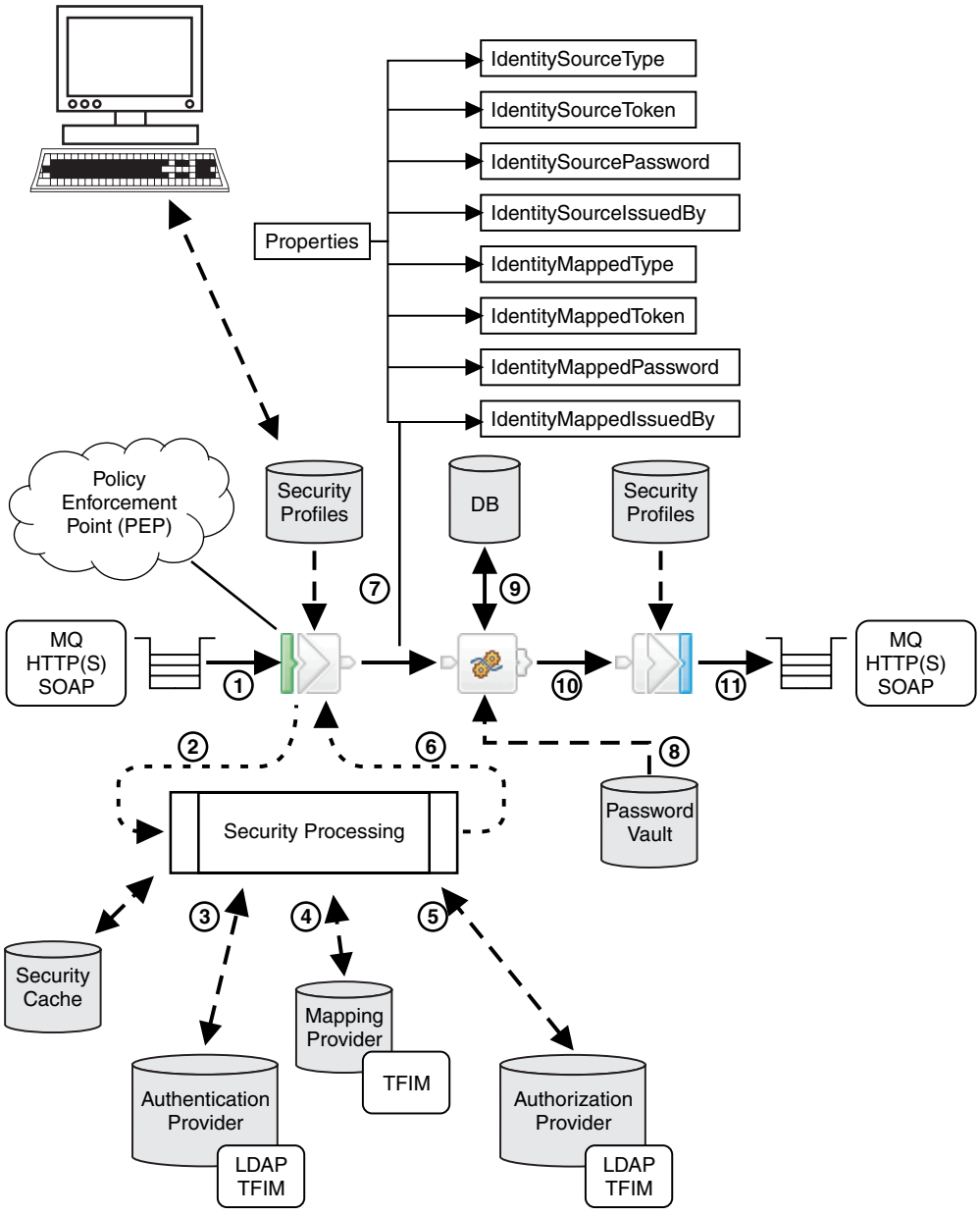
Instead of delegating this authority to the transport mechanism, the security manager enables the broker to:

- Extract the identity from an inbound message
- Authenticate the identity (using an external security provider)
- Map the identity to an alternative identity (using an external security provider)
- Check that either the alternative identity or the original identity is authorized to access the message flow (using an external security provider)
- Propagate either the alternative identity or the original identity with an outbound message.

The security functions that are associated with a message flow are controlled by using "Security profiles" on page 7, which are created by the broker administrator and accessed by the security manager at run time. Two external security providers are supported: Lightweight Directory Access Protocol (LDAP) for authentication and authorization, and Tivoli® Federated Identity Manager (TFIM) for authentication, mapping, and authorization.

The input nodes that support runtime security are MQInput, HTTPInput, and SOAPInput. However, the support for treating security exceptions as normal exceptions is provided only by the MQInput and HTTPInput nodes; it is not available in the SOAPInput node.

The output nodes that support identity propagation are MQOutput, HTTPRequest, SOAPRequest, and SOAPAsyncRequest. If the message flow is a Web Service that is implemented by using the broker SOAP nodes, the identity can be taken from the WS-Security header tokens that are defined through appropriate Policy sets and bindings.



The following steps explain the sequence of events that occur when a message arrives at a message flow. The numbers correspond to those in the preceding diagram:

1. When a message arrives at an input node, a security profile associated with the node indicates whether runtime security is configured. The broker's security manager is called to read the profile, which specifies the combination of propagation, authentication, authorization, and mapping to be performed with the identity of the message. It also specifies the external security provider to be used.

   You can create security profiles by using either the mqsicreateconfigurableservice command or an editor in the Broker Administration perspective of the workbench. You then use the Broker Archive editor to configure the security profile on the message flow.

If you configure a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity), you must also define and specify an appropriate policy set and bindings. For more information, see Policy sets.

2. If a security profile is present, the security manager extracts the identity information from the input message and sets it in a group of new elements in the Properties folder. This source identity information can be in a message header or in the message body itself, or a mixture of the two.
3. If authentication is specified in the security profile, the security manager calls the provider to authenticate the identity. A failure results in a SecurityException being thrown. The security providers that are supported by Message Broker for authentication are LDAP and TFIM.
4. If identity mapping is specified in the security profile, the security manager calls the provider to map the identity to an alternative identity. A failure results in a SecurityException being thrown. Otherwise, the mapped identity information is set in a group of new elements in the Properties folder. The provider that is supported by Message Broker for identity mapping is TFIM.
5. If authorization is specified in the security profile, the security manager calls the provider to authorize that the identity has access to this message flow. A failure results in a SecurityException being issued. Message Broker supports LDAP and TFIM for authorization.
6. When all security processing is complete, control returns to the input node.
7. The message, including the Properties folder and its source and mapped identity information, is propagated down the message flow.
8. At subsequent nodes in the message flow an identity might be required to access a resource such as a database. The identity used to access such a resource is a proxy identity, either the broker's identity or an identity configured for the specific resource by using the mqsisetdbparms command.
9. When you are developing a message flow you can use the identity fields in the Properties folder for application processing (for example, identity-based routing or content building based on identity). Also, as an alternative to invoking mapping through TFIM, you can set the mapped identity fields in a user-defined node.
10. When the message reaches an output node, a security profile associated with the node indicates whether an identity is to be propagated when the message is sent.

    If the security profile indicates that propagation is required, the mapped identity is used. If the mapped identity is not set, the source identity is used. If no identity is set, a SecurityException is issued.
11. The propagated identity is included in the appropriate message header when it is sent.

To improve performance, the authentication, authorization, and mapping information from the configured providers is cached for reuse. You can use the mqsireloadsecurity command to reload the cache and the mqsichangeproperties command to set the expiry and sweep intervals for the cache.

For a SOAPRequest and SOAPAsyncRequestnode, an appropriate policy set and bindings can be defined to specify how the token is placed in the WS-Security header (rather than the underlying transport headers). For more information, see Policy sets.

The following topics in this section provide more detailed information about message flow security:

Security profiles

A security profile defines the security operations to be performed by input and output nodes.

Security profiles are created by the broker administrator before deploying a message flow, and are accessed by the security manager at run time.

A security profile allows a broker administrator to specify whether identity propagation, authentication, authorization, and mapping are performed on the identity of messages in the message flow, and if so, which external security provider is used. Lightweight Directory Access Protocol (LDAP) is supported for authentication and authorization, and IBM® Tivoli Federated Identity Manager (TFIM) is supported for authentication, authorization, and mapping.

Security properties apply to input, output, and request nodes, and are configured by the administrator at deployment time in the Broker Archive editor. Input, output, and request nodes have a **Security Profile** property (in the Broker Archive editor), which can be set to a specific profile name, left blank so that they inherit the Security Profile property set at the message flow level, or set to *No Security* to explicitly turn off security for the node. For input nodes, the **Security Profile** property determines whether runtime security is configured. If the property does not resolve to a real security profile, identity tokens are not extracted to the identity fields in the Properties folder.

The security profile also specifies whether propagation is required. A pre-configured profile that specifies propagation is provided for use by output and request nodes. This profile is the Default Propagation security profile.

Security profiles contain values for the following properties:

authentication
: Defines the type of authentication that is performed on the source identity. This property applies only to input nodes. For more information, see "Authentication" on page 10.

authenticationConfig
: Defines the information that the broker needs to connect to the provider, and the information needed to look up the identity tokens. It is a provider-specific configuration string. This property applies only to input nodes.

mapping
: Defines the type of mapping that is performed on the source identity. This property applies only to input nodes. For more information, see "Identity mapping" on page 12.

mappingConfig
Defines how the broker connects to the provider, and contains additional information required to look up the mapping routine. It is a provider-specific configuration string. This property applies only to input nodes.

authorization
Defines the types of authorization checks that are performed on the mapped or source identity. This property applies only to input nodes. For more information, see "Authorization" on page 11.

authorizationConfig
Defines how the broker connects to the provider, and contains additional information that can be used to check access (for example, a group that can be checked for membership). It is a provider-specific configuration string. This property applies only to input nodes.

passwordValue
Defines how passwords are treated when they enter a message flow. If *PLAIN* is selected, the password appears in the Properties folder in plain text. If *OBFUSCATE* is selected, the password appears in the Properties folder in base64 encoding. If *MASK* is selected, the password appears in the Properties folder as four asterisks (****). This property applies only to input nodes.

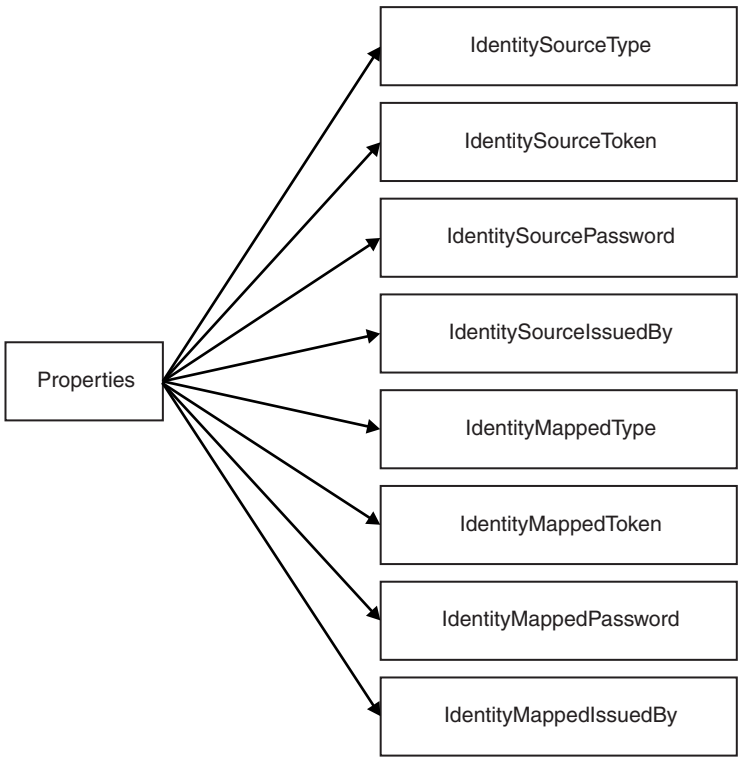
Propagation
Enables or disables identity propagation on output and request nodes. For more information, see "Identity propagation" on page 16.

For information on configuring a security profile for LDAP or TFIM, see "Creating a security profile" on page 28.

Identity

An identity is a piece of information that can uniquely identify an individual or object.

When a supported input node is configured with a security profile, the extracted identity is held in the broker as eight properties in the Properties folder of the message tree structure. These properties define two identities in the broker: source and mapped. For both the source and mapped identities, values are held for **Type**, **Token**, **Password**, and **IssuedBy** properties:



The **Type** property defines the format of the token, and valid values are *Username*, *Username and Password* and *X.509 Certificate*. The **Type** property can also have a value of *Transport Default*, which causes the transport's default to be used. For WebSphere MQ the default is *Username*, and for HTTP the default is *Username and Password*.

The **Token** property holds the security token and, in the case of a *Username and Password* token, the Password field contains the attached password. The **IssuedBy** property defines where the token was created. For example, for an *X.509 Certificate* this value could be "IBM" (the Common Name of the Certifying Authority). For *Username* and *Username and Password* formats, the value is transport specific unless the IssuedBy property is set on the node. For more information, see "Configuring identity extraction in a message flow" on page 34.

The following table shows the support that is provided (by the broker security manager and external security providers) for the different security token types:

Table 1. Support for security token types

Token type (format)	Broker security manager support	External security provider support
Username	Token obtained from one of the following transport headers: • MQ • HTTP • SOAP or from an MQ or HTTP message body.	LDAP: Authorization

Table 1. Support for security token types (continued)

Token type (format)	Broker security manager support	External security provider support
Username and password	Token obtained from one of the following transport headers: • HTTP • SOAP or from an MQ or HTTP message body.	LDAP: • Authentication • Authorization TFIM: • Authentication • Mapping • Authorization.
X.509 Certificate	Token obtained from a SOAP transport header or from an MQ or HTTP message body.	TFIM: • Authentication • Mapping • Authorization.

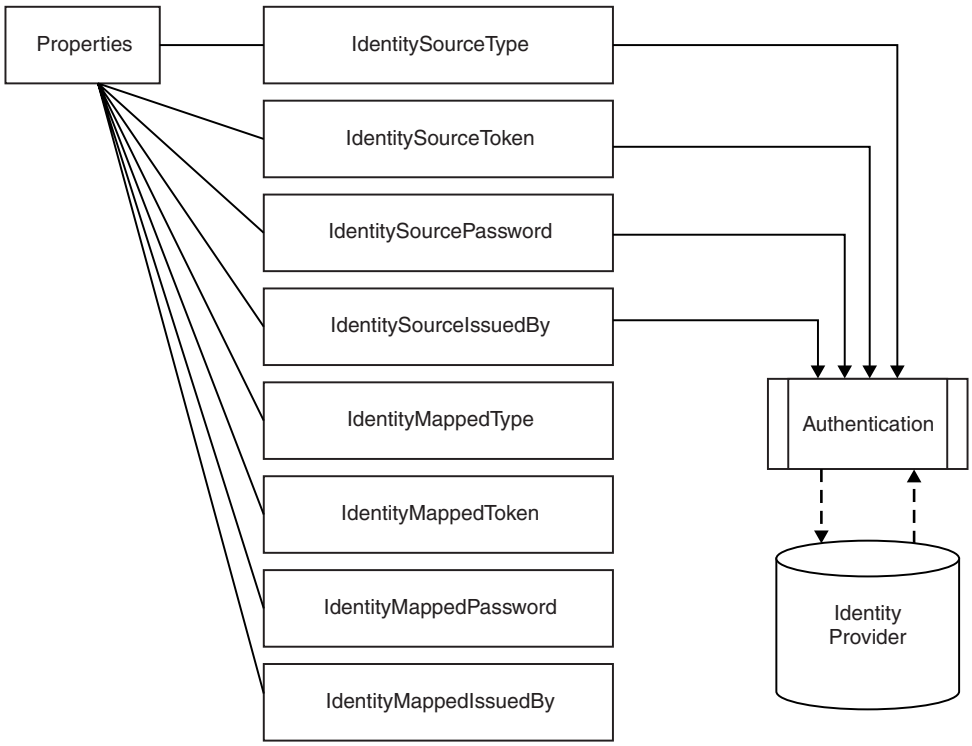
The source identity is set by the input node only if a security profile is associated with the node. The information to complete these fields is typically found in the headers of a message but can also be located in the body (as long as the node has been configured with an ESQL Path or XPath reference for the various properties). If multiple identities are available (for example, if you are using message aggregation), the first identity is used. The token extraction is transport specific and can be performed only using transports that support the flow of identities. These transports are: Websphere MQ, HTTP(S), and SOAP. See MQInput node and HTTPInput node for more information.

You can modify the values in the properties (for example, from ESQL), but do not write to the *IdentitySource** values. For example, you can create a custom identity mapping routine in ESQL or Java™ by using the *IdentitySource** values to create custom *IdentityMapped** values.

Authentication

Authentication is the process of establishing the identity of a user or system and verifying that the identity is valid.

In WebSphere Message Broker, authentication is the process of passing the identity source type and token to an external authenticator. For more information about source type and token, see “Identity” on page 8.



The two external security providers supported for authentication are Lightweight Directory Access Protocol (LDAP) and Tivoli Federated Identity Manager (TFIM). The external security provider checks the identities and returns a value to confirm whether the identity is authentic. If the identity is not authentic, a security exception is raised.

Some identity providers support only a single type of authentication token. If a token of another type is passed into the message flow, an exception is raised. For example, LDAP supports only a *Username and password* token.

If you want to use LDAP to authenticate an incoming identity token, the identity must be a member of the relevant group configured in the security profile. The LDAP server must be LDAP Version 3 compliant.

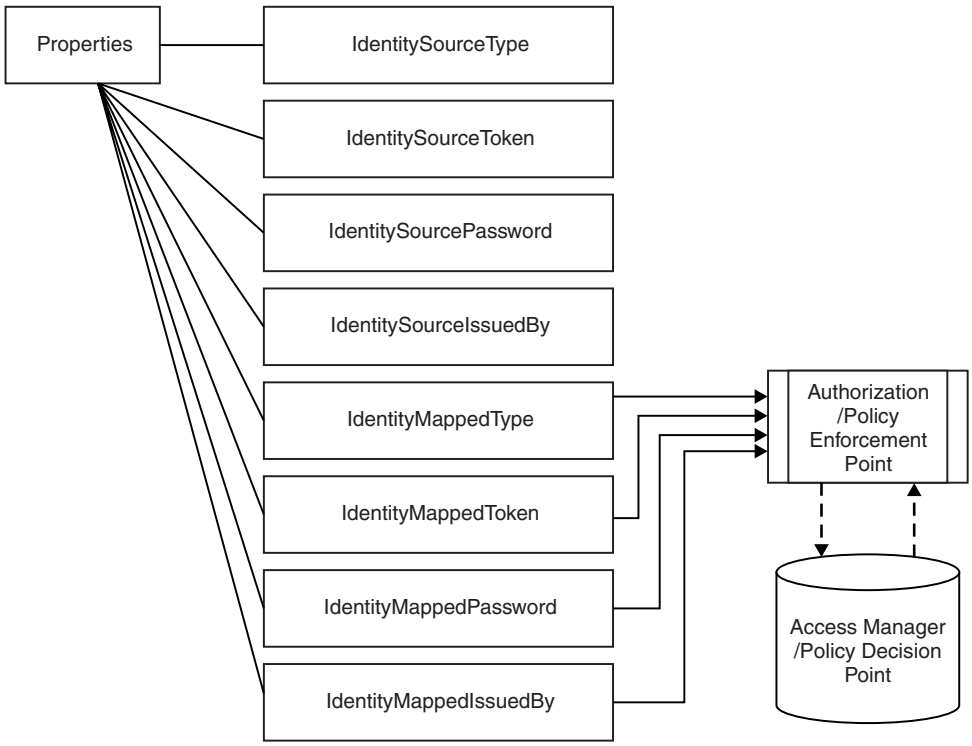
You can also select TFIM for authentication in the profile, but the TFIM module chain is invoked only once and the single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

For more information about using TFIM for authentication, see "Authentication, Mapping, and Authorization with TFIM and TAM" on page 14.

Authorization

Authorization is the process of verifying that an identity has permission to start a message flow.

If authentication and mapping are configured, they are used to verify the identity before it is authorized.



If a mapped identity exists, authorization is applied to the mapped identity. If a mapped identity does not exist, the source identity is used.

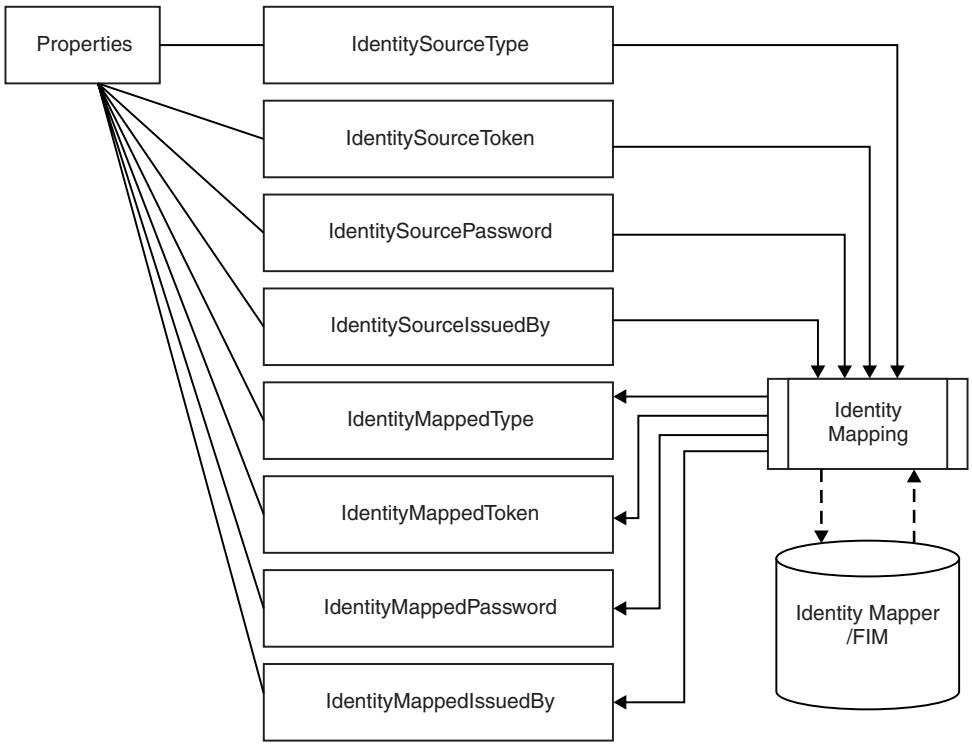
If you use LDAP for authorization, the identity must be a member of the relevant group configured in the security profile. The LDAP server must be LDAP Version 3 compliant.

Alternatively, you can select TFIM for authorization in the profile. However, the TFIM module chain is invoked only once and the single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

For more information about using TFIM for authorization, see "Authentication, Mapping, and Authorization with TFIM and TAM" on page 14.

Identity mapping

Identity mapping is the transformation of an identity in one format to an identity in another format, or the transformation of an identity in one realm to an identity in another realm.



TFIM mapping

The mapper that is supported by WebSphere Message Broker is IBM Tivoli Federated Identity Manager (TFIM). Mapping is performed only for input nodes with an associated security profile that specifies that TFIM is to be used for mapping. TFIM can also be selected for authentication and authorization in the profile, but the TFIM module chain is invoked only once and must be configured to perform all the required operations. Mapping is not performed in output nodes, even if the node has been configured with a security profile.

In the broker, identity mapping is performed at the input node, after authentication but before authorization. The source identity is passed to an identity mapper (also known as a Federated Identity Manager) for processing.

WebSphere Message Broker supports the mapping from a *Username* token to a *Username* token, and the mapping from an *X.509 certificate* to a *Username* token. No support exists for mapping to an X.509 token. When mapping from an X.509 certificate, TFIM can validate the certificate but cannot verify the identity of the original sender. However, if it is required, this verification integrity check can be performed by the SOAPInput node. For more information, see WS-Security mechanisms.

For more information about using TFIM for mapping, see “Authentication, Mapping, and Authorization with TFIM and TAM” on page 14.

User-defined mapping

When you develop a message flow, you can implement a custom token mapping to be used for identity propagation. For example, you can implement a custom token mapping using a JavaCompute node, following the input node. In the

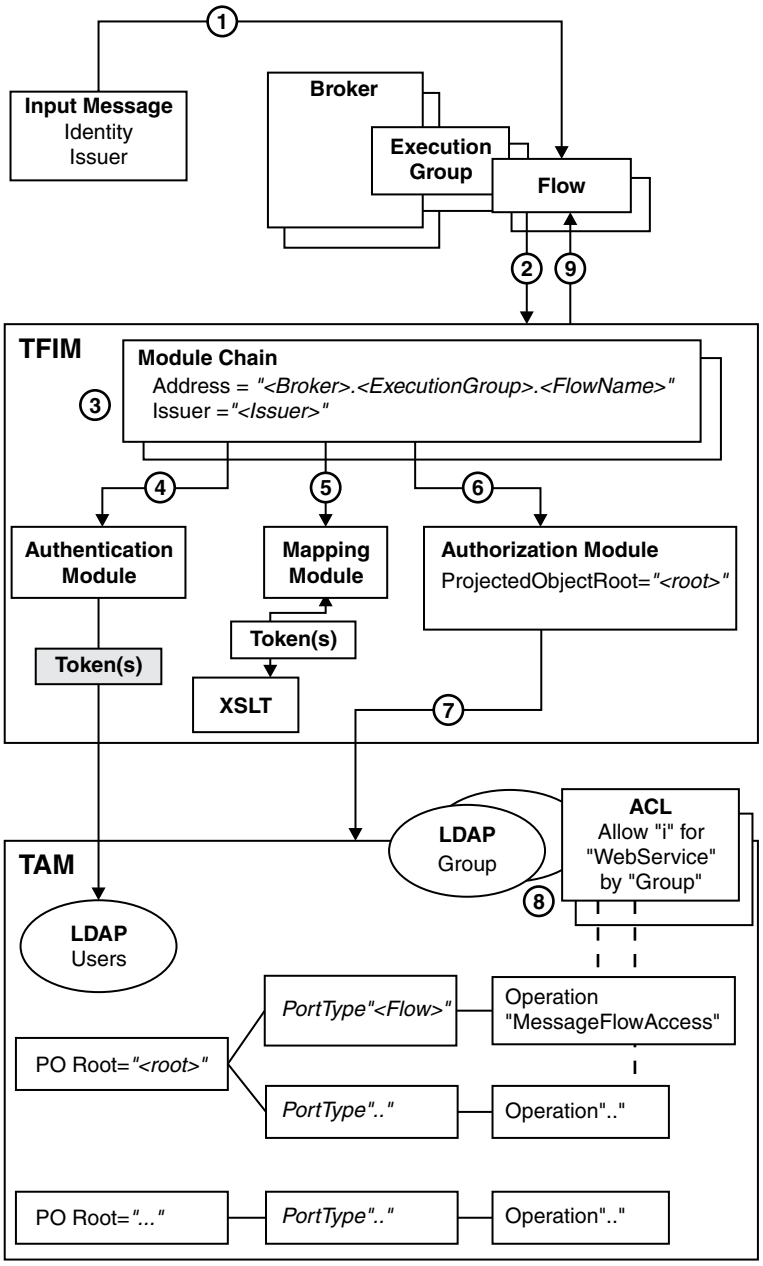
JavaCompute node you read the source identity values from the Properties folder, process them, then write the new identity values to the mapped identity fields. The mapped identity fields are then used in preference to the source identity fields by subsequent nodes. However, in this case, any authorization configured on the input node will already have been performed using the source identity.

Authentication, Mapping, and Authorization with TFIM and TAM

Use WebSphere Message Broker, Tivoli Federated Identity Manager (TFIM), and Tivoli Access Manager (TAM) to control authentication, mapping, and authorization.

WebSphere Message Broker makes a single TFIM WS-Trust call for an input node that is configured with a TFIM security profile, which means that a single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

The following diagram shows the configuration of WebSphere Message Broker, TFIM, and TAM to enable authentication, mapping, and authorization of an identity in a message flow:



The numbers in the preceding diagram correspond to the following sequence of events:

1. A message enters a message flow.
2. A WS-Trust request is issued by the broker, with these properties:
   - RequestType = Validate
   - Identity = Token(s) from input message
   - Issuer = Issuer from input message
   - AppliesTo Address = *"Broker.ExecutionGroup.FlowName"*
   - PortType = *"FlowName"*
   - Operation = "MessageFlowAccess"

3. TFIM selects a module chain to process the WS-Trust request, based on the AppliesTo Address and Issuer properties of the request.
4. A module chain can perform authentication if it includes a module (such as a UsernameTokenSTSModule or X509STSModule) in `validate` mode.
5. A module chain can perform mapping by using an XSLTransformationModule in `mapping` mode to manipulate the identity information.
6. A module chain can perform authorization by using an AuthorizationSTSModule in `other` mode. The module chain must be configured with a *Protected Object Root* value.
7. The AuthorizationSTSModule performs the authorization check by making a request to TAM with these properties:
   - Action = "i" (invoke)
   - Action Group = "WebService"
   - Protected Object = *"ProtectedObjectRoot.FlowName.MessageFlowAccess"*

     where *"i"* and *"WebService"* are default values used by an AuthorizationSTSModule; and *FlowName* and *MessageFlowAccess* are the WS-Trust request PortType and Operation values.
8. TAM processes the authorization request by:
   a. Finding the Access Control Lists (ACLs) associated with protected object *"<ProtectedObjectRoot>.<FlowName>.MessageFlowAccess"*.
   b. Checking whether or not the ACLs grant action "i" on action group "WebService" to the user (with the user either named directly, or by membership of a named group).
9. The WS-Trust reply is returned to the broker. If this action is the result of a mapping request, the WS-Trust reply contains the mapped identity token.

For further information about how to configure TFIM, see the IBM Tivoli Federated Identity Manager information center.

For information about how to configure TAM, see the IBM Tivoli Access Manager information center.

Identity propagation

Identity propagation enables a logical identity to be maintained throughout a message flow.

In an enterprise system, you can use different physical identities (such as user names and certificates) to represent a single logical identity through different parts of the enterprise. Identity propagation ensures that the logical identity is kept throughout the system by mapping between the various physical forms as necessary. For example, a message might enter the system using a certificate, but a user name token might be required for server processing of the message. Identity mapping is used to convert from the certificate to the username token, and identity propagation ensures that the mapped identity is placed in the correct place for the outbound transport.

For information about how to configure a message flow to propagate a message identity, see "Configuring for identity propagation" on page 43. For more information about how one physical identity is converted to another, see "Identity mapping" on page 12.

Security exception processing

A security exception is raised when a runtime security failure occurs during security processing in an input node.

Security exceptions are processed in a different way from other errors on the input node. An error is typically caught on the input node and routed down the Failure terminal for error processing, but security exceptions are not processed in the same way. By default, the broker does not allow security exceptions to be caught within the flow, but backs the message out or returns an error (as in the case of HTTP). Security exceptions are managed in this way to prevent a security denial of service attack filling the logs and destabilizing the system.

If you have designed the flow to be in a secure area and you want to explicitly perform processing of security exceptions, you can select the **Treat Security Exceptions as normal exceptions** property on the MQInput or HTTPInput nodes. This property causes security exceptions to be processed in the same way as other exceptions in the message flow.

For information on how to diagnose the causes of security exceptions, see "Diagnosing security problems" on page 44.

Authorization for configuration tasks

Authorization is the process of granting or denying access to a system resource. For WebSphere Message Broker, authorization is concerned with controlling who has permission to access WebSphere Message 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" on page 513.
- 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 69

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 Message Broker to address this environment over and above those available for a production environment.

Security for runtime resources: Access control lists

Access control list (ACL) entries allow or deny a user access to specific runtime resources. Runtime resources are WebSphere Message Broker objects that exist at run time in the broker domain.

Each runtime object has an ACL that determines which users and groups can access the object. Using ACL entries, you can control users' access to specific objects in the broker domain, and permit a user or group to view, modify, or deploy the object. You can manipulate the ACL entries by using the workbench,

the Java Configuration Manager Proxy API, or the mqsicreateaclentry, mqsideleteaclentry, and mqsilistaclentry commands.

For example, user USER1 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.

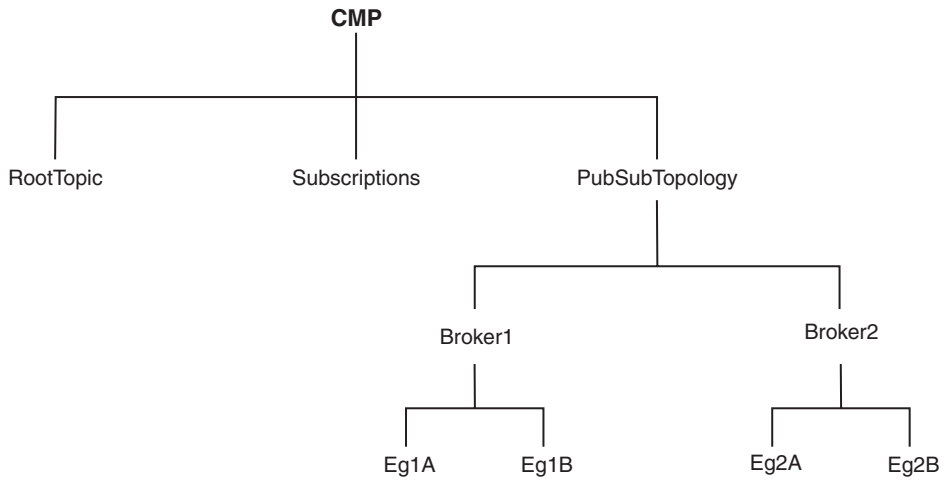
When you try to view or modify an object for which you require permission, the following information is passed to the Configuration Manager:
- Object type
- Object name
- Requested action
- Your user ID.

The Configuration Manager checks the ACL table and, if your user ID is included in the ACL entry for the named object, you are authorized to perform the operation.

There are four different access levels that can be granted for a user or group: Full control, View, Deploy, and Edit. Not all access levels are valid for all object types; see "ACL permissions" on page 723 for a list of the permissions that can be applied to each object type and for a summary of the actions that the user or group can perform.

An ACL entry contains the user name and can specify a host name or domain name. For example, it is possible for a user to gain access to the objects by creating an account on a computer that has a host name that is the same as an authorized Windows domain name. Use ACL entries to control access to the objects in the broker domain but do not rely on ACL entries to secure your broker domains; use SSL or security exits to secure the channels between components in the broker domain. Although ACL entries permit or deny access to an object based on the user ID, they do not secure the object because an ACL entry cannot verify the user's identity.

To reduce the number of ACL entries that a broker administrator must create, the ACL permissions operate 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, permission is granted to that object and to all objects beneath it in the hierarchy, unless it is overridden by another ACL entry. The following diagram shows an example hierarchy of access control list entries:



For examples showing how this hierarchy works in practice, see "Configuring security for domain components" on page 54.

To change the access control entries for an object, a user must have Full authority for that object or any parent in the hierarchy. This means that the permission to change the ACLs themselves works in the same way as described previously, 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.

On z/OS®, you must define an OMVS segment for user IDs and groups, so that a Configuration Manager can obtain user ID and group information from the External Security Manager (ESM) database.

ACL entries and groups

In previous versions of WebSphere Message Broker, access to runtime objects was controlled by defining a set of groups and assigning users to those groups. ACL entries enable you to control access with more granularity than groups. ACL entries also enable a single Configuration Manager to manage development, test, and production systems separately by configuring users' access to each broker. Using groups, you would have to place the development, test, and production systems in separate broker domains, each controlled by a separate Configuration Manager.

If you migrate a Configuration Manager from a previous version of WebSphere Message Broker, ACL entries are automatically defined for the following groups:

- mqbrkrs
- mqbrops
- mqbrdevt
- mqbrasgn
- mqbrtpic

Without these ACL entries, users that belong to these groups do not have authority to perform actions on the objects in the domain.

When you use single user ACLs, you must define the users on the workstation that is accessing the objects (that is, the machine on which the Toolkit is running), but you do not need to define them on the workstation that is running the Configuration Manager. However, when you are using Group ACLs, you must define the users on both workstations and then define the groups on the workstation that is running the Configuration Manager, before adding the users to the groups. This is necessary because no group information is passed between the workstations.

Security exits

Use security exit programs to verify that the partner at the other end of a connection is genuine.

When you start the Message Broker Toolkit a security exit to monitor the connection to the Configuration Manager is not started by default. If you want to protect access to the Configuration Manager from client programs, you can use the WebSphere MQ security exit facility.

If you want to use a security exit to provide connection security, you must define one when you create a new domain connection in the Domains view of the Message Broker Toolkit. You can enable security exits on the connection from the Message Broker Toolkit to the Configuration Manager.

- Set up a security exit on the channel at the Configuration Manager end. This security exit does not have any special requirements; you can provide a standard security exit.
- Set up a security exit in the Message Broker Toolkit. Identify the security exit properties when you create the domain connection.

The security exit is a standard WebSphere MQ security exit, written in Java.

For an overview of security exits and details in their implementation, see ″Channel security exit programs″ in the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

Public key cryptography

All encryption systems rely on the concept of a key. A key is the basis for a transformation, usually mathematical, of an ordinary message into an unreadable message. For centuries, most encryption systems have relied on private key encryption. Public key encryption is the only challenge to private key encryption that has appeared in the last 30 years.

Private key encryption

Private key encryption systems use a single key that is shared between the sender and the receiver. Both must have the key; the sender encrypts the message by using the key, and the receiver decrypts the message with the same key. Both the sender and receiver must keep the key private to keep their communication private. This kind of encryption has characteristics that make it unsuitable for widespread general use:

- Private key encryption requires a key for every pair of individuals who need to communicate privately. The necessary number of keys rises dramatically as the number of participants increases.

- Keys must be shared between pairs of communicators, therefore the keys must be distributed to the participants. The need to transmit secret keys makes them vulnerable to theft.
- Participants can communicate only by prior arrangement. You cannot send a usable encrypted message to someone spontaneously. You and the other participant must make arrangements to communicate by sharing keys.

Private key encryption is also called symmetric encryption because the same key is used to encrypt and decrypt the message.

Public key encryption

Public key encryption uses a pair of mathematically-related keys. A message that is encrypted with the first key must be decrypted with the second key, and a message that is encrypted with the second key must be decrypted with the first key.

Each participant in a public key system has a pair of keys. The symmetric (private) key is kept secret. The other key is distributed to anyone who wants it; this key is the public key.

To send an encrypted message to you, the sender encrypts the message by using your public key. When you receive the message, you decrypt it by using your symmetric key. To send a message to someone, you encrypt the message by using the recipient's public key. The message can be decrypted with the recipient's symmetric key only. This kind of encryption has characteristics that make it very suitable for general use:

- Public key encryption requires only two keys per participant. The increase in the total number of keys is less dramatic as the number of participants increases, compared to symmetric key encryption.
- The need for secrecy is more easily met: only the symmetric key needs to be kept secret, and because it does not need to be shared, the symmetric key is less vulnerable to theft in transmission than the shared key in a symmetric key system.
- Public keys can be published, which eliminates the need for prior sharing of a secret key before communication. Anyone who knows your public key can use it to send you a message that only you can read.

Public key encryption is also called asymmetric encryption, because the same key cannot be used to encrypt and decrypt the message. Instead, one key of a pair is used to undo the work of the other.

With symmetric key encryption, beware of stolen or intercepted keys. In public key encryption, where anyone can create a key pair and publish the public key, the challenge is in verifying the identity of the owner of the public key. Nothing prevents a user from creating a key pair and publishing the public key under a false name. The listed owner of the public key cannot read messages that are encrypted with that key because the owner does not have the symmetric key. If the creator of the false public key can intercept these messages, that person can decrypt and read messages that are intended for someone else. To counteract the potential for forged keys, public key systems provide mechanisms for validating public keys and other information with digital certificates and digital signatures.

Digital certificates

Certificates provide a way of authenticating users. Instead of requiring each participant in an application to authenticate every user, third-party authentication relies on the use of digital certificates.

A digital certificate is equivalent to an electronic ID card. The certificate serves two purposes:

- Establishes the identity of the owner of the certificate
- Distributes the owner's public key

Certificates are issued by trusted parties, called certificate authorities (CAs). These authorities can be commercial ventures or local entities, depending on the requirements of your application. The CA is trusted to adequately authenticate users before issuing certificates. A CA issues certificates with digital signatures. When a user presents a certificate, the recipient of the certificate validates it by using the digital signature. If the digital signature validates the certificate, the certificate is recognized as intact and authentic. Participants in an application need to validate certificates only; they do not need to authenticate users. The fact that a user can present a valid certificate proves that the CA has authenticated the user. The descriptor, trusted third-party, indicates that the system relies on the trustworthiness of the CAs.

Contents of a digital certificate

A certificate contains several pieces of information, including information about the owner of the certificate and the issuing CA. Specifically, a certificate includes:

- The distinguished name (DN) of the owner. A DN is a unique identifier, a fully qualified name including not only the common name (CN) of the owner but also the owner's organization and other distinguishing information.
- The public key of the owner.
- The date on which the certificate is issued.
- The date on which the certificate expires.
- The distinguished name of the issuing CA.
- The digital signature of the issuing CA. The message-digest function creates a signature based upon all the previously listed fields.

The core idea of a certificate is that a CA takes the public key of the owner, signs the public key with its own private key, and returns the information to the owner as a certificate. When the owner distributes the certificate to another party, it signs the certificate with its private key. The receiver can extract the certificate that contains the CA signature with the public key of the owner. By using the CA public key and the CA signature on the extracted certificate, the receiver can validate the CA signature. If valid, the public key that is used to extract the certificate is considered good. The owner signature is validated, and if the validation succeeds, the owner is successfully authenticated to the receiver.

The additional information in a certificate helps an application decide whether to honor the certificate. With the expiration date, the application can determine if the certificate is still valid. With the name of the issuing CA, the application can check that the CA is considered trustworthy by the site.

A process that uses certificates must provide its personal certificate, the one containing its public key, and the certificate of the CA that signed its certificate,

called a signer certificate. In cases where chains of trust are established, several signer certificates can be involved.

Requesting certificates

To get a certificate, send a certificate request to the CA. The certificate request includes:

- The distinguished name of the owner or the user for whom the certificate is requested
- The public key of the owner
- The digital signature of the owner

The message digest function creates a signature based on all the previously listed fields.

The CA verifies the signature with the public key in the request to ensure that the request is intact and authentic. The CA then authenticates the owner. Exactly what the authentication consists of depends on a prior agreement between the CA and the requesting organization. If the owner in the request is authenticated successfully, the CA issues a certificate for that owner.

Using certificates: Chain of trust and self-signed certificate

To verify the digital signature on a certificate, you must have the public key of the issuing CA. Public keys are distributed in certificates, therefore you must have a certificate for the issuing CA that is signed by the issuer. One CA can certify other CAs, so a chain of CAs can issue certificates for other CAs, all of whose public keys you need. Eventually, you reach a root CA that issues itself a self-signed certificate. To validate a user certificate, you need certificates for all of the intervening participants back to the root CA. You then have the public keys that you need to validate each certificate, including the user certificate.

A self-signed certificate contains the public key of the issuer and is signed with the private key. The digital signature is validated like any other, and if the certificate is valid, the public key it contains is used to check the validity of other certificates issued by the CA. However, anyone can generate a self-signed certificate. In fact, you can probably generate self-signed certificates for testing purposes before installing production certificates. The fact that a self-signed certificate contains a valid public key does not mean that the issuer is a trusted certificate authority. To ensure that self-signed certificates are generated by trusted CAs, such certificates must be distributed by secure means; for example, hand-delivered on floppy disks, downloaded from secure sites, and so on.

Applications that use certificates store these certificates in a keystore file. This file typically contains the necessary personal certificates, its signing certificates, and its private key. The private key is used by the application to create digital signatures. Servers always have personal certificates in their keystore files. A client requires a personal certificate only if the client must authenticate to the server when mutual authentication is enabled.

To allow a client to authenticate to a server, a server keystore file contains the private key and the certificate of the server and the certificates of its CA. A client truststore file must contain the signer certificates of the CAs of each server to which the client must authenticate.

If mutual authentication is needed, the client keystore file must contain the client private key and certificate. The server truststore file requires a copy of the certificate of the client CA.

Digital signatures

A digital signature is a number attached to a document. For example, in an authentication system that uses public-key encryption, digital signatures are used to sign certificates.

This signature establishes the following information:

- The integrity of the message: Is the message intact? That is, has the message been modified between the time it was digitally signed and now?
- The identity of the signer of the message: Is the message authentic? That is, was the message actually signed by the user who claims to have signed it?

A digital signature is created in two steps. The first step distills the document into a large number. This number is the digest code or fingerprint. The digest code is then encrypted, which results in the digital signature. The digital signature is appended to the document from which the digest code is generated.

Several options are available for generating the digest code. This process is not encryption, but a sophisticated checksum. The message cannot regenerate from the resulting digest code. The crucial aspect of distilling the document to a number is that if the message changes, even in a trivial way, a different digest code results. When the recipient gets a message and verifies the digest code by recomputing it, any changes in the document result in a mismatch between the stated and the computed digest codes.

To stop someone from intercepting a message, changing it, recomputing the digest code, and retransmitting the modified message and code, you need a way to verify the digest code as well. To verify the digest code, reverse the use of the public and private keys. For private communication, it makes no sense to encrypt messages with your private key; these keys can be decrypted by anyone with your public key. This technique can be useful for proving that a message came from you. No one can create it because no one else has your private key. If some meaningful message results from decrypting a document by using someone's public key, the decryption process verifies that the holder of the corresponding private key did encrypt the message.

The second step in creating a digital signature takes advantage of this reverse application of public and private keys. After a digest code is computed for a document, the digest code is encrypted with the sender's private key. The result is the digital signature, which is attached to the end of the message.

When the message is received, the recipient follows these steps to verify the signature:

1. Recomputes the digest code for the message.
2. Decrypts the signature by using the sender's public key. This decryption yields the original digest code for the message.
3. Compares the original and recomputed digest codes. If these codes match, the message is both intact and authentic. If not, something has changed and the message is not to be trusted.

SSL authentication

SSL authentication is available for the Real-time nodes, the HTTP listener, and the WebSphere MQ Java Client.

SSL authentication for the Real-time nodes

SSL authentication in WebSphere Message Broker supports an authentication protocol known as *mutual challenge-response password authentication*. This support 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. Although this protocol is both secure and convenient to administer, you might prefer to use the industry standard SSL protocol exactly as defined, especially if a public key cryptography infrastructure is already deployed for other purposes. Two standardized versions of SSL are provided:

Asymmetric SSL
: This version is used by most Web browsers. In this protocol, 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 by 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 version uses a shared symmetric key for both participants, which is used to encrypt and decrypt messages. 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 incurs 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 85). 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 keyring 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 keyring 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, therefore keyring 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, keyring 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 keyring file. However, only the passphrase file must be secure in this way, and the keyring file is protected by the passphrase. Only private keys are sensitive. Other information in the keyring file, such as the broker's certificates, can be revealed without compromising security.

For more information on SSL authentication for the Real-time nodes, see “Enabling SSL for the Real-time nodes” on page 60.

SSL authentication for the HTTP listener

For information on SSL authentication for the HTTP listener, see “Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 63, and “Configuring an HTTPRequest node to use SSL (HTTPS)” on page 66.

SSL authentication for the JMS nodes

For information on SSL authentication for the JMS nodes, see “Configuring the broker to use SSL with JMS nodes” on page 58.

SSL authentication for the WebSphere MQ Java Client

For information on SSL authentication for the WebSphere MQ Java Client, see “Enabling SSL on the WebSphere MQ Java Client” on page 57.

Tunneling

When implementing WebSphere Message 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 Message 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 for Message Broker Toolkit users on Windows

If you enable domain awareness, access to the Message Broker Toolkit is not restricted to users from one Windows® domain.

The Configuration Manager is always domain aware; when a Configuration Manager on Windows receives user information, it verifies the user's domain membership.

You can configure the Message Broker Toolkit to send domain aware information about the user to the Configuration Manager, if required. If the user is logged on to a Windows domain and the Message Broker Toolkit sends domain aware information, the Message Broker Toolkit sends the user's domain membership information and the user's ID. If the user is not logged on to a Windows domain, the Message Broker Toolkit sends only the user's ID and the computer name.

If you enable domain awareness, users from different Windows domains, who are either trusted by the primary domain, or in a Windows transitive trust relationship, can perform Message Broker 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 or object level security groups is not restricted to users from one domain.
- Nesting in user role definition or object level security groups is supported.

Planning for security when you install WebSphere Message Broker

The Installation Guide describes the security tasks that you must complete before, during, and after installation.

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

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

After installation, refer to “Creating user IDs” on page 46 for further security considerations.

For an introduction to various aspects of security, see “Security overview” on page 3.

Setting up message flow security

Set up security on a message flow to control access based on the identity of a message passing through the message flow.

You can configure the broker to perform end-to-end processing of an identity carried in a message through a message flow. Administrators can configure security at message flow level, controlling access based on the identity flowed in a message. This security mechanism is independent of both the transport and the message format.

To set up security for a message flow, perform the tasks described in the following topics:

1. “Creating a security profile”
2. “Configuring identity extraction in a message flow” on page 34
3. “Configuring identity authentication” on page 35
4. “Configuring identity mapping” on page 38
5. “Configuring authorization” on page 39
6. “Configuring for identity propagation” on page 43.

If the message flow is a Web service implemented using the broker SOAP nodes and the identity is to be taken from the WS-Security header tokens, you must also create appropriate Policy sets and bindings and then configure them on the relevant SOAP Nodes (in addition to the security profile). See Associating policy sets and bindings with message flows and nodes.

To work with a Username and Password identity, you need to configure the policy sets and bindings for Username token capabilities. To work with an X.509 Certificate identity, you need to configure them for X.509 certificate token capabilities. In the policy set binding, the Certificates mode of the X.509 certificate authentication token must be set as Trust Any (rather than Trust Store) so that the certificate is passed to the security provider defined by the security profile. Setting *Trust Store* causes the certificate to be validated in the local broker trust store. For more information, see Policy Sets and Policy Set Bindings editor: Authentication and Protection Tokens panel.

Creating a security profile

You can create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Tivoli Federated Identity Manager (TFIM) by using either the mqsicreateconfigurableservice command or an editor in the Broker Administration perspective of the Message Broker Toolkit.

Before you can enable security on a node or a flow, you need to create a security profile that defines the operations you want to perform.

You can create a security profile for use with either LDAP or TFIM to provide the required security enforcement and mapping. If you want to extract and propagate an identity without security enforcement or mapping, you can use the supplied security profile called Default Propagation. The Default Propagation profile is a predefined profile that requests only identity propagation. To create a security profile, see:

- "Creating a security profile for LDAP"
- "Creating a security profile for TFIM" on page 32.

Creating a security profile for LDAP

Create a security profile for use with Lightweight Directory Access Protocol (LDAP) by using either the mqsicreateconfigurableservice command or an editor in the Broker Administration perspective of the Message Broker Toolkit.

Before you start:

Ensure that you have an LDAP server that is LDAP Version 3 compliant, for example:

- IBM Tivoli Directory Server
- Microsoft® Active Directory
- OpenLDAP.

If your LDAP directory does not permit login by unrecognized user IDs, and does not grant search access rights on the subtree, you must also set up a separate authorized login ID that the broker can use for the search. For information on how to do this, see "Configuring authorization with LDAP" on page 39 or "Configuring authentication with LDAP" on page 36.

Creating a security profile using mqsicreateconfigurableservice:

You can use the mqsicreateconfigurableservice command to create a security profile that uses LDAP for authentication, authorization, or both. The security profile ensures that each message has an authenticated ID and is authorized for the message flow.

1. Open a command window that is configured for your environment.
2. Enter the mqsicreateconfigurableservice command on the command line. For example:

   ```
   mqsicreateconfigurableservice WBRK_BROKER -c SecurityProfiles -o LDAP
   -n authentication,authenticationConfig,authorization,authorizationConfig
   -v "LDAP,\"ldap://ldap.acme.com:389/ou=sales,o=acme.com\",LDAP,
   \"ldap://ldap.acme.com:389/cn=All Sales,ou=acmegroups,o=acme.com\""
   ```

   You must enclose the LDAP URL (which contains commas) with escaped double quotation marks (\" and \") so that the URL commas are not confused with the comma separator of the value parameter of mqsicreateconfigurableservice.

   If the LDAP URL includes an element name with a space, in this case `cn=All Sales`, the set of values after the -v flag must be enclosed by double quotes, (")

   For more information about the structure of the command, refer to the "mqsicreateconfigurableservice command" on page 532.

   You can define the security-specific parts of the command in the following way:

   a. Set the **authentication** to *LDAP*. This ensures that the incoming identity is validated.
   b. Set the **authenticationConfig** using the following syntax:

ldap[s]**://***server*[**:***port*]**/***baseDN*[**?[***uid_attr*]**[?[base|sub]]]**

For example:

```
ldap://ldap.acme.com:389/ou=sales,o=acme.com
ldaps://localhost:636/ou=sales,o=acme?cn?base
```

ldap: (Required) Fixed protocol string.

s: (Optional) Specifies whether SSL should be used. Default is not to use SSL.

server: (Required) The name or IP address of the LDAP server to contact.

port: (Optional) The port to connect to. Default is 389 (non-SSL). For LDAP servers with SSL enabled, the port is typically 636.

baseDN
(Required) String defining the base distinguished name (DN) of all users in the directory. If users exist in different subtrees, specify a common subtree under which a search on the username uniquely resolves to the required user entry, and set the *sub* attribute.

uid_attr:
(Optional) String defining the attribute to which the incoming username maps, typically uid, CN, or e-mail address. Default is *uid*.

base|sub:
(Optional) Defines whether to perform a base or subtree search. If base is defined, the authentication is faster because the DN of the user can be constructed from the *uid_att*, *username*, and *baseDN* values. If *sub* is selected, a search must be performed before the DN can be resolved. Default is *sub*.

c. Set the **authorization** to *LDAP*. This ensures that the incoming identity is checked for group membership in LDAP.

d. Set the **authorizationConfig** using the following syntax:

```
ldap[s]://server[:port]/groupDN[?[member_attr]
[?[base|sub][?[x-userBaseDN=baseDN,
x-uid_attr=uid_attr]]]]
```

For example:

```
ldap://ldap.acme.com:389/cn=All Sales,ou=acmegroups,
o=acme.com?uniquemember?sub?x-userBaseDN=ou=sales%2co=ibm.com,
x-uid_attr=emailaddress
```

ldap: (Required) Fixed protocol string

s: (Optional) Specifies whether SSL is used. Default is not to use SSL.

server: (Required) The name or IP address of the LDAP server to contact.

port: (Optional) The port to connect to. Default is 389 (non-SSL). For LDAP servers with SSL enabled, the port is typically 636.

groupDN
(Required) Fully defined distinguished name of the group in which users must be members to be granted access.

member_attr:
(Optional) The attribute of the group used to filter the search. Default is to look for both **member** and **uniquemember** attributes.

The following options are required only if authentication has not preceded the authorization, and if the authentication configuration string has not been specified. If the authentication configuration string has been specified,

the following parameters are ignored and those provided by the **baseDN**, **uid_attr**, and **[base|sub]** for authentication are used instead:

base|sub:
(Optional) Defines whether to perform a base or subtree search. If base is defined, the authentication is faster because the DN of the user can be constructed from uid_att + username + baseDN. If sub is selected, a search must be performed before the DN can be resolved. Default is sub.

baseDN
(Optional) String defining the base distinguished name of all users in the directory. Must be preceded by the string `x-userBaseDN`. Any commas in the BaseDN must be rendered as `%2c`.

x-uid_attr:
(Optional) String defining the attribute to which the incoming username should map, typically uid, CN, or email address. Default is uid. Must be preceded by the string `x-uid_attr`.

When you submit the command from a batch (.bat) file or command (.cmd) file, if the LDAP URL includes an extension with LDAP URL "percent hex hex" escaped characters (for example, a comma replaced by `%2c`, or a space replaced by `%20`), the percent signs must be escaped from the batch interpreter with an extra percent sign (`%%`). For example:

```
mqsicreateconfigurableservice WBRK_BROKER -c SecurityProfiles -o LDAP_URI_FUN
-n authentication,authenticationConfig,authorization,authorizationConfig -v
"LDAP,\"ldap://ldap.acme.com:389/ou=sales,o=acme.com?emailaddress?sub\",TRUE,
LDAP,\"ldap://ldap.acme.com:389/cn=All Sales,ou=acmegroups,
o=acme.com?report?base?x-BaseDN=ou=sales%%2co=acme.com,
x-uid_attr=emailaddress\""
```

The selected group must be defined on the LDAP server, and all of the required users must be members of the group.

3. If you need to reconfigure the security profile after it has been created, use the mqsichangeproperties command.

Creating a security profile using the Message Broker Toolkit:

You can use the Broker Administration perspective in the Message Broker Toolkit to create a security profile for LDAP.

1. In the workbench, switch to the Broker Administration perspective.
2. Right-click the broker in the Domains view. A menu is displayed.
3. Click **Open Security Profiles** in the menu. The Security Profiles window is displayed, containing a list of existing security profiles for the broker on the left, and a pane in which you can configure the profile on the right.
4. Click **Add** to create a new profile and add it to the list. You can edit the name of the security profile by highlighting it in the list and pressing **F2**.
5. Configure the security profile using the entry fields on the right side of the pane:
   a. Select the type of **Authentication** required. This can be LDAP, TFIM, or NONE.
   b. If you have selected LDAP for authentication, edit the following fields in the **LDAP Parameters** section:
      - LDAP Host
      - LDAP baseDN
      - LDAP uid attr

- LDAP search Scope

The values that you enter in the **LDAP Parameters** fields create a configuration string, which is displayed in the **Authentication Config** field. For information about the valid values for the parameters, see "Creating a security profile using mqsicreateconfigurableservice" on page 29.

c. Select the type of **Mapping** required. This can be either TFIM or NONE.

d. If you have selected TFIM for mapping, type the URL of the TFIM server in the **TFIM Configuration** field of the **TFIM Parameters** section.

The value that you specify in the **TFIM Configuration** field creates a configuration string, which is displayed in the **Mapping Config** field.

e. Select the type of **Authorization** required. This can be LDAP, TFIM, or NONE.

f. If you have selected LDAP for authorization, edit the following fields in the **LDAP Parameters** section:

- LDAP Host
- LDAP baseDN
- LDAP uid attr
- LDAP search Scope
- LDAP group baseDN
- LDAP group member.

The values that you enter in the **LDAP Parameters** fields create a configuration string, which is displayed in the **Authorization Config** field. For information about the valid values for the parameters, see "Creating a security profile using mqsicreateconfigurableservice" on page 29.

g. In the **Propagation** field, specify whether or not you require the identity to be propagated. The default is False.

h. In the **Password Value** field, select the way in which the password is displayed in the properties folder. The options are:

PLAIN
The password appears in the Properties folder as plain text.

OBFUSCATE
The password appears in the Properties folder as base64 encoding.

MASK
The password appears in the Properties folder as four asterisks (****).

6. Click **Finish** to deploy the security profile to the broker.

To delete an existing security profile, select the profile in the list and then click **Delete**.

Creating a security profile for TFIM

You can create a security profile for Tivoli Federated Identity Manager (TFIM) for any combination of the following functions: authentication, authorization, and mapping. You can use either the mqsicreateconfigurableservice command or an editor in the Broker Administration perspective of the Message Broker Toolkit to create the security profile.

Creating a profile using mqsicreateconfigurableservice:

To create a security profile that uses TFIM, you can use the mqsicreateconfigurableservice command by setting the configuration parameter to the URL of the TFIM server. For example: `http://tfimserver.mycompany.com:9080`

To create a security profile that uses TFIM for mapping, enter the following command:

```
mqsicreateconfigurableservice brokername -c SecurityProfiles -o profilename
-n mapping,mappingConfig -v TFIM,http://tfimserver.mycompany.com:9080
```

If the URL specifies an address beginning with `https://`, an SSL secured connection is used for requests to the TFIM server. For example, to create a security profile that uses an HTTPS connection to TFIM for mapping, enter the following command:

```
mqsicreateconfigurableservice brokername -c SecurityProfiles -o profilename
-n mapping,mappingConfig -v TFIM,https://tfimserver.mycompany.com:9443
```

where `https://tfimserver.mycompany.com:9443` is the address of the TFIM server. If TFIM is selected for more than one operation (for example, for authentication and mapping), the TFIM server URL must be identical for all the operations, and is therefore specified only once.
The following example creates a security profile that uses TFIM for authentication, mapping, and authorization:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c SecurityProfiles -o TFIM
-n authentication,mapping,authorization,propagation,mappingConfig
-v TFIM,TFIM,TFIM,TRUE,http://tfimhost1.ibm.com:9080
```

Creating a profile for TFIM using the Message Broker Toolkit:

You can use the Broker Administration perspective in the Message Broker Toolkit to create a security profile for using TFIM.

1. Right-click the broker in the Domains view of the Broker Administration perspective. A menu is displayed.
2. Click **Open Security Profiles** in the menu. The Security Profiles window is displayed, containing a list of existing security profiles for the broker on the left and, on the right, a pane in which you can configure the profile.
3. Click **Add** to create a new profile and add it to the list. You can edit the name of the security profile by highlighting it in the list and pressing **F2**.
4. Configure the security profile using the entry fields on the right side of the pane:
   a. Select the type of **Authentication**, **Mapping**, and **Authorization** required. If you select TFIM for any of these options, the **TFIM Configuration** field at the bottom of the pane is enabled.
   b. If you have selected TFIM for authentication, mapping or authorization, type the URL of the TFIM server into the **TFIM Configuration** field. The URL that you enter forms a configuration string, which is displayed in one or more of the configuration fields (**Authentication Config**, **Mapping Config**, and **Authorization Config**) depending on the entry fields that have TFIM selected.

      For more information about the valid values for the configuration parameter, see “Creating a profile using mqsicreateconfigurableservice” on page 32.
   c. In the **Propagation** field, specify whether you require the identity to be propagated. The default is False.

d. In the **Password Value** field, select the way in which the password is displayed in the properties folder. The options are:

PLAIN
The password is shown in the Properties folder as plain text.

OBFUSCATE
The password is shown in the Properties folder as base64 encoding.

MASK
The password is shown in the Properties folder as four asterisks (****).

5. Click **Finish** to deploy the security profile to the broker.

Configuring identity extraction in a message flow

Configure the broker to extract the identity from a message and pass it through the message flow.

Before you start:

Check that an appropriate security profile exists or create a new security profile. See "Creating a security profile" on page 28.

Input nodes can retrieve identity from the bitstream.

- An MQInput node retrieves the UserIdentifier element from the message descriptor (MQMD) and puts it into the Identity Source Token element of the Properties folder. At the same time, it sets the Identity Source Type element to username and the Identity Source Issued By element to MQMD.PutApplName (the put application name).
- An HTTPInput node retrieves the BasicAuth header from the HTTP request, decodes it, and puts it into the Identity Source Token and Password elements in the Properties folder. At the same time, it sets the Identity Source Type element to username + Password and the Identity Source Issued By element to the HTTP header UserAgent property.
- A SOAPInput node retrieves either the appropriate tokens as defined by the configured WS-Security policy sets and bindings, or (if they are not set) the underlying transport headers definition, and then populates the Identity Source fields in the Properties folder.

In some cases, the information extracted from the transport headers is not set or is insufficient to perform authentication or authorization. For example, for authentication to occur, a Username + Password type token is required; however, with WebSphere MQ, only a username is available, which means that the incoming identity has to be trusted. However, you can increase security by applying transport-level security using WebSphere MQ Extended Security Edition.

If the transport header cannot provide the required identity credentials, the information must be provided as part of the body of the incoming message. To enable the identity information to be taken from the body of the message, you must specify the location of the information by using the **Security** tab on the HTTP or MQ input nodes:

1. In **Identity Token Type**, specify the type of identity token that is in the message. The type can have one of the following values:
   - Transport Default
   - Username

- Username + Password
- X.509 Certificate

The default is *Transport Default*.

2. In **Identity Token Location**, specify the location in the message where the identity is specified. This string is in the form of an ESQL path or XPath expression, and must resolve to a token with the type Username, Username + Password, or X.509 Certificate.

   If you leave this option blank, the identity is retrieved from the transport header location. For example, for MQ the identity is retrieved from the MQMD.UserIdentifier transport header.
3. In **Identity Password Location**, enter the location in the message where the password is specified. This string is in the form of an ESQL path or XPath expression, and must resolve to a string containing a password. This option can be set only if the **Identity Token Type** is set to Username + Password.

   If you leave this option blank, the password is taken from the transport header if it is provided. For example, with Websphere MQ it is not set.
4. In **Identity IssuedBy Location**, specify a string or path expression to show where (in the message) information about the issuer of the identity is held. This string is in the form of an ESQL Path, XPath expression, or literal string defining where the identity was defined.

   If you leave this blank, the transport header value is used, if there is one. For example, for MQ the MQMD.PutApplName value is used.
5. (Optional) Ensure that all input nodes share the same information by promoting the properties to the message flow.

To enable the extraction of an identity in the message flow, select a security profile:

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile.
6. Save the BAR file.

Configuring identity authentication

You can configure a message flow to use either Lightweight Directory Access Protocol (LDAP) or Tivoli Federated Identity Manager (TFIM) for identity authentication.

Before you start:

Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 28.

For information about configuring authentication with LDAP or TFIM, see:

- “Configuring authentication with LDAP” on page 36
- “Configuring authentication with TFIM” on page 37.

Configuring authentication with LDAP

This topic describes how to configure a message flow to perform identity authentication using Lightweight Directory Access Protocol (LDAP).

Before you start:

Before you can configure a message flow to perform identity authentication using LDAP, you need to check that an appropriate security profile exists, or create a new security profile. See "Creating a security profile for LDAP" on page 29.

To authenticate the identity of a user or system, the broker attempts to connect to the LDAP server using the username and password associated with the identity. To do this, the broker needs the following information:

- To resolve the username to an LDAP entry, the broker needs to know the base distinguished name (base DN) of the accepted login IDs. This is required to enable the broker to differentiate between different entries with the same name.
- If the identities do not all have a common base DN, but can be uniquely resolved from a subtree, the DN can be specified in the broker configuration. When a subtree search has been specified, the broker must first connect to the LDAP server and search for the given username in order to obtain the full username distinguished name (DN) to be used for authentication. If your LDAP directory does not permit login of unrecognized IDs, and does not grant search access rights on the subtree, you must set up a separate authorized login ID that the broker can use for the search. Use the mqsisetdbparms command to specify a username and password. For example:

  ```
  mqsisetdbparms -n ldap::LDAP -u username -p password
  ```

  or

  ```
  mqsisetdbparms -n ldap::<servername> -u username -p password
  ```

  where <*servername*> is your base LDAP server name, for example, `ldap.mydomain.com`.

  If you specify `ldap::LDAP`, it creates a default setting for the broker, which the broker attempts to use if you have not explicitly used the mqsisetdbparms command to create a login ID for a specific <*servername*>. All servers that do not have an explicit `ldap::servername` entry then start using the credentials in the `ldap::LDAP` entry. This means that any servers that were previously using anonymous bind by default will start using the details in `ldap::LDAP`.

  The username that you specify in the **-u** parameter must be recognized by the LDAP server as a complete user name. In most cases this means that you need to specify the full DN of the user. Alternatively, by specifying a username to be anonymous, you can force the broker to bind anonymously to this LDAP server. This might be useful if you have specified a non-anonymous bind as your default (ldap::LDAP). For example:

  ```
  mqsisetdbparms -n ldap::<servername> -u anonymous -p password
  ```

  In this case, the value specified for *password* is ignored.

Steps for enabling LDAP authentication:

To enable an existing message flow to perform identity authentication, use the Broker Archive editor to select a security profile that uses LDAP for authentication. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses LDAP for authentication.
6. Save the BAR file.

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

If the message identity does not contain enough information for authentication, the information must be taken from the message body. For example, if a password is required for authentication but the message came from WebSphere MQ with only a username, the password information must be taken from the message body. For more information, see "Configuring identity extraction in a message flow" on page 34.

Configuring authentication with TFIM

This topic describes how to configure a message flow to perform identity authentication using Tivoli Federated Identity Manager (TFIM).

Before you start:

Before you can configure a message flow to perform identity authentication, you need to check that an appropriate security profile exists, or create a new security profile. See "Creating a security profile for TFIM" on page 32.

When you use TFIM for authentication, a request is made to the TFIM trust service with the following three parameters, which select the module chain:

- Issuer = Properties.IdentitySourceIssuedBy
- Applies To = The Fully Qualified Name of the Flow: <*Brokername*>.<*Execution Group Name*>.<*Message Flow Name*>
- Token = Properties.IdentitySourceToken

For more information about these parameters, see "Authentication, Mapping, and Authorization with TFIM and TAM" on page 14.

For further information about how to configure TFIM, see the IBM Tivoli Federated Identity Manager information center.

Steps for enabling TFIM authentication:

To enable an existing message flow to perform identity authentication, use the Broker Archive editor to select a security profile that uses TFIM for authentication. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.

2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses TFIM for authentication.
6. Save the BAR file.

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

If the message identity does not contain enough information for authentication, the information must be taken from the message body. For example, if a password is required for authentication but the message came from WebSphere MQ with only a username, the password information must be taken from the message body. For more information, see "Configuring identity extraction in a message flow" on page 34.

Configuring identity mapping

This topic explains how to configure a message flow to perform identity mapping.

Before you start:

Before you can configure a message flow to perform identity mapping, you need to check that an appropriate security profile exists, or create a new security profile. See "Creating a security profile" on page 28.

Configure Tivoli Federated Identity Manager (TFIM) to map the incoming security token and, if required, to authenticate and authorize it. The security token is the X.509 Certificate, Username, or Username + Password. To configure TFIM to map the incoming security token, you need to create a custom module chain in TFIM, which performs the security operations. The TFIM configuration controls the token type that is returned from the mapping.

When you use TFIM for mapping, a request is made to the TFIM trust service with the following three parameters, which select the module chain:

- Issuer = Properties.IdentitySourceIssuedBy
- Applies To = The Fully Qualified Name of the Flow: <*Brokername*>.<*Execution Group Name*>.<*Message Flow Name*>
- Token = Properties.IdentitySourceToken

For information on how to configure TFIM, see the IBM Tivoli Federated Identity Manager information center.

Follow these steps to enable an existing message flow to perform identity mapping.

Using the Broker Archive editor, select a security profile that has mapping enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, enter the name of a security profile that has mapping enabled.
6. Save the BAR file.

Configuring authorization

You can configure the broker to use either Lightweight Directory Access Protocol (LDAP) or Tivoli Federated Identity Manager (TFIM) to authorize an identity in a message flow.

Before you start:

Check that an appropriate security profile exists, or create a new security profile. See "Creating a security profile" on page 28.

For information about configuring authorization for LDAP or TFIM, see:

- "Configuring authorization with LDAP"
- "Configuring authorization with TFIM" on page 40.

Configuring authorization with LDAP

This topic describes how to configure a message flow to perform authorization on an identity using Lightweight Directory Access Protocol (LDAP).

Before you start:

Before you can configure a message flow to perform authorization, you need to check that an appropriate security profile exists, or create a new security profile. See "Creating a security profile for LDAP" on page 29.

When LDAP is used for authorization, the broker needs to determine whether the incoming username is a member of the given group. To do this, the broker requires the following information:

- To resolve the username to an LDAP entry, the broker needs to know the base distinguished name (Base DN) of the accepted login IDs. This is required to enable the broker to differentiate between different entries with the same name.
- To get an entry list from a group name, the group name must be the distinguished name of the group, not just a common name. An LDAP search is made for the group, and the username is checked by finding an entry matching the distinguished name of the user.
- If your LDAP directory does not permit login by unrecognized IDs, and does not grant search access rights on the subtree, you must set up a separate authorized login ID that the broker can use for the search. Use the mqsisetdbparms command to specify a username and password:

  ```
  mqsisetdbparms -n ldap::LDAP -u username -p password
  ```

  or

  ```
  mqsisetdbparms -n ldap::<servername> -u username -p password
  ```

where <*servername*> is your base LDAP server name. For example: `ldap.mydomain.com`.

If you specify `ldap::LDAP`, it creates a default setting for the broker, which the broker attempts to use if you have not explicitly used the mqsisetdbparms command to create a login ID for a specific <*servername*>. All servers that do not have an explicit `ldap::servername` entry then start using the credentials in the `ldap::LDAP` entry. This means that any servers that were previously using anonymous bind by default will start using the details in `ldap::LDAP`.

The username that you specify in the **-u** parameter must be recognized by the LDAP server as a complete user name. In most cases this means that you need to specify the full DN of the user. Alternatively, by specifying a username to be anonymous, you can force the broker to bind anonymously to this LDAP server. This might be useful if you have specified a non-anonymous bind as your default (ldap::LDAP). For example:

```
mqsisetdbparms -n ldap::<servername> -u anonymous -p password
```

In this case, the value specified for *password* is ignored.

Steps for enabling LDAP authorization:

To enable an existing message flow to perform authorization using LDAP, use the Broker Archive editor to select a security profile that has authorization enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that uses LDAP for authorization.
6. Save the BAR file.

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

Configuring authorization with TFIM

This topic describes the steps involved in configuring a message flow to perform authorization on an identity using Tivoli Federated Identity Manager (TFIM).

Before you start:

Before you configure a message flow to perform authorization with TFIM:

- Check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile for TFIM” on page 32.
- Define the required users and groups in TFIM.

The broker security manager issues an authorization request to the TFIM trust service with the following three parameters, which select the TFIM module chain to be used:

- Issuer = Properties.IdentitySourceIssuedBy
- Applies To = The Fully Qualified Name of the Flow: <*Brokername*>.<*Execution Group Name*>.<*Message Flow Name*>
- Token = Properties.IdentitySourceToken

Authorization is performed with TFIM using an instance of the TFIM AuthorizationSTSModule in the selected module chain. The TFIM AuthorizationSTSModule must be set with `Mode = Other`. This AuthorizationSTSModule authorizes a user by checking an Access Control List (ACL) from Tivoli Access Manager (TAM). TFIM performs the authorization check by verifying that the action ″i″ (invoke) has been granted in an ACL for the `WebService` action group.

The ACL is found starting from the root of the TAM object space using a path formed from the Authorization module **Web service protected object name** parameter, followed by the **Port Type** and **Operation Name** from the authorization request. When the broker makes an authorization request to TFIM, the **Port Type** and **Operation Name** parameters have the following values:

- PortType:<*Message flow name*>
- Operation ″MessageFlowAccess″

Therefore, the ACL is found at this location in the TAM object space:

```
/<WSProtectedObjectName>.<MessageFlowName>."MessageFlowAccess"
```

For more information about this process and the parameters, see “Authentication, Mapping, and Authorization with TFIM and TAM” on page 14.

Steps for enabling TFIM authorization:

To enable an existing message flow to perform authorization with TFIM, use the Broker Archive editor to select a security profile that has authorization enabled. You can set a security profile on a message flow or on individual input nodes. If no security profile is set for the input nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that has authorization enabled.
6. Save the BAR file.

For a SOAPInput node to use the identity in the WS-Security header (rather than an underlying transport identity) an appropriate policy set and bindings must also be defined and specified. For more information, see Policy sets.

In addition to configuring Message Broker to perform authorization with TFIM, you must configure TFIM and TAM. For information about how to do this, see the following topics:

- "Creating a module chain in TFIM"
- "Configuring TAM."

For further information on how to configure TFIM, see the IBM Tivoli Federated Identity Manager information center.

Creating a module chain in TFIM:

This topic describes how to create a module chain in Tivoli Federated Identity Manager (TFIM).

To enable Message Broker to use TFIM for authorization, you need to configure TFIM to process the security request from the message flow. To do this you need to create a module chain in TFIM to handle the request:

1. Create a *Custom* module chain, and ensure that the chain performs all the actions required (Authenticate, Map, Authorize).
2. Set the *Issuer* and *AppliesTo* properties of the module chain, so that it is invoked for the requests from the message flow. When the broker makes a request to TFIM, the **Port Type** and **Operation Name** parameters have the following values:
   - PortType:<*Message flow name*>
   - Operation "MessageFlowAccess"

   The *RequestType* is always set to *Validate*.
3. To perform authorization in a module chain, add an instance of the Authorization module in *other* mode, which allows the module parameter **Web Service protected object name** to be set for the Tivoli Access Manager (TAM) configuration.

When you have created the module chain in TFIM, see "Configuring TAM" for information on how to configure TAM to process authorization requests from TFIM.

Configuring TAM:

This topic describes how to configure Tivoli Access Manager (TAM) to enable authorization using Tivoli Federated Identity Manager (TFIM).

To configure TAM to process an authorization request from TFIM, complete the following steps. The examples relate to the TAM Version 6.01 pdadmin utility:

1. Check that the **action group** used by the TFIM authorization module is available. The action group used is *WebService*:

   ```
   action group list
   ```

   If *WebService* is not listed, create it:

   ```
   action group create WebService
   ```
2. Display the **action** in the action group used by the TFIM authorization module. The action used is "*i*":

   ```
   action list WebService
   ```

   If action "*i*" <*label*> *0* is not listed, create it. The value of <*label*> can vary:

   ```
   action create i <label> 0 WebService
   ```

3. Create the Access Control List (ACL) that will be used to grant access to one or more message flows. First, create the ACL and give the administrators access to it. In this example, `iv-admin` is the administration group and `sec_master` is the main administrator:

   ```
   acl create <AclName>
   acl modify <AclName> set Group iv-admin TcmdbsvaBRxl[WebService]i
   acl modify <AclName> set User sec_master TcmdbsvaBRxl[WebService]i
   ```

4. Grant access to all authenticated users, or specific groups, by adding them to the ACL. Grant any authenticated identity access:

   ```
   acl modify <AclName> set Any-other Trx[WebService]i
   ```

   To add a specific group:

   ```
   acl modify <AclName> set group <GroupName> Trx[WebService]i
   ```

5. Define protected object spaces in TAM for authorization of message flows:
   a. Create the *application container object* as the root of the protected object space. This is the name that is used to link an instance of a TFIM AuthorizationSTSModule (within a module chain) into the TAM object space. The container object name is specified to match the **Web Service protected object name** parameter on a TFIM Authorization module.

      ```
      objectspace create /<ContainerObjectName> <Description> 14
      ```

   b. Create the container objects in the tree for each broker message flow that is being authorized. The message flow name is used by TFIM to locate a point in the TAM Object Space tree for Authorization, through the attached ACL. The message flow name is passed as the **PortType** in the WS-Trust request to TFIM. Use the following command to create the object tree node representing each flow to be authorized:

```
object create /<ContainerObjectName>/<FlowName> <Description> 11 ispolicyattachable yes
```

      The **ispolicyattachable** parameter applies to all levels, so you can attach an ACL at any level.

   c. Create the leaf object that represents the authorized object to grant access to the message flow. This is the fixed string *MessageFlowAccess*, which the broker sends to TFIM through the TFIM **OperationName** extension to the WS-Trust request. A fixed name (MessageFlowAccess) is used instead of a true operation name, because the broker does not necessarily know at the input node which operation a flow is going to perform. The command syntax is:

```
object create /<ContainerObjectName>/<FlowName>/MessageFlowAccess <Description> 12 ispolicyattachable yes
```

      where *<FlowName>* has been created in a previous step.

6. Attach the ACL to the relevant node in the protected object space tree. Each node in the object space inherits ACLs from its parent, and a lower level ACL can override a higher level one. Use the following command syntax to attach an ACL to a node in the object space:

   ```
   acl attach /<ObjectSpacePath> <AclName>
   ```

   To attach an ACL to the leaf node:

   ```
   acl attach /<ContainerObjectName>/<FlowName>/MessageFlowAccess <AclName>
   ```

For further information about configuring TAM, see the IBM Tivoli Access Manager information center.

Configuring for identity propagation

This topic describes the steps involved in configuring a message flow to propagate a message identity.

Before you start:

Before you can configure a message flow to perform identity propagation, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 28.

To enable a message flow to perform identity propagation the input nodes must extract the identity and the output node must propagate it. An input node extracts security tokens if it is configured with a security profile at deployment time. An output node propagates an identity if it is configured with a security profile that enables propagation at deployment time.

To enable a message flow to perform identity propagation:

Using the Broker Archive editor, select a security profile that has identity propagation enabled. You can set a security profile on a message flow or on individual input and output nodes. If no security profile is set for the input and output nodes, the setting is inherited from the setting on the message flow.

1. Switch to the Broker Application Development perspective.
2. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
3. Click the **Manage and Configure** tab.
4. Click the flow or node on which you want to set the security profile. The properties that you can configure for the message flow or for the node are displayed in the **Properties** view.
5. In the **Security Profile Name** field, select a security profile that has identity propagation enabled.
6. Save the BAR file.

For a SOAPRequest or SOAPAsyncRequest node, you can define an appropriate policy set and bindings to specify how the propagated identity is placed in the WS-Security header (rather than the underlying transport headers). For more information, see Policy sets.

If the message identity does not contain enough information for identity propagation, you can use any of the following methods to acquire the necessary information:

- Take the information from the message body. For example, if the message comes from WebSphere MQ with only a username token, and the output is an HTTP request node requiring a Username + Password token, the password might be present in the body of the incoming message. For more information, see “Configuring identity extraction in a message flow” on page 34.
- Configure an identity mapper using TFIM. For more information, see the IBM Tivoli Federated Identity Manager information center.
- Use ESQL or Java to set the Mapped Identity fields in the Properties tree.

Diagnosing security problems

This topic explains how to find out why access to a secured flow has been denied.

By default, security exceptions are not processed in the same way as other errors (see “Security exception processing” on page 17). Security exceptions are not logged to the system event log, to prevent a security denial of service attack filling the logs and destabilizing the system.

This means that, by default, you cannot diagnose security exceptions in the same way as other errors. To see what might be causing the security exceptions, you can do either of the following things:

- Select the **Treat Security Exceptions as normal exceptions** property on the input nodes.
- Use the user trace.

The following steps show you how to use the user trace to find out why access to a secured message flow has been denied:

1. Use the mqsireloadsecurity command to clear the security cache, so that the traced request goes to the security provider rather than using a result held in the cache. This ensures that the reason codes returned from the security provider are displayed in the traced exception.
2. Enable user trace for the message flow, using either the workbench or the mqsichangetrace command (see Starting user trace for more information).
3. Resend the request that has been rejected by the security provider.
4. Stop the user trace, using either the workbench or the mqsichangetrace command.
5. Use the mqsireadlog command to examine the trace information that was recorded by the user trace. This trace information contains the error codes provided by the broker and the security provider.

Setting up broker domain security

You must consider several security aspects when you are setting up a broker domain that includes brokers running on Windows, Linux, or UNIX platforms.

For an introduction to various aspects of security, see “Security overview” on page 3.

This section does not apply to z/OS. Refer to “Setting up z/OS security” on page 73 and “Summary of required access (z/OS)” on page 673 for information about setting up broker domain security on z/OS.

Before you start setting up security for your broker domain, refer to “Planning for security when you install WebSphere Message Broker” on page 27, which contains links to security information that you need before, during, and after installation of WebSphere Message Broker.

Use the following list of tasks as a security checklist. Each item comprises a list of reminders or questions about the security tasks to consider for your broker domain. The answers to the questions provide the security information that you require to configure your domain components , and also give you information about other security controls that you might want to employ.

- “Creating user IDs” on page 46
- “Considering security for a broker” on page 49
- “Considering security for a Configuration Manager” on page 51
- “Considering security for the workbench” on page 46
- “Configuring security for domain components” on page 54
- “Enabling topic-based security” on page 69
- “Using security exits” on page 71
- “Database security” on page 72

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 process broker commands" on page 49
  - "Deciding which user accounts can execute User Name Server commands" on page 70
  - "Deciding which user accounts can process Configuration Manager commands" on page 52
- Service user IDs under which components run. Refer to:
  - "Deciding which user account to use for the broker service ID" on page 50
  - "Deciding which user account to use for the User Name Server service ID" on page 70
  - "Deciding which user account to use for the Configuration Manager service ID" on page 52

  On all platforms, 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 broker and user databases" on page 122 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 69.

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 and message sets 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

Set up appropriate levels of security for the workbench.

When the workbench is started, domain information is sent with the user ID of the workbench user to the Configuration Manager. You can choose whether to check that information or ignore it when a user is being verified.

When you create an access control list (ACL), you can use either the **-a** or the **-m** parameter of the mqsicreateaclentry command to specify whether the domain information is to be checked as part of the user verification. If you specify the **-a** parameter, the specified user name can connect from any computer. If you specify the **-m** parameter, you allow the user to connect from only the specified computer.

Consider the following factors when you are setting up security for the workbench:

1. "Checking domain information"
2. "Ignoring domain information" on page 48
3. "Securing the channel between the workbench and the Configuration Manager" on page 48

For the highest level of security, check the domain information for each user and configure security for the connection between the Configuration Manager and the workbench.

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

Checking domain information

Increase security by specifying that the domain information that is sent with the user name to the Configuration Manager is checked as part of the user verification.

For example, 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. Complete the following steps:

1. Add any domain users or groups to the local group names that you use in your ACLs.
2. When you create the Configuration Manager, specify the **-m** parameter on the mqsicreateaclentry command to ensure that the domain is considered when verifying the user. The **-m** parameter allows the user to connect from only the specified computer.

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 you are trying to deploy message flows and message sets to the broker. To avoid this problem:

- 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 user ID on the computer where the broker is running.
- Ensure that all user 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 48.

Ignoring domain information

If you choose not to check the domain information associated with the user name, security is reduced.

You can allow users to be verified without checking the domain information by specifying the **-a** parameter of the mqsicreateaclentry command. This parameter allows the specified user name to connect from any computer.

If your workbench users are on a local domain, add them to the local groups that you use in your ACLs, and then follow the steps in "Securing the channel between the workbench and the Configuration Manager."

If your users are from another domain, make the other domain a trusted domain of the computer on which the Configuration Manager is running, then add the groups and users from the trusted domain to the local groups of the Configuration Manager.

Securing the channel between the workbench and the Configuration Manager

If you want to secure the connection, you must update the configuration of the SVRCONN channel between the Configuration Manager and the workbench to include the security options that you want.

When you create the Configuration Manager, a default SVRCONN channel, SYSTEM.BKR.CONFIG, is created; you can use this channel, or create a new one. If you use a different channel, you must set the new name in the connection properties.

Implement one or both of the following options:

- Implement SSL security on the channel. You must have the appropriate software to manage SSL certificate stores; for example, you can install either the WebSphere MQ Client or the Server, and use the IBM Key Management tools for the client (workbench). You can use either JKS or PKCS12 stores.
  - Use WebSphere MQ Explorer or the runmqsc command to update the SVRCONN definition to specify the required value in the SSLCIPH attribute.
  - In the workbench, switch to the Broker Administration perspective.
    - Right-click the domain connection and click **Properties**.
    - Select the cipher suite that matches the value you set for SSLCIPH.
    - Enter the full path and name for the Key Store and Trust Store, or click **Browse** to search for them.
  - Add the queue manager certificate to the workbench truststore.
  - If you want server-only (one way) certification, set the SVRCONN channel attribute SSLCAUTH to OPTIONAL.
  - If you want mutual (two way) certification:
    - Set the SVRCONN channel attribute SSLCAUTH to REQUIRED.
    - Add the client (workbench) certificate to the queue manager's truststore.
  - Start the domain connection in the workbench.

  For further details about setting up SSL configuration, see "Enabling SSL on the WebSphere MQ Java Client" on page 57.

  For more information about setting up connections secured with SSL, see the WebSphere MQ Java Client developerWorks® article.

- Create and enable a pair of security exits to run at the workbench and Configuration Manager ends of the SVRCONN channel that connects the two components. Program these exits to verify workbench users with the security manager on the computer on which the Configuration Manager is running.

  For more information about creating and enabling security exits, refer to "Security exits" on page 20.

Considering security for a broker

Several factors must be considered when you are deciding which users can execute broker commands and which users can control security for other broker resources.

When you are deciding which users are to perform the different tasks, consider the following steps:

1. "Deciding which user accounts can process broker commands"
2. "Deciding which user account to use for the broker service ID" on page 50
3. "Setting security on the broker's queues" on page 51
4. "Enabling topic-based security in the broker" on page 51
5. "Securing the broker registry" on page 51

Deciding which user accounts can process broker commands

Decide which 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: Ensure that your user ID has the following characteristics:
      - It is a member of the mqbrkrs group
      - If it will be used to create or delete a broker, it must be a member of the mqm group

      Go to "Deciding which user account to use for the broker service ID" on page 50.
2. Are you processing 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.

      Ensure that your user ID has the following characteristics:
      - It is a member of the mqbrkrs group.
      - If it will be used to create a broker, the user ID must be defined in your local domain.
      - If it will be used to create or start a broker, the user ID must be a member of the Administrators group (for example, `WKSTN1\Administrators`).
      - If it will be used to create or delete a broker, the user ID must be a member of the mqm group.

      Go to "Deciding which user account to use for the broker service ID" on page 50.
3. Are you processing 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.

   Ensure that the user ID has the following characteristics:

   - It is a member of the mqbrkrs group.
   - If it will be used to create a broker, the user ID must be defined in your local domain.
   - If it will be used to create or start a broker, the user ID must be a member of the Administrators group. For example, if you create a broker using `DOMAIN1\user1`, ensure that `DOMAIN1\user1` is a member of `WKSTN1\Administrators`.
   - If it will be used to create or delete a broker, the user ID must be a member of the mqm group.

   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 run the mqsistart command with a user ID that is a member of the mqm and mqbrkrs groups, the user ID under which you run the mqsistart command becomes the user ID under which the broker component process will run.

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: Ensure that the user ID is a member of the mqbrkrs group.

      Go to "Setting security on the broker's queues" on page 51.
2. Do you have any existing brokers running on this Windows system?
   a. No: You can choose a service ID for the broker. Go to the next question.
   b. Yes: On the Windows platform, all brokers must run with the same service ID. Use your existing service ID when you create the new broker.
3. 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 has the following characteristics:
      - It is defined in your local domain.
      - It is a member of the mqbrkrs group.
      - It has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel → Performance and maintenance → Administrative Tools → Local Security Policy**.

      Go to "Setting security on the broker's queues" on page 51.
4. 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:
      - Your user ID has been granted the Logon as a service privilege (from the Local Security Policy).
      - DOMAIN1\user1 is a member of DOMAIN1\Domain mqbrkrs.
      - DOMAIN1\Domain mqbrkrs is a member of WKSTN1\mqbrkrs.

- The user ID has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel → Performance and maintenance → Administrative Tools → Local Security Policy**.

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

Setting security on the broker's queues

When you run the mqsicreatebroker command, the local mqbrkrs group is granted access to internal queues whose names begin with the characters SYSTEM.BROKER. Do no change this ACL, because it is required for the broker to function correctly.

The Configuration Manager that controls the broker puts messages to SYSTEM.BROKER.ADMIN.QUEUE. If your Configuration Manager is on the same computer as your broker, its service ID is in the mqbrkrs group, therefore no further action is required. If the Configuration Manager is on a different computer, ensure that its service ID is defined to the computer that is running the broker, and ensure that it has WebSphere MQ access to put messages to SYSTEM.BROKER.ADMIN.QUEUE.

If you use collectives for publish/subscribe, other brokers in your domain must put messages to SYSTEM.BROKER.INTERBROKER.QUEUE. Therefore their service IDs require authority to put messages to that queue.

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 69.
2. No: Go to "Considering security for a Configuration Manager."

Securing the broker registry

Broker operation depends on the information in the broker registry, which you must secure to guard against accidental corruption. The broker registry is stored in the Windows registry or the Linux or UNIX file system. Set your security options so that only user IDs that are members of the group mqbrkrs can read from or write to *brokername*/CurrentVersion and all sub-keys.

Considering security for a Configuration Manager

Determine the security characteristics and group membership required for user IDs to perform tasks associated with the Configuration Manager.

Consider the characteristics and group membership required for user IDs that perform the following functions:

- Run as a service user ID (running the Configuration Manager)
- Process Configuration Manager commands
- Access the Configuration Manager 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.

Read the appropriate sections in this list:

1. "Deciding which user accounts can process 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 53
4. "Running the Configuration Manager in a domain environment" on page 54

Deciding which user accounts can process 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. Is your Configuration Manager running on a Linux or UNIX operating system?
   a. No: Go to the next question.
   b. Yes: Ensure that your user ID is a member of the mqbrkrs group.

      Go to "Deciding which user account to use for the Configuration Manager service ID"
2. Are you running 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"
3. Are you running 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 run the mqsistart command with a user ID that is a member of the mqm and mqbrkrs groups, the user ID under which you run the mqsistart command becomes the user ID under which the Configuration Manager component process will run.

Answer the following questions:

1. Is your Configuration Manager running on a Linux or UNIX operating system?

a. No: Go to the next question.
b. Yes: Ensure that your user ID is a member of the mqbrkrs group.

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

2. 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 has the following characteristics:
      - It is defined in your local domain
      - It is a member of the mqbrkrs group
      - It is a member of the mqm group
      - It is a member of the Administrators group
      - It has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**.

      Go to "Setting security on the Configuration Manager's queues."
3. 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:
      1) user1 is defined in DOMAIN1
      2) DOMAIN1\user1 is a member of DOMAIN1\Domain mqbrkrs
      3) DOMAIN1\user1 is a member of WKSTN1\mqm
      4) DOMAIN1\Domain mqbrkrs is a member of WKSTN1\mqbrkrs
      5) DOMAIN1\user1 is a member of WKSTN1\Administrators
      6) The user ID (user1) has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**.

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

Setting security on the Configuration Manager's queues

When you run the mqsicreateconfigmgr command, the **mqbrkrs** group is granted 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.

Brokers and User Name Servers communicate with the Configuration Manager through these queues. If they run on the same computer as the Configuration Manager, you do not need to do anything else to enable them to communicate. However, if they are running on a different computer, you must ensure that their service ID exists on the computer that is running the Configuration Manager, and either add that user account to the mqbrkrs group or grant it explicit MQ access to put messages to the Configuration Manager's queues.

Administrators using either commands or the Toolkit need the authority to put messages to the Configuration Manager's queues. You can use the mqsicreateaclentry command to create the required access to WebSphere MQ.

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 "Ignoring domain information" on page 48.
- If you want to *disable* domain awareness, go to "Checking domain information" on page 47.

Configuring security for domain components

Configure access control lists (ACLs) to control access to runtime resources.

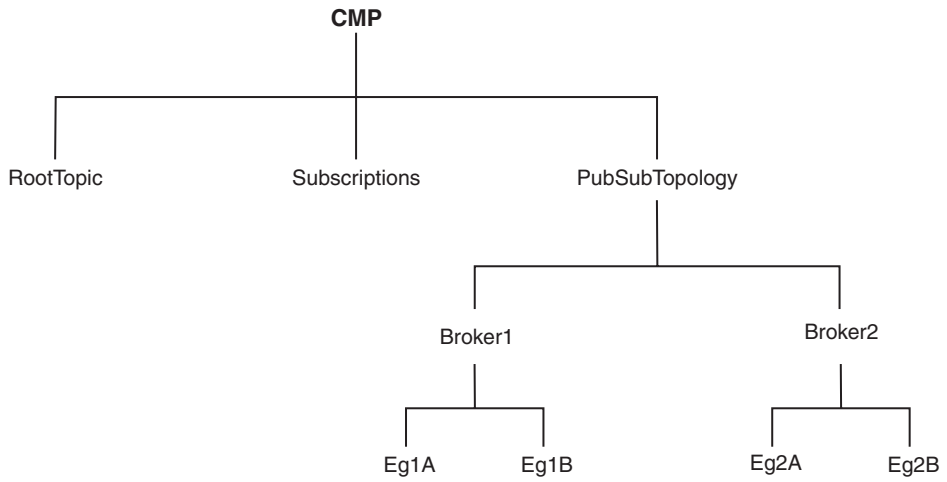
When you have created the runtime resources (for example, the brokers and execution groups) and secured the transport connection, you must configure access control lists (ACLs) to control which objects can be accessed by which user IDs. Default access, which you have until you have configured the ACLs, is Full control access for the Configuration Manager service ID only.

To configure your ACLs:

1. Decide which objects you want to control, by referring to the hierarchy of ACLs that can be defined.
2. Use the mqsicreateaclentry command to define permissions for each object that requires non-default access.
3. Optional: If you are a publish/subscribe user, you can define other ACLs for application-level access to publish and subscribe on specified topics. These are also controlled by the mqsicreateaclentry command, but are not required for administration.

When the ACLs have been configured, users of the workbench and commands are able to work with objects associated with the domain. The control that the users have over the objects (Full control, View, Deploy, or Edit) depends on the access that you have granted to them in the access control list entries.

The following diagram shows an example hierarchy of access control list entries:



The following examples show how this hierarchy works in practice.

Example 1

UserA has no access control entries. Therefore, UserA cannot manipulate any objects in the hierarchy, or see any of the objects defined in it.

Example 2

UserB has an ACL entry that gives Deploy authority to the execution group Eg1A. This entry gives UserB implied View authority to PubSubTopology and Broker1. UserB must be able to view PubSubTopology and Broker1 (for example, in the Message Broker Toolkit) to be able to deploy to Eg1A.

Because UserB does not have any ACL entries for PubSubTopology or Broker1, UserB does not inherit access to the other broker or execution groups in the hierarchy. In practice, this means that UserB can see that there is another execution group defined on the broker Broker1 but cannot see any details (including the name of the execution group). Similarly, UserB can see that another broker exists within the topology, but cannot see any details. UserB has no access to RootTopic or to Subscriptions (the subscriptions table).

The following command creates the ACL entry for UserB:

```
mqsicreateaclentry testcm -u UserB -a -x D -b Broker1 -e Eg1A
```

The mqsilistaclentry command then displays the following information:

```
BIP1778I: userb -USER  -  D  -  Broker1/Eg1A      -  ExecutionGroup
```

Example 3

UserC has an ACL entry that gives View authority for the Configuration Manager Proxy (CMP), and an ACL entry that gives Full authority for Broker1. These entries give UserC the following authorities:

CMP View
RootTopic View
Subs View

Topology View
Broker1 Full
Eg1A Full
Eg1B Full
Broker2 View
Eg2A View
Eg2B View

The following commands create the ACL entries for UserC:

```
mqsicreateaclentry testcm -u UserC -a -x V -p
mqsicreateaclentry testcm -u UserC -a -x F -b Broker1
```

The mqsilistaclentry command then displays the following information:

```
BIP1778I: userc - USER - V - ConfigManagerProxy - ConfigManagerProxy
BIP1778I: userc - USER - F - Broker1 - Broker
```

Example 4

UserD has an ACL entry that gives Full control authority for the CMP API, and an ACL entry that gives View authority for Broker1. The View authority to access Broker1 means that UserD does not inherit Full control authority for Broker1. This use of View ACL entries is useful because it allows users who usually have full control over a given object to reduce their access temporarily, to prevent accidental deletion or deployment. If users need full control of the object, removing the View entry restores Full control authority, so that they can perform the operations that they need, and then restore the View entry. UserD has the following authorities:

CMP Full
RootTopic Full
Subs Full
Topology Full
Broker1 View
Eg1A View
Eg1B View
Broker2 Full
Eg2A Full
Eg2B Full

The following commands create the ACL entries for UserD:

```
mqsicreateaclentry testcm -u UserD -a -x F -p
mqsicreateaclentry testcm -u UserD -a -x V -b Broker1
```

The mqsilistaclentry command then displays the following information:

```
BIP1778I: userd - USER - F - ConfigManagerProxy - ConfigManagerProxy
BIP1778I: userd - USER - V - Broker1 - Broker
```

The following command can then delete the ACL entries for UserD:

```
mqsideleteaclentry testcm -u UserD -a -b Broker1
```

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 499.

Implementing SSL authentication

The following topics contain instructions for implementing SSL authentication:

- "Enabling SSL on the WebSphere MQ Java Client"
- "Configuring the broker to use SSL with JMS nodes" on page 58
- "Enabling SSL for the Real-time nodes" on page 60
- "Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)" on page 63
- "Configuring an HTTPRequest node to use SSL (HTTPS)" on page 66

Enabling SSL on the WebSphere MQ Java Client

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between an application and the queue manager. Configure SSL support for connections between applications that use the CMP API (including the Message Broker Toolkit) and a Configuration Manager.

For one-way authentication, when the client CMP application authenticates the Configuration Manager, perform the following steps:

1. Generate or obtain all the appropriate keys and certificates. You must include 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 JSSE truststore of the Java Virtual Machine (JVM) at the CMP application end using a tool such as Keytool.
4. Decide which cipher suite to use and change the properties on the server-connection channel by using WebSphere MQ Explorer, to specify the cipher suite to be used. This channel has a default name of `SYSTEM.BKR.CONFIG`; this name is used unless you have specified a different name on the Create a Domain Connection panel or Domain Properties panel ; see "Creating a domain connection" on page 251 and "Modifying domain connection properties" on page 253.
5. Add the required parameters (cipher suite, for example) to the CMP application. 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 CMP application connects to the Configuration Manager if it has a valid signed key that has been signed by a trusted certificate authority.

For two-way authentication, when the Configuration Manager also authenticates the CMP application), perform the following additional steps:

1. Generate or obtain all the appropriate keys and certificates. You must include 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 by using the standard WebSphere MQ facilities.
3. Set the server-connection channel to always authenticate. Specify SSLCAUTH(REQUIRED) in runmqsc, or in WebSphere MQ Explorer.

4. Add the pkcs12 certificate to the JSSE keystore of the JVM at the CMP application end by using a tool such as Keytool.
5. If you are not using the default keystore, its full path must be passed into the CMP through the keystore parameter

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

You can make further restrictions by 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 CMP applications and the Configuration Manager; see "Using security exits" on page 71.

Configuring the broker to use SSL with JMS nodes

Configure your broker to work with a JMS provider that supports JMS clients that can connect by using the Secure Sockets Layer (SSL) protocol.

The JMS 1.1 Specification states that JMS does not provide features for controlling or configuring message integrity or message privacy. JMS providers typically support these additional features, and provide their own administration tools to configure these services. Clients can get the proper security configuration as part of the administered objects they use.

If you want to apply SSL security to the JMS connections created by the three built-in nodes JMSInput, JMSOutput, and JMSReply, check the documentation supplied by your chosen JMS provider. The configuration of the JNDI administered objects that are used by the JMS nodes is specific to each JMS provider.

The three built-in nodes JMSInput, JMSOutput, and JMSReply are referred to in this topic by the generic term JMS nodes; apply the information and instructions here to the specific type of node that you are using.

One example of a JMS provider that provides SSL support for connecting JMS clients is TIBCO Enterprise Message Service (EMS). The following sections describe the authentication model used for JMS nodes, with specific reference to TIBCO EMS, and provide information about how to connect JMS nodes to a TIBCO EMS JMS Server securely by using SSL:

1. "SSL authentication model for the JMS nodes"
2. "Configuring your JMS nodes to use SSL-enabled JNDI administered objects" on page 59
3. "Creating a keystore file to store the broker's certificates" on page 59

SSL authentication model for the JMS nodes:

The JMS provider TIBCO EMS supports Java clients that can use either the Java Secure Sockets Extension (JSSE) Java package, or an SSL implementation supplied by Entrust. For details about the services provided, see the documentation provided with your chosen package.

TIBCO EMS supports a number of different authentication scenarios, but JMS nodes can use only client authentication to the server. In this scenario, the TIBCO EMS server requests the client's digital certificate during an SSL handshake, and

checks its issuer against the server's list of trusted Certificate Authorities. If the authority is not in the server's list, further communications are prevented with the JMS node.

Therefore, you must configure the EMS server to explicitly enable client authentication of the SSL certificates in its configuration file; configure the JNDI administered SSL JMS connection factories for the same level of support.

Configuring your JMS nodes to use SSL-enabled JNDI administered objects:

The JMS nodes use JNDI to look up a connection factory object that is used to create JMS connections to a TIBCO EMS server.

1. Configure the JMS node property Connection factory name to specify a pre-configured connection factory that is enabled for SSL connectivity.

   Make sure that you have set the appropriate parameters in the corresponding SSL JMS connection factory definition:

   - Enable client authentication
   - Specify the SSL protocol in the server URL
   - Set other parameters to define the support your require.

   See the provider's documentation for information about how to generate this JNDI administered object:
2. Configure the JMS node property Location JNDI Bindings with the URL that points to the JNDI bindings containing the JNDI administered objects for SSL connectivity.

   For TIBCO EMS , this URL takes the following format:

   `tibjmsnaming://`*`server_name`*`:`*`ssl_port`*

   - *server_name* is the host name of the computer where the server is installed.
   - *ssl_port* is the server port for SSL connectivity; typically, this is port 7243 for a TIBCO EMS server.
3. Make the TIBCO EMS client JAR files available to the broker to which you deploy the message flow that includes your JMS nodes. Use the mqsicreateconfigurableservice or the mqsichangeproperties command to set the JMSProviders configurable service property jarsURL to point to the directory that contains the JMS provider's client JAR files and the SSL vendor's JAR files.

   If you are using JSSE for the SSL support, the following JAR files are typically located in the `jarsURL` directory:

   - `jsse.jar`
   - `net.jar`
   - `jcert.jar`
   - `tibcrypt.jar`

   You can find standard non-SSL client JAR files in the same location.

Creating a keystore file to store the broker's certificates:

To work with keystore files, use an administrative program such as keytool, which is supplied as part of the Java Runtime Environment (JRE) and therefore installed with the broker.

To create the keystore file by using keytool, complete the following steps:

1. Select **Start** → **IBM WebSphere Message Broker 6.1** → **Command Console** to open the broker command console.

2. In the command console, type the following command:
   ```
   "%MQSI_FILEPATH%\jre\bin\keytool"
   ```
   This command displays the help options, and therefore validates that the command is working.
3. In the command console, type the following command:
   ```
   "%MQSI_FILEPATH%\jre\bin\keytool" -genkey -keypass password
    -keystore keystore_file -alias jms
   ```
   - *password* is the password used for the keystore.
   - *keystore file* is the fully qualified name of the keystore file. This file is typically called `.keystore`, and is located in the home (installation) directory.

   You must ensure that the server key and the cipher that you are using are compatible. For example, if the server is using a DSA key, but the cipher that the client is using is SSL_DHE_RSA_WITH_AES_128_CBC_SHA, you must change the server to use an RSA key. To change the server key to RSA, add the option `-keyalg RSA` before `-alias` on the keytool command.

   When you specify the `-genkey` parameter, all the certificate files that are required for a JMS connection are generated; however, these files are not official certificates and are appropriate for test systems only. For 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 that is generated by a certificate authority, use the `-import` option in place of the `-genkey` option.
4. Enter personal details in response to the command prompt. Your personal details are used to generate the certificates, or added to an existing keystore. You can set these details to values that suit your configuration, but you must update the properties on the broker to reflect these values.

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

Enabling SSL for the Real-time nodes

Use optional authentication services between JMS clients and Real-timeInput and Real-timeOptimizedFlow nodes.

In a default configuration, SSL authentication services are disabled.

To configure the product to use the SSL 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 is 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.
- For Real-time nodes when using multiple execution groups, set the **sslKeyRingFile** and **sslPassphraseFile** properties on each execution group separately using the mqsichangeproperties command.

Configuring the User Name Server:

The User Name Server distributes to the brokers passwords that are required to support these authentication protocols.

To configure the User Name Server to support authentication, specify the following two parameters on either the mqsicreateusernameserver or the mqsichangeusernameserver command:

- *AuthProtocolDataSource* describes the location of an local file that contains the information that is required to support the authentication protocols.
- The -j flag indicates whether the file that is pointed to by the*AuthProtocolDataSource* parameter contains group and group membership information in addition to 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 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 Message Broker home directory. If you store the file in another location, specify the full path definition of the location of the file.
- Stop and restart the User Name Server to implement the changes.

Use the -d flag on the mqsichangeusernameserver command to disable this option.

Configuring a broker:

Configure a broker to support WebSphere Message Broker authentication services. 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.

1. Switch to the Broker Application Development perspective.
2. For each message flow in the Message Flow Topology:
   a. Select the Real-timeInput or Real-timeOptimizedFlow node to open the Properties view. The node properties are displayed.
   b. Select **Authentication**.
3. For each broker in the Broker Topology:
   a. Select the broker to open the Properties view. The broker properties are displayed.
   b. Enter the required value in **Authentication Protocol Type**.

      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 always to support the strongest protocol that the client supports, choose RSMP.
   c. If you have chosen S or R as one of the options in **Authentication Protocol Type**, specify the **SSL Key Ring File Name** and the **SSL Password File Name**.
   d. Click **OK**.

e. Use the mqsicreatebroker or mqsichangebroker command, with the following two parameters, to configure the broker:

UserNameServerQueueManagerName (-s)
This parameter defines the name of the queue manager that it associated with the User Name Server. Specify this parameter if you require authentication services, publish/subscribe access control services, or both.

Publish/Subscribe Access Control Flag (-j)
Set this flag in addition to specifying the **UserNameServerQueueManagerName** parameter if you want to use publish/subscribe access control services.

Use of the authentication services in the broker is enabled at the IP input node level, not by a parameter on these commands.

Sample password files:

Two sample files, `password.dat` and `pwgroup.dat`, are supplied with WebSphere Message 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.

The file `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 that is retrieved by the User Name Server from the operating system. User names that are defined in the file, but are not defined in 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.

The file `pwgroup.dat` contains group information in addition to user and password information. Each user entry includes a list of group names that specify the groups that contain the user.

The file `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 the brokers, update the file, and then restart the User Name Server and the 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 6.0 or later, the client application supports two levels of authentication.

You can configure a *TopicConnectionFactory* 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 you have successfully configured authentication services for a Real-timeInput node, a JMS client application must specify its credentials when creating a connection. To make a connection for this configuration, the JMS client application supplies a user ID and password combination to the *TopicConnectionFactory.createTopicConnection* method; for example:

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

If the application does not specify these credentials, or specifies them incorrectly, it receives a JMS wrapped exception containing the MQJMS error text.

Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)

Configure the HTTPInput and HTTPReply nodes to communicate with other applications that use HTTPS by creating a keystore file, configuring the broker to use SSL, and creating a message flow to process HTTPS requests.

Follow these steps to configure the HTTPInput and HTTPReply nodes to communicate with other applications using HTTP over SSL:

1. "Create a keystore file to store the broker's certificates"
2. "Configuring the broker to use SSL on a particular port" on page 64
3. "Creating a message flow to process HTTPS requests" on page 65
4. "Testing your example" on page 65

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

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

1. Select **Start → IBM WebSphere Message Broker 6.1 → 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 keystore and generate a new self-signed certificate (or key pair):
   a. In the command console, type the following command:

      ```
      "%MQSI_FILEPATH%\jre\bin\keytool" -genkey -keystore keystore file
       -storepass password -alias mykey
      ```

      where *keystore file* is the fully qualified name of the keystore file, *password* is the password used for the keystore, and *mykey* is the name (or label) given to the new key pair that is being generated. The keystore file is typically called `.keystore` and is located in the WebSphere Message Broker home directory.
   b. Enter the personal details that are required to generate the certificates.
   c. When you are prompted to enter a key password for the alias, press **Enter** to use the same password as the keystore. The keystore is either generated or updated (if it already exists).

   This command creates a keystore of type JKS, which is the only store type supported by the broker.

   You can set these parameters to any values, but the properties on the broker must be changed to reflect them.

   The **-genkey** parameter generates all the certificate files necessary to enable HTTPS to work for testing purposes, but they are not suitable for use in a production system. You must purchase an official certificate from a certification authority. Consult your system administrator to check your company policy for certificate creation.

   To import a certificate generated by a certification authority, use the **-import** parameter instead of the **-genkey** parameter.

   You must ensure that the server key and the cipher that you are using are compatible with each other. For example, if the server is using a DSA key, and the client is using a SSL_DHE_RSA_WITH_AES_128_CBC_SHA cipher, you need to use an RSA key on the server.

   To change the server key to RSA, add the following parameter to the keytool command before the **-alias** parameter:

   ```
   -keyalg RSA
   ```

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 to make use of HTTP over SSL. All of these properties can be changed using the mqsichangeproperties command. Change the properties as follows:

- 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 Message Broker will listen for HTTPS requests

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

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

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

Ensure that each of these properties is set with correct values for your system. Only the **enableSSLConnector** property must be set; the other three properties have default values. The mqsichangeproperties command lists the default values for all the properties.

On UNIX systems, only processes that run under a privileged user account (in most cases, root) can bind to ports lower than 1024. For the broker to listen on these ports, the broker's service user ID must be root.

Creating a message flow to process HTTPS requests:

You can create a simple message flow to use HTTPS by connecting an HTTPInput node to an HTTPReply node. The two most important properties to set on the HTTPInput node are:

- **Path suffix for URL**; for example, `/*` or `/testHTTPS`.
- **Use HTTPS**.

`/*` means that the HTTPInput node matches against any request that is sent to the HTTP listener on a designated port. This option is useful for testing purposes, but is not suitable for production systems.

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

You can now test the system.

Testing your example:

The simplest method of testing whether HTTPS is configured correctly is to use a Web browser to make a request to the broker over HTTPS.

Start a Web browser and enter the following URL:

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

Change any values in the URL to reflect changes that you have made in your broker configuration. When a window is displayed asking you to accept the certificate, select **Yes**. The browser refreshes the window and displays an empty HTML page. In Mozilla browsers, the empty HTML page looks like the following example:

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

and in Internet Explorer, the following information is displayed:

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

These responses mean that a blank page was returned, indicating that the setup worked correctly. To add content to the empty page, 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 to configure it to make client connections over SSL.

You can also use another HTTPS client, such as a Java or .net client, instead of the Web browser. Depending on the type of client, you might need to export the certificate (which was created with keytool) from the HTTP listener's keystore file and then import it into the client's keystore. Read the client documentation to find out how to configure the client to make client connections over SSL.

Configuring an HTTPRequest node to use SSL (HTTPS)

Configure the HTTPRequest node to communicate with other applications that use HTTP over SSL by adding certificates to the cacerts file and creating a message flow to make HTTP requests.

This topic describes the steps that you need to follow when configuring an HTTPRequest node on a Windows system. The steps that you must follow on other operating systems are almost identical.

To enable an HTTPRequest node to communicate using HTTP over SSL, an HTTPS server application is required. The information provided in this topic shows how to use the HTTPInput node for SSL as the server application, but the same details also apply when you are using any other server application.

Complete the following sub-tasks:

1. "Adding certificates to the cacerts file"
2. "Creating a message flow to make HTTPS requests" on page 68
3. "Testing your example" on page 68.

Adding certificates to the cacerts file:

You must add the certificate for the server application to be called (and trusted) to the cacerts file for WebSphere Message Broker. This file is the default trust store for the broker and is located in the broker's JRE security directory. To find the cacerts file on Windows:

1. Select **Start → IBM WebSphere Message Broker 6.1 → Command Console** to open a broker command console.
2. In the command console, type the following command to change to the directory in which the cacerts files is stored, for example:

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

   or

   ```
   cd "%MQSI_FILEPATH%\jre15\lib\security"
   ```

On UNIX systems, the cacerts file is stored in the following directory:

```
/opt/IBM/mqsi/6.1/jre15/ppc64/lib/security
```

or

```
$MQSI_FILEPATH/jre15/ppc64/lib/security
```

Importing a certificate into the cacerts file

Use the keytool command to modify the cacerts file:

1. Click **Start → IBM WebSphere Message Broker 6.1 → 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 -keystore cacerts
-storepass changeit
```

where:

name of certificate file
is the fully qualified name of the certificates file. This file is typically found in the message broker user's home directory.

changeit
is the default password for the cacerts file. You can use the keytool command to change the password, but, because it is not a configurable property of the broker, the broker always attempts to access the cacerts file using the default password *changeit*.

If you must change the cacerts password, or if you must use a different trust store, you can pass the information to the broker's Java Virtual Machine (JVM) by setting the following environment variable:

On Windows, AIX®, and Linux:

```
IBM_JAVA_OPTIONS=
-Djavax.net.ssl.trustStore=<trustStore_path>/<trustStore_filename>
-Djavax.net.ssl.trustStorePassword=<trustStore_password>
```

On Solaris and HP-UX:

```
_JAVA_OPTIONS=
-Djavax.net.ssl.trustStore=<trustStore_path>/<trustStore_filename>
-Djavax.net.ssl.trustStorePassword=<trustStore_password>
```

Use caution when using this environment variable, because if the setting is not valid, the broker's execution groups might be unable to create their JVM, and therefore cannot start successfully. Do not use the **IBM_JAVA_OPTIONS** (or **_JAVA_OPTIONS**) environment variable if you are also using SSL authentication with Real-time nodes, or with the WebSphere MQ Java Client.

Extracting a certificate from another keystore

1. Click **Start → IBM WebSphere Message Broker 6.1 → Command Console** to open a broker command console.
2. In the command console, type the following command:

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

where

name of certificate file
is the fully qualified name of the certificate file. This file typically has the extension `.keystore` and is located in the broker user's home directory.

keystore file
is the fully qualified name of the keystore file. This file is typically found in the broker user's home directory.

mykey is the alias name for the keystore entry (certificate).

changeit
is the password for the keystore file.

You must import the correct certificate (which the HTTP server uses to present its authentication credentials) into the cacerts file.

Creating a message flow to make HTTPS requests:

The following message flow creates a generic message flow for converting a WebSphere MQ message into an HTTP Request:

1. Create a message flow with the nodes MQInput->HTTPRequest->Compute->MQOutput.
2. On the MQInput node, set the queue name to HTTPS.IN1 and create the WebSphere MQ queue.
3. On the MQOutput node, set the queue name to HTTPS.OUT1 and create the WebSphere MQ queue.
4. On the HTTPRequest node, set the Web Service URL to point to the HTTP server to call. To call the HTTPInput node, use `https://localhost:7083/testHTTPS`.
5. On the Advanced properties tab of the HTTPRequest node, set the **Response message location in tree** property to *OutputRoot.BLOB*.
6. On the Compute node, add the following ESQL code:

```
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 the instructions in "Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)" on page 63, including testing the example.

2. Deploy the HTTPRequest message flow.
3. Put a message to the WebSphere MQ queue HTTPS.IN1. If successful, a message appears on the output queue. If the process fails, an error appears in the local error log (which is the event log on Windows).

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.

Before you start:

Before you create a User Name Server, refer to "Considering security for a User Name Server" on page 70.

To enable topic-based security, complete the following steps:

1. Create a User Name Server. For more information, refer to "Creating a User Name Server" on page 202.
2. On the mqsicreatebroker command (or the mqsichangebroker command if you are using an existing broker), select the -j flag and set the -s parameter to the name of the queue manager for the User Name Server .
3. Set the -s parameter on the mqsicreateconfigmgr or mqsichangeconfigmgr command to the name of the queue manager for the User Name Server.
4. Create ACLs for the topics that require additional security. For more information, see "Creating ACL entries" on page 89.
5. Ensure that the broker's service user ID has authority to perform the following actions:
   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.
6. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put messages to and get messages 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 messages to the SYSTEM.BROKER.CONTROL.QUEUE queue.
   c. Authorize subscribe applications to get messages from the queue to which messages are published.
   d. Authorize publish and subscribe applications to get messages 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 25, "Quality of protection" on page 26, and "Authentication services" on page 83.

Go to "Considering security for a Configuration Manager" on page 51.

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 51.
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 which 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: Ensure that the user ID is a member of the mqbrkrs group.

      Go to "Deciding which user account to use for the User Name Server service ID."
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: Ensure that the user ID is a member of the mqbrkrs group.

      Go to "Setting security on the User Name Server's queues" on page 71

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 has the following characteristics:
      - It is defined in your local domain
      - It is a member of the mqbrkrs group
      - It has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**.

      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:
      - The user ID (user1) has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**
      - DOMAIN1\user1 is a member of DOMAIN1\Domain mqbrkrs
      - 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.

Go to "Running the User Name Server in a domain environment."

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

Define a security exit on the WebSphere MQ channel when you create a domain connection

To create a security exit on the WebSphere MQ channel that you define for communications between the Message Broker Toolkit and the Configuration Manager that controls the domain, you must define a security exit when you create the connection.

1. Switch to the Broker Administration perspective.

2. Click **File** → **New** → **Domain Connection**. The Create a Domain Connection wizard opens.
3. Enter the values for **Queue Manager Name**, **Host**, and **Port** that you want to use.
4. Enter the Security Exit **Class** name. The name must be a valid Java class name.
5. Set the **JAR File location** for the Security Exit that is required on this connection. Click **Browse** to find the file location.

The security exit is started every time a message passes across the connection.

Alternatively, use SSL to communicate between the Configuration Manager Proxy (CMP) and the Configuration Manager; see "Implementing SSL authentication" on page 57.

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 437.

Once set, HTTP tunneling is administered using the "mqsichangeproperties command" on page 437 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 437.

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

You must set up a database for each broker, and you can access databases from the message flows that you deploy to your brokers, and must therefore consider the steps you might want to take to secure that access.

You must authorize the following IDs for each broker:
- Authorize each broker service user ID for create and update tasks on the database that contains the broker internal tables.
- Authorize all data source user IDs that you have specified by using mqsicreatebroker or mqsisetdbparms command to access those databases from deployed message flows.

If you have run the Default Configuration wizard or the "mqsicreatedb command" on page 536 on Windows to create a Derby database, you do not have to set up security authorization. Refer to "Using Derby databases on Windows" on page 119 for more information.

You can define and modify database security by using a number of commands, depending on the database that you are working with:
- "mqsicreatebroker command" on page 513
- "mqsichangebroker command" on page 404
- "mqsisetdbparms command" on page 646

WebSphere Message Broker does not provide any special commands to administer databases. Discuss your database security requirements with the database administrator for the database manager that you are using, or refer to the documentation provided by your database supplier.

For further information about possible security requirements, and examples of setting up security for DB2 and Oracle databases, see "Authorizing access to broker and user databases" on page 122.

Setting up z/OS security

On z/OS, you must complete several security configuration tasks before WebSphere Message Broker can work correctly.

The steps you need to follow are described in this topic and also in the following topics:
- "Setting up DB2 security on z/OS" on page 75
- "Setting up WebSphere MQ" on page 76
- "Setting up workbench access on z/OS" on page 77
- "Creating Publish/Subscribe user IDs" on page 77
- "Enabling the Configuration Manager on z/OS to obtain user ID information" on page 77

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 Message Broker PDSEs. A typical name might be `WMQI.MQP1BRK.CNTL` or `MQS.MQP1UNS.BIPCNTL`, where `MQP1` is the queue manager name. You must give the WebSphere Message 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 ID 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 Message 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 Message 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 require a minimum of 400 000 blocks available 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 Message Broker and z/OS administrators must agree on the name of the started task.

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

The started task user IDs and the WebSphere Message 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.

For more information on various aspects of security, see "Security overview" on page 3.

Setting up DB2 security on z/OS

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 must have *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 must have 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. 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 and gives details of the authorities that your user ID needs to perform the required operations.

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 190 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. These names cannot be changed. Restrict access to the broker, Configuration Manager, and User Name Server started task user IDs, and to WebSphere Message Broker administrators.

If you are running a Configuration Manager on z/OS, remote users connecting from either the Message Broker Toolkit or from a Configuration Manager Proxy application need to be authorized to connect to the queue manager through the channel initiator and require *PUT* and *GET* access to SYSTEM.BROKER.CONFIG.QUEUE and SYSTEM.BROKER.CONFIG.REPLY

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 on x86. The workbench must run on Windows or Linux on x86.

See "Considering security for a Configuration Manager" on page 51 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))`.

Enabling the Configuration Manager on z/OS to obtain user ID information

This topic lists the steps you need to complete, to enable the Configuration Manager on z/OS to correctly obtain the list of user IDs for a particular group from the External Security Manager (ESM) database.

When connecting to the Configuration Manager, the local user ID on the connecting machine is sent to the Configuration Manager for the purposes of broker domain authorization. This user ID is checked against the Configuration Manager Access Control Lists (ACL) to determine the level of authorization.

For any group ACLs defined, the Configuration Manager queries the local ESM database for a list of user IDs defined to that group. The Configuration Manager then tries to match any user ID connecting to it, with this list, to grant the correct authorization to the broker domain.

For the Configuration Manager on z/OS to obtain this list of user IDs, the group and any user IDs must have an OMVS segment defined.

User IDs

- If you have suitable authorization, you can use the following Security Server (formerly RACF) command to display OMVS information about a user:

  ```
  LU id OMVS
  ```

- If you have suitable authorization, you can use the following Security Server command to give a user ID an OMVS segment:

  ```
  ALTUSER id  OMVS(UID(xxx))
  ```

Groups

- If you have suitable authorization, you can use the following Security Server command to display OMVS information about a user:

  ```
  LG group OMVS
  ```

- If you have suitable authorization, you can use the following Security Server command to give a group an OMVS segment:

```
ALTGROUP id  OMVS(GID(xxx))
```

See 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 the group, or any of the defined user IDs in that group, are not found by the Configuration Manager (either because they do not exist, or because they do not have an OMVS segment), the Configuration Manager is not able to authorize the user attempting the connection.

Publish/subscribe security

Security services are provided for your publish/subscribe domain.

- Topic-based security

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

  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 using real-time nodes

  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 create a User Name Server and include it in your domain.

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 queue manager that hosts the User Name Server.

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 Message 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 Message 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 Message 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 Message 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 can only published from, and subscribed to, 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 the ACLs are ignored.

Setting access control on topics

Any user that has an object-level security ACL that gives full control 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:

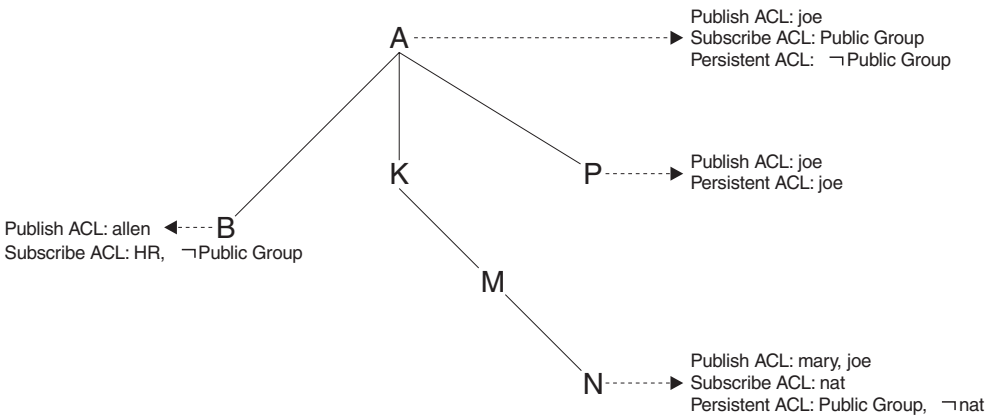
- Clients that are denied Publish access have their publication messages refused.
- Clients that are denied Subscribe access do not receive the publication.
- The persistent access control does not deny the message to subscribers, but denies them persistence, so denied subscribers always receive messages, 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 Message 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 Message 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 Message Broker workbench.

You must have an object-level security ACL that gives full control 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 Message Broker Real-timeInput and Real-timeOptimizedFlow nodes.

The WebSphere Message 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 Message 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 84 and “Mutual challenge-response password authentication” on page 85 respectively. Asymmetric and symmetric SSL protocols are described in “SSL authentication” on page 25.

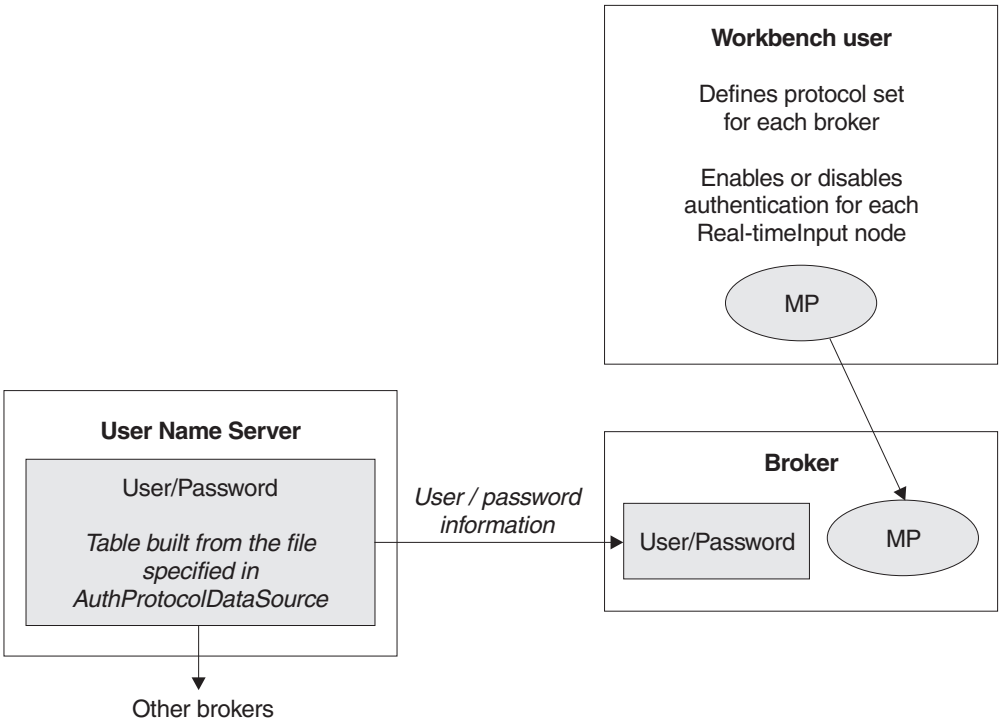
The protocols vary in strength, in terms of providing protection against participants that are not valid participants in the session; P is the weakest and R is the strongest.

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

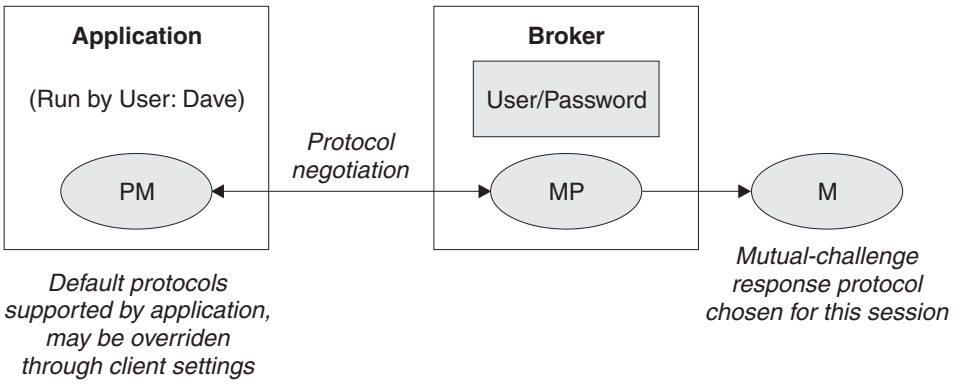
authentication is enabled at a Real-timeInput node, that node supports the full set of protocols specified for its corresponding broker. The configuration options are illustrated in the following diagrams:

Overview of authentication configuration

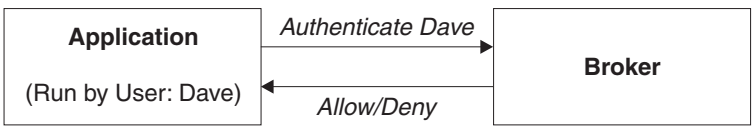


Two-stage runtime authentication process

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 Message 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 Message 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

Use appropriate options to secure your publish/subscribe domain.

The following topics describe the actions that you can take to secure your publish/subscribe domain:

- "Enabling topic-based security" on page 69
- "Creating ACL entries" on page 89
- "Enabling SSL for the Real-time nodes" on page 60
- "Using message protection" on page 92
- "Securing WebSphere MQ resources" on page 93

For more information, see "Publish/subscribe security" on page 78 and "Security overview" on page 3.

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.

Before you start:

Before you create a User Name Server, refer to “Considering security for a User Name Server” on page 70.

To enable topic-based security, complete the following steps:

1. Create a User Name Server. For more information, refer to “Creating a User Name Server” on page 202.
2. On the mqsicreatebroker command (or the mqsichangebroker command if you are using an existing broker), select the -j flag and set the -s parameter to the name of the queue manager for the User Name Server .
3. Set the -s parameter on the mqsicreateconfigmgr or mqsichangeconfigmgr command to the name of the queue manager for the User Name Server.
4. Create ACLs for the topics that require additional security. For more information, see “Creating ACL entries” on page 89.
5. Ensure that the broker’s service user ID has authority to perform the following actions:
   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.
6. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put messages to and get messages 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 messages to the SYSTEM.BROKER.CONTROL.QUEUE queue.
   c. Authorize subscribe applications to get messages from the queue to which messages are published.
   d. Authorize publish and subscribe applications to get messages 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 25, “Quality of protection” on page 26, and “Authentication services” on page 83.

Go to “Considering security for a Configuration Manager” on page 51.

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 51.
2. Yes: You need a User Name Server. Go to “Deciding which user accounts can execute User Name Server commands” on page 70.

Deciding which user accounts can execute User Name Server commands

During this task you decide which 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: Ensure that the user ID is a member of the mqbrkrs group.

      Go to "Deciding which user account to use for the User Name Server service ID" on page 70.
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 70.
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 70.

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: Ensure that the user ID is a member of the mqbrkrs group.

      Go to "Setting security on the User Name Server's queues" on page 71
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 has the following characteristics:
      - It is defined in your local domain
      - It is a member of the mqbrkrs group
      - It has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**.

      Go to "Setting security on the User Name Server's queues" on page 71
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:
   - The user ID (user1) has been granted the *Logon as a service* privilege in the Local Security Policy in Windows, which you can access by selecting **Control Panel** > **Performance and maintenance** > **Administrative Tools** > **Local Security Policy**
   - DOMAIN1\user1 is a member of DOMAIN1\Domain mqbrkrs
   - DOMAIN1\Domain mqbrkrs is a member of WKSTN1\mqbrkrs.

   Go to "Setting security on the User Name Server's queues" on page 71.

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.

Go to "Running the User Name Server in a domain environment" on page 71.

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

You must create an access control list (ACL) for each new topic. See "Adding a new topic" on page 279 for more information about doing this.

Enabling SSL for the Real-time nodes

Use optional authentication services between JMS clients and Real-timeInput and Real-timeOptimizedFlow nodes.

In a default configuration, SSL authentication services are disabled.

To configure the product to use the SSL 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 is 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.
- For Real-time nodes when using multiple execution groups, set the **sslKeyRingFile** and **sslPassphraseFile** properties on each execution group separately using the mqsichangeproperties command.

Configuring the User Name Server

The User Name Server distributes to the brokers passwords that are required to support these authentication protocols.

To configure the User Name Server to support authentication, specify the following two parameters on either the mqsicreateusernameserver or the mqsichangeusernameserver command:

- *AuthProtocolDataSource* describes the location of an local file that contains the information that is required to support the authentication protocols.
- The -j flag indicates whether the file that is pointed to by the*AuthProtocolDataSource* parameter contains group and group membership information in addition to 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 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 Message Broker home directory. If you store the file in another location, specify the full path definition of the location of the file.
- Stop and restart the User Name Server to implement the changes.

Use the -d flag on the mqsichangeusernameserver command to disable this option.

Configuring a broker

Configure a broker to support WebSphere Message Broker authentication services. 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.

1. Switch to the Broker Application Development perspective.
2. For each message flow in the Message Flow Topology:
   a. Select the Real-timeInput or Real-timeOptimizedFlow node to open the Properties view. The node properties are displayed.
   b. Select **Authentication**.
3. For each broker in the Broker Topology:
   a. Select the broker to open the Properties view. The broker properties are displayed.
   b. Enter the required value in **Authentication Protocol Type**.

      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 always to support the strongest protocol that the client supports, choose RSMP.

c. If you have chosen S or R as one of the options in **Authentication Protocol Type**, specify the **SSL Key Ring File Name** and the **SSL Password File Name**.

d. Click **OK**.

e. Use the mqsicreatebroker or mqsichangebroker command, with the following two parameters, to configure the broker:

UserNameServerQueueManagerName (-s)
This parameter defines the name of the queue manager that it associated with the User Name Server. Specify this parameter if you require authentication services, publish/subscribe access control services, or both.

Publish/Subscribe Access Control Flag (-j)
Set this flag in addition to specifying the **UserNameServerQueueManagerName** parameter if you want to use publish/subscribe access control services.

Use of the authentication services in the broker is enabled at the IP input node level, not by a parameter on these commands.

Sample password files

Two sample files, `password.dat` and `pwgroup.dat`, are supplied with WebSphere Message 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.

The file `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 that is retrieved by the User Name Server from the operating system. User names that are defined in the file, but are not defined in 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.

The file `pwgroup.dat` contains group information in addition to user and password information. Each user entry includes a list of group names that specify the groups that contain the user.

The file `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 the brokers, update the file, and then restart the User Name Server and the 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 6.0 or later, the client application supports two levels of authentication.

You can configure a *TopicConnectionFactory* 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 you have successfully configured authentication services for a Real-timeInput node, a JMS client application must specify its credentials when creating a connection. To make a connection for this configuration, the JMS client application supplies a user ID and password combination to the *TopicConnectionFactory.createTopicConnection* method; for example:

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

If the application does not specify these credentials, or specifies them incorrectly, it receives a JMS wrapped exception containing the MQJMS error text.

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 broker domain configuration requires.

This information does not apply to z/OS.

Runtime components depend on a number of WebSphere MQ resources to operate successfully. You must control access to these resources to ensure that the 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.ADAPTER.FAILED
  - SYSTEM.BROKER.ADAPTER.INPROGRESS
  - SYSTEM.BROKER.ADAPTER.NEW
  - SYSTEM.BROKER.ADAPTER.PROCESSED
  - SYSTEM.BROKER.ADAPTER.UNKNOWN
  - SYSTEM.BROKER.ADMIN.QUEUE
  - SYSTEM.BROKER.AGGR.CONTROL
  - SYSTEM.BROKER.AGGR.REPLY
  - SYSTEM.BROKER.AGGR.REQUEST
  - SYSTEM.BROKER.AGGR.TIMEOUT
  - SYSTEM.BROKER.AGGR.UNKNOWN
  - SYSTEM.BROKER.CONTROL.QUEUE
  - SYSTEM.BROKER.EDA.COLLECTIONS
  - SYSTEM.BROKER.EDA.EVENTS
  - SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
  - SYSTEM.BROKER.EXECUTIONGROUP.REPLY
  - SYSTEM.BROKER.INTERBROKER.MODEL.QUEUE
  - SYSTEM.BROKER.INTERBROKER.QUEUE
  - SYSTEM.BROKER.MODEL.QUEUE
  - SYSTEM.BROKER.TIMEOUT.QUEUE
  - SYSTEM.BROKER.WS.ACK
  - SYSTEM.BROKER.WS.INPUT
  - SYSTEM.BROKER.WS.REPLY
- When you issue the command mqsicreateconfigmgr 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
- 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

- If you have created 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 by using a WebSphere MQ client/server connection. For details of WebSphere MQ channel security, see "Setting up WebSphere MQ client security" in the *Clients* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.
- When you create and deploy a message flow that includes nodes which reference WebSphere MQ queues, grant get, inq, and put authority to the user ID under which the broker is running.

Part 2. Configuring the broker domain

Configuring WebSphere Message Broker

Configure WebSphere Message Broker by creating and setting up all the databases, components, and connections that are required for a broker domain, to which you can deploy message flow applications.

If you want to create a simple configuration on Windows or Linux on x86 to learn about WebSphere Message Broker, and to run the samples in the Message Broker Toolkit Samples Gallery, run the Default Configuration wizard. The Default Configuration wizard creates the Default Configuration, which is a basic broker domain, including a broker database, a broker, a Configuration Manager, and a queue manager. See "Using the Default Configuration wizard" on page 217.

To configure WebSphere Message Broker:

1. Plan the system.
2. Create and configure the databases.
3. If you are configuring a broker domain on z/OS, Customize the z/OS environment.
4. Ensure that you have the correct authorization and permissions to create and access components. For more information, see "Authorization for configuration tasks" on page 17 and "Setting up broker domain security" on page 45.
5. Create the components.
6. If you are using the Message Broker Toolkit, Configure the broker domain in the workbench.
7. If you want the broker domain to support publish/subscribe messaging, Configure the domain for publish/subscribe messaging.

Planning a broker domain

When you start to plan a broker domain, you must first consider your resource naming conventions, the design of the WebSphere MQ infrastructure, and possible performance issues.

The following topics describe these considerations:

- "Considering resource naming conventions" on page 98
- "Designing the WebSphere MQ infrastructure" on page 100
- "Considering performance in the broker domain" on page 105

You must also consider your domain component requirements:

1. **Brokers:** The number of brokers that you require in your broker domain depends on the following factors:
   a. Performance: What is the required message throughput? See "Optimizing message flow throughput" on page 106.

      What is the size of the messages that are being processed? Larger messages take longer to process. A small number of brokers handling many messages might impact overall performance of the broker domain. See "Considering performance in the broker domain" on page 105.
   b. Operation mode: The mode in which your broker is working, can affect the number of execution groups and message flows that you can deploy, and

the types of node that you can use. See “Restrictions that apply in each operation mode” on page 368. (This option does not apply to brokers that you create on z/OS systems.)

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

d. Do the brokers need to handle publish/subscribe? See Developing publish/subscribe applications.

2. **Configuration Manager:** This component is an interface between the configuration repository, 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.

   If you think your configuration might experience these problems, consider dividing the brokers into more than one domain where related brokers are kept together. You can then establish connections with each domain; see “Creating a domain connection” on page 251.

3. **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 to the User Name Server can be handled more quickly if you configure 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 Message 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.

Considering resource naming conventions

When you plan a new WebSphere Message 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

Naming conventions for product components and associated resources

Establish a naming convention for all the WebSphere Message Broker resources in your broker domain to ensure that names are unique, and that users creating new resources can be confident of not introducing duplication or confusion.

Consider the names that you use for the following product components and resources:

Configuration Manager components

When you create a Configuration Manager, give it a name that is unique on your system. Names must be unique between Configuration Manager components, and between Configuration Manager components and brokers. Configuration Manager names are case-sensitive except on Windows platforms.

Broker components

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 workbench (in the broker domain topology). 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.

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 one.

Message flow names must be unique within the broker domain. All references to that name within the broker domain are 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 a name when you create it. It is identified only by the name of the WebSphere MQ queue manager that hosts the services it provides.

WebSphere MQ naming conventions

All WebSphere Message 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 Message 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 Message 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 Message 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 Message Broker for z/OS" on page 671.

Database naming conventions

Consider the naming conventions you use for databases that you create for WebSphere Message Broker brokers, and for databases that you create for application use.

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

Ensure that the databases that you use for application data, which are accessed through your deployed message flows, are uniquely named throughout your network, so that confusion and errors are avoided by all your users.

Designing the WebSphere MQ infrastructure

When you design your broker domain, consider the WebSphere MQ resources that you need to support the components in your domain configuration, and the applications that connect to the brokers to supply or receive messages.

WebSphere Message Broker runtime components

The WebSphere Message Broker runtime components use WebSphere MQ to exchange internally-generated messages that provide information, status, and instructions that concern the operation of those components. Connections are also required by each workbench with each Configuration Manager with which it want to communicate.

Some of these resources that are required are created for you when you create the components that depend on them. Others depend on the configuration of your broker domain; you must create these resources yourself.

The requirements associated with each type of component are described in the following topics:

- "WebSphere MQ resources for the broker" on page 102

- "WebSphere MQ resources for the Configuration Manager" on page 103
- "WebSphere MQ resources for the User Name Server" on page 104

Applications and message flows

Your applications exchange messages and other data by communicating with message flows that are running in the broker. You can choose to connect your applications to the broker with any one of the supported communications methods. If your applications are written to use WebSphere MQ, the requirement for the channels or client connections are determined by the types of nodes that you include in your message flows. These resources are application specific, and you must create these resources yourself.

The following nodes might require WebSphere MQ resources:

- MQInput, MQOutput, MQReply, MQOptimizedFlow, and MQGet
- Real-timeInput and Real-timeOptimizedFlow
- SCADAInput and SCADAOutput

For more information about creating resources, see the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

Planning for publish/subscribe application support

WebSphere Message Broker supports WebSphere MQ Version 7.0; both the broker, and the queue manager it runs on, support publish/subscribe engines, and you can choose the most suitable option for your publish/subscribe applications.

If you install WebSphere MQ Version 7.0, and either create or migrate your broker queue managers to this version, you can choose whether the broker's or the queue manager's publish/subscribe engine controls all publish/subscribe application messages.

The state of the broker publish/subscribe engine overrides the state of the queue manager publish/subscribe engine. When you start a broker on a computer on which you have installed WebSphere MQ Version 7.0, the broker checks the status of its predefined publish/subscribe configurable service; the default initial configuration is for the broker to control all publish/subscribe application messages. In this default state, the broker checks, and if necessary, resets, the status of the queue manager's publish/subscribe engine, so that control is retained by the broker.

If you migrate your broker domain from WebSphere MQ Version 6.0 to WebSphere MQ Version 7.0, your publish/subscribe clients and applications are unaffected because the default state of the broker's configurable service ensures that the publish/subscribe engine of all broker queue managers is disabled.

When you install or migrate to WebSphere MQ Version 7.0, you can choose to use the queue manager's publish/subscribe engine. Although the two engines provide similar support, you might prefer to enable the queue manager's publish/subscribe engine if your environment confirms to one or more of the following scenarios:

- Your applications typically generate a large number of subscriptions. The broker's publish/subscribe engine can support a maximum of about 25000 subscriptions, but the queue manager's publish/subscribe engine has no restrictions.

- You want to use high performance publish/subscribe messages with streaming and non-queue based (put to topic) messages.
- You want a consistent model across your WebSphere MQ domain; for example, you want to use the WebSphere MQ security model.
- You have a large publish/subscribe application network already in place, based on WebSphere MQ functions. If you change this network to use the broker's publish/subscribe engine, you cannot migrate the resources associated with your applications; that is, the registrations, subscriptions, and retained subscriptions. In addition, the ACLs you have defined are disabled. You would have to manually recreate these resources in the broker's match space.

If you decide that the queue manager's publish/subscribe engine provides the best solution for your environment, the following restrictions apply in the broker application environment:

- The Real-timeInput, Real-timeOptimizedFlow, SCADAInput, and SCADAOutput nodes are no longer supported. You can deploy message flows that include these nodes, but when a message is received, the broker throws a recoverable exception.
- Samples that use these nodes no longer work; for example, the Scribble sample.
- You cannot use the following views in the Broker Administration perspective of the workbench to manage publish/subscribe resources associated with the broker:
  - Publish/Subscribe Topology
  - Topics
  - Subscriptions

  These views continue to display and interact with the state of publish/subscribe resources that are associated with the broker's publish/subscribe engine, which are inactive and irrelevant after you have disabled that engine.

  Use WebSphere MQ Explorer for all actions related to resources associated with the queue manager's publish/subscribe engine.

For more information about the broker's publish/subscribe engine, see "Modifying the broker's publish/subscribe engine" on page 228.

For more information about the queue manger's publish/subscribe engine, access What's new in publish/subscribe in WebSphere MQ V7.0. For details of other enhancements and additions, see What's new in WebSphere MQ V7.0.

WebSphere MQ resources for the broker

Each broker depends on a number of WebSphere MQ resources: some are required, others are optional, and depend on your broker domain requirements. Some of these resources are created for you, but others you must define for yourself.

WebSphere MQ resources created for you

When you create a broker, the following WebSphere MQ resources are created for you:

- On distributed systems, the broker's queue manager. Each broker must be associated with a dedicated queue manager. Specify a queue manager name when you create the broker. If this queue manager does not exist, it is created for you.

  You can specify the same queue manager for a single broker, a Configuration Manager, and the User Name Server if you create all three components on the same computer.

- Fixed-name queues on the queue manager that hosts this broker. These queues are used by the broker to exchange messages with other components in your broker domain.

WebSphere MQ resources that you must create yourself

Depending on the setup of your broker, you might need to create some WebSphere MQ resources yourself. You might need some or all of the following resources:

- If you create a broker on z/OS, you must create the queue manager. See "Creating a broker on z/OS" on page 183 for more details.
- If the broker and Configuration Manager do not share a queue manager, define the channels and transmission queues to support communication between the two queue managers.
- If the broker and User Name Server do not share a queue manager, define the channels and transmission queues to support communication between the two queue managers.
- Define listener connections on the broker's queue manager. You must define one listener connection for every protocol that your applications use; for example, TCP/IP.
- If the broker is to communicate with other brokers, for example in a publish/subscribe network, define the channels and transmission queues to permit two-way communication between the broker queue managers.

Create WebSphere MQ resources by using one of the following commands and utilities:

- runmqsc
- The PCF interface
- WebSphere MQ Explorer

For more information about creating resources, see the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

WebSphere MQ resources for the Configuration Manager

Each Configuration Manager depends on a number of WebSphere MQ resources: some are required, others are optional and depend on your broker domain requirements. Some of these resources are created for you: you must define others yourself.

WebSphere MQ resources created for you

When you create a Configuration Manager, the following WebSphere MQ resources are created for you:

- On distributed systems, the queue manager that will host this Configuration Manager. Each Configuration Manager must be associated with a queue manager. Specify a queue manager name when you create the Configuration Manager. If this queue manager does not exist, it is created for you.

  You can specify the same queue manager for a single broker, a Configuration Manager, and the User Name Server if you create all three components on the same computer.
- Fixed-name queues on the queue manager that hosts this Configuration Manager. These queues are used by the Configuration Manager to exchange messages with other components in your broker domain.

- A server connection for use by the Workbench.

WebSphere MQ resources that you must create yourself

Depending on the setup of your broker, you might need to create some WebSphere MQ resources yourself. You might need some or all of the following resources:

- If you create a Configuration Manager on z/OS, you must create the queue manager. See "Creating a Configuration Manager on z/OS" on page 197 for more details.
- Define the transmission queues and channels between the Configuration Manager and each broker in your domain that the Configuration Manager will manage, except for a broker that shares its queue manager with this Configuration Manager.
- If the Configuration Manager and the User Name Server do not share a queue manager, define the channels and transmission queues to support communication between the two queue managers.
- Define listener connections on the queue manager associated with this Configuration Manager for the workbench.

Create WebSphere MQ resources by using any of the following commands and utilities:

- runmqsc
- The PCF interface
- WebSphere MQ Explorer

For more information about creating resources, see the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

WebSphere MQ resources for the User Name Server

Each User Name Server depends on a number of WebSphere MQ resources: some are required, others are optional and depend on your broker domain requirements. Some of these resources are created for you: you must define others yourself.

WebSphere MQ resources created for you

When you create a User Name Server, the following WebSphere MQ resources are created for you:

- On distributed systems, the queue manager that will host this User Name Server. Each User Name Server must be associated with a queue manager. Specify a queue manager name when you create the User Name Server. If this queue manager does not exist, it is created for you.

  You can specify the same queue manager for a single broker, a Configuration Manager, and the User Name Server if you create all three components on the same computer.
- Fixed-name queues on the queue manager that hosts this User Name Server. These queues are used by the User Name Server to exchange messages with other components in your broker domain.

WebSphere MQ resources that you must create yourself

Depending on the setup of your broker, you might need to create some WebSphere MQ resources yourself. You might need some or all of the following resources:

- If you create a User Name Server on z/OS, you must create the queue manager. See "Creating a User Name Server on z/OS" on page 206 for more details.
- If the Configuration Manager and the User Name Server do not share a queue manager, define the channels and transmission queues to support communication between the two queue managers.
- Define the transmission queues and channels between the User Name Server and every broker in your domain (apart from the broker that shares a queue manager with the User Name Server).
- You must define listener connections on the User Name Server's queue manager for components that do not share its queue manager. You must define one listener connection for every protocol used.

Create WebSphere MQ resources by using any of the following commands and utilities:

- runmqsc
- The PCF interface
- WebSphere MQ Explorer

For more information about creating resources, see the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

Considering performance in the broker domain

When you design your broker domain, and the resources associated with the components, 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. The message must be parsed by the input node, although some parsers support partial parsing which might reduce processing, because only the parts of the message that are referenced are parsed. Other processing in a message flow that might affect performance are the amount, efficiency, and complexity of ESQL, access to databases, and how many message tree copies are made.

You must 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 and message models

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, they have to be stored for safekeeping.

You might need to process messages with a well defined structure; if so, you can create MRM models for your messages. If you do not plan to interrogate the structure, you can work with undefined messages, such as BLOB mesages.

If you are working in XML, be aware that it can be verbose, and therefore produce large messages, but XML message content is easier to understand than other formats, such as CWF. Field size and order might be important; these factors can be included in your MRM model.

For more information about these factors, see Optimizing message flow response times and Performance considerations for regular expressions in TDS messages.

Broker configuration and domain topology
You can configure your broker domain to include multiple brokers, multiple systems, multiple execution groups, and so on. Your configuration decisions can influence message flow performance, and how efficiently messages can be processed.

For more information about these factors, see "Tuning the broker" on page 223, "Optimizing message flow throughput," 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 Message Broker and performance, review these sources:

- The Business Integration Zone on developerWorks. This site 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. This design might result in very complex message flows that include large numbers of nodes that can cause a performance overhead, and might create potential bottlenecks. You can increase the number of message flows that process your messages to provide the opportunity for parallel processing and therefore improved throughput.

The mode that your broker is working in can affect the number of message flows that you can use; see "Restrictions that apply in each operation mode" on page 368.

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 behavior is the default. 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 option 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 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). Therefore 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 situation 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 *Application Programming Reference* and *Application Programming Guide* sections in the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online for programs written to the MQI, and the *WebSphere MQ Application Messaging Interface* book for programs written to the AMI.

The *WebSphere MQ Application Messaging Interface* book is available from the WebSphere MQ library Web page (listed under Version 5.3), or from SupportPac MA0F on the WebSphere MQ SupportPacs Web page.

See also the *Publish/Subscribe User's Guide* section in the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

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. The broker does not provide message ordering for messages received across WebSphere MQ Web Services Transport, 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 option has similar effects to increasing the number of processing threads in a single message flow, although typically provides less noticeable gains.

This option also removes the ability to determine the order in which the messages are processed, 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 might 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 option 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.

The scope of the message flow

You might find that, in some circumstances, you can split a single message flow into several different flows to reduce the scope of work that each message flow performs. If you do split your message flow, be aware that it is not possible to run the separate message flows in the same unit of work, and if transactional aspects to your message flow exist (for example, the updating of multiple databases), this option does not provide a suitable solution.

The following two examples show when you might want to split a message flow:

1. In a message flow that uses a RouteToLabel node, the input queue has become a bottleneck. You can use another copy of the message flow in a second execution group, but this option is not appropriate if you want all of the messages to be handled in the order in which they are shown on the queue. You can consider splitting out each branch of the message flow that starts with a Label node by providing an input queue and input node for each branch. This option might be appropriate, because when the message is routed by the RouteToLabel node to the relevant Label node, it has some level of independence from all other messages.

   You might also need to provide another input queue and input node to complete any common processing that the Label node branches connect to when unique processing has been done.
2. If you have a message flow that processes very large messages that take a considerable time to process, you might be able to:
   a. Create other copies of the message flow that use a different input queue (you can set this option up in the message flow itself, or you can update this property when you deploy the message flow).
   b. Set up WebSphere MQ queue aliases to redirect messages from some applications to the alternative queue and message flow.

You can also create a new message flow that replicates the function of the original message flow, but only processes large messages that are immediately passed on to it by the original message flow, that you modified to check the input message size and redirect the large messages.

The frequency of commits

If a message flow receives input messages on a WebSphere MQ queue, you can improve its throughput for some message flow scenarios by modifying its default properties after you have added it to a BAR file. (These options are not available if the input messages are received by other input nodes; commits in those message flows are performed for each message.)

The following properties control the frequency with which the message flow commits transactions:

- Commit Count. This property represents the number of messages processed from the input queue before an MQCMIT is issued.
- Commit Interval. This property represents the time interval that elapses before an MQCMIT is started.

Configuring broker and user databases

A broker stores internal performance and operational data in a database that you must create and configure before you create the broker. You might also choose to access databases to hold application or business data.

Multiple brokers that are at the same version can store their tables in a single database schema, if appropriate, even if they are not on the same computer.

Databases that hold application or business data are known as *user databases*, and are read from and written to by nodes within the message flows that you deploy to one or more brokers in your domain.

In some situations, and for some applications, you might need to ensure the integrity of the data that you hold in user databases across multiple systems and resource managers by coordinating table updates and the writing to one database with the deletion of data in another. To achieve these goals, you must configure your databases, your brokers, and your message flows to be globally coordinated.

For more information about the requirement for, and set up of, broker and user databases, and the restrictions that apply, see “Databases overview” on page 110.

z/OS For additional information about setting up databases on z/OS, see “DB2 planning on z/OS” on page 165 and “Customizing the z/OS environment” on page 152.

The process of making databases available has the following phases:

1. Required: Create and configure a broker database.

   If you are migrating from a previous release, you can continue to use the same broker database, but you must check and update your ODBC configuration to ensure continued database access. Database access requires different configuration in each release; see Migrating from Version 6.0 products or Migrating from Version 5.0 products.

If you are not migrating, you must create and configure a database for each broker that you create, and you must configure the ODBC resources required by the broker to connect to that database.

Optional: if you want to deploy publish/subscribe message flow applications, and you have created a broker database on a Sybase database instance, you must modify the database to operate successfully in this environment.

2. Optional: Create and configure user databases. If your message flows interact with databases, you must create and configure those databases ready for connection by the broker on behalf of the message flows. For user databases, you can configure ODBC and JDBC connections.
3. Optional: If your user databases contain critical information, coordinate their updates through a transaction manager.

   On distributed systems, the WebSphere MQ queue manager is the transaction manager that interacts with the resource managers (the database providers). On z/OS, RRS provides equivalent coordination.

To complete these phases:

1. For a new installation, create and configure the broker database:
   a. Create the databases.
   b. Authorize access to the database.
   c. Enable the ODBC connection.
   d. Optional: If you are creating a publish/subscribe broker domain *and* you are using a Sybase broker database, configure the database to support retained publications.

   For an existing broker that you are migrating, follow the instructions in the appropriate topics in Migrating and upgrading.

   You have now completed the only mandatory step for database configuration; now you can create and configure your broker domain components:
   a. If you have one or more components on z/OS, customize the z/OS environment.
   b. Configure broker domain components.
   c. Configure a broker domain in the workbench.
   d. Optional: If you want to establish a publish/subscribe network, configure a publish/subscribe topology.
2. Optional: If you want to access user databases from your deployed message flows, create and configure additional databases and connections:
   a. Create the databases.
   b. Authorize access to the databases.
   c. Optional: If you want your databases to participate in globally coordinated transactions, configure the databases for global coordination.
   d. Enable an ODBC connection or enable a JDBC connection to each database.
3. Optional: If you want your databases to participate in globally coordinated transactions, configure the environment for global coordination.

Databases overview

WebSphere Message Broker brokers use databases for two purposes: *broker databases* are used to store internal data about the broker, and *user databases* contain your business data. You must create and configure the broker database before you can create a broker. If you have user databases, you must configure them also before you can access them from your message flow.

WebSphere Message Broker supports the databases that are listed in Supported databases for both broker and user databases. If you configure your message flows to access user databases, you cannot access some of the data types that are supported by these databases. The supported data types are defined in Data types of values from external sources.

Broker databases

A broker stores configuration and control information in its database. You must create the broker database before you can create the broker because when the broker is created, the broker's tables are automatically created and initialized.

Linux UNIX If you create a broker on Linux or UNIX systems, depending on your operating system, you can create the broker database in DB2, Oracle, SQL Server, or Sybase.

Windows On Windows, you can create the broker database in DB2, Oracle, SQL Server, Sybase, or Derby. See Supported databases to check which databases are supported on your operating system.

If you create a 64-bit execution groups in the broker, the broker database must be a 64-bit database instance.

When you create a broker, the required database tables are created in the default schema that is associated with the user ID used to access the database. Specify this user ID when you run the mqsicreatebroker command.

- For DB2 and Oracle, the default behavior is for the schema name to be the same as the user ID that is used to access the database.
- For Sybase and SQL Server, the typical behavior is to use the database-owning schema, *dbo*.

You can create a database schema for each broker, or you can configure brokers to share a database schema if all brokers are at the same version.

WebSphere Message 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.

The size of the broker database is not fixed; it depends on the complexity of your message flows and message sets. If you develop message flows that support many publishers or subscribers, you might need to increase your initial sizings.

When you have created a broker database, you must enable a connection from the broker to the database. On all platforms, the broker connects to databases using ODBC. For 32-bit brokers (Windows and Linux on x86), you must always enable a 32-bit ODBC connection. For 64-bit brokers (all other platforms), you must always enable a 64-bit ODBC connection. ODBC drivers are supplied with WebSphere Message Broker.

For more information about enabling 32-bit and 64-bit connections to the broker database see "Broker database connections" on page 112

User databases

User databases are the databases in which you store the business data that is processed by message flow applications. You can create user databases using any of the database managers that you can use for broker databases. Additional local and remote database managers might also be supported for your computer. For more information, see Supported databases and Database locations.

You must set up connections to the user databases so that the broker can access the databases on behalf of its deployed message flows. Both ODBC and JDBC connections are supported; some restrictions apply on some platforms, as described in the topics in this section. ODBC drivers are supplied and installed with the broker component. JDBC drivers are not supplied by WebSphere Message Broker; you must obtain these files from your database vendor. Supported drivers are listed in Supported databases.

ODBC connections are specific to either 32-bit or 64-bit mode. You must set up connections to the user databases depending on whether the message flows that access the user databases are deployed to 32-bit or 64-bit execution groups, and whether the message flow transactions are globally coordinated by a 32-bit or a 64-bit queue manager. JDBC connections are not dependent on 32-bit or 64-bit mode except where stated.

For information about 32-bit and 64-bit connections to user databases see "User database connections" on page 114.

Databases created by the Default Configuration wizard

On Windows or Linux on x86, if you use the Default Configuration wizard to create the Default Configuration, the wizard automatically creates a broker database for the broker. On Linux systems, the wizard creates the broker database using DB2; on Windows, if DB2 is not installed, the wizard uses the Derby database manager by default, although you can choose to use DB2 if it is installed.

Broker database connections

Create ODBC connections from each broker to its database.

Broker components and execution groups read and write data about internal operations to a broker database. The number of connections needed depends on the actions of the message flows that the broker processes. Each broker needs the following connections:

- Five for internal broker threads.
- One for each Publish/Subscribe neighbor, if you are using retained publications and the topology has been deployed.
- One for each message flow thread that contains a Publication node, if you are using retained publications.
- 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, include all brokers in your calculations.

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

The broker also opens connections to WebSphere MQ queues and to user databases when it needs to use them, and these connections remain open until:

The connection has been idle for one minute

The broker is stopped

Linux UNIX Windows On Linux, UNIX, and Windows systems, to avoid breaking global coordination, database connections are released only for message flows that are not globally coordinated.

z/OS On z/OS, database connections for globally coordinated message flows are also released if the database has not been accessed for one minute.

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. Increase this parameter and the associated parameter **maxagents** to new values based on your calculations, if appropriate.

32-bit and 64-bit considerations

On all platforms that support 32-bit execution groups, the broker needs 32-bit access to the data source. You must therefore always define a 32-bit ODBC data source name (DSN) for the broker to connect to the broker database; this definition is required even if the broker has a 64-bit database, in which case you must present the broker with an environment that provides a 32 bit-compatible interface to the database (see “Setting your environment to support 32-bit access to databases” on page 143).

Execution groups on a broker must also be able to connect to the broker database. A 32-bit execution group on a 32-bit broker can connect to the broker database using the same 32-bit DSN definition that the broker uses. A 64-bit execution group on a 32-bit broker, however, needs a 64-bit ODBC connection to be able to connect to the broker database, therefore you must define a 64-bit ODBC DSN for the broker database in addition to the broker’s 32-bit DSN definition.

When message flow transactions are globally coordinated, the queue manager must also be able to connect to the broker database; if the transactions are globally coordinated by a 64-bit queue manager (all WebSphere MQ Version 6 queue managers on 64-bit platforms are 64-bit), you must define a 64-bit ODBC DSN for the broker database, even if the broker and the execution group are 32-bit applications.

On HP-UX on Itanium, Linux on POWER®, Linux on System z®, and Solaris on x86-64, a 64-bit broker supports only 64-bit execution groups, therefore an

execution group can access the broker database using the same 64-bit DSN definition that the broker uses; a 32-bit DSN definition is not required.

For 32-bit and 64-bit considerations when connecting to user databases, see "User database connections."

For help when you are deciding whether to create 32-bit DSNs, 64-bit DSNs, or both, for your broker database, see "Enabling ODBC connections to the databases" on page 127.

User database connections

User databases contain your business data that is written and accessed by deployed message flows. You must create connections from the broker to the user database using ODBC or JDBC.

The broker requires a database connection for each data source name (DSN) that is referenced in the message flow, even if different DSNs resolve to the same physical database.

The number of connections to a user database that a broker requires depends on the actions of the message flows that access the database. For each message flow thread, a broker that accesses a user database makes one connection for each data source name (DSN). If a different node on the same thread uses the same DSN, the same connection is used, unless a different transaction mode is used, in which case another connection is required. For further information about transactions, see Database connections for coordinated message flows.

When you start a broker, and while it is running, it opens connections to WebSphere MQ queues and to databases. The broker makes the connections when it needs to use them, and they remain open until:

The message flow has been idle for one minute

The message flow is stopped

The broker is stopped

Database connections from message flows that are not globally coordinated are released when a flow has no work. For example, a connection is released if the message flow input queue has no messages, and the database has not been accessed for one minute.

Linux UNIX Windows On Linux, UNIX, and Windows systems, to avoid breaking global coordination, database connections are released only for message flows that are not globally coordinated.

z/OS On z/OS, database connections for globally-coordinated message flows are also released if the database has not been accessed for one minute.

If you are using the same database for business data and for broker internal data, add the two connection requirements together when you calculate how many connections are required. For details of broker database connection requirements, see "Broker database connections" on page 112.

If you are using DB2 for your database, the default action 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 that the connections that

the broker might require exceeds the value for **maxappls**, increase this and the associated parameter **maxagents** to new values based on your calculations.

If you are using another database, check the database documentation for information about connections and the limits or restrictions that might apply.

When a message flow is idle, the execution group periodically releases database connections. Therefore, connections held by the broker reflect the broker's current use of these resources. This situation allows the broker to respond when a database quiesces, provided that the database manager supports quiescing. Not all databases support the quiesce function, and not all databases quiesce in the same way. Check your database documentation for information about database quiescing.

32-bit and 64-bit considerations

If you are creating ODBC connections to your user databases, ensure that you correctly create 32-bit, 64-bit, or both, connections for each DSN. If you are creating JDBC connections, their use is independent of 32-bit or 64-bit mode.

If a message flow that accesses the user database is deployed to a 32-bit execution group, you must define a 32-bit ODBC data source name (DSN) for the user database so that the broker can connect to the user database on behalf of the message flow.

If the message flow is deployed to a 32-bit execution group and the message flow transactions are globally coordinated by a 64-bit queue manager (all WebSphere MQ Version 6 and Version 7 queue managers on 64-bit platforms are 64-bit), you must define both a 32-bit ODBC DSN and a 64-bit ODBC DSN for the user database. You must also define a 64-bit ODBC DSN for the broker database; see "Broker database connections" on page 112.

If a message flow that accesses the user database is deployed to a 64-bit execution group, define a 64-bit ODBC DSN for the user database so that the broker can connect to the user database on behalf of the message flow. You cannot use a 32-bit queue manager to globally coordinate a message flow that is deployed to a 64-bit execution group, therefore you do not need to define a 32-bit ODBC DSN for the user database.

For 32-bit and 64-bit considerations when connecting to the broker database, see "Broker database connections" on page 112.

For help when you are deciding whether to create 32-bit DSNs, 64-bit DSNs, or both, for your user databases, see "Enabling ODBC connections to the databases" on page 127.

Creating the broker and user databases

Before you can create a broker, you must create a database in which the broker can store its internal data. If your message flows create, update, retrieve, or delete application and business data in one or more user databases, create the databases before you deploy the message flows to a broker.

If you create more than one broker, you can set their properties to share, and store their tables in, a single database.

For information about which databases you can use, see Supported databases.

z/OS For information about setting up databases on z/OS, see “DB2 planning on z/OS” on page 165 and “Customizing the z/OS environment” on page 152.

To create the databases on distributed systems:

1. If you are creating Oracle databases for 32-bit brokers on UNIX systems, run the mqsi_setupdatabase command before you create a database. For details, see the “mqsi_setupdatabase command” on page 653. Do not complete this step for any other database manager, or for Oracle databases on Linux or Windows.
2. Create the databases that you require. Choose a unique name for each broker database, for example WBRKBKDB, and keep a note of it for when you create the broker.
   - For DB2, follow the instructions in “Creating a DB2 database on Windows” or “Creating a DB2 database on Linux and UNIX systems” on page 117.
   - For Derby, see the “mqsicreatedb command” on page 536. Derby database support is described in “Using Derby databases on Windows” on page 119.
   - For other supported database managers, see the documentation supplied with that database manager.
3. If you are creating Sybase databases on AIX, run the Sybase profile before you run mqsiprofile. For more information about using mqsiprofile to initialize command environments on Linux and UNIX platforms, see Command environment: Linux and UNIX systems.

You have now created a database for your broker or business data.

Next: If you have been following the instructions in “Configuring broker and user databases” on page 109, the next task is “Authorizing access to broker and user databases” on page 122.

Creating a DB2 database on Windows

Use the mqsicreatedb command or the DB2 Control Center to create a DB2 database.

When you create a broker, you specify the user name and password that are used to connect to the broker database. The process of creating a broker creates the required broker tables in the user’s schema within the broker database if the tables do not already exist.

DB2 authenticates the user name by using the operating system’s user management; you do not have to define the user name to DB2 itself.

Using the mqsicreatedb command:

To create a database by using the mqsicreatedb command, enter the command at the command line for each database that you want to create, specifying the appropriate parameters. You must provide a user name and password that is known to DB2. Specify that you want to create a DB2 database (not a Derby database).

For more information about these and other parameters, and for examples of command use, see the “mqsicreatedb command” on page 536.

Using the DB2 Control Center:

To create a database by using the DB2 Control Center:

1. Start the DB2 Control Center by clicking **Start → Programs → IBM DB2 → General Administration Tools → Control Center**
2. Expand **All Systems** in the object tree in the DB2 Control Center until you find **Databases**.
3. Right-click **Databases** and click **Create Database → Standard**.
4. 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`.
5. Click **Done**.
6. When you have completed these steps for each database you create, click **OK**.
7. Increase the database heap size to ensure that it is sufficient for the broker. This task is described in "Changing the database heap size on DB2 broker databases" on page 119.

Using other DB2 options:

If you prefer, you can use any other method supported by DB2 to create a database (including command line or batch files); refer to the DB2 documentation for details of how to do this.

If you use the DB2 command line to create the databases, you must bind the db2cli package to the database. You do not have to do this if you used the DB2 Control Center wizard, the mqsicreatedb command, or if you created the broker with the Default Configuration wizard.

To bind the package to the database:

1. Open a DB2 Command Line Processor window.
2. Connect to the database using the following command:

   ```
   db2cmd db2 CONNECT to YourDatabaseName
   ```

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 quotation marks.

   ```
   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 each database.

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, the next task is "Changing the database heap size on DB2 broker databases" on page 119.

Creating a DB2 database on Linux and UNIX systems

Create a DB2 database for a broker or user database.

When you create a broker, you specify the user name and password that are used to connect to the broker database. The process of creating a broker creates the necessary broker tables in the user's schema within the broker database, if the tables do not already exist.

DB2 authenticates the user name by using the operating system's user management; you do not have to define the user name to DB2 itself.

On 64-bit platforms, WebSphere MQ (Version 6.0 and 7.0) performs all transaction coordination in 64-bit mode. If you require XA coordination, all data sources connected to DB2 from message flows in both 32-bit and 64-bit execution groups must connect to 64-bit DB2 instances.

To create a DB2 database on Linux or UNIX:

1. Log on as root.
2. Create a database instance. Use the commands shown here for guidance for the different platforms.
   - On AIX:

     ```
     /usr/lpp/db2_09_01/instance/db2icrt -u fence_userID username
     ```

   - On Linux, Solaris, or HP-UX:

     ```
     /opt/IBM/db2/V9.1/instance/db2icrt -u fence_userID username
     ```

   The *username* that you specify on this command determines the nominated owner of the database instance. 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 topic assume that you are logged on as *username*, and use the tilde (~) character to indicate this user ID 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.

   The *fence_userID* refers to the user ID under which stored procedures run. You can specify a different ID to the instance owner ID for the database for extra security and protection, which is achieved because the stored procedure runs under a different ID, and therefore in a different process, to the database instance itself.

   For further explanation of database ownership, refer to the DB2 library.
3. Log on as *username*.
4. Create a database (in this example called `WBRKBKDB`) by using the following commands (on some platforms, an explicit path name is required).

   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 that it is sufficient for the broker. This task is described in “Changing the database heap size on DB2 broker databases” on page 119.
6. If you are using 32-bit execution groups, set the environment variable `MQSI_LIBPATH32` to include the 32-bit database libraries. See “Enabling ODBC connections to the databases” on page 127 for further information about how to set this variable.

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

Next: If you have been following the instructions in “Configuring broker and user databases” on page 109, the next task is “Changing the database heap size on DB2 broker databases” on page 119.

Changing the database heap size on DB2 broker databases

Increase the value of the database heap size parameter for each broker database to ensure that the heap size is sufficient.

Each database in a DB2 instance is associated with a heap (temporary storage) but the default size of the heap (measured in 4 KB pages) is too low for the broker to use. Therefore, if you create broker databases in DB2, you must increase the heap size to at least 900, which gives the broker database 3600 KB of storage space (that is, 900 multiplied by 4 KB pages).

Using the DB2 Control Center:

To change the database heap size using the DB2 Control Center:

1. Start the DB2 Control Center.
   - On Windows, click **Start → Programs → IBM DB2 → General Administration Tools → Control Center**
   - On Linux and UNIX , enter the following command (ensure that you have already run the DB2 profile so that the shell is running the DB2 environment):

     ```
     db2cc
     ```
2. For each broker database that you have created:
   a. In the tree, right-click the database name, then click **Configure Parameters**.
   b. In the Database Configuration window, click **Performance → DBHEAP**.
   c. Click the cell in the **Value** column, then click the button labeled **...**
   d. In the Change Database Configuration Parameters window, set the value of the DBHEAP parameter to at least 900 (the maximum value that you can enter is 60000), then click **OK**. The new value is displayed in the cell in the **Pending Value** column in the Database Configuration window.
   e. Click **OK**. The new value is applied and a message is displayed to show that the heap size was changed successfully.
   f. Click **Close** to close the message and return to the main DB2 Control Center window.

The heap size of the broker database is changed.

Using the DB2 command line:

1. Open a DB2 command window.
2. Enter the following command. In this example, WBRKBKDB is the name of the broker database, and `900` is the number of 4 KB pages:

   ```
   db2 update database configuration for WBRKBKDB using dbheap 900
   ```

The heap size of the broker database is changed.

Next: If you have been following the instructions in “Configuring broker and user databases” on page 109, the next task is “Authorizing access to broker and user databases” on page 122.

Using Derby databases on Windows

Set up your environment to access a Derby database.

Derby refers to the DB2 database product that is based on the Apache Derby open source project from the Apache Software Foundation. Derby database support is embedded in the broker component on Windows only.

This topic describes the processes, services, IP ports, and database files that are required to support Derby on Windows.

Security

The Derby database has no associated security controls, and no optimizations have been performed. For these reasons, do not use Derby in a production environment.

DB2 Run-Time Client use

A broker uses ODBC to connect to 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 a broker and the Derby database. It does not provide a DB2 database and therefore does not consume the resources that a full DB2 installation typically requires.

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

You must create and start a network server to enable access to Derby databases through ODBC 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. It starts automatically when Windows starts, and starts the network server. The service runs under the user name that you supplied with the mqsicreatedb command.

All Derby databases that you create using the mqsicreatedb command are served by one instance of the Database Instance Manager 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 port when you create a 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.

Run the command `mqsilist DatabaseInstanceMgr` to produce a list all of the databases that have been created by the mqsicreatedb command. You can remove the Database Instance Manager and the network server after the last Derby database has been deleted, using the mqsideletedb command.

If you change the password for 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 Database Instance Manager component.

Multiplicity (brokers, Database Instance Managers, installations, databases)

The number of databases that you can create with the mqsicreatedb command is limited only by availability of system resources. A maximum of one Database Instance Manager is created irrespective of how many databases have been created. If you have installed multiple instances of WebSphere Message Broker on a single computer, all installations use a single instance of the Database Instance Manager component.

Removing databases and the Database Instance Manager component

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

Issuing database commands on Windows

On Windows, use special commands to create and delete databases for use by a broker or by applications.

Only DB2 and Derby databases are supported with the supplied commands:

- "mqsicreatedb command" on page 536
- "mqsideletedb command" on page 563
- "mqsichangedbimgr command" on page 420

The mqsilist command lists the databases that have been created by the mqsicreatedb command. Only databases created by the mqsicreatedb command can be deleted by 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 by using the mqsilist command and specifying the parameter **DatabaseInstanceMgr**.

Use the mqsisetdbparms command 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 topic applies only to the mqsicreatedb, mqsideletedb, and mqsichangedbimgr commands.

Supported database engines

If DB2 version 8.1 Fix Pack 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 is DB2 unless only DB2 Run-time Client Version 8.2 is installed, in which case a Derby database is created.

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. No process or Windows service is required for the Database Instance manager component, and if you start the component it is not 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. At most one Database Instance manager Windows service exists, even if you install WebSphere Message Broker 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 command `mqsilist DatabaseInstanceMgr6` lists all the databases that have been created by the mqsicreatedb command on this Windows computer. Use the mqsichangedbimgr command to change the user name and password under which the Database Instance manager service runs. Run this command only if passwords change, or if user names are updated after the initial installation and configuration. For more information, see "Using Derby databases on Windows" on page 119.

Creating and deleting databases

Use the mqsicreatedb command 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. When the database is created (in either DB2 or Derby), the ODBC data source name (DSN) is also created (with the same name).

Because the data source names and the Database 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 Message Broker. The mqsicreatedb command warns you if you try to do so. A database created by the mqsicreatedb command can be deleted by the mqsideletedb command, even if that database is in use by a broker. See the command descriptions for more information.

Authorizing access to broker and user databases

When you have created a broker or user database, you must authorize the broker and its execution groups to access it.

Before you start: create the databases.

When you run the mqsicreatebroker command, you must specify at least one user ID for runtime authorization (the service user ID); you can optionally specify a second user ID that the broker uses when connecting to databases (the data source user ID). If you do not specify a separate data source user ID for connecting to databases, the broker uses its service user ID for database access as well.

You specify the service user ID and its password with the **-i** and **-a** parameters, and the optional database connection user ID and password with the **-u** and **-p** parameters.

To assign a specific user ID and password for a particular database, you can set up or change the authorization by using the mqsisetdbparms command.

If you want to change the service user ID or password, or the data source password, after you have created the broker, use the mqsichangebroker command; you cannot change the data source user ID.

The user ID that the broker uses to access databases must have the following authorizations:

- The user ID must be authorized to connect to the database.
- Before you can create a broker, the user ID must have authorization to create tables in the broker database.
- The user ID must have appropriate privileges on the user database objects that are accessed by the message flow application; for example, tables, procedures, and indexes.

If you expect to deploy message flows that participate in globally coordinated transactions to a broker, you must provide additional authorization. For more information, see "Configuring databases for global coordination of transactions" on page 124.

The way that you authorize access depends on the database manager that you are using, and the platform on which you have created the database. The instructions might also vary from release to release of a single database. Consult your database administrator, or see the documentation for the appropriate database when you perform this task.

The following sections provide examples of the steps that you can take to provide the required authorization for specific databases:

- "DB2 authorization"
- "Oracle authorization" on page 124

DB2 authorization

To authorize access to a DB2 database, you can use either the DB2 Control Center or the DB2 command line:

- To use the DB2 Control Center:
  1. Start the DB2 Control Center.
  2. Expand the object tree until you find the database that you created.
  3. Expand the tree under the database then click the **User and Group Objects** folder. The **DB Users** and **DB Groups** folders are displayed in the right pane.
  4. In the right pane, right-click the **DB Users** folder then click **Add**. The Add User notebook opens.
  5. From the list, click the user ID that you want to authorize to access the database (for example, `mqsiuid`). The user ID that you select must be the user ID that you specify to be used for access to this database when you create the broker or run the mqsisetdbparms command. The user ID must exist on the operating system before you can select it; if it does not exist, define the user ID on the operating system.
  6. Select the appropriate options from the choices in the dialog box that is labeled **Choose the appropriate authorities to grant to the selected user** for the database. The following options are available:
     - Connect database
     - Create tables
     - Create packages
     - Register functions to run in database manager's process
  7. Click **OK**. The authorities are granted. The dialog box closes.
  8. Close the DB2 Control Center.

- To use the DB2 command line:
  1. Open a DB2 command window.
  2. Connect to the database with a user ID that has DB2 system administration (SYSADM or DBADM) authority. Substitute the correct database and ID in the following example command:

     ```
     db2 connect to broker_db user SysAd_id
     ```

  3. 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 the following example command, if you are not using `mqsiuid`:

```
db2 grant connect, createtab, bindadd, create_external_routine on
   database to user mqsiuid
```

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, the next task is "Configuring databases for global coordination of transactions."

Oracle authorization

You must have database administrator (DBA) privileges to authorize access to an Oracle database.

To authorize access to an Oracle database:

1. Log on as the Oracle database administrator (DBA) to the database using SQL*Plus.
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. If you are authorizing access to the broker database, give the user ID sufficient privilege to allow the creation of, and updates to, the broker tables:

   ```
   GRANT CREATE SESSION TO dbid;
   GRANT CREATE TABLE TO dbid;
   ```

3. If appropriate, increase the quota (disk space) available for table spaces associated with this database.

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, the next task is "Configuring databases for global coordination of transactions."

Configuring databases for global coordination of transactions

If your message flow interacts with a user database, and you want to globally coordinate the updates made to the database with other actions within the message flow, configure your databases for global coordination.

Before you start: Create your database and authorize access to it.

If you restart a user database while the broker is still running, you must also restart the broker. The broker cannot detect that the database has stopped, and WebSphere MQ therefore retains its old connections to the database. When the database starts again, the broker tries, and fails, to use these connections.

To configure databases for coordinated message flows, follow the instructions relevant to your database manager:

- DB2
- Informix®

- Oracle
- Sybase

Configuring DB2 for global coordination of transactions

You must complete these steps for databases that you connect to with an ODBC or a JDBC connection.

You must have database administrator (DBA) privileges to perform the following tasks.

To configure DB2 database instances for global coordination of transactions:

1. Windows Linux Windows and Linux on x86 systems only: for each 32-bit instance that will be involved in the global coordination, run the following commands to set the Transaction Process Monitor name (TP_MON_NAME) to MQ:

   ```
   db2 update dbm cfg using TP_MON_NAME MQ
   db2stop
   db2start
   ```

   UNIX Linux On Linux and UNIX systems (except for Linux on x86), do not set this variable for 32-bit or 64-bit instances.
2. Ensure that you have adequate connection resources and find out from the broker administrator whether the broker will use TCP/IP or shared memory to connect to databases.

   To use TCP/IP connections, see the example in the section about message SQL1224N in Resolving problems when using databases.

   To enable extended shared memory:

   a. On the DB2 server, run the following commands:

      ```
      export EXTSHM=ON
      db2set DB2ENVLIST=EXTSHM
      db2stop
      db2start
      ```

   b. Ensure that shared memory support is enabled in the broker environment. For more information, see “Configuring global coordination with DB2 using a 32-bit queue manager” on page 285 or “Configuring global coordination with DB2 using a 64-bit queue manager” on page 287.
3. If you are connecting a broker on a distributed platform to a DB2 instance on z/OS, you must configure DB2® Connect™ to enable support for global coordination. Ensure that you have already configured a DB2 alias to represent the database using DB2 Connect.

   Perform the following tasks on the system that hosts the broker:

   a. Turn on the Connection Concentrator by configuring the DB2 database manager’s configuration parameters so that the value of the **MAX_CONNECTIONS** parameter is greater than the value of the **MAX_COORDAGENTS** parameter:

      ```
      db2 update dbm cfg using MAX_CONNECTIONS max_connections_value
      ```

      where *max_connections_value* is greater than the existing value of the **MAX_COORDAGENTS** parameter.

   b. Define the SPM name as the name of the system that hosts the broker:

      ```
      db2 update dbm cfg using SPM_NAME host_name
      ```

      where *host_name* is the TCP/IP name of the system that hosts the broker.

c. Stop, then restart DB2 on the system that hosts the broker to apply the changes:

```
db2stop
db2start
```

DB2 Connect is now configured to enable global coordination of message flows that are deployed to the broker (on a distributed platform) and that access DB2 on z/OS.

The DB2 database instances are now configured for global coordination.

Next: See "Configuring global coordination of transactions (two-phase commit)" on page 281.

Configuring Informix for global coordination of transactions

You must complete these steps for databases that you connect to with an ODBC connection only.

You must have database administrator (DBA) privileges to perform the following task.

To configure Informix databases for global coordination of transactions ensure that the databases are created with a log option; for example:

```
CREATE DATABASE mydb WITH LOG
```

The Informix databases are now configured for global coordination.

Next: See "Configuring global coordination of transactions (two-phase commit)" on page 281.

Configuring Oracle for global coordination of transactions

You must complete these steps for databases that you connect to with an ODBC connection only.

You must have database administrator (DBA) privileges to perform the following tasks.

To configure Oracle databases for global coordination of transactions:

1. If you are using WebSphere MQ Version 6 to globally coordinate transactions, ensure that the JAVA_XA package is present on the Oracle database. Typically you can perform this task by running the scripts `initjvm.sql` and `initxa.sql`, which are supplied with the Oracle installation; your database administrator can tell you if these scripts have been run. For more information, see the Oracle product documentation.
2. Ensure that the user ID that the broker uses to access the database has the necessary Oracle privileges to access the DBA_PENDING_TRANSACTIONS view. You can grant the required access using, for example, the following Oracle SQLPLUS command:

   ```
   grant select on DBA_PENDING_TRANSACTIONS to userid;
   ```

The Oracle databases are now configured for global coordination.

Next: See "Configuring global coordination of transactions (two-phase commit)" on page 281.

Configuring Sybase for global coordination of transactions

You must complete these steps for databases that you connect to with an ODBC connection only.

You must have database administrator (DBA) privileges to perform the following tasks.

To configure Sybase databases for global coordination of transactions, ensure that the user ID that the broker uses to access the database has been granted the Sybase role of `dtm_tm_role`.

The Sybase databases are now configured for global coordination.

Next: See "Configuring global coordination of transactions (two-phase commit)" on page 281.

Enabling ODBC connections to the databases

Set up the resources and environment that the broker requires for Open Database Connectivity (ODBC) connections to broker and user databases on distributed systems.

The broker uses ODBC to connect to the broker database. You must define ODBC data source names (DSNs) for the broker database on each computer that hosts a broker. You can configure both ODBC and Java Database Connectivity (JDBC) connections for access to user databases:

- To set up ODBC connections to the broker database, and to user databases, follow the instructions in this section.
- To set up JDBC connections to user databases, see "Enabling JDBC connections to the databases" on page 143.
- On z/OS systems, see Data sources on z/OS for information about enabling connections to databases. You do not have to follow the tasks described in this section.

Linux UNIX On Linux and UNIX systems, the actions that you must take to define the connections between the broker and the broker database depend on whether the broker core components operate in 32-bit or 64-bit mode. You must define either a 32-bit ODBC DSN, or a 64-bit ODBC DSN, as appropriate, for the broker to connect to the broker database.

All execution groups that you create on a broker must also be able to connect to the broker database. If the execution group matches the operational mode of the broker core, the execution group can connect to the broker database using the same DSN definition that the broker uses. If a 64-bit broker also supports 32-bit execution groups (for example, on AIX), you must also define a 32-bit ODBC connection to support connections from 32-bit execution groups to the broker database. This 32-bit connection is required in addition to the 64-bit ODBC definition, even if you do not create 32-bit execution groups.

Because 64-bit connections need 64-bit database instances, you can use 32-bit database instances only on computers with a 32-bit broker core. Migrate all 32-bit database instances to 64-bit instances on all platforms that support 64-bit operation.

When you define the DSNs, consider the following two factors that determine whether you must define a 32-bit DSN for the database, a 64-bit DSN, or both:

- Whether the execution group and the database instance operate in 32-bit or 64-bit mode
- Whether you plan to globally coordinate message flow transactions

Linux UNIX On Linux and UNIX systems, DSNs are defined in a plain text file on the computer that hosts the broker:

- Define 32-bit DSNs in the file that is referenced by your ODBCINI32 environment variable.
- Define 64-bit DSNs in the file that is referenced by your ODBCINI environment variable.

Set your ODBCINI environment variable to point to the ODBC `.ini` file that contains the DSN that is defined for the broker to connect to the broker database and all user databases; the file must be a copy of `odbc64.ini` on all platforms except Linux on x86 where it must be a copy of `odbc32.ini`. If you need to define 32-bit DSNs in your copy of `odbc32.ini` on 64-bit platforms, you must also set your ODBCINI32 environment variable to point to your copy of `odbc32.ini`. Delete the ODBCINI64 variable if it exists; it is not required by Version 6.1 brokers.

For more information about the 32-bit and 64-bit considerations, see "Broker database connections" on page 112 and "User database connections" on page 114.

When you have defined the appropriate DSNs, you must also configure the environment so that the broker can access the correct database libraries; for more information, see "Setting your environment to support 32-bit access to databases" on page 143.

The sample ODBC `.ini` files that are supplied, and the information contained in these configuration topics, include all the connection parameters that are supported for connections to your databases. Additional parameters that are provided by the DataDirect drivers are not tested or supported in a broker environment; consider your requirements carefully before specifying other parameters in your tailored ODBC `.ini` files.

To enable connections on distributed systems:

1. Define the ODBC DSNs according to your platform:

   Windows **On Windows:**
   Windows provides only 32-bit support. Follow the instructions in "Connecting to a database from Windows systems" on page 130.

   Linux UNIX **On Linux and UNIX systems:**
   Depending on your broker configuration, for each database you might need to define a 32-bit ODBC DSN, a 64-bit ODBC DSN, or both.

   Use the following tables to check which DSNs you must define, and follow the links for the appropriate instructions.

	32-bit execution group	**64-bit execution group**
32-bit broker Linux on x86	Broker database: 32-bit User database: 32-bit	Not possible

	32-bit execution group	64-bit execution group
64-bit broker AIX, Solaris on SPARC, HP-UX on PA-RISC, Linux on x86-64	Broker database: 32-bit and 64-bit User database: 32-bit	Broker database: 64-bit User database: 64-bit
64-bit broker HP-UX on Itanium, Linux on POWER, Linux on System z, Solaris on x86-64	Not possible	Broker database: 64-bit User database: 64-bit

The following tables provide links to topics for connecting databases when you are using global coordination and a 64-bit queue manager. All WebSphere MQ Version 6 and Version 7 queue managers on 64-bit platforms run in 64-bit mode.

	32-bit execution group	64-bit execution group
64-bit broker AIX, Solaris on SPARC, HP-UX on PA-RISC, Linux on x86-64	Broker database: 32-bit and 64-bit User database: 32-bit and 64-bit	Broker database: 64-bit User database: 64-bit
64-bit broker HP-UX on Itanium, Linux on POWER, Linux on System z, Solaris on x86-64	Not possible	Broker database: 64-bit User database: 64-bit

The following table provides links to topics for connecting databases when you are using global coordination and a 32-bit queue manager. All WebSphere MQ Version 6 queue managers on 32-bit platforms run in 32-bit mode.

	32-bit execution group	64-bit execution group
32-bit broker Linux on x86	Broker database: 32-bit User database: 32-bit	Not possible
64-bit broker All other platforms	Not possible	Not possible

You have now configured the ODBC DSN for your broker database and the ODBC DSNs for your user databases.

2. Configure the environment for issuing console commands and for running the broker so that it can access the required database libraries. For more information, see “Setting your environment to support 32-bit access to databases” on page 143.

You have now enabled the broker to make connections to the broker database and to your user databases.

Next: If you have been following the instructions in “Configuring broker and user databases” on page 109, and you use Sybase for your broker database, the next task is “Using retained publications with a Sybase database” on page 151 (optional). If you do not use Sybase for your broker database, the next task is “Configuring global coordination of transactions (two-phase commit)” on page 281 (optional).

Connecting to a database from Windows systems

To enable a broker to connect to a database, define the ODBC data source name (DSN) for the database.

Before you start: check that you have set up your environment so that the broker can connect to the database. Most database managers set up the required environment when you install, but others supply a database profile that you must run. For information about environments and running database profiles, see Setting up a command environment: Windows platforms.

Configure an ODBC data source using the ODBC Data Source Administrator:

1. Click **Start** → **Control Panel** → **Administrative Tools** → **Data Sources (ODBC)**.
2. Click the **System DSN** tab and click **Add**.
3. Complete the steps in the following sections for the databases that you are working with.

   If you need more information about a particular database product, see the product-specific documentation.

The Default Configuration wizard and the database commands to create a broker, or a database, on Windows, automatically create the ODBC data source names (DSNs) for you.

DB2 UDB

Define a data source for DB2 UDB:

1. Select the driver `IBM DB2 ODBC DRIVER`.
2. Enter the data source name (DSN) and description.
3. Select the correct database alias from the list.
4. Click **Finish** to save your definition.
5. Click **OK** to close the ODBC Data Source Administrator.

You must register the data source as a system data source.

If you prefer, you can use the Configuration Assistant instead of the ODBC Data Source Administrator:

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**. Select the system data source option.
5. Click **Finish**.
6. The Test Connection dialog opens automatically and you can test the various connections.

Informix Dynamic Server

Define a data source for Informix Dynamic Server:

1. Select the driver `IBM INFORMIX ODBC DRIVER`.
2. On the **Connection** tab, specify:
   - The Informix server name.
   - The machine host name.
   - The Informix network service name (as defined in the services file).
   - The network protocol (for example, `olsoctcp`).
   - The Informix data source name.

- The user identifier to access the data source within.
- The password for that user identifier.

3. Click **Apply**.
4. Click **Test Connection** to check your supplied values.
5. Click **OK** to close the ODBC Data Source Administrator.

Microsoft SQL Server

Define a data source for Microsoft SQL Server:

1. Select the driver for the version of SQL Server that you are using:
   - `SQL Server` for SQL Server 2000. The driver level must be Version 3.60, or later.
   - `SQL Native Client` for SQL Server 2005.
2. Specify a name and description.
3. Select the correct server from the list.
4. Click **Finish** to save your definition.
5. Click **OK** to close the ODBC Data Source Administrator.

Oracle

Define a data source for Oracle:

1. Select the driver `MQSeries DataDirect Technologies 5.30 32-BIT Oracle`.

   The ODBC Oracle Driver Setup dialog box opens.
2. On the **General** tab:
   a. 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).
   b. Select the appropriate Oracle client version from the list.
3. On the **Advanced** tab:
   a. Select **Enable SQLDescribeParam**.
   b. Select **Procedure Returns Results**. The resultant ODBC definition in the Windows registry has a string value called **ProcedureRetResults** with the value 1.
4. Click **OK** to close the ODBC Data Source Administrator.
5. Click **Start** → **Run**.
6. Type `REGEDIT` in the **Open** field and click **OK**.
7. In the Registry Editor, navigate to the correct location.
   - On Windows 32–bit editions: `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI`
   - On Windows 64–bit editions: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ODBC\ODBC.INI`
8. Expand that location, and right-click your DSN entry. Select **New** → **String Value**.
9. Specify **WorkArounds** for the string name.
   a. Right-click **WorkArounds**.
   b. Select **Modify**.
   c. Type the data value `536870912`.
10. Close the Registry Editor.

Sybase Adaptive Server Enterprise

Define a data source for Sybase Adaptive Server Enterprise:

1. Select the driver `MQSeries DataDirect Technologies 5.30 32-BIT Sybase Wire Protocol`.
2. Enter the DSN name, description, and network address of the server.
3. Select **Enable Describe Parameter**. This parameter is on the **Advanced** tab.
4. Ensure the **Prepare Method** setting is `1 - Partial`. This parameter is on the **Performance** tab.
5. Click **Start** → **Run**.
6. Type `REGEDIT` in the **Open** field and click **OK**.
7. In the Registry Editor, navigate to the correct location:
   - On Windows 32–bit editions: `HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\ODBC.INI`
   - On Windows 64–bit editions: `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ODBC\ODBC.INI`
8. Expand that location, and right-click your DSN entry. Select **New** → **String Value**. Specify **SelectUserName** for the string, and set the value to 1.
9. Right-click your DSN again, and select **New** → **String Value**. Specify **EnableSPColumnTypes** for the string, and set the value to 2.
10. Close the Registry Editor.

You have now configured your ODBC data source names on Windows.

Next: Configure the environment for issuing console commands, and for running the broker, so that the broker can access the required database libraries. For more information, see "Setting your environment to support 32-bit access to databases" on page 143.

Connecting to a database from Linux and UNIX systems: 32-bit considerations

To enable a broker to connect to a database, define the ODBC data source name (DSN) for the database.

Before you start:

- Ensure that the database has been created, see "Creating the broker and user databases" on page 115.
- Ensure that the broker is authorized to access the database, see "Authorizing access to broker and user databases" on page 122.
- Check that you have set up your environment so that the broker can access the database; you might have to run a database profile supplied by the database vendor. For further information, see Setting up a command environment: Linux and UNIX systems.

On Linux and UNIX systems, an ODBC Driver Manager exists, but no graphical application is available to help you configure the ODBC DSNs. To enable a 32-bit ODBC connection, define each database as a DSN in a plain text file (called `odbc32.ini`) on the computer that hosts the broker.

Define 32-bit DSNs in the following situations:

- If a message flow is deployed to a 32-bit execution group, define a 32-bit DSN for the broker and user databases.

- If you are using Linux on x86; the broker runs as a 32-bit application on this 32-bit platform.

To configure a 32-bit DSN for a database:

1. Copy the `odbc32.ini` sample file that is supplied in the *install_dir*`/ODBC32/V5.3` directory to a location of your choice; for example, copy the file to your user ID's home directory. Each broker service user ID on the system can therefore use its own DSN definitions.

   See the sample file contents in "odbc32.ini sample file" on page 343.
2. Ensure that the `odbc32.ini` file has file ownership of `mqm:mqbrkrs`, and has the same permissions as the supplied `odbc32.ini` sample file.
3. Set the ODBCINI32 environment variable to point to your `odbc32.ini` file, specifying a full path and file name.

   For Linux on x86 only, use the `ODBCINI` environment variable.
4. On AIX, HP-UX, Solaris on SPARC, and Linux on x86-64, set the 32-bit library search path environment variable MQSI_LIBPATH32 to show the location of the 32-bit libraries for the database manager that you are using. For example, on AIX:

   ```
   export
   MQSI_LIBPATH32=$MQSI_LIBPATH32:DB2_instance_directory/sqllib/lib32
   ```

   On Linux on x86 only, set the 32-bit library search path LD_LIBRARY_PATH to show the location of the 32-bit libraries for the database manager.
5. If you are using a DB2 database instance that is installed on AIX, a single process can make a maximum of 10 connections using shared memory to a DB2 database. Therefore, 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 resolve this issue, use a TCP/IP connection to the database instance; see DB2 error message SQL1224N is issued when you connect to DB2 for details.
6. Edit the final stanza in the `odbc32.ini` file (the [ODBC] stanza) to specify the location of the ODBC Driver Manager, and to control tracing. The exact details in the stanza depend on the operating system.

   To ensure that you edit the correct `odbc32.ini` file, open the file in the vi text editor by using the following command:

   ```
   vi $ODBCINI32
   ```

   or `vi $ODBCINI` on Linux on x86.

   a. In **InstallDir**, add the WebSphere Message Broker installation location to complete the fully qualified path to the ODBC directory. If you do not set this value correctly, the ODBC definition will not work.

   b. In **Trace**, set the value to `0`; if your IBM service representative asks you to enable ODBC trace, set the value to `1`.

   c. In **TraceFile**, type the fully-qualified path and file name to which the ODBC trace is written. Trace files can become large; specify a directory with plenty of free disk space.

   d. In **TraceDll**, add the WebSphere Message Broker installation location to complete the fully qualified path to the ODBC trace DLL.

   e. Accept the default values that are shown in the sample `odbc32.ini` file for all the other entries in the stanza. For example:

      - On AIX:

```
;#######################################
;###### Mandatory information stanza #####
;#######################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

7. Edit the first stanza in the `odbc32.ini` file (the [ODBC Data Sources] stanza) to list the DSN of each database. For example:
   - On AIX:

     ```
     ;#######################################
     ;###### List of data sources stanza #######
     ;#######################################
     [ODBC Data Sources]
     DB2V8DB=IBM DB2 Version 8 ODBC Driver
     DB2V9DB=IBM DB2 Version 9 ODBC Driver
     ORACLEDB=DataDirect 5.3 Oracle Driver
     ORACLERACDB=DataDirect 5.3 Oracle Driver (Real Application Clusters)
     SYBASEDB=DataDirect 5.3 Sybase Wire Protocol
     SYBASEDBUTF8=DataDirect 5.3 Sybase UTF8 Wire Protocol
     SQLSERVERDB=DataDirect 5.3 SQL Server Wire Protocol
     INFORMIXDB=IBM Informix ODBC Driver
     ```

   List all your DSNs in your `odbc32.ini` file, regardless of the database manager. You can define multiple DSNs to resolve to the same database; however, if you are using global coordination of transactions, do not use this option because it might cause data integrity problems.
8. For each database that you listed in the [ODBC Data Sources] stanza, create a stanza in the `odbc32.ini` file after the [ODBC Data Sources] stanza. The entries in the stanza depend on the database manager. The information for different operating systems can differ; for example, the file paths to the drivers.

   For a DB2 Version 8 database instance

   a. In **Driver**, add the full path of your DB2 installation.
   b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.
   c. In **Database**, type the DB2 alias. The data source name must be the same as the database alias name. If you are using a remote DB2 database, set up your client-server connection to resolve this alias to the correct database. For more information, see the DB2 documentation.

      For example, on AIX:

      ```
      ;# DB2 version 8 stanza
      [DB2V8DB]
      Driver=<Your DB2 v8 install directory>/lib/libdb2.a
      Description=DB2V8DB DB2 ODBC Database
      Database=DB2V8DB
      ```

   For a DB2 Version 9 database instance

   a. In **Driver**, add the full path of your DB2 installation.
   b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.

c. In **Database**, type the DB2 alias. The data source name must be the same as the database alias name. If you are using a remote DB2 database, set up your client-server connection to resolve this alias to the correct database. For more information, see the DB2 documentation.

   For example, on AIX:

```
;# DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.a
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB
```

For an Oracle database:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver that is shown in the sample `odbc32.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.

c. In **ServerName**, type the Oracle Service Name or Connect Descriptor that resolves to the target Oracle database; for example, through a mapping in the `TSNAMES.ORA` file.

d. Accept the default values that are shown in the sample `odbc32.ini` file for all the other entries in the stanza. For example:

   - On AIX:

```
;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

For an Oracle database that uses Real Application Clusters:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver that is shown in the sample `odbc32.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.

c. In **ServerName**, type the Oracle Real Application Cluster Service Name or Connect Descriptor that resolves to the target Oracle database; for example, through a mapping in the `TSNAMES.ORA` file.

d. Accept the default values that are shown in the sample `odbc32.ini` file for all the other entries in the stanza.

   For example on AIX:

```
;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name defined for the RAC>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

For a Sybase database:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver that is shown in the sample `odbc32.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.

c. In **Database**, type the name of the database to which to connect by default. If you do not specify a value, the default is the database that is defined by your system administrator for each user.

d. In **NetworkAddress**, type the network address of your Sybase ASE server (which is required for local and remote databases). Specify an IP address or server name in the following format:

```
<Your Sybase Server Name>,<Your Sybase Port Number>
```

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 named `interfaces` on Linux and UNIX systems.

e. Accept the default values that are shown in the sample `odbc32.ini` file for all the other entries in the stanza. For example:

- On AIX:

```
;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
```

If you want to use a UNICODE UTF8 Sybase data source, add the following line to the end of your Sybase stanza:

```
Charset=UTF8
```

For an SQL Server database:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver that is shown in the sample `odbc32.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only, and does not affect the connection.

c. In **Address**, type the network address of your database server (which is required for local and remote databases). Specify an IP address or server name in the following format:

```
<Your SQLServer Machine Name or IP address>,<Your SQLServer Port Number>
```

d. In **Database**, type the name of the database to which to connect by default. If you do not specify a value, the default is the database that is defined by your system administrator for each user.

e. Accept the default values that are shown in the sample `odbc32.ini` file for all the other entries in the stanza.

For example, on AIX:

```
;# UNIX to SQL Stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1
```

For an Informix database:

a. In **Driver**, add the full path of your Informix installation to complete the fully qualified path to the driver shown in the sample odbc32.ini file.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **ServerName**, type the name of the Informix IDS server.

d. In **Database**, type the name of the database on the IDS server.

For example, on AIX:

```
;# Informix stanza
[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.so
Description=IBM Informix ODBC Driver
ServerName=<YourServerName>
Database=<Your Database Name>
```

9. Ensure that you have edited all three parts of the `odbc32.ini` file:
   - The [ODBC Data Source] stanza at the top of the `odbc32.ini` file.
   - A stanza for each data source.
   - The [ODBC] stanza at the end of the `odbc32.ini` file.

   If you do not configure all three parts correctly, the ODBC DSNs are not valid, and the broker cannot connect to the database.

You have now configured 32-bit ODBC database connections.

Next: Configure the environment for issuing console commands, and for running the broker, so that the broker can access the required database libraries. For more information, see “Setting your environment to support 32-bit access to databases” on page 143.

Connecting to a database from Linux and UNIX systems: 64-bit considerations

To enable a broker to connect to a database, define the ODBC data source name (DSN) for the database.

Before you start:

- Ensure that the database has been created, see “Creating the broker and user databases” on page 115.
- Ensure that the broker is authorized to access the database, see “Authorizing access to broker and user databases” on page 122.
- Check that you have set up your environment so that the broker can access the database; you might have to run a database profile supplied by the database vendor. For further information, see Setting up a command environment: Linux and UNIX systems.

Important:

- Before you can create a broker, you must define the 64-bit ODBC data source name (DSN) that the broker will use to connect to the broker database, because the broker is a 64-bit application.
- You must define a 64-bit ODBC DSN for the broker database even if you intend to use 32-bit execution groups, because parts of the broker need 64-bit access to the data source. For more information, see "Connecting to a database from Linux and UNIX systems: 32-bit considerations" on page 132.
- If the message flows that are deployed to the broker access one or more user databases, you must define a 64-bit DSN for each user database.
- If you are using 32-bit execution groups, you might also need to define a 32-bit DSN for your broker and user databases. For more information, see "Enabling ODBC connections to the databases" on page 127.

The ODBC Driver Manager has no graphical application to help you to configure the ODBC DSNs. To enable a 64-bit ODBC connection, you must define each database as a DSN in a plain text file called `odbc64.ini` on the computer that hosts the broker. For more information, see "Enabling ODBC connections to the databases" on page 127.

To configure a 64-bit DSN for a database:

1. Copy the `odbc64.ini` sample file that is supplied in the *`install_dir`*`/ODBC64/V5.3` directory to a location of your choice; for example, copy the file to your user ID's home directory. Each broker service user ID on the system can therefore use its own DSN definitions.

   See the sample file contents in "odbc64.ini sample file" on page 351.
2. Ensure that the `odbc64.ini` file has file ownership of `mqm:mqbrkrs` and has the same permissions as the supplied sample file.
3. Set the ODBCINI environment variable to point to your `odbc64.ini` file, specifying a full path and file name.
4. Set the library search path environment variable to show the location of the 64-bit libraries for the database manager that you are using. Ask your database administrator (DBA) for information about the database manager that you are using.

   For more information about the library search path, see the database manager's documentation.

   The library search path environment variable depends on your platform:

   - Linux Solaris On Linux and Solaris: LD_LIBRARY_PATH
   - HP-UX On HP-UX: SHLIB_PATH
   - AIX On AIX: LIBPATH
5. If you are using a DB2 database instance that is installed on AIX, a single process can make a maximum of 10 connections using shared memory to a DB2 database. Use TCP/IP mode to connect to the database instance; see DB2 error message SQL1224N is issued when you connect to DB2.
6. Edit the final stanza in the `odbc64.ini` file, the [ODBC] stanza, to specify the location of the ODBC Driver Manager and to control tracing. The exact details in the stanza depend on the platform.

   To ensure that you edit the correct `odbc64.ini` file, you can open the file in the vi text editor using the following command:

   ```
   vi $ODBCINI
   ```

a. In **InstallDir**, add the WebSphere Message Broker installation location to complete the fully qualified path to the ODBC directory. If you do not specify this value correctly, the ODBC definition will not work.
b. In **Trace**, set the value to `0`; if your IBM service representative asks you to enable ODBC trace, set the value to `1`.
c. In **TraceFile**, type the fully-qualified path and file name to which the ODBC trace is written. Trace files can become quite large; specify a directory with plenty of free disk space.
d. In **TraceDll**, add the WebSphere Message Broker installation location to complete the fully qualified path to the ODBC trace DLL.
e. Accept the default values shown in the sample `odbc64.ini` file for all the other entries in the stanza.

   For example on AIX:

```
;########################################
;###### Mandatory information stanza ######
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

7. Edit the first stanza in the `odbc64.ini` file, the [ODBC Data Sources] stanza, to list the DSN of each database.

   For example on AIX:

```
;#####################################
;#### List of data sources stanza ######
;#####################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 64bit SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver
```

   List all your DSNs in your `odbc64.ini` file, regardless of the database manager. You can define multiple DSNs to resolve to the same database; however, if you are using global coordination of transactions, do not use this option because it might cause data integrity problems.

8. For each database that you listed in the [ODBC Data Sources] stanza, create a stanza in the `odbc64.ini` file after the [ODBC Data Sources] stanza. The entries in the stanza depend on the database manager. Slight differences also occur between operating systems, for example the file paths to the drivers.

   For a DB2 database instance:

   a. In **Driver**, accept the value as shown in the sample `odbc64.ini` file.
   b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.
   c. In **Database**, type the DB2 alias. The data source name must be the same as the database alias name. 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.

For example, on AIX:

```
;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB
```

For an Oracle database:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver shown in the sample `odbc64.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **HostName**, type the name or IP address of the machine that is hosting your Oracle system.

d. In **PortNumber**, type the number of the port on which your Oracle server is listening on the machine you specified in **HostName**.

e. In **SID**, type the Oracle service name that you want to connect to on the system you specified in **HostName**.

f. Accept the default values shown in the sample `odbc64.ini` file for all the other entries in the stanza.

For example on AIX:

```
;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

For an Oracle database that uses Real Application Clusters:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver shown in the sample `odbc64.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **HostName**, type the name or IP address of the machine that is hosting your Oracle system.

d. In **PortNumber**, type the number of the port on which your Oracle server is listening on the machine you specified in **HostName**.

e. In **ServiceName**, type the Oracle Real Application Cluster service name that you want to connect to on the system you specified in **HostName**.

f. Accept the default values shown in the sample `odbc64.ini` file for all the other entries in the stanza.

For example on AIX:

```
;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

For a Sybase database:

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver shown in the sample `odbc64.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **Database**, type the name of the database to which you want to connect by default. If you do not specify a value, the default value is the database defined by your system administrator for each user.

d. In **NetworkAddress**, type the network address of your Sybase ASE server (this address is required for local and remote databases). Specify an IP address or server name as follows:

   ```
   <Your Sybase server name or IP address>,<Your Sybase port number>
   ```

   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 that is named `interfaces`.

e. Accept the default values shown in the sample `odbc64.ini` file for all the other entries in the stanza.

   For example on AIX:

   ```
   ;# Sybase Stanza
   [SYBASEDB]
   Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
   Description=DataDirect 5.3 64bit Sybase Wire Protocol
   Database=<Your Database Name>
   ApplicationUsingThreads=1
   EnableDescribeParam=1
   OptimizePrepare=1
   SelectMethod=0
   NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
   SelectUserName=1
   ColumnSizeAsCharacter=1
   EnableSPColumnTypes=2
   ```

   If you want to use a UNICODE UTF8 Sybase data source, add the following line to the end of your Sybase stanza:

   ```
   Charset=UTF8
   ```

For an SQL Server database

a. In **Driver**, add the WebSphere Message Broker installation location to complete the fully qualified path to the driver shown in the sample `odbc64.ini`.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **Address**, type the network address of your database server (this address is required for local and remote databases). Specify an IP address or server name as follows:

```
<Your SQLServer machine name or IP address>,<Your SQLServer port number>
```

d. In **Database**, type the name of the database to which you want to connect by default. If you do not specify a value, the default value is the database defined by your system administrator for each user.

e. Accept the default values shown in the sample `odbc64.ini` file for all the other entries in the stanza.

For example, on AIX:

```
;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 64bit SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
AnsiNPW=Yes
Database=db
QuotedId=No
ColumnSizeAsCharacter=1
```

For an Informix database

a. In **Driver**, accept the value as shown in the sample `odbc64.ini` file.

b. In **Description**, type a meaningful description of the database. This field is for information only and does not affect the connection.

c. In **ServerName**, type the name of the Informix IDS server.

d. In **Database**, type the name of the database to which you want to connect by default. If you do not specify a value, the default value is the database that is defined by your system administrator for each user.

For example, on AIX:

```
;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>
```

9. Ensure that you have edited all three parts of the `odbc64.ini` file:
   - The [ODBC Data Source] stanza at the top of the `odbc64.ini` file.
   - A stanza for each data source.
   - The [ODBC] stanza at the end of the `odbc64.ini` file.

   If you do not configure all three parts correctly, the ODBC DSNs do not work and the broker is unable to connect to the database.

10. If you are running DB2 Version 9.1 on HP-UX on PA-RISC, export environment variable MQSI_SIGNAL_EXCLUSIONS in the broker environment:

```
export MQSI_SIGNAL_EXCLUSIONS=10
```

You have now configured database connections from 64-bit applications on Linux and UNIX.

Next: Configure the environment for issuing console commands, and for running the broker, so that the broker can access the required database libraries. For more information, see "Setting your environment to support 32-bit access to databases."

Setting your environment to support 32-bit access to databases

When you have configured your ODBC data source names (DSNs), you must also configure the environment so that you can issue console commands, and the brokers that you start can access the required database libraries. For example, if you have a DB2 database, you must add the DB2 client libraries to your library search path.

Windows On Windows platforms, the environment is typically set up for you when you install the database product, and no further action is required. However, some database managers provide a database profile that you must run to enable the connection from the broker; for further information, see Setting up a command environment: Windows platforms.

Linux UNIX On Linux and UNIX systems:

- You must run a profile for each database you want to access. For example, on DB2 you must run `db2profile`; other database vendors have similar profiles. For further information, see Setting up a command environment: Linux and UNIX systems.
- If you intend to use only 64-bit execution groups, the correct environment is set up by the database profiles, and no further action is required.
- If you intend to use 32-bit execution groups on AIX, HP-UX on PA-RISC, Solaris on SPARC, or Linux on x86-64, you must ensure that the broker has an environment that presents 32-bit database libraries; the default database profile might present 64-bit libraries.

To set the broker's environment to present 32-bit database libraries:

- If you are using a 64-bit DB2 instance, add *DB2 instance directory*`/sqllib/lib` for DB2 Version 8 or *DB2 instance directory*`/sqllib/lib32` for DB2 Version 9 to the end of the `MQSI_LIBPATH32` library search path environment variable after running `db2profile`.
- If the broker accesses an Informix user database, add `$`*INFORMIX*`/lib:$`*INFORMIX*`/lib/esql:/$`*INFORMIX*`/lib/cli` to the `MQSI_LIBPATH32` library search path environment variable, where *INFORMIX* is your 32–bit Informix Client SDK install location. Set the environment variable `INFORMIXDIR32` to point to the location of your 32–bit Informix Client SDK installation tree.
- If you are using a 64-bit Oracle instance, you must add `$ORACLE_HOME/lib32` to the end of the `MQSI_LIBPATH32` library search path environment variable.

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, and you use Sybase for your broker database, the next task is "Using retained publications with a Sybase database" on page 151 (optional). If you do not use Sybase for your broker database, the next task is "Configuring global coordination of transactions (two-phase commit)" on page 281 (optional).

Enabling JDBC connections to the databases

Configure connections to a user database through a JDBCProvider service.

Use a JDBC connection from Java programs that are associated with a JavaCompute node or a user-defined node that is written in Java.

You must also set up JDBC connections if you include DatabaseRetrieve or DatabaseRoute nodes in your message flows.

If you configure a JDBC type 4 connection from an application running on a Linux, UNIX, or Windows system, you can configure your broker and queue manager to include interactions with the databases in globally-coordinated transactions. On z/OS, JDBC connections can be broker-coordinated only.

The information provided in this section is independent of whether your operating systems, brokers, execution groups, queue managers, and databases operate in 32-bit or 64-bit mode, except where stated.

When you write Java classes for a JavaCompute node or a user-defined node, your code must comply with the following restrictions:

- Do not include any code that performs a COMMIT or a ROLLBACK function.
- Do not close the connection to the database. The broker manages all connections, and closes a connection if it is idle for approximately one minute, or if the message flow completes.

To configure JDBC type 4 connections:

1. Set up your JDBC provider definition.
2. Optional: Set up security.
3. Optional: Configure for global-coordination of transactions.
4. Optional: If your broker is running on a Windows system, authorize access to JDBCProvider resources.

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, the next task is "Setting up a JDBC provider for type 4 connections."

When you have completed configuration of the databases, add or modify Java code in your JavaCompute or user-defined nodes to access the database that is identified in the JDBCProvider service.

Setting up a JDBC provider for type 4 connections

Use the mqsicreateconfigurableservice or the mqsichangeproperties command to configure a JDBC provider service.

Before you start:

- Create a broker
- Create your database following the database documentation.

When you include a DatabaseRetrieve, DatabaseRoute, JavaCompute, or Java user-defined node in a message flow, and interact with a database in that node, the broker must establish a connection with the database to fulfill the operations that are performed by the node. You must define a JDBCProvider configurable service to provide the broker with the information that it needs to complete the connection.

A JDBCProvider configurable service supports connections to one database only; you must create a service for each database that your Java applications connect to.

1. Identify the type of database for which you require a JDBC provider. The following databases are supported with JDBC connections:
   - DB2
   - Informix
   - Oracle
   - SQL Server
   - Sybase

   Supported JDBC drivers are shown in Supported databases. JDBC type 4 connections to a SQLServer databases cannot participate in globally-coordinated (XA) transactions.
2. Run the mqsireportproperties command to view the list of available JDBC providers. Substitute the name of your broker in place of *broker_name*.

```
mqsireportproperties broker_name -c JDBCProviders -a -o AllReportableEntityNames
```

   This command displays a list of all the JDBC provider configurable services that are defined. If you have not created any new definitions, the following list of default supplied providers is shown:
   - DB2
   - Informix
   - Informix_With_Date_Format
   - Microsoft_SQL_Server
   - Oracle
   - Sybase_JConnect6_05

   If you are connecting to an Informix database:
   - Use Informix_With_Date_Format for compatibility with client applications that are dependent on the date format connection attribute that was used by earlier versions of Informix servers.
   - Use Informix for client applications that are not dependent on the date format attribute.
3. View the contents of the relevant definition. For example, run the following command to display the supplied Oracle definition:

```
mqsireportproperties broker_name -c JDBCProviders -o Oracle -r
```

   The command returns a list of all the properties for the Oracle definition. If you have not changed this definition, the properties are set to initial values, some of which you must change to create a viable definition. For example, the property **databaseName** is set to default_Database_Name, and you must change it to identify the specific database that you want to connect to.

   A JDBCProvider has the following properties:
   - **connectionUrlFormat**. A pattern that represents the connection URL definition, which is specific to a particular database type. For example, the pattern for DB2 is defined with the following content:

     ```
     jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];
     ```

     The pattern is used and completed by the broker at run time when it connects to the database. The values in brackets, for example `[serverName]`, are substituted by the broker into the pattern by using the values that you have specified on the mqsicreateconfigurableservice or mqsichangeproperties commands.

     Do not use the mqsichangeproperties command to change the pattern itself; changes made to the pattern might cause unpredictable results.
   - **connectionUrlFormat Attr1-5**. If the defined URL pattern for a database contains non-standard JDBC data source properties, such as a server identifier, specify these properties in addition to the standard attributes by using one of five general purpose connection URL attributes. For example:

  - If **connectionURLFormat** = jdbc:oracle:thin:[user]/[password]@[serverName]:[portNumber]:[connectionUrlFormatAttr1], **connectionUrlFormatAttr1** must contain an Oracle server identifier, which you must supply by defining the value for the property **connectionUrlFormatAttr1** on the mqsicreateconfigurableservice or mqsichangeproperties command. The broker can then substitute all the required values into the required pattern.
  - If **connectionURLFormat** = jdbc:informix-sqli://[serverName]:[portNumber]/[databaseName]:informixserver=[connectionUrlFormatAttr1];user=[user];password=[password], **connectionUrlFormatAttr1** must contain the name of the Informix instance on the server (typically specified by the INFORMIXSERVER environment variable). This value is case-sensitive.
- **databaseName**. The name of the database to which the data source entry enables connections; for example, employees.
- **databaseType**. The database type; for example, `DB2`.
- **databaseVersion**. The database version; for example, 9.1.
- **description**. An optional property to describe the data source definition.
- **jarsURL**. The local directory path, on the system on which the broker is running, where the JAR file that contains the type 4 driver class is located.

  In addition, a Storage Area Network disk can be used for the directory path, but a mapped network drive to a remote machine cannot be used.
- **portNumber**. The port number on which the database server is listening; for example, 50000.
- **securityIdentity**. A unique security key to perform a second broker registry lookup to find an entry under the broker's security identities, which store the encrypted password for the user on their associated host system; for example, mysecurityIdentity.

  Create a new security identity by using the mqsisetdbparms command, as described in "Securing a JDBC type 4 connection" on page 147. The value of **securityIdentity** (for example, mysecurityIdentity) must match the value that you specify following the prefix `jdbc::` for the parameter **-n** on that command.

  The security identity provides a user ID and password value pair, which are used to access the specified data source defined for a given JDBC provider entry. This property is ignored if the connection URL does not contain both a user ID and password pair, which require property values to be substituted for such inserts. The DataSourceUserId and DataSourcePassword properties under which the broker was created are used under the following conditions:
  - If the **securityIdentity** is blank, or if you have not changed it from the default value default_User@default_Server, but the identity is required for the connection URL pattern.
  - If you have entered a valid unique security identity key, but it cannot be found under the DSN key.

  z/OS On z/OS, you must specify a user ID and password with appropriate authorization to access the database. Do not use the broker-started-task user ID for this purpose. (Assign a password to a started-task user ID only after considering all of the potential security implications.)
- **serverName**. The name of the server; for example, host1.

- **type4DatasourceClassName**. The name of the JDBC data source class name that is used to establish a type 4 connection to a remote database, and to coordinate transaction support. For example, specify com.ibm.db2.jcc.DB2XADataSource for DB2, or specify oracle.jdbc.xa.client.OracleXADataSource for Oracle. You must always specify the XA class name, even if you do not use coordinated transactions.
- **type4DriverClassName**. The name of the JDBC Type 4 driver class name that is used to establish a connection. For example, specify com.ibm.db2.jcc.DB2Driver for DB2, or specify oracle.jdbc.OracleDriver for Oracle.

4. If you want to use the provided definition, run the mqsichangeproperties command to replace default values with those specific to your database and environment. If you are in any doubt about the required values, consult your database administrator, or check the documentation that is provided with your chosen database. Some values depend on how and where you have installed the database product; for example, the property **jarsURL** identifies the location of the JAR files supplied and installed by the database provider.
5. If you want to create a new configurable service, perhaps because you want to retain the supplied service as a template for future definitions, run the mqsicreateconfigurableservice command to create the new definition.

```
mqsicreateconfigurableservice broker_name -c JDBCProviders -o provider_name
-n list of properties -v list of values
```

Enter the command on a single line; the example has been split to enhance readability.

Specify all the properties that are required by the database provider that you have chosen. To specify a list of properties and values, separate the items after each flag with a comma. For example, `-n databaseName,databaseType -v EmployeeDB,DB2`. If you do not specify all the properties on the mqsicreateconfigurableservice command, you can update them later with the mqsichangeproperties command.

6. When you have set up your JDBCProvider service, you must stop and restart the broker.

Next: if required, set up security for the JDBC connection, set up the environment to include the JDBCProvider service in globally-coordinated transactions, or both.

Securing a JDBC type 4 connection

Set up security for the JDBC connection if required by the database provider.

Before you start: Set up your JDBC provider definition.

Some databases require all access to be associated with a known user ID; for others this association is optional. For example, DB2 requires a data source login name and password on all connections. If the database requires secure access to be defined, or if you choose to implement security in an optional situation, complete the task described here.

1. Identify the user ID that you want to associate with the JDBC connection, or create a new user ID with a password, following the appropriate instructions for your operating system and database.
2. Run the mqsisetdbparms command to associate the user ID and password with the security identity that is associated with the database that you will access using the JDBCProvider configurable service. Use the following command format:

```
mqsisetdbparms broker_name -n security_identity -u userID -p password
```

For example, if you want user ID myuserid with a password of secretpw to access a database on broker BROKER1, run the following command:

```
mqsisetdbparms BROKER1 -n jdbc::mySecurityIdentity -u myuserid -p secretpw
```

In the example, the *mySecurityIdentity* is prefixed with `jdbc::` to indicate the type of the connection for which the user ID and password are defined.

3. Update the corresponding **securityIdentity** property for the JDBCProvider configurable service to associate the connection with the security identity that you have just defined. Use the following command format:

```
mqsichangeproperties broker_name -c JDBCProviders -o service_name -n securityIdentity -v security_identity
```

For example, if you are using the supplied JDBCProvider definition for Oracle:

```
mqsichangeproperties BROKER1 -c JDBCProviders -o Oracle -n securityIdentity -v mySecurityIdentity
```

You can use the same user ID and password definition for more than one JDBCProvider if appropriate; specify the same security identity that you specified on the mqsisetdbparms command as the value for the **securityIdentity** property in each JDBCProvider definition that uses the same access security.

Next: if you are setting up a connection on Windows, see "Authorizing access to JDBC type 4 JDBCProvider resources on Windows" on page 150. Otherwise, see "Configuring a JDBC type 4 connection for globally-coordinated transactions."

Configuring a JDBC type 4 connection for globally-coordinated transactions

If you want the database that you access through a JDBC type 4 connection to participate in globally-coordinated transactions, set up the appropriate environment.

Before you start: Set up your JDBC provider definition.

Updates that you make to a database across a JDBC type 4 connection can be coordinated with other actions taken within the message flow, if you set up the resources to support coordination.

Complete the following steps:

1. Check that the definition of your JDBCProvider service is appropriate for coordinated transactions.

   For example, to set up the required JDBC classes:

   - For DB2, set **type4DatasourceClassName** to `com.ibm.db2.jcc.DB2XADataSource` and **type4DriverClassName** to `com.ibm.db2.jcc.DB2Driver`
   - For Oracle, set **type4DatasourceClassName** to `oracle.jdbc.xa.client.OracleXADataSource` and **type4DriverClassName** to `oracle.jdbc.OracleDriver`

   Consult your database administrator or the documentation provided by your database supplier, to confirm that all the JDBCProvider service properties are set appropriately. For example, a database supplier might require secure access if it is participating in coordinated transactions.
2. Define the switch file and the database properties:

a. Linux UNIX On Linux and UNIX systems, open the `qm.ini` file for the broker's queue manager with a text editor. Add the following stanza for each database:

```
XAResourceManager:
        Name=Database_Name
        SwitchFile=JDBCSwitch
        XAOpenString=JDBC_DataSource
        ThreadOfControl=THREAD
```

Database_Name is the database name (DSN) of the database defined to the JDBCProvider configurable service (for example, specified by `-n databaseName -v` *Database_Name* on the mqsichangeproperties command).

`JDBCSwitch` is a fixed generic name that represents the switch file for XA coordination. Use this value, or another single fixed value, in each stanza; the specific switch file that the queue manager uses is defined by the symbolic links you create in the next step.

JDBC_DataSource is the identifier of the JDBCProvider configurable service (the value that you specified for the **-o** parameter on the mqsichangeproperties command).

Define a stanza for each database (DSN) that you connect to from this broker. You must create separate definitions even if the DSNs resolve to the same physical database. Therefore, you must have a stanza for each JDBCProvider configurable service that you have defined, because each service can define the properties for a single database.

b. Windows On Windows systems, open WebSphere MQ Explorer and select the queue manager for your broker, for example BROKERQM.

Open the **XA resource manager** page, and modify the attributes to create the definition of the database. The attributes are the same as those shown for Linux and UNIX; **Name**, **SwitchFile**, **XAOpenString**, and **ThreadofControl**. Leave the additional attribute, **XACloseString**, blank.

Enter the fully qualified file name in **SwitchFile**; *install_dir*`\bin\JDBCSwitch.dll`.

3. Set up queue manager access to the switch file:

a. Linux UNIX On Linux and UNIX systems, create a symbolic link to the switch files that are supplied in your *install_dir*`/lib` directory.

install_dir is the directory to which you installed the broker component. The default location for this directory is `/opt/ibm/mqsi/6.1` on Linux or `/opt/IBM/mqsi/6.1` on UNIX systems.

Set up links in the `/var/mqm/exits` directory, or the `/var/mqm/exits64` directory, or both. The file names for each platform are shown in the following table.

Platform	32-bit file	64-bit file
AIX	`libJDBCSwitch.so`	`libJDBCSwitch64.so`
HP-UX on Itanium		`libJDBCSwitch.so`
HP-UX on PA-RISC	`libJDBCSwitch.sl`	`libJDBCSwitch64.sl`
Linux on POWER		`libJDBCSwitch.so`
Linux on System z		`libJDBCSwitch.so`
Linux on x86	`libJDBCSwitch.so`	
Linux on x86-64	`libJDBCSwitch.so`	`libJDBCSwitch64.so`

Platform	32-bit file	64-bit file
Solaris on SPARC	libJDBCSwitch.so	libJDBCSwitch64.so
Solaris on x86-64		libJDBCSwitch.so

Specify the same name of the switch file, JDBCSwitch or your own value, in both the /exits and /exits64 directories. For example, on AIX:

```
ln -s install_dir/lib/libJDBCSwitch.so /var/mqm/exits/JDBCSwitch

and

ln -s install_dir/lib/libJDBCSwitch64.so /var/mqm/exits64/JDBCSwitch
```

b. Windows On Windows systems, copy the JDBCSwitch.dll file from the *install_dir*\bin directory to the \exits subdirectory in the WebSphere MQ installation directory.

4. Configure the message flow that includes one or more nodes that access databases that are to participate in a globally-coordinated transaction.
   a. Open a workbench session.
   b. Switch to the Broker Application Development perspective.
   c. Add the message flow that includes the node or nodes that connect to the database that is to participate in a globally-coordinated transaction to a new or existing BAR file.
   d. Build the BAR file.
   e. Click the **Configure** tab, select the message flow that you have added, and select the **Coordinated Transaction** check box.

Next: If your broker is running on Windows, authorize the broker and its queue manager to access resources associated with the JDBCProvider configurable service.

If you have been following the instructions in "Configuring broker and user databases" on page 109, and your broker is running on Linux or UNIX, the next task is "Using retained publications with a Sybase database" on page 151 (optional).

Authorizing access to JDBC type 4 JDBCProvider resources on Windows

Authorize the broker and queue manager to access shared resources that are associated with the JDBCProvider. This task is required only if you want the database updates to be included in globally-coordinated transactions on Windows systems.

Before you start: Set up your JDBC provider definition.

When the queue manager coordinates transactions, both queue manager and broker access shared memory to control a connection to the databases with which the message flow interacts. Therefore, they require the same access control of the shared memory. One method to achieve this control is to use the same ID for the broker service ID and the queue manager's administrative ID.

Complete the following steps on the Windows system on which the broker is running:

- If you defined the broker queue manager when you created the broker by running the mqsicreatebroker command, the two components share the same administrative ID, defined as the broker service ID, and you do not have to take any further action.
- If you specified an existing queue manager when you created the broker, check that its administrative ID is the same ID as that used for the service ID of the broker. If the ID is not the same, change the queue manager ID to be the same as the broker service ID:
  1. Click **Start** → **Run** and enter dcomcnfg. The Component Services window opens.
  2. In the left pane, expand **Component Services** → **Computers** → **My Computer** and click **DCOM Config**.
  3. In the right pane, right-click the WebSphere MQ service labeled **IBM MQSeries Services**, and click **Properties**.
  4. Click the **Identity** tab.
  5. Select **This user** and enter the user ID and password for the broker service ID to associate that ID with the queue manager.
  6. Click **OK** to confirm the change.

Next: If you have been following the instructions in "Configuring broker and user databases" on page 109, the next task is "Using retained publications with a Sybase database" (optional).

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, 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 this 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 broker database name for `WBRKBKDB` in this 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 this 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`.

Customizing the z/OS environment

If you are planning to use a z/OS environment consider whether to create your components on z/OS. There are also a number of tasks you must complete to configure your environment.

Although you might be installing only one broker initially, you might want to consider how the product will be used in your organization in a few years time. Planning ahead makes developing your WebSphere Message Broker configuration easier.

You might consider creating the Configuration Manager on z/OS to manage the broker domain:

- In a new installation of WebSphere Message Broker, or
- If you are migrating from an earlier version of the product, where the Configuration Manager was previously on Windows.

If you want to run a Configuration Manager on z/OS, you can either:

- Connect to the Message Broker Toolkit directly, if you have the optional WebSphere MQ Client Attach feature installed; see “Connecting directly to a Configuration Manager on z/OS” on page 174, or
- Connect through an intermediate queue manager (for example, on Windows) and define the necessary WebSphere MQ components to communicate with the z/OS queue manager; see “Connecting to a z/OS Configuration Manager through an intermediate queue manager” on page 175.

If you are using publish/subscribe with security, you also require a User Name Server, which can be on z/OS or another platform.

The following rules apply:

- Queue managers must 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. See Database contents for details of the DB2 database user tables that are created.
- A Configuration Manager and User Name Server require 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 Message Broker as SYSTEM.BROKER queues, but you can use shared queues for your message flow queues.

You can find details of the WebSphere MQ queues that are created and used by WebSphere Message Broker on z/OS in “mqsicreatebroker command” on page 513.

When planning to work in a z/OS environment, you must complete the following tasks:

- Create started task procedures for each broker, User Name Server, and Configuration Manager that you plan to use. These procedures must be defined, in the started task table, with an appropriate user ID.
- Decide on your recovery strategy. As part of your systems architecture, you must 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 Message Broker to use ARM.
- Plan for corequisite products, including UNIX System Services, Resource Recovery Services, DB2, WebSphere MQ, and Java.
- Ensure that the runtime library system (RTLS) for the broker is turned off in the default options of the language environment for the system. This setting is required because the broker code is compiled using `XPLINK`, and `XPLINK` applications cannot be started while RTLS is active.
- Collect broker statistics on z/OS.

See the following topics for more information:

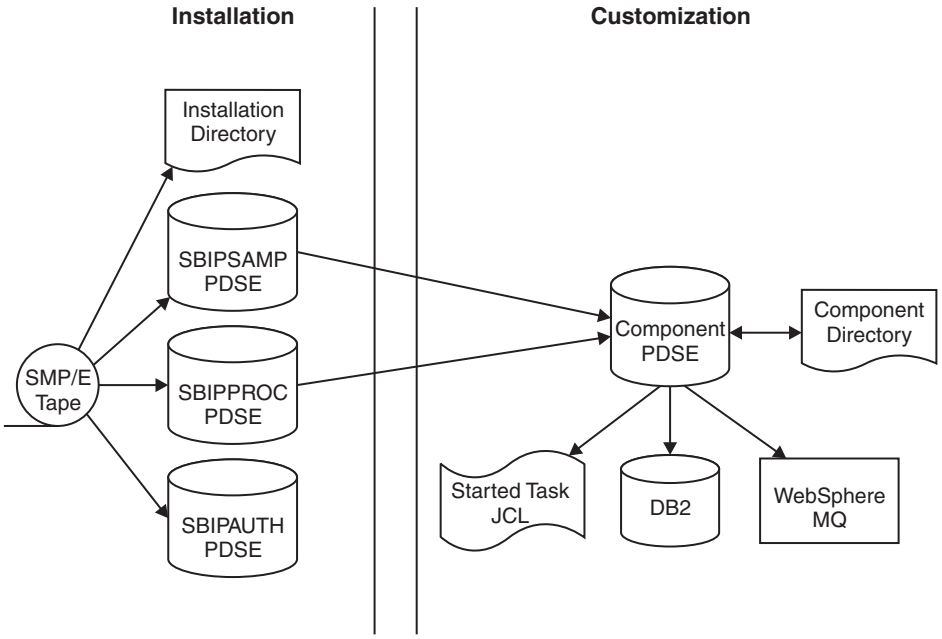
- "z/OS customization overview"
- "Customizing UNIX System Services on z/OS" on page 163
- "DB2 planning on z/OS" on page 165
- "WebSphere MQ planning for z/OS" on page 168
- "Resource Recovery Service planning on z/OS" on page 169
- "Defining the started tasks to z/OS Workload Manager (WLM)" on page 169
- "Automatic Restart Manager planning" on page 169
- "Mounting file systems" on page 170
- "Checking the permission of the installation directory" on page 171
- "Customizing the version of Java on z/OS" on page 171
- "Checking APF attributes of `bipimain` on z/OS" on page 172
- "Collecting broker statistics on z/OS" on page 172
- "Configuring an execution group address space as non-swappable on z/OS" on page 172

For an overview of how to create WebSphere Message Broker components, see "Creating WebSphere Message Broker components on z/OS" on page 173. To verify your configuration, see "WebSphere Message Broker and WebSphere MQ setup verification" on page 176.

z/OS customization overview

After you have used SMP/E to install WebSphere Message 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 following diagram gives an overview of the post-installation process.



To perform the customization, update and submit the required JCL. All necessary JCL is supplied to create the runtime environments of your broker, Configuration Manager, and User Name Server. You start the broker, Configuration Manager, or User Name Server using one of the supplied JCL files, which is run as a started task.

For more information, see:

- "Installation directory on z/OS"
- "Components on z/OS"
- "Component directory on z/OS" on page 155
- "Component PDSE on z/OS" on page 155
- "XPLink on z/OS" on page 156
- "Binding a DB2 plan to use data-sharing groups on z/OS" on page 156
- "Using the file system on z/OS" on page 157
- "Event log messages on z/OS" on page 157
- "Security considerations on z/OS" on page 157
- "Overview of message serialization on z/OS" on page 158

Installation directory on 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).

For further information on mounting file systems and allocating space see "Mounting file systems" on page 170

Components on z/OS

A *component* is a set of runtime processes that perform a specific set of functions, and can be 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 Message 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.

However, 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 on z/OS

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

The *component directory* is also referred to as `ComponentDirectory` in some instances within the code. Both the WebSphere Message Broker administrator and the component require read and write access to the component directory.

An example directory for each of the three components follows:

- `/mqsi/brokers/MQP1BRK` for the broker
- `/mqsi/uns/MQP1UNS` for the User Name Server
- `/mqsi/configmgrs/MQP1CMGR` for the Configuration Manager

For further information on mounting file systems and allocating space see "Mounting file systems" on page 170

Component PDSE on z/OS

On z/OS, the *component PDSE* contains jobs customized for a single component. These jobs are used to create and administer the component.

The members specific to a component type are copied from<hlq>.SBIPSAMP and <hlq>.SBIPPROC to the component PDSE. These are then customized for the component.

The broker started-task user ID requires read access to its component PDSE at runtime.

XPLink on z/OS

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 Message Broker for z/OS, use XPLink technology to improve their performance. To ensure the highest possible performance gains, WebSphere Message 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 Message 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 on z/OS

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 )
```

Using the file system on z/OS

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.

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

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).

For further information on mounting file systems and allocating space see "Mounting file systems" on page 170

Space requirements:

For details of the disk space required, see "Disk space requirements on z/OS" on page 677.

Event log messages on z/OS

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

On z/OS, all address spaces have a job log where BIP messages issued by WebSphere Message Broker appear. Additionally, all messages appear on the `syslog` and important operator messages are filtered to the console using MPF (Message Processing Facility).

To prevent the operator's console receiving unnecessary BIP messages, you must configure MPF to suppress all BIP messages, with the exception of important messages. Note that you do not need to have the `USS SYSLOG` configured.

Security considerations on z/OS

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

The role of the WebSphere Message Broker administrator includes customizing and configuring, running utilities, performing problem determination, and collecting diagnostic materials. People involved in these activities need WebSphere Message Broker authorities. You must set up some security for WebSphere Message Broker to work properly. The information that you need to do this is in "Setting up z/OS security" on page 73.

Overview of message serialization on z/OS

Some messaging transactions depend upon the exact sequence of messages from a queue, and for that sequence to be maintained in the event of a failure of the queue manager. In these instances you must serialize the access to those messages.

Serialization of messages is achieved through the use of specialized connection options, and a unique connection token when the application that empties the messages from a queue issues a connect call to the WebSphere MQ queue manager that owns that queue.

A typical situation in which WebSphere Message Broker can exploit this feature is the case where multiple brokers, with multiple execution groups, are each running message flows that empty from a shared input queue. If one broker queue manager fails, then the message flow can be automatically started on another broker while maintaining the transactional integrity and original sequencing of the messages on the shared queue.

The following examples demonstrate how these features can be applied:

1. "Serialization of input between separate brokers on z/OS"
2. "Serialization of input between separate execution groups running on the same broker on z/OS" on page 160
3. "Serialization of input within an execution group on z/OS" on page 161

Note the following:

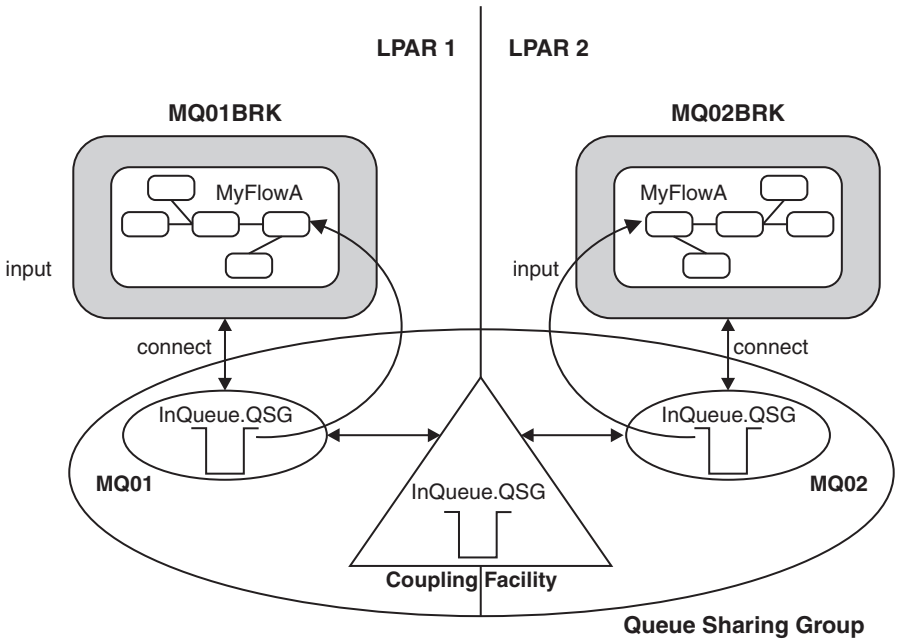
- In each of the first two examples there are two brokers configured, referred to as `MQ01BRK` and `MQ02BRK`; the broker's respective queue managers are called `MQ01` and `MQ02`.
- The queue managers participate in the same queue sharing group. Each queue manager has a shared queue `INQueue.QSG` that has been defined with a disposition of `QSG`, and a local queue called `INQueue`
- The queue managers can be running in the same Logical Partition (LPAR) or separate LPARs.
- The Coupling Facility shown in the following diagrams is a zSeries® component that allows z/OS WebSphere MQ queue managers in the same system image, or different system images, to share queues.

Serialization of input between separate brokers on z/OS: This example demonstrates that only one input node at a time takes messages from a shared queue when the same serialization token is used by message flows running on separate brokers.

An identical message flow `MyFlowA` is deployed to an execution group called `MYGroupA` on each broker. Note that the message flows do not have to be identical; the significant point is that an identical serialization token is used in both flows.

The simple message flow in this example consists of an `MQInput` node connected to an `MQOutput` node. The `MQInput` node in both message flows gets messages from the shared queue `INQueue.QSG`; the node attribute `Serialization Token` is configured as `MyToken123ABC` in both `MQInput` nodes.

The message flow property `additional Instances` takes the default value of zero in both message flows, which ensures that input is serialized within the flow.



A typical sequence of events for this example follows:

1. The first broker MQ01BRK starts and runs message flow MyFlowA in execution group MyGroupA. The input node MyInputNode connects to queue manager MQ01 using a serialization token MyToken123ABC. The input node successfully opens shared queue INQUeue.QSG and gets input messages.
2. The second broker MQ02BRK starts and begins to run its copy of message flow MyFlowA in execution group MyGroupA. The Input node MyInputNode attempts to connect to queue manager MQ02 , also using a serialization token MyToken123ABC.

   The following SDSF console message is logged:

   ```
   BIP2656I MQ02BRK MyGroupA 17 UNABLE TO OPEN QUEUE
   'INQueue.QSG' ON WEBSPHERE BUSINESS INTEGRATION QUEUE
   MANAGER 'MQ02': COMPLETION CODE 2; REASON CODE 2271.
   :ImbCommonInputNode(759) BECAUSE SERIALIZATION TOKEN
   MyToken123ABC is already in use. NO USER ACTION REQUIRED.
   ```

   Note that this message is output every 30 minutes.

   Message flow MyFlowA in execution group MyGroupA running on broker MQ02BRK is unable to process input because the serialization token it has passed is already in use within the queue sharing group. This is indicated by the reason code 2271 (MQRC_CONN_TAG_IN_USE) in message bip2623.
3. Broker MQ01BRK stops. Message flow MyFlowA in execution group MyGroupA in broker MQ02BRK2 is now able to get messages from the shared queue INQueue.QSG.

   A sequence of SDSF console messages is logged, of which the following two are relevant:

   ```
    BIP2091I MQ02BRK MyGroupA 17 THE BROKER HAS
   RECONNECTED TO WEBSPHERE BUSINESS INTEGRATION
   SUCCESSFULLY : ImbCommonInputNode(785)

    BIP9142I MQ01BRK 0 THE COMPONENT HAS STOPPED. :
   ImbControlService(594)
   ```

The preceding sequence of events also occurs should broker MQ01BRK fail, rather than stop through a request from the operator, or if a new broker configuration is deployed to MQ01BRK that deletes or modifies message flow MyFlowA.

This arrangement can also be used where the requirement is to migrate message processing between brokers running in different z/OS system images that are attached to the same Coupling Facility.

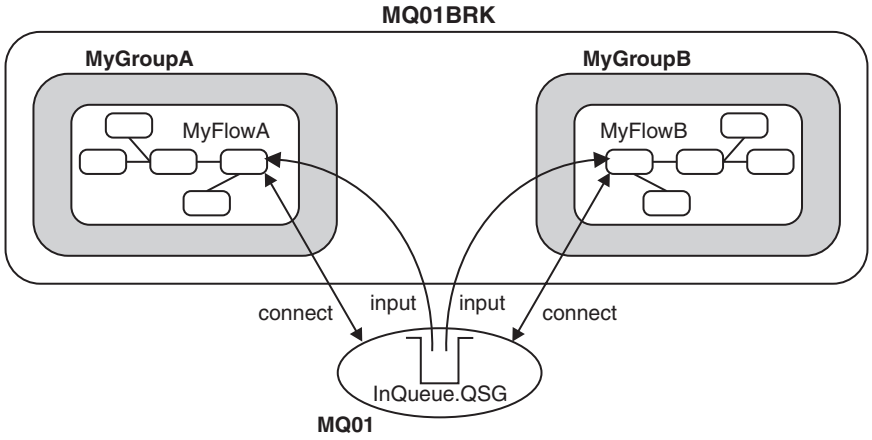
Serialization of input between separate execution groups running on the same broker on z/OS: This example demonstrates that only one MQInput node at a time is allowed to take messages from a shared queue when the same serialization token is used by message flows running in separate execution groups on the same broker.

An identical message flow MyFlowA is deployed to two execution groups called MYGroupA and MYGroupB on broker MQ01BRK.

In this case it is not a requirement that the queue manager participates in a queue sharing group. The input queue INQueue is defined as local with disposition QMGR.

As in "Serialization of input between separate brokers on z/OS" on page 158:

- Note that the message flows do not have to be identical; the significant point is that an identical serialization token is used in both flows.
- The simple message flow in this example consists of an MQInput node connected to an MQOutput node. The MQInput node in both message flows gets messages from the shared queue INQueue.QSG; the node attribute Serialization Token is configured as MyToken123ABC in both MQInput nodes.
- The message flow property additional Instances takes the default value of zero in both message flows, which ensures that input is serialized within the flow.



A typical sequence of events for this example follows:

1. Broker MQ01BRK starts and the first message flow to begin is MyFlowA in execution group MyGroupA. The MQInput node MyInputNode connects to queue manager MQ01 using the serialization token MyToken123ABC. The MQInput node successfully opens shared queue INQUeue and gets input messages.
2. The second execution group MyGroupB starts and message flow MyFlowA in execution group MyGroupB begins. The MQInput node MyInputNode now attempts

to connect to queue manager MQ01 using serialization token MyToken123ABC. The following SDSF console message is logged:

```
BIP2656I MQ01BRK MyGroupB 11 UNABLE TO OPEN QUEUE
'INQueue' ON WEBSPHERE BUSINESS INTEGRATION QUEUE
MANAGER 'MQ01':  BECAUSE SERIALIZATION TOKEN
MyToken123ABC is already in use. NO USER ACTION REQUIRED
```

Message flow MyFlowA in execution group MyGroupB is unable to process input because the serialization token it has passed is already in use within the queue manager (by the MQInput node in message flow MyFlowA in execution group MyGroupA). This is indicated by the reason code 2271 (MQRC_CONN_TAG_IN_USE) in message bip2623.

3. The first execution group is deleted or cancelled.

   If the first execution group is cancelled by the operator, abends, or is deleted during a redeployment of the broker configuration, then the input node in the second execution group is now able to get input messages from queue INQueue.

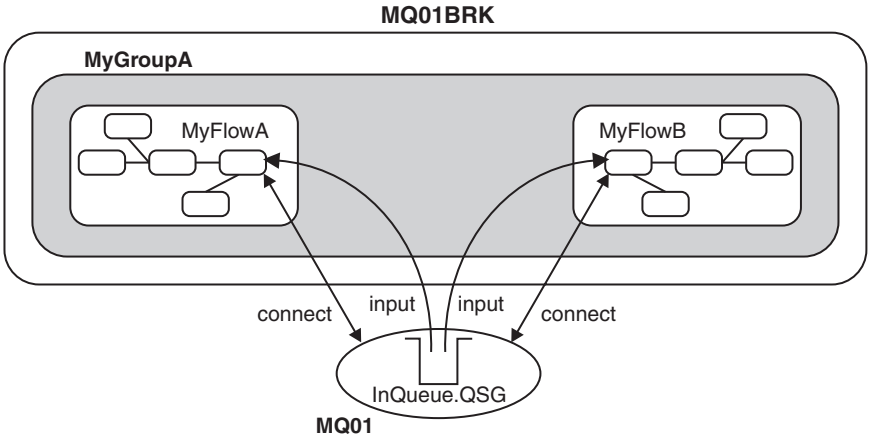
   A sequence of SDSF console messages is logged, of which the following one is relevant:

```
 BIP2091I MQ01BRK MyGroupB 11 THE BROKER HAS
RECONNECTED TO WEBSPHERE BUSINESS INTEGRATION
SUCCESSFULLY : ImbCommonInputNode(785)
```

Message flow MyFlowA in execution group MyGroupB is now able to recover processing of messages from the shared queue INQueue.QSG.

Note that, although serialization of input can be achieved in a similar manner by configuring the input queue for exclusive input, this does not ensure message integrity during a recovery situation. This can be achieved only through the use of the serialization token as described in this example.

Serialization of input within an execution group on z/OS: To allow concurrent processing within a message flow, while still serializing messages between message flows in separate execution groups, the scope of the serialization token is restricted within a single execution group.



This example demonstrates that the serialization token is restricted within a single execution group running on a broker::

- Two MQInput nodes in separate message flows (in this case MyFlowA and MyFlowB) are running within the same execution group MyGroupA. Both MQInput nodes concurrently get messages from the shared input queue even though they are using the same serialization token.
- If serialization is required within a single message flow then the message flow attribute additional instances must be set to zero which is the default setting. However, if greater throughput is required and serialization of input within the flow is not important, you can set additional instances to a value greater than zero.
- The use of the serialization token attribute on the MQInput node does not serialize input between message flows operating within the same execution group. However, setting the attribute has no adverse affect on the processing within that execution group
- In this way it is possible to maximize throughput in a message flow on one broker while still serializing input between brokers. This is useful where the requirement is to have one or more brokers acting as an immediate standby, should the currently active broker need to be stopped for servicing, or fail unexpectedly.

Serialization token - user tasks on z/OS:

Configure shared input queues and define serialization tokens for message flows.

Configure a shared input queue for message flows

The broker makes use of WebSphere MQ queue-sharing groups on z/OS.

Queue managers that can access the same set of shared queues form a group called a queue-sharing group (QSG) and they communicate with each other by means of a coupling facility (CF) that stores the shared queues. A shared queue is a type of local queue whose messages can be accessed by one or more queue managers that are in a QSG.

To further enhance the availability of messages in a QSG, WebSphere MQ detects if another queue manager in the group disconnects from the CF in an unusual way, and completes pending units of work for that queue manager where possible; this is known as peer recovery.

To understand more fully the concepts of shared-queues and queue-sharing groups, see the *Concepts and Planning Guide* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online, and perform the following steps:

- Add a QSG to the DB2 tables
- Add a queue manager to a QSG
- Create the shared queue as a member of the QSG

Define a serialization token

Define the same value for the serialization token attribute for each MQInput node that is required to access the shared queue.

For the situations described in the preceding text to work you must:

- Ensure that the Coupling Facility Structure is at CFLEVEL(3) or above, and that you set RECOVER=YES.

If you do not do this, when an MQInput node attempts to get a message from the shared queue, the action fails with the WebSphere MQ return code BIP2048 (MQRC_PERSISTENT_NOT_ALLOWED)

- Set the Backout Threshold for the shared queue to at least 2.

  This value prevents input messages that are in progress being sent to the Dead Letter Queue because, during recovery, a message is restored to the input queue before another broker is able to get it and resume processing.

Customizing UNIX System Services on z/OS

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

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

Use the D OMVS,O command to display your current OMVS options.

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 parameters shown in the following table.

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	> 1 073 741 824 A minimum value of 393 216 000 bytes 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. To calculate the value needed 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, then add 10 to the sum. 3. Add to the sum the number of threads used for each HTTP listener.

Deploying a message flow that starts an execution group in a new address space uses USS Semaphore and SharedMemorySegment resources. Each new address space uses a semaphore and SharedMemorySegment. The SharedMemorySegment is deleted immediately after the new address space has started, but the semaphore remains for the life of the new address space.

Certain USS system parameters can affect the start of a new execution group address-space, if you set them incorrectly. These parameters include:

- IPCSEMNIDS
- IPCSHMNIDS
- IPCSHMNSEGS

You need a minimum of three semaphores for each execution group address-space that is started.

You must set IPCSEMNIDS to a value four times the number of potential execution group address-spaces on a system.

You need one SharedMemorySegment for each execution group address-space that is started. You must set IPCSEMNIDS to a value that exceeds the number of potential execution group address-spaces on a system.

A control address space (BIPSERVICE and BIPBROKER processes) can be attached to many SharedMemorySegments - potentially, one for each execution group address-space started for that broker. You must set IPCSHMNSEGS to a value that exceeds the potential number of execution groups for each broker.

Ensuring sufficient space for temporary files

The environment variable TMPDIR is the path name of the directory being used for temporary files. If it is not set, the z/OS shell uses /tmp.

When starting WebSphere Message Broker components, sufficient space is required in the directory referenced by TMPDIR. In particular, Java needs sufficient space to hold all JAR files required by WebSphere Message Broker.

If you do not allocate sufficient space, the execution group address-spaces abends with a 2C1 code.

Allow at least 50 MB of space in this directory for broker components and 10 MB of space for Configuration Manager components. More space might be needed if you deploy large user-defined nodes or other JAR files to the broker component.

Defining WebSphere Message Broker files as shared-library programs

If you plan to deploy to more than one execution group on z/OS, the amount of storage required by the execution group address-spaces can be reduced by setting the shared-library extended attribute on the following WebSphere Message Broker files:

```
/usr/lpp/mqsi/bin/*
/usr/lpp/mqsi/lil/*
/usr/lpp/mqsi/lib/*
/usr/lpp/mqsi/lib/wbirf/*
/usr/lpp/mqsi/lib/wbimb/*
```

To set the shared-library attribute, use the **extattr** command with the +1 option. For example:

```
extattr +l /usr/lpp/mqsi/bin/*
```

To find out if the shared-library extended attribute has been set, use the `ls -E` command. For example, use the command `ls -E bipimain` to generate the following response:

```
-rwxr-x---  a-l-  1 USER  GROUP   139264 Mar 15 10:05 bipimain
```

where `l` (lowercase L, as in `a-l-`) shows that the program is enabled to run in a shared address space.

Use the following command to check that you have enough SHRLIBRGNSIZE to contain all of the shared-library programs that are to be used on the system:

```
/D OMVS,LIMITS
```

DB2 planning on z/OS

This is part of the larger task of customizing your z/OS environment and is relevant only to the broker.

The Configuration Manager and User Name Server do not require access to DB2.

WebSphere Message Broker for z/OS accesses DB2 tables by using ODBC. To connect to DB2 using ODBC, the location name of the DB2 subsystem is used.

See the *DB2 Administration > Data Sharing* section of the DB2 information center (z/OS) for more details.

You must give certain user IDs access to DB2 resources:

- DB2 systems administrator
  1. Create database, storage groups, and table spaces (`BIPCRDB`).
  2. Drop database (`BIPDLDB`)
- Administrator for the broker database (DBA). This should be the WebSphere Message Broker administrator.
  1. Create tables and indexes (`BIPCRBK`)
  2. Create tables, drop tables, and create indexes (`BIPMGCMP`)
- Broker started task user ID:
  1. SQL to select, insert, and delete rows from the broker database tables, and select from DB2 system tables.
- WebSphere Message Broker administrator and other users
  1. SQL to select, insert, and delete rows from the broker database tables, and select from DB2 system tables.

When your DB2 system starts up, message `DSNL004I  DDF START COMPLETE` is displayed. 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 *Programming DB2 > Programming for ODBC* section of the DB2 information center (z/OS) for more details.

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 *Programming DB2 > Programming for ODBC* section of the DB2 information center (z/OS) for more details.

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';
```

Rebind if the location column is blank and not '*'.

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

You can get significant performance benefits if you use the CACHE DYNAMIC SQL facility of DB2, because this eliminates the need to reprocess DB2 statements.

See CACHEDYN=YES in the *Programming DB2 > Application Programming and SQL* section of the DB2 information center (z/OS) for more details.

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.

You can use the DB2 security mechanism, or, if you are using z/OS 1.6 and DB2 Version 8, use an external security manager; for example, RACF.

DB2 security mechanism

The most practical way of managing access to a broker's DB2 resources is to define two RACF groups and connect users to these groups. For example, RACF groups MQP1ADM and MQP1USR are defined for broker MQP1BRK as follows:

- For group MQP1ADM

1. Grant this group DBADM authority for the broker database.
2. Typically owned by the WebSphere Message Broker administrator; user IDs must be added to this group who need to submit BIPCRBK to create a broker, or BIPMGCMP to migrate a broker.

- For group MQP1USR
  1. Give this group access to manipulate rows in the broker tables and allow select access to DB2 system tables. For example:

```
GRANT DELETE, INSERT, SELECT, UPDATE
     ON TABLE
              DB2_TABLE_OWNER.BSUBSCRIPTIONS
             ,DB2_TABLE_OWNER.BPUBLISHERS
             ,DB2_TABLE_OWNER.BCLIENTUSER
             ,DB2_TABLE_OWNER.BTOPOLOGY
             ,DB2_TABLE_OWNER.BNBRCONNECTIONS
             ,DB2_TABLE_OWNER.BRETAINEDPUBS
             ,DB2_TABLE_OWNER.BACLENTRIES
             ,DB2_TABLE_OWNER.BMQPSTOPOLOGY
             ,DB2_TABLE_OWNER.BUSERNAME
             ,DB2_TABLE_OWNER.BGROUPNAME
             ,DB2_TABLE_OWNER.BUSERMEMBERSHIP
             ,DB2_TABLE_OWNER.BROKERAA
             ,DB2_TABLE_OWNER.BROKERAAEG
             ,DB2_TABLE_OWNER.BROKERRESOURCES
             ,DB2_TABLE_OWNER.BRMINFO
             ,DB2_TABLE_OWNER.BRMRTDINFO
             ,DB2_TABLE_OWNER.BRMRTDDEPINFO
             ,DB2_TABLE_OWNER.BRMWFDINFO
             ,DB2_TABLE_OWNER.BRMPHYSICALRES
             ,DB2_TABLE_OWNER.BAGGREGATE
             ,DB2_TABLE_OWNER.BMULTICASTTOPICS
     TO MQP1USR;

GRANT SELECT
     ON TABLE
              SYSIBM.SYSTABLES
             ,SYSIBM.SYSSYNONYMS
             ,SYSIBM.SYSDATABASE
     TO MQP1USR;
```

  2. If a message flow issues a CALL statement to invoke a stored procedure, add select access to the following DB2 system tables:

```
GRANT SELECT
     ON TABLE
              SYSIBM.SYSROUTINES
             ,SYSIBM.SYSPARMS
     TO MQP1USR;
```

  3. Connect the broker started task user ID and the WebSphere Message Broker administrator to this group, and connect any other users who might need access to the tables, for example those submitting BIPRELG to run the mqsireadlog command.

The following conditions apply:

- When accessing DB2 resources, you specify a CURRENT SQLID as a DB2 command, or through the BIPDSNAO member.

  If this ID is a group, DB2 checks to see if your user ID is connected to this group, and if it is, you inherit the access from the group; if the user ID is not in the group, you get SQL code -551. If your ID is in multiple groups, the highest authorities are used.
- For BIPCRDB and BIPDLDB the CURRENT SQLID is specified as a command in the JCL. For all other JCL it is specified in BIPDSNAO.

- If you do not use groups to define permissions, but use a specific user ID to define the permissions to individual user IDs, if the granting user ID is removed from DB2, the permissions that it gave are also removed. This action can prevent other users working.
- When the BIPDLDB job drops the broker DB2 database, it also deletes any Image Copy references to itself that you currently have. If you restore the broker in future you need to reinstate the Image Copy references.

If this ID is a group, DB2 checks to see if your user ID is connected to this group, and if it is, you inherit the access from the group; if the user ID is not in the group, you get SQL code -551. If your ID is in multiple groups, the highest authorities are used.

If you do not use groups to define permissions, but use a specific user ID to define the permissions to individual user IDs, if the granting user ID is removed from DB2, the permissions that it gave are also removed. This action can prevent other users working.

See "z/OS JCL variables" on page 683 for further information on the WebSphere Message Broker for z/OS jobs that are supplied.

WebSphere MQ planning for 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 Message 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 251 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 see the *z/OS System Administration Guide* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online 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 on z/OS

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

WebSphere Message 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.

Automatic Restart Manager planning

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

WebSphere Message Broker for z/OS allows you to register a component to the Automatic Restart Manager (ARM).

When customizing a component, you register it with ARM by editing the following environment variables in the component's profile:

Description	Name
Switch that determines whether ARM will be used (YES or NO).	MQSI_USE_ARM
ARM element name	MQSI_ARM_ELEMENTNAME
ARM element type	MQSI_ARM_ELEMENTTYPE

By default components do not register to ARM; the initial setting is MQSI_USE_ARM=NO. You can override this by setting MQSI_USE_ARM=YES and providing an ARM element name and type.

MQSI_ARM_ELEMENTNAME must be a maximum of 8 characters in length, because WebSphere Message Broker adds a prefix of SYSWMQI. For example, if you supply the value MQP1BRK to ARM_ELEMENTNAME, the element you define in your ARM policy is SYSWMQI_MQP1BRK.

To enable automatic restart you must also:

- Set up an ARM couple data set.
- Define the automatic restart actions that you want z/OS to perform in an ARM policy.
- Start the ARM policy.

The following manuals provide detailed information about ARM couple data sets, including samples:

- *z/OS MVS Programming: Sysplex Services Guide*
- *z/OS MVS Programming: Sysplex Services Reference*
- *z/OS MVS Setting up a Sysplex*

You can access these manuals from the z/OS V1R7.0 LibraryCenter

Mounting file systems

For directories such as ++HOME++, ++COMPONENTDIRECTORY++, and ++INSTALL++ you must either create a directory, or mount a file system of this name, before using the directory.

To create a directory in an already mounted file system use the mkdir command. For example:

```
mkdir -p /mqsi/brokers/MQP1BRK
```

To mount a new file system, follow the instructions given in the *z/OS UNIX System Services Planning* manual.

From USS, use the following instruction:

```
mkdir -p /mqsi/brokers/MQP1BRK
```

From TSO, use the following instructions:

```
ALLOCATE DATASET('MQSI.BROKER.MQP1BRK') DSNTYPE(HFS) SPACE(5,5) DIR(1) CYL
FREE DATASET('MQSI.BROKER.MQP1BRK')
MOUNT FILESYSTEM('MQSI.BROKER.MQP1BRK') TYPE(HFS)
      MOUNTPOINT('/mqsi/brokers/MQP1BRK')
```

Note that the preceding ALLOCATE command is an example; the dataset should be allocated the correct amount of storage as described in "Disk space requirements on z/OS" on page 677

Checking the permission of the installation directory

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

You must ensure that the appropriate user IDs, for example, the WebSphere Message Broker Administrator and any component Started Task user IDs have READ and EXECUTE permission to the WebSphere Message Broker installation directory.

You are recommended to set these permissions using the group access control.

1. Display the permissions on the installation directory using the ls command.

   ```
   ls -l /usr/lpp/mqsi
   ```

   This command displays lines similar to the following:

   ```
   drwxr-xr-x   2 TSOUSER  MQM     8192 Jun 17 09:54 bin
   ```

   In this example, MQM is the group associated with the directory. Those user IDs requiring permission to the directory must be a member of this group.

   This example also shows the permissions defined for the directory. User TSOUSER has rwx (READ, WRITE and EXECUTE), group MQM has rx, and any other user ID has rx.

2. Ensure that the user IDs requiring permission have a group that matches that of the installation directory. Use the following command, where userid is the ID you want to check:

   ```
   id <userid>
   ```

3. If the installation directory does not have a valid group, use the command chgrp to set the group of the directory:

   ```
   chgrp -R <group> <pathname>
   ```

   For example:

   ```
   chgrp -R MQSI /usr/lpp/mqsi
   ```

   You must be the owner of the group or have superuser authority to use this command.

4. If the installation directory does not have the correct permissions for the group (READ / EXECUTE), use the command chmod to change the permissions:

   ```
   chmod -R g=rx <pathname>
   ```

   For example:

   ```
   chmod -R g=rx /usr/lpp/mqsi
   ```

Customizing the version of Java on z/OS

How you check the version of Java in your enterprise, and how you change the version, if necessary.

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

WebSphere Message Broker supports Java version 5 (SR5) (also known as Java 1.5.)

To check the current version of Java in a broker component:

1. Change to the `bin` directory in which you installed Java.

   For example, if Java is installed in `/usr/lpp/java`, change to the `/usr/lpp/java/bin` directory and type `./java -fullversion`. You receive a response similar to the following:

```
java version "1.5.0"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0)
Classic VM (build 1.5.0 J2RE 1.5.0 IBM z/OS Persistent Reusable VM build cm142-20040917
 (JIT enabled: jitc))
```

   The response in this example confirms that Java 1.5.0 is the current version. which is the minimum level required for this platform.

2. If you want to set, or change, the current Java version, edit JAVAHOME in the profile for the component (BIPBPROF for a broker, BIPUPROF for a User Name Server, BIPCPROF for a Configuration Manager).

   Because the version of Java used is defined in the component profile, you can specify a different version for each component.

   For further details about component profiles, see "Creating WebSphere Message Broker components on z/OS" on page 173.

3. Run BIPGEN to incorporate any changes that you make.
4. Restart the component to pick up changes to the ENVFILE.

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/mqsi/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/mqsi/bin/bipimain
```

You must have the appropriate authorization to issue this command.

Collecting broker statistics on z/OS

If you need to write broker statistics to SMF, you need to have the library `<HLQ>.SBIPAUTH` in your `STEPLIB`.

This library and all the libraries in the `STEPLIB` concatenation need to be APF authorized.

You can use the "mqsichangeflowstats command" on page 426 with `o=SMF` for this purpose.

Configuring an execution group address space as non-swappable on z/OS

Because broker execution groups run as processes in UNIX System Services they cannot be set as `NOSWAP` in the PPT.

Instead, you can set the following environment variables in the broker environment file so that some, or all, of the address spaces of the broker execution groups request that they become non-swappable by the system; see "Creating the environment file" on page 188 for further information on adding an environment variable to a broker.

```
MQSI_NOSWAP=yes
```

sets the address spaces of all the execution groups to be non-swappable.

```
MQSI_NOSWAP_egname=yes
```

issues a request to the system, for each execution group labelled `egname`, that the address space be set as non-swappable.

```
MQSI_NOSWAP_uuid=yes
```

issues a request to the system, for each execution group with the UUID labelled `uuid`, that the address space be set as non-swappable.

In order for the above requests to succeed, the broker's started task ID needs `READ` access to the `BPX.STOR.SWAP` facility class through their external security manager, for example, RACF.

When an application makes an address space non-swappable, it can cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this service can reduce the installation's ability to re-configure storage in the future.

Creating WebSphere Message Broker components on z/OS

An overview of how you create WebSphere Message Broker components on z/OS.

Before you start

Before starting this task, you must have installed:

- WebSphere MQ for z/OS, with the optional JMS feature applied; for example, mounted at `/usr/lpp/mqm`.
- WebSphere Message Broker for z/OS, with a broker file system mounted; for example, `/usr/lpp/mqsi`.

If you want to connect a Message Broker Toolkit directly to the Configuration Manager on z/OS, you *must* install the optional WebSphere MQ Client Attach feature.

If you do not have the WebSphere MQ Client Attach feature installed, you can connect the Message Broker Toolkit through an intermediate queue manager.

You should also read through all the sub topics in the "Customizing the z/OS environment" on page 152 section, and follow the guidance within those topics.

1. Determine the customization information for your environment. The following list of topics contains tables of information that needs to be gathered before you can proceed with creating a broker domain on z/OS. Complete the information that your enterprise requires.

   If necessary, discuss the requirements with your system, DB2, and WebSphere MQ administrators.

   - Broker:

- "Installation information - broker and User Name Server" on page 184
- "DB2 information" on page 184
- "Component information - broker" on page 185

- Configuration Manager:
  - "Installation information - Configuration Manager" on page 198
  - "Component information - Configuration Manager" on page 198
- User Name Server:
  - "Installation information - broker and User Name Server" on page 184
  - "Component information - User Name Server" on page 207

2. Set up security for the started task user IDs. Start with "Setting up z/OS security" on page 73.
3. Plan your DB2 requirements. Start with "DB2 planning on z/OS" on page 165.
4. Create the broker by carrying out the tasks listed in "Creating a broker on z/OS" on page 183. Start with "Creating the broker PDSE" on page 186.
5. Create the Configuration Manager by carrying out the tasks listed in "Creating a Configuration Manager on z/OS" on page 197. Start with "Creating the Configuration Manager PDSE" on page 199.
   a. Set up the connections between the Message Broker Toolkit and the Configuration Manager.
      - If you are using the WebSphere MQ Client Attach feature, see "Connecting directly to a Configuration Manager on z/OS."
      - If you are connecting through an intermediate queue manager; see "Connecting to a z/OS Configuration Manager through an intermediate queue manager" on page 175.
6. Optionally, create the User Name Server by carrying out the tasks listed in "Creating a User Name Server on z/OS" on page 206. Start with "Creating the User Name Server PDSE" on page 208.

Connecting directly to a Configuration Manager on z/OS

You can create a direct connection to a Configuration Manager in several ways.

You can connect from:

- The Message Broker Toolkit.
- A Configuration Manager Proxy (CMP) application.
- One of the WebSphere Message Broker commands used for the Configuration Manager; for example, mqsicreateexecutiongroup.

All of these options connect over a WebSphere MQ server-connection channel. The default name for this is `SYSTEM.BKR.CONFIG`, but you can use another value. You can connect directly to this channel only if you have the optional WebSphere MQ Client Attach feature installed.

Alternatively, you can connect through an intermediate queue manager; see "Connecting to a z/OS Configuration Manager through an intermediate queue manager" on page 175.

Before you start

Before starting this task, you must have set up your system components, as described in "Creating WebSphere Message Broker components on z/OS" on page 173.

Set up the connections between the Message Broker Toolkit and the Configuration Manager by carrying out the following tasks:

1. Verify that your queue manager and channel initiator are running and that the channel initiator is listening on the appropriate port.

   For more information see your WebSphere MQ documentation.
2. Ensure that your Configuration Manager is running; see "Starting and stopping a Configuration Manager on z/OS" on page 326.
3. Ensure that your user ID has been given the appropriate authorization on the z/OS Configuration Manager.

   In SDSF, grant FULL domain access to user ID `test1`. For all machines enter:

   ```
   '/F WMQxCFG,CA U=test1,A=YES,P=YES,X=F'
   ```

   To grant access to a specific machine for user `test1`, enter:

   ```
   '/F WMQxCFG,CA U=test1,M=mymachine,P=YES,X=F'
   ```
4. Create a new domain connection; see "Creating a domain connection" on page 251.

Connecting to a z/OS Configuration Manager through an intermediate queue manager

Connect a Message Broker Toolkit to a z/OS Configuration Manager without the need for the WebSphere MQ Client Attach feature.

You must use an intermediate queue manager on another platform, for example, Windows, to make this connection.

When carrying out this task, see your WebSphere MQ documentation for more information.

Before you start

Before starting this task, you must have set up your system components as described in "Creating WebSphere Message Broker components on z/OS" on page 173.

1. Set up the connections between the Message Broker Toolkit and the Configuration Manager by carrying out the following tasks:
   a. Verify that your queue manager and channel initiator are running and that the channel initiator is listening on the appropriate port.

      For more information, see your WebSphere MQ documentation.
   b. Ensure that your user ID has been given the appropriate authorization on the z/OS Configuration Manager.

      In SDSF, grant FULL domain access to user Id `test1`. For all machines enter:

      ```
      '/F WMQxCFG CA U=test1,A=YES,P=YES,X=F'
      ```

      To grant access to a specific machine for user `test1`, enter:

      ```
      '/F WMQxCFG CA U=test1,M=mymachine,P=YES,X=F'
      ```
2. Connect the intermediate queue manager to the z/OS queue manager using WebSphere MQ channels and a transmission queue. The connection must be bidirectional.
3. Ensure that you have started all the channels.
4. Create a server connection on the Windows queue manager. The default name for this connection is `SYSTEM.BKR.CONFIG`, but you can use another value for the

WebSphere Message Broker client connection. The Message Broker Toolkit connects to a server connection of this name on the identified queue manager.

5. Start the Configuration Manager on z/OS; see "Starting and stopping a Configuration Manager on z/OS" on page 326.
6. Start the Message Broker Toolkit on Linux on x86 or Windows.
7. Create a new domain connection; see "Creating a domain connection" on page 251.
8. Enter the following connection parameters for the domain connection:

```
hostname = (intermediate machine name, for example, localhost)
port = (listener port for intermediate queue manager, for example, 1414)
queue manager = (z/OS queue manager, for example, MQ05)
```

9. Right click **Domain** and select **Connect** to connect the Message Broker Toolkit to the z/OS Configuration Manager.

WebSphere Message Broker and WebSphere MQ setup verification

Whenever a component (that is a broker, User Name Server, or Configuration Manager) starts, a basic component verification runs automatically.

This verification checks:

- Basic WebSphere MQ information
- DB2 information
- Registry information
- Setup verification

If an error is found the component does not start. All output from the verification step is written to the component's JOBLOG.

You also need to validate that the entire WebSphere Message Broker system you installed is running correctly; that is, there are no errors in the system and that the system is performing as expected. This check is especially important where you are installing WebSphere Message Broker for the first time.

You should download and review *WebSphere Message Broker SupportPac IP13*.

This SupportPac provides:

- Tools to help test WebSphere Message Broker flows running on z/OS. These tools provide:
  - Put and get user messages
  - Measurement of the elapsed time
  - CPU time used to process the messages
  - Recommendations on the environment configuration.
- Recommendations on designing flows for evaluation, to reduce the need for applications outside of the WebSphere Message Broker environment.
- WebSphere Message Broker flows, which allow you to compare the throughput you achieve in your environment with that achieved at IBM. Using the tools supplied in *WebSphere Message Broker SupportPac IP13* can give suggestions as to why the results might be different.

Select the following link to access *WebSphere Message Broker SupportPac IP13* - IP13.

Configuring broker domain components

Create and configure the components that you want on the operating system of your choice.

Before you start:

Ensure that the following requirements are met:

- Your user ID has the correct authorizations to perform the task. The authorizations are defined in "Security requirements for administrative tasks" on page 723.
- Windows On Windows: you have created a new user ID, the service user ID. This ID is specified during component creation and is used to run the component (the Configuration Manager, broker, and User Name Server).

  For more information about user ID authorization and creation, refer to "Planning for security when you install WebSphere Message Broker" on page 27.
- You have initialized the command environment on distributed systems; see Setting up a command environment.

To create, modify, or delete a WebSphere Message Broker component:

1. Select the task for the component and action that you require from the list of tasks in this topic.
2. Select the operating system that you require from within the task.

Alternatively:

1. In the information center's table of contents, select the top-level container for the component and the action that you require (for example, "Creating a Configuration Manager" on page 192).
2. Expand the container.
3. Select the operating system that you require from the list of topics.

On Windows, you can create, modify, and delete physical components using the Command Assistant wizard.

If you want a default broker domainconfiguration on Linux on x86 or Windows, you can use the Default Configuration wizard. The Default Configuration wizard creates all the components that you need to start exploring WebSphere Message Broker and run the supplied samples. For more information, see "Using the Default Configuration wizard" on page 217.

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

The following set of tasks describes how to create and configure component databases, and how to create, modify, and delete the physical broker domain components and associated resources by using the command line. For information about how to complete these tasks by using the CMP API, see Developing applications using the CMP.

This collection of tasks uses specific resource names and user IDs. These names are examples only; you can use your own names. Follow existing naming conventions for WebSphere MQ and other resources.

- "Configuring broker and user databases" on page 109

- "Creating a broker"
- "Adding an execution group to a broker using the command line" on page 191
- "Adding an execution group to a broker on z/OS" on page 192
- "Creating a Configuration Manager" on page 192
- "Enabling a User Name Server" on page 202
- "Creating a User Name Server" on page 202
- "Using the Default Configuration wizard" on page 217
- "Using the Command Assistant wizard" on page 219
- "Verifying components" on page 221
- "Connecting components" on page 221
- "Tuning the broker" on page 223
- "Preparing the environment for WebSphere Adapters nodes" on page 230
- "Modifying a broker" on page 232
- "Modifying a Configuration Manager" on page 237
- "Modifying a User Name Server" on page 241
- "Moving from WebSphere Message Broker on a distributed platform to z/OS" on page 243
- "Deleting a broker" on page 244
- "Deleting an execution group from a broker using the command line" on page 244
- "Deleting a Configuration Manager" on page 246
- "Disabling a User Name Server" on page 248
- "Deleting a User Name Server" on page 248

Creating a broker

You can create brokers on every platform that is supported by WebSphere Message Broker. The broker runs as a 64-bit application on all platforms except Linux on x86, Windows, and z/OS.

Before you start:

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 723.
- On distributed systems, you must set up your command-line environment before creating a broker, by running the product profile or console; refer to Setting up a command environment.
- On z/OS, you must create and start the queue manager for this broker before you create the component.
- You must create the broker database before you create the broker; the tables in which the broker stores its internal data are created automatically when you create the first broker to use that database. Subsequent brokers that you create specifying the same database and database user ID share these tables; the rows within the tables are qualified by a unique identifier (UUID) for each broker.

  Although you can install different versions of the product on a single computer, you must ensure that brokers that share a database are at the same version (and are migrated at the same time, if appropriate).

Create a broker by using the command line on the computer on which you have installed the broker component. On Windows and Linux on x86, you can alternatively use the Command Assistant in the Message Broker Toolkit to complete this task.

- You must give the broker a name that is unique within the broker domain. You must also ensure that the broker name is unique on a single computer, if you have created multiple domains on that computer.
- Broker names are case-sensitive on all supported platforms, except Windows.
- You must associate each broker with its own dedicated WebSphere MQ queue manager.

The mode in which your broker is working, can affect the number of execution groups and message flows that you can deploy, and the types of node that you can use. See “Restrictions that apply in each operation mode” on page 368.

To create a broker, follow the link for the appropriate platform.

- Linux and UNIX
- Windows
- z/OS

Using WebSphere MQ trusted applications

Configure a broker to run as a WebSphere MQ trusted application.

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 723.
- 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 *Application Programming Guide* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

You can configure a broker to run as a trusted (fastpath) application on all supported platforms, with the exception of z/OS where the option is not applicable. If the broker is configured as a trusted application, it runs in the same process as the WebSphere MQ queue manager agent, and all broker processes benefit from an improvement in the overall system performance.

A broker does not run as a trusted application by default; you either create a trusted application by using the “mqsicreatebroker command” on page 513, or modify an existing broker by using the “mqsichangebroker command” on page 404.

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 the *Intercommunication* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

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 178 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.

     Refer to "Modifying a broker" on page 232 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 Linux and UNIX systems

On Linux and UNIX systems, create brokers on the command line; on Linux on x86, you can also create brokers in the Message Broker Toolkit by using the Command Assistant wizard.

Before you start:

- Ensure that you have already created the broker database. If you are not sure, check with your database administrator (DBA).
- If you want to configure the broker as a WebSphere MQ trusted application, see "Using WebSphere MQ trusted applications" on page 179.
- Read "Considering security for a broker" on page 49.
- Check which operation mode you are licensed to use. If you do not set a mode, the automatic default is enterprise mode; see Operation modes.

When you create a broker, the command creates the specified queue manager if it does not already exist. The broker database must already exist, but the tables in which the broker stores its internal data are created automatically when the first broker to use that database is created. Subsequent brokers that are created using the same database and database user ID will share these tables.

To create a broker:

1. Ensure that the user ID that the broker uses to connect to the broker database is authorized to create tables in the broker database. If you are not sure, check with your database administrator (DBA). The broker connects to the broker database by using the user ID and password that you specify in the **-i** and **-u** parameters of the mqsicreatebroker command when you create the broker.

   For more information, see "Authorizing access to broker and user databases" on page 122.
2. Define the ODBC data source name (DSN) of the broker database to enable the broker to make a connection. Multiple brokers on the same host can use the same ODBC DSN to connect to the same broker database.

   For more information, see "Enabling ODBC connections to the databases" on page 127.
3. Ensure that you are logged in using a user ID that has authority to run the mqsicreatebroker command.
4. Run the mqsiprofile script to set up the command environment for the broker:

   ```
   . install_dir/bin/mqsiprofile
   ```

   You must run this script before you can run any of the WebSphere Message Broker commands.

   For more information, see Setting up a command environment.
5. Run the SQL profile that was created when the broker database was created. For example, if the broker database is a DB2 instance, run the `db2profile`. For more information, see "Setting your environment to support 32-bit access to databases" on page 143.
6. Use the mqsicreatebroker command to create the broker.

   For example, if you want to create a broker called WBRK_BROKER on a queue manager called WBRK_QM with a broker database that has the data source name WBRKBKDB, enter the following command:

   ```
   mqsicreatebroker WBRK_BROKER -q WBRK_QM -i wbrkuid -a wbrkpw -n WBRKBKDB -u dbuid -p dbpw
   ```

   where:

   - *wbrkuid* and *wbrkpw* are the user name and password under which the broker runs.
   - *dbuid* and *dbpw* are the user name and password that the broker uses to access the broker database and create tables to store its internal data.

   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. For more information, see "Enabling a User Name Server" on page 202.

   For more information about the command options, and for more examples, see "mqsicreatebroker command" on page 513.
7. To enable function that becomes available in WebSphere Message Broker fix packs, use the **-f** parameter on the mqsichangebroker command. For more information, see "mqsichangebroker command" on page 404.

You have created a broker.

Next: Complete the following tasks:

1. Start the broker by using the mqsistart command.
2. Create other resources and components that you need.
3. Create a WebSphere MQ infrastructure to connect the components together; see "Connecting components" on page 221.

4. Add the broker to the broker domain:
   - To add the broker by using the workbench, see "Adding a broker to a broker domain" on page 255.
   - To add the broker by using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy.

When you have completed these tasks, you can create the resources that you want to associate with the broker; for example message flows. You can create and work with resources by using either the workbench or the CMP API.

Creating a broker on Windows

On Windows, you can create brokers on the command line or by using the Command Assistant wizard in the workbench.

Before you start:

- Ensure that the broker database has been created. If you are not sure, check with your database administrator (DBA).
- If you want to configure the broker as a WebSphere MQ trusted application, see "Using WebSphere MQ trusted applications" on page 179.
- Read "Considering security for a broker" on page 49.
- Check which operation mode you are licensed to use. If you do not set a mode, the automatic default is enterprise mode; see Operation modes.

When you create a broker, if the WebSphere MQ queue manager does not already exist, the queue manager is automatically created. The broker database must already exist but the tables in which the broker stores its internal data are created automatically when the first broker to use that database is created. Subsequent brokers that you create specifying the same database and database user ID share these tables.

To create a broker by using the Command Assistant wizard, see "Using the Command Assistant wizard" on page 219.

To create a broker by using the command line, complete the following steps:

1. Ensure that the user ID that the broker uses to connect to the broker database is authorized to create tables in the broker database. If you are not sure, check with your database administrator (DBA). The broker connects to the broker database using the user ID and password that you specify in the **-i** and **-u** parameters of the mqsicreatebroker command when you create the broker.

   For more information, see "Authorizing access to broker and user databases" on page 122.
2. Define the ODBC data source name (DSN) of the broker database to enable the broker to make a connection. Multiple brokers on the same host can use the same ODBC DSN to connect to the same broker database.

   For more information, see "Enabling ODBC connections to the databases" on page 127.
3. Open a WebSphere Message Broker command prompt for the runtime installation in which you want to create the broker. For more information about initializing the runtime environment, see Command environment: Windows platforms.
4. Use the mqsicreatebroker command to create the broker.

For example, if you want to create a broker called WBRK_BROKER on a queue manager called WBRK_QM with a broker database that has the data source name WBRKBKDB, enter the following command:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
-q WBRK_QM -n WBRKBKDB -u dbuid -p dbpw
```

where:

- *wbrkuid* and *wbrkpw* are the user name and password under which the broker runs.
- *dbuid* and *dbpw* are the user name and password that the broker uses to access the broker database and create tables to store its internal data.

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. For more information, see "Enabling a User Name Server" on page 202.

For more information about the command options, see "mqsicreatebroker command" on page 513.

5. To enable function that becomes available in WebSphere Message Broker fix packs, use the **-f** parameter on the mqsichangebroker command. For more information, see "mqsichangebroker command" on page 404.

You have created a broker.

Next: Complete the following tasks:

1. Start the broker by using the mqsistart command.
2. Create other resources and components that you need.
3. Create a WebSphere MQ infrastructure to connect the components together; see "Connecting components" on page 221.
4. Add the broker to the broker domain:
   - To add the broker by using the workbench, see "Adding a broker to a broker domain" on page 255.
   - To add the broker by using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy.

When you have completed these tasks, you can create the resources that you want to associate with the broker; for example message flows. You can create and work with resources by using either the workbench or the CMP API.

Creating a broker on z/OS

Create the broker component and the other resources on which it depends.

To create your broker, perform the following tasks in order:

1. "Collecting the information required to create a broker" on page 184
2. "Creating the broker PDSE" on page 186
3. "Creating the broker directory on z/OS" on page 186
4. "Customizing the broker component data set" on page 187
5. "Customizing the broker JCL" on page 187
6. "Creating the environment file" on page 188
7. "Priming DB2" on page 189
8. "Creating the broker component" on page 190
9. "Copying the broker started task to the procedures library" on page 190

To enable function that becomes available in WebSphere Message Broker fix packs, use the **-f** parameter on the mqsichangebroker command. For more information, see "mqsichangebroker command" on page 404.

Collecting the information required to create a broker:

This is part of the larger task of creating a broker on z/OS.

You must complete the information in each of the tables, at the following links, before continuing:

- "Installation information - broker and User Name Server"
- "DB2 information"
- "Component information - broker" on page 185

Installation information - broker and User Name Server:

Decide on the values for the list of the JCL variables for your system; an example installation value is provided for each one.

Collect the information shown in the Description column, and complete the values that you require for your particular system. You can see a complete list of the variables that you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example installation value	Your installation value
Fully qualified name of the product's SBIPPROC data set	N/A	<hlq>.SBIPPROC	
Fully qualified name of the product's SBIPSAMP data set	N/A	<hlq>.SBIPSAMP	
File system directory where the product has been installed	++INSTALL++	/usr/lpp/mqsi	
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	
Location of Java installation	++JAVA++	/usr/lpp/java/IBM/J1.5	
WebSphere MQ high-level qualifier	++WMQHLQ++	MQM.V600	
Location of IBM XML Toolkit installation	++XMLTOOLKIT++	/usr/lpp/ixm/IBM/xml4c-5_6	

Note: You must include the XML Toolkit Library in the BROKER STEPLIB or the system LINKLIST.

DB2 information:

Collect the information explained in the Description column and complete the values you require for your particular system. You can see a complete list of variables that you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example installation value	Your installation value
DB2 high-level qualifier	++DB2HLQ++.	SYS2.DB2.V810	
DB2 run library value	++DB2RUNLIB++	DSN810PK.RUNLIB.LOAD	
DB2 subsystem identifier	++DB2SUBSYSTEM++	DFK4	
DB2 plan name	++DB2DSNACLIPLAN++	DSNACLI	
DB2 program value	++DB2SAMPLEPROGRAM++	DSNTEP2	
DB2 plan value	++DB2SAMPLEPROGRAMPLAN++	DSNTEP81	
DB2 location value of the DB2 subsystem	++DB2LOCATION++	DSN810PK	
DB2 converter	++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 shown in the Description column and complete the values you require for your particular system. You can see a complete list of variables you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example component value	Your component value
Home directory of the file system for the broker's user ID	++HOME++	/u/mqp1brk	
Queue Manager associated with the broker	++QUEUEMANAGER++	MQP1	
File system directory where the broker is to exist	++COMPONENTDIRECTORY++	/mqsi/brokers/MQP1BRK	
Broker name	++COMPONENTNAME++	MQP1BRK	
Data set where all JCL relevant to the broker is saved	++COMPONENTDATASET++	TESTDEV.BROKER.MQP1BRK	
Profile name	++COMPONENTPROFILE++	BIPBPROF	**BIPBPROF**
Name of the Started Task JCL; can be a maximum of 8 characters	++STARTEDTASKNAME++	MQP1BRK	
Name of the broker DB2 database	++DB2DATABASE++	DMQP1BRK	
Name of the broker DB2 storage group	++DB2STORAGEGROUP++	MQP1STOR	
Name of the broker DB2 buffer pool	++DB2BUFFERPOOL++	BP0	
DB2 index buffer pool	++DB2INDEXBP++	BP0	
DB2 LOB table buffer pool	++DB2LOBBP++	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)
- A 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 task, you must have completed "Collecting the information required to create a broker" on page 184 and "Creating the broker PDSE."

1. Use the TSO command OMVS to get into OMVS.
2. Create the broker root directory using the command:

   ```
   mkdir -p <ComponentDirectory>
   ```

   The name of the directory must be the same as the JCL variable `++COMPONENTDIRECTORY++`.
3. Display the contents of the directory, which is currently empty, using the command:

   ```
   ls -dl /var/wmqi/MQP1BRK
   ```
4. Display the permissions on the directory using the command:

   ```
   ls -al /var/wmqi/MQP1BRK
   ```
5. Ensure that the user ID of the person doing the customization has a group that matches the group of the directory. Use the following command, where `userid` is the ID you want to check:

   ```
   id <userid>
   ```
6. Check that the directory has a valid group, and that the group has rwx permissions. If they do not, use the following command to set the group of the directory:

   ```
   chgrp <group> <pathname>
   ```

   For example:

   ```
   chgrp WMQI /var/wmqi/MQP1BRK
   ```

   You must be the owner of the group, or have superuser authority, to use this command.

7. To give the group READ, WRITE, and EXECUTE access, use the following command:

   ```
   chmod g=rwx <pathname>
   ```

   For example:

   ```
   chmod g=rwx /usr/wmqi/MQP1BRK
   ```

8. To display the amount of space used and available, use the following command:

   ```
   df -P /var/wmqi/MQP1BRK
   ```

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 task, you must have completed "Collecting the information required to create a broker" on page 184.

Create the broker data set in TSO, identified by ++COMPONENTDATASET++, as instructed below:

1. Copy the members specified in "Broker PDSE members originating in <hlq>.SBIPSAMP" on page 679 from <hlq>.SBIPSAMP to ++COMPONENTDATASET++. Ensure that you copy only the listed files.
2. Copy the members specified in "Broker PDSE members originating in <hlq>.SBIPPROC" on page 679 from <hlq>.SBIPPROC to ++COMPONENTDATASET++. Ensure that you copy only the listed files.
3. Customize the JCL.

Customizing the broker JCL:

This is part of the larger task of creating a broker on z/OS.

Before you start

Before starting this task, you must have completed "Customizing the broker component data set."

All JCL has a standard header, comprising:

- A brief description of its function.
- A description where further information can be found, relating to the function of the JCL.
- If appropriate, a topic number.
- The section listing the JCL variables themselves.

Each JCL file defines its own STEPLIB. 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, because 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 you can use to help you customize your JCL. After you have customized BIPEDIT you can run this REXX program against the other JCL files to change their JCL variables.

When you update BIPBPROF (the broker profile), the changes are not accessible until you run BIPGEN to copy the profile to the file system and create the ENVFILE. You must do this each time you update BIPBPROF for the changes to take effect.

1. Customize the renamed BIPEDIT file. Use the information you collected in:
   - "Installation information - broker and User Name Server" on page 184
   - "DB2 information" on page 184
   - "Component information - broker" on page 185
2. 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.
3. Edit each JCL file. Run the renamed BIPEDIT exec by typing its name on the command line (for example MQ01EDBK). Instead of editing a member, you might want to View it until you have resolved any problems in your REXX program. Alternatively, you can Cancel the Edit session instead of saving it.

   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; you must replace them with additional optional parameters specific to the command on z/OS, or remove them. Typically, 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 of 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 you intend to have different administrators create the DB2 and WebSphere MQ resources, you can consider using one of these optional parameters; see "mqsicreatebroker command" on page 513 for further information.

You need to be aware that another process might be using the current ENVFILE, therefore you need to consider whether updating the current ENVFILE in the file system 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 task, you must have completed "Customizing the broker JCL" on page 187

1. Review the BIPBPROF member. If you define parameters for all users, you can configure BIPBPROF to use these parameters.

   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.

Priming 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). Your DB2 systems administrator will tell you which buffer pools and storage groups to use.

The BIPCRDB job issues commands that require the following authorities:

- 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 job again when you have resolved the problems.

Before you start

Before starting this task, you must have completed "Creating the environment file" on page 188.

1. Edit the BIPCRDB job.
2. Review and change the BUFFERPOOL and INDEXBP buffer pools on the CREATE DATABASE statement
3. Change the buffer pool specification on the CREATE LOB TABLESPACE statements to a value suitable for your enterprise.
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 task, you must have completed "Priming DB2" on page 189.

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 "mqsicreatebroker command" on page 513 for further information.

1. Submit job BIPCRBK with option -1. This job creates the files and directories that 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.
4. Ensure that the jobs have run successfully by:

   Checking the STDOUT stream in the JOBLOG.

   Viewing STDOUT for any errors and checking for`BIP8071I: Successful command completion`.

If you encounter any problems, delete the broker and recreate it using the following procedure. You must have the appropriate authority to run the jobs.

1. Edit and configure job BIPDLBK.
2. Run job BIPDLBK with the same option, or options, that caused the problems when you ran the BIPCRBK job.
3. Correct the problems and run the BIPCRBK job again.

Copying the broker 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 task, you must have completed "Creating the broker component."

1. Ensure that the user ID for the broker started task 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 73 and "Summary of required access (z/OS)" on page 673 for more information.

2. Copy the Started Task JCL (BIPBRKP) to the procedures library, for example USER.PROCLIB.

Adding an execution group to a broker using the command line

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly.

Before you start:

You must complete the following tasks:

- "Adding a broker to a broker domain" on page 255

The mode that your broker is working in can affect the number of execution groups that you can use; see "Restrictions that apply in each operation mode" on page 368.

You can use one of three methods to complete this task:

- The workbench
- The mqsicreateexecutiongroup command
- The CMP API

This task describes the second method. For information about creating an execution group in the workbench, see "Adding an execution group to a broker in the workbench" on page 259. For information about using the CMP API, see Developing applications that use the Configuration Manager Proxy API.

For details about why you might want to create multiple execution groups, see Execution groups.

To add an execution group to a broker using the command line on Linux, UNIX and Windows systems:

1. Open a command prompt that has the environment configured for your current installation.
2. Enter the following command to add the execution group:

   ```
   mqsicreateexecutiongroup -i host -p 1414 -q QMGR -b BROKER –e EG1
   ```

   where

   host The host name or IP address of the Configuration Manager for the domain on which the broker resides.

   1414 The port on which the queue manager for the Configuration Manager is listening.

   QMGR
   The name of the queue manager for the Configuration Manager.

   BROKER
   The name of the broker.

   EG1 The name of the execution group that you want to create.

To add an execution group to a broker on z/OS systems, see "Adding an execution group to a broker on z/OS" on page 192.

When you've completed this task the execution group has been created in the workbench and you must deploy to the broker to update the broker's configuration.

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 255

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 Broker 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 Broker Toolkit see "Adding an execution group to a broker in the workbench" on page 259

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

You can create a Configuration Manager on every platform that is supported by WebSphere Message Broker.

Before you start:

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 723.
- For platforms other than z/OS, see "Considering security for a Configuration Manager" on page 51 for information about security matters relevant to a Configuration Manager during the configuration task.

  To create a Configuration Manager, follow the link for the appropriate platform.
- On distributed systems, you must set up your command-line environment before creating a Configuration Manager, by running the product profile or console; refer to Setting up a command environment.
- On z/OS, you must create and start the queue manager for this Configuration Manager before you create the component.

Create a Configuration Manager using the command line on the system where the Configuration Manager component is installed. On Windows and Linux on x86, you can alternatively use the Command Assistant in the Message Broker Toolkit to complete this task.

To create a Configuration Manager, follow the link for the appropriate platform.

- AIX
- HP-UX
- Linux
- Solaris
- Windows
- z/OS

Creating a Configuration Manager on AIX

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 Message 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 Message 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 221.

2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 251.

Creating a Configuration Manager on HP-UX

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 Message 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 Message 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 221.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 251.

Creating a Configuration Manager on Linux

Follow the steps detailed in this task for creating a Configuration Manager onLinux platforms.

Linux 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 Message 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 Message 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 221.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 251.

Creating a Configuration Manager on Solaris

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 Message 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 Message 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 221.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 251.

Creating a Configuration Manager on Windows

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 Message 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 Message 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 221.
2. Create a broker domain connection using the workbench. Refer to "Creating a domain connection" on page 251.

Creating a Configuration Manager on z/OS

Create the Configuration Manager component and the other resources on which it depends.

To create your Configuration Manager, perform the following tasks in order.

1. "Collecting the information required to create a Configuration Manager on z/OS"
2. "Creating the Configuration Manager PDSE" on page 199
3. "Creating the Configuration Manager directory on z/OS" on page 199
4. "Customizing the Configuration Manager component data set" on page 199
5. "Customizing the Configuration Manager JCL" on page 200
6. "Creating the Configuration Manager component" on page 201
7. "Copying the Configuration Manager started task to the procedures library" on page 202

Collecting the 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"
- "Component information - Configuration Manager"

Installation information - Configuration Manager:

This topic gives installation details of the Configuration Manager.

Collect the information shown in the Description column and complete the values you require for your particular system. You can see a complete list of variables you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example installation value	Your installation value
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	
WebSphere MQ high-level qualifier	++WMQHLQ++	MQM.V600	
Location of Java installation	++JAVA++	/usr/lpp/java/IBM/J1.5	
Location of IBM XML Toolkit installation	++XMLTOOLKIT++	/usr/lpp/ixm/IBM/xml4c-5_6	
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 file system install directory	++MQPATH++	/usr/lpp/mqm	

Component information - Configuration Manager:

Collect the information shown in the Description column and complete the values you require for your particular system. You can see a complete list of variables you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example component value	Your component value
Configuration Manager name	++COMPONENTNAME++	MQP1CMGR	
Configuration Manager's user ID file system home directory	++HOME++	/u/mqp1cmgr	
File system directory where the Configuration Manager is to exist	++COMPONENTDIRECTORY++	/mqsi/configmgrs/MQP1CMGR	

Data set where all JCL relevant to the Configuration Manager is saved	++COMPONENTDATASET++	TESTDEV.CMGR.MQP1CMGR	
Profile name	++COMPONENTPROFILE++	BIPCPROF	**BIPCPROF**
Name of the Started Task JCL; can be a maximum of 8 characters	++STARTEDTASKNAME++	MQP1CMGR	
mqsicreateconfigmgr options	++OPTIONS++	Any additional optional parameters for the **mqsicreateconfigmgr** command	

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)
- A data set type of `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 tasks:

- "Collecting the information required to create a Configuration Manager on z/OS" on page 197
- "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 186 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 task, you must have completed "Collecting the information required to create a Configuration Manager on z/OS" on page 197.

Create the Configuration Manager data set in TSO, identified by ++COMPONENTDATASET++, as instructed below:

1. Copy BIPCPROF from <hlq>.SBIPSAMP to ++COMPONENTDATASET++.
2. Copy the members specified in "Configuration Manager PDSE members originating in <hlq>.SBIPPROC" on page 682 from <hlq>.SBIPPROC to ++COMPONENTDATASET++. Ensure that you copy only the listed files.
3. Customize the JCL.

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 199.

- A brief description of its function.
- A description where further information can be found, relating to the function of the JCL.
- If appropriate, a topic number.
- The section listing the JCL variables themselves.

Each JCL file defines its own STEPLIB. 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, because 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 you can use to help you customize your JCL. After you have customized BIPEDIT you can run this REXX program against the other JCL files to change their JCL variables.

When you update BIPCPROF (the Configuration Manager profile), the changes are not accessible until you run BIPGEN to copy the profile to the file system 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.

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.

1. Customize the renamed BIPEDIT file. Use the information you collected in:
   - "Installation information - Configuration Manager" on page 198
   - "Component information - Configuration Manager" on page 198
2. 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.

3. Edit each JCL file. Run the renamed BIPEDIT exec by typing its name on the command line (for example MQ01EDCM). Instead of editing a member, you might want to View it until you have resolved any problems in your REXX program. Alternatively, you can Cancel the Edit session instead of saving it.

   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; you must replace them with additional optional parameters specific to the command on z/OS, or remove them. 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 of the members except the edit macro itself.

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

Complete "Customizing the Configuration Manager JCL" on page 200.

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 you expect that a different administrator will create the WebSphere MQ resources, consider using one of these optional parameters; see the "mqsicreateconfigmgr command" on page 525 for further information.

1. Submit job BIPCRCM with option -1. This job creates the Configuration Manager together with the files and directories that are placed in the registry. To run this job, you must have 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.
3. Ensure that the jobs have run successfully:

   Check the STDOUT stream in the JOBLOG.

   View STDOUT for any errors and checking for`BIP8071I: Successful command completion.`

If you encounter any problems, delete the Configuration Manager and recreate it using the following procedure. You must have the appropriate authority to run the jobs.

1. Edit and configure job BIPDLCM.
2. Run job BIPDLCM with the same option, or options, that caused the problems when you ran the BIPCRCM job.
3. Correct the problems and run the BIPCRCM job again.

The BIPCRCM job can take several minutes to run, depending on the content of the remote database.

Copying the Configuration Manager 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 task, you must have completed "Creating the Configuration Manager component" on page 201.

Copy the Started Task JCL (BIPCMGRP) to the procedures library, for example USER.PROCLIB.

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 232.
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 237.

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

You can create a User Name Server on every platform that is supported by WebSphere Message Broker.

Before you start:

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 723.

- 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" on page 202.
- On distributed systems, you must set up your command-line environment before creating a User Name Server, by running the product profile or console; refer to Setting up a command environment
- On z/OS, you must create and start the queue manager for this User Name Server before you create the component.

Create a User Name Server using the command line on the system where the User Name Server component is installed. On Windows and Linux on x86, you can alternatively use the Command Assistant in the Message Broker Toolkit to complete this task.

To create a User Name Server, follow the link for the appropriate platform.

- AIX
- HP-UX
- Linux
- Solaris
- Windows
- z/OS

Creating a User Name Server on AIX

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 Message 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 221.

Creating a User Name Server on HP-UX

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 Message 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 221.

Creating a User Name Server on Linux

Follow the steps detailed in this task for creating a User Name Server on Linux platforms.

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 Message 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 221.

Creating a User Name Server on Solaris

Solaris The following steps describe 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 Message 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 221.

Creating a User Name Server on Windows

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 Message 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 221.

Creating a User Name Server on z/OS

Create the User Name Server component and the other resources on which it depends.

To create your User Name Server, perform the following tasks in order.

1. "Collecting the information required to create a User Name Server on z/OS" on page 207

2. "Creating the User Name Server PDSE" on page 208
3. "Creating the User Name Server directory on z/OS" on page 208
4. "Creating the User Name Server runtime environment on z/OS" on page 209
5. "Customizing the User Name Server component data set" on page 209
6. "Customizing the User Name Server JCL" on page 210
7. "Creating the User Name Server component" on page 211
8. "Copying the User Name Server started task to the procedures library" on page 212

Collecting the 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 184
- "Component information - User Name Server"

Installation information - broker and User Name Server:

Decide on the values for the list of the JCL variables for your system; an example installation value is provided for each one.

Collect the information shown in the Description column, and complete the values that you require for your particular system. You can see a complete list of the variables that you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example installation value	Your installation value
Fully qualified name of the product's SBIPPROC data set	N/A	<hlq>.SBIPPROC	
Fully qualified name of the product's SBIPSAMP data set	N/A	<hlq>.SBIPSAMP	
File system directory where the product has been installed	++INSTALL++	/usr/lpp/mqsi	
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	
Location of Java installation	++JAVA++	/usr/lpp/java/IBM/J1.5	
WebSphere MQ high-level qualifier	++WMQHLQ++	MQM.V600	
Location of IBM XML Toolkit installation	++XMLTOOLKIT++	/usr/lpp/ixm/IBM/xml4c-5_6	

Note: You must include the XML Toolkit Library in the BROKER STEPLIB or the system LINKLIST.

Component information - User Name Server:

Collect the information shown in the Description column and complete the values you require for your particular system. You can see a complete list of variables you can customize in "z/OS JCL variables" on page 683.

Description	JCL variable	Example component value	Your component value
User Name Server's user ID file system home directory	++HOME++	/u/mqp1uns	
File system directory where the User Name Server is to exist	++COMPONENTDIRECTORY++	/mqsi/uns/MQP1UNS	
Data set where all JCL relevant to the User Name Server is saved	++COMPONENTDATASET++	TESTDEV.UNS.MQP1UNS	
Profile name	++COMPONENTPROFILE++	BIPUPROF	**BIPUPROF**
Name of the Started Task JCL; can be a maximum of 8 characters	++STARTEDTASKNAME++	MQP1UNS	
mqsicreateusernameserver options	++OPTIONS++	Any additional optional parameters for the **mqsicreateusernameserver** command	

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)
- A data set type of `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 tasks:

- "Collecting the information required to create a User Name Server on z/OS" on page 207
- "Creating the User Name Server PDSE"

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 186 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 208.

Use the **mqsicreateusernameserver** command to create a User Name Server and its runtime environment. The command syntax is:

```
mqsicreateusernameserver -q QueueManagerName [-r RefreshInterval] [-1] [-2]
```

where:

-q *QueueManagerName*
is a required parameter and is the name of the WebSphere MQ queue manager associated with your User Name Server, for example MQP1.

-r *RefreshInterval*
is an optional parameter and is 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 is an optional parameter and is the registry pass; this creates only the User Name Server registry.

-2 is an optional parameter and is the WebSphere MQ pass; this creates only the User Name Server WebSphere MQ queues.

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, run **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 Integrator 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.

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 task, you must have completed "Collecting the information required to create a User Name Server on z/OS" on page 207.

Create the User Name Server data set in TSO, identified by ++COMPONENTDATASET++, as instructed below:

1. Copy BIPUPROF from <hlq>.SBIPSAMP to ++COMPONENTDATASET++.

2. Copy the members specified in "User Name Server PDSE members originating in <hlq>.SBIPPROC" on page 681 from <hlq>.SBIPPROC to ++COMPONENTDATASET++. Ensure that you copy only the listed files.
3. Customize the JCL.

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 task, you must have completed "Customizing the User Name Server component data set" on page 209.

All JCL has a standard header, comprising:

- A brief description of its function.
- A description where further information can be found, relating to the function of the JCL.
- If appropriate, a topic number.
- The section listing the JCL variables themselves.

Each JCL file defines its own STEPLIB. 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, because 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.

When you update BIPUPROF (the User Name Server profile), the changes are not accessible until you run BIPGEN to copy the profile to the file system and create the ENVFILE. You must do this each time you update BIPUPROF for the changes to take effect.

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.

1. Customize the renamed BIPEDIT file. Use the information you collected in:
   - "Installation information - broker and User Name Server" on page 184
   - "Component information - User Name Server" on page 207
2. 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.
3. Edit each JCL file. Run the renamed BIPEDIT exec by typing its name on the command line (for example MQ01EDUN). Instead of editing a member, you might

want to View it until you have resolved any problems in your REXX program. Alternatively, you can Cancel the Edit session instead of saving it.

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.

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 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 task, you must have completed "Customizing the User Name Server JCL" on page 210.

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 "mqsicreateusernameserver command" on page 540 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.
3. Ensure that the jobs have run successfully by:

   Checking the STDOUT stream in the JOBLOG.

   Viewing STDOUT for any errors and checking for`BIP8071I: Successful command completion.`

If you encounter any problems, delete the User Name Server and recreate it using the following procedure. Note that you need the appropriate authority to run the jobs.

1. Edit and configure job BIPDLUN.
2. Run job BIPDLUN with the same option, or options, that caused the problems when you ran the BIPCRUN job.
3. Correct the problems and run the BIPCRUN job again.

Copying the User Name Server 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 task, you must have completed "Creating the User Name Server component" on page 211

Copy the Started Task JCL (`BIPUNSP`) to the procedures library, for example `USER.PROCLIB`.

Connecting the User Name Server to the WebSphere Message 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 and stopping the User Name Server on z/OS" on page 328.

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 221.

The Configuration Manager requests user IDs and group information from the User Name Server. The WebSphere Message 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 213
- "Starting the WebSphere MQ channels and listeners" on page 214
- "Enabling applications to use Publish/Subscribe" on page 216
- "Enabling applications to use Publish/Subscribe security on z/OS" on page 216

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 202

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 69.

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 Message
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 206

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 69.

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 Message
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 213.

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.

UNIX systems:

1. To start a listener enter the following command in a shell window:

```
runmqlsr -t tcp -p 1414 -m WBRK_QM
```

2. To start a sender channel, enter the following command in a shell window:

```
runmqchl -c BROKER.CONFIG -m WBRK_QM
```

LINUX systems:

- If you are using WebSphere MQ Version 6.0, listeners and channels can be started using the WebSphere MQ Explorer, in the same way as with Windows platforms, as described in "Windows platforms using WebSphere MQ Version 6."
- If you are using WebSphere MQ Version 5, listeners and channels must be started by entering commands in a shell window, as described in this section.

1. To start a listener, enter the following command in a shell window:

```
runmqlsr -t tcp -p 1414 -m WBRK_QM
```

2. To start a sender channel, enter the following command in a shell window:

```
runmqchl -c BROKER.CONFIG -m WBRK_QM
```

Windows platforms using WebSphere MQ Version 6:

If you are using WebSphere MQ Version 6.0, start listeners and channels using WebSphere MQ Explorer.

1. To start a listener as a background task:
   a. Click **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**.
   b. In the left pane expand the queue manager, expand **Advanced**, and select **Listeners**
   c. Right-click **Listeners** → **New** → **TCP Listener...**, enter a name for the listener, then click **Finish**. A new listener is created with transport type TCP and (default) port number 1414.
   d. Right-click the new listener and click **Start** to start it.
2. To start the sender channels as background tasks using WebSphere MQ Explorer expand the queue manager, expand **Advanced**, and select **Channels**.
3. If you prefer, you can start listeners and channels as foreground tasks:
   a. To start a listener, enter the following command on the command line:

```
runmqlsr -t tcp -p 1414 -m WBRK_CONFIG_QM
```

   b. To start channels, enter the following commands:

```
runmqchl -m WBRK_UNS_QM -c WBRK_UN_TO_BR

runmqchl -m WBRK_QM -c WBRK_BR_TO_UN
```

Windows platforms using WebSphere MQ Version 5:

If you are using WebSphere MQ Version 5, start the listeners using WebSphere MQ Services, and start channels by entering commands on the command line.

1. To start a listener as a background task:
   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 and select **All Tasks** → **Start** to start the listener.
   d. If the listener is not displayed:

1) Right-click the queue manager and select **New → Listener**. A new listener is created with (default) transport type TCP and (default) port number 1414.
2) Right-click the new listener and select **Start** to start it.

2. If you prefer, you can start a listener as a foreground task. Enter the following command on the command line:

```
runmqlsr -t tcp -p 1414 -m WBRK_CONFIG_QM
```

3. To start the channels as foreground tasks, enter the following 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

Use the Default Configuration wizard to set up and test a basic broker domain configuration.

Before you run the Default Configuration wizard:

- You must install the Message Broker Toolkit and all of the runtime components on this computer. Access the Default Configuration wizard through the Message Broker Toolkit, which is available on Linux on x86 and Windows only.
- Windows On Windows, you must have Administrator privileges, and your user ID must be a local ID (not a domain ID).
- A database must be available, and your user ID must be authorized to create databases:
  - Linux On Linux on x86, install DB2 Enterprise Server.
  - Windows On Windows, install either DB2 Enterprise Server or the ODBC Drivers for Apache Derby (to use the embedded Derby database).

Using the Default Configuration wizard, you set up a basic configuration on your local machine so that you can explore the product and run the samples that are supplied in the Samples Gallery. You can also remove the default configuration, if it already exists, that has been set up on your logon account.

- "Creating the default configuration"
- "Removing the default configuration" on page 218

The default configuration is described in more detail in the Installation Guide, which also describes how you can verify your installation by using the sample programs; see Installation Guide.

Creating the default configuration

The wizard creates the following resources:

- A broker domain with a broker named WBRK61_DEFAULT_BROKER and a Configuration Manager named WBRK61_DEFAULT_CONFIGURATION_MANAGER.
- A WebSphere MQ queue manager named WBRK61_DEFAULT_QUEUE_MANAGER
- A database named DEFBKD61 to be used by the broker .

  On Windows, the wizard defaults to whatever database manager is available. The database that is used is recorded on the Default Configuration Summary page. Details of the database manager are also written to the wizard's log file.

1. On Windows, the Default Configuration wizard starts automatically at the end of the installation of WebSphere Message Broker. You can start the wizard manually from the Message Broker Toolkit Welcome page, which is displayed

the first time you start the Message Broker Toolkit. If the Welcome page is not displayed, open it in the Message Broker Toolkit by clicking **Help** → **Welcome**.

2. The Welcome page 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 checks that the default configuration is not already installed.

3. The Default Configuration Summary page lists the resources that will be created.The information field in this page confirms whether Derby has been set as the default broker database on Windows. It also suggests an alternative option of installing and configuring an enterprise database server instead. Click **Next** to continue.

4. The Default Configuration Progress page lists the background configuration actions as they occur, and indicates successful completion. You can cancel the creation of the default configuration at this point by clicking **Cancel**. The wizard backs out all of the configuration tasks and then displays the progress and success of the process. The configuration process is written to a log file in the Eclipse workspace directory:

   - Linux *user_home_directory*\IBM\wmbt61\workspace\.metadata\DefaultConfigurationWizard.log
   - Windows *user_home_directory*/IBM/wmbt61/workspace/.metadata/DefaultConfigurationWizard.log

   If the default configuration is set up successfully, you see an appropriate message. If errors occur, you see an appropriate message and the wizard backs out all of the 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. **Launch Samples Wizard when finished** is selected by default. Click **Finish** to open the Prepare the Samples wizard.

   If you do not want to open the Prepare the Samples wizard, clear **Launch Samples Wizard when finished** before clicking **Finish**.

   If you are viewing this information from within the Message Broker Toolkit, you can open the samples manually by clicking on the following sample:

   - Pager samples

   Alternatively, click **Help** → **Samples Gallery**, and expand **Application samples** → **Message Broker**.

   You can view samples only when you use the information center that is integrated with the Message Broker Toolkit.

Removing the default configuration

1. Start the Default Configuration wizard from the Message Broker Toolkit Welcome page, which is displayed after you start the Message Broker Toolkit. If the Welcome page is not displayed, open it in the Message Broker Toolkit by clicking **Help** → **Welcome**.

2. The Welcome page 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 checks that the default configuration is already installed.

3. The Remove Default Configuration Summary page lists the resources that will be removed. Click **Next** to continue.

4. The Default Configuration Progress page lists the removal actions as they occur, and indicates successful completion. The removal process is written to a log file in the Eclipse workspace directory:
   - Linux *user_home_directory*\IBM\wmbt61\workspace\.metadata\DefaultConfigurationWizard.log
   - Windows *user_home_directory*/IBM/wmbt61/workspace/.metadata/DefaultConfigurationWizard.log
5. A message confirms that the default configuration has been removed successfully. Click **Finish** to close the wizard.

   If errors occur during the removal of the default configuration, the wizard displays the errors, and also writes them to the log file. Follow the advice in the log, and try each step again.

If you experience problems when you are using the wizard to remove the default configuration, you might have to remove the default configuration manually. For more information, see You experience problems with the default configuration.

Using the Command Assistant wizard

Use the Command Assistant wizard to create, modify, and delete physical runtime components.

Before you start:

- The Message Broker Toolkit and all runtime components must be installed on this system. The Command Assistant wizard is available through the toolkit, and is available on Windows only.
- You must have administration privileges.

Use the Command Assistant wizard to create, modify, and delete the following physical runtime components:
- Brokers
- Configuration Managers
- User Name Servers

By using the wizard, you access the equivalent command line command through a graphical interface:
- "mqsicreatebroker command" on page 513, "mqsicreateconfigmgr command" on page 525, and "mqsicreateusernameserver command" on page 540
- "mqsichangebroker command" on page 404, "mqsichangeconfigmgr command" on page 415, and "mqsichangeusernameserver command" on page 499
- "mqsideletebroker command" on page 555, "mqsideleteconfigmgr command" on page 557, and "mqsideleteusernameserver command" on page 566

The wizard displays a series of panels that lead you through the task that you want to complete. The wizard provides help, in the banner at the top of each panel, that indicates what actions you should take to complete the panel and continue. Not every optional parameter on each of these commands is supported through the wizard; if you are using some of the more advanced features (for example, setting or modifying LDAP directory access for a broker), you must use the command line interface.

Use the buttons displayed at the bottom of each panel to move **Back** to the previous panel, to move to the **Next** panel, to **Finish** working with the wizard, or to **Cancel** the current action and end the wizard.

1. Switch to the Broker Application Development perspective or the Broker Administration perspective.
2. Select **File** → **New** → **Other**. The **New** dialog opens.
3. Select **Command Assistant Wizard** within the **Broker Administration - Getting Started** category and click **Next**. The wizard opens and displays its first panel.
4. Select the type of component that you want to work with. The wizard checks which components have been installed on this system. You must select a value from the available list; you cannot enter a different value.
5. Navigate to the **Name** input field. If the wizard has found any existing components of the type that you entered, it shows these in this field. If no components of this type exist, or you want to create a new component, enter a new unique name in this field, following the naming restrictions enforced by the product and any naming conventions that are in use in your environment.

   The name of a resource is case insensitive. If a resource of the same name exists, but with characters in a different case to those that you typed into this field, the name that you typed is overwritten with the existing name.
6. Navigate to the **Action** input field and choose the action that you want to complete. The wizard prevents you from entering an invalid action for the resource that you have entered in the **Name** field. For example, if you have entered a name that the wizard has not found on this system, you can choose only to create a new resource. If the resource already exists, you can choose to modify it or delete it.
7. If you have more than one installation of the product on this system, select the correct value in the **Location** field. This field displays the home directory of the installation identified by the wizard:
   - If only one installation exists on this system, the directory is displayed and the field is read-only.
   - If you have specified an existing, uniquely named, resource in the **Name** field, the wizard displays the location of the installation that is associated with that resource, and the field is read-only.
   - If more than one installation exists, and might be the target for your request, select the correct location in this field.
8. Click **Next**. The wizard displays the next panel, the content of which depends on your choices so far.

   Use the help that is displayed by the wizard on each panel, and navigate through the entry fields, selecting or entering text where appropriate.

   If you enter a password, the characters are displayed as asterisk characters in the entry field to increase security.
9. When you have completed the entry field on the panel, click **Next**. The wizard displays a summary that shows you the commands that will be invoked, and any additional actions that will be taken.
10. Check the information in the summary; if it is correct, click **Next**.

    If you want to change anything, click **Back** to return to a previous panel and change your input.
11. The wizard starts processing your request. If the action succeeds, the wizard displays messages in the summary panel.

    If an action fails, the wizard reports the error in a message dialog. If you know what is causing the error, and can fix it, correct the error and click **Yes**. The wizard reissues the command.

If you do not know what is causing the error, or you cannot fix it, click **No**. The wizard backs out any actions that have already completed and returns your system to its initial state.

12. Click **Finish** to end the wizard, or click **Next** to return to the first page and select another task to complete.

Verifying components

To verify that the WebSphere Message 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

Create connections between the Configuration Manager, the brokers, and the User Name Server, if you have created a User Name Server as a component of your broker domain.

Before you start:

Complete the following tasks:

- "Creating a Configuration Manager" on page 192
- "Creating a broker" on page 178
- (Optional) "Creating a User Name Server" on page 202

The following steps describe how to make connections between the Configuration Manager, the brokers, and the User Name Server, if you have created a User Name Server as a component of your broker domain.

If the components in your broker domain are supported by different queue managers, you must establish WebSphere MQ connections between those queue managers to enable messages to be exchanged. Every broker must be 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 those components. If you have more than one broker, each broker must have its own queue manager; brokers cannot share a queue manager.

To achieve the required connection, complete the following steps. All of the steps 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`.

In the following steps the value of 104857600 for maxmsgl is an example. Check the appropriate WebSphere MQ documentation to confirm the value for maxmsgl that you can use on your platforms.

You must set the maxmsgl attribute only on the transmission queue that sends messages from the queue manager associated with the Configuration Manager to the queue manager associated with the broker.

1. Define a transmission queue on each component's queue manager. These transmission queues collect messages ready for transmission between components. The transmission queue must have the same name as the queue manager to which it transmits messages (that is COMP1 and COMP2 for this example). Set the maxmsgl attribute to its maximum value.

   For example, on queue manager COMP1:

   ```
   define qlocal('COMP2') usage(XMITQ) maxmsgl (104857600) replace
   ```

   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 Message Broker components.

   a. Define the sender channel on the first component's queue manager (Sender(3)). This sender channel transports 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')
      maxmsgl (104857600) replace
      ```

      where the command parameters have the following meanings:

      - channel and chltype define the name and type of the channel
      - trtype defines the transmission protocol
      - conname defines the host name of the target computer and the port number that the computer is listening on
      - xmitq names the transmission queue for the channel
      - maxmsgl defines the maximum supported message length
      - replace specifies that any existing definition of the named channel is replaced

      For more information about WebSphere MQ commands and parameters, see the *Script (MQSC) Command Reference* section of the WebSphere MQ Version 7 information center online or WebSphere MQ Version 6 information center online.

   b. Define a receiver channel on the first component's queue manager (Receiver(2)). Messages sent by the second component to the first component are 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)
      maxmsgl (104857600) replace
      ```

c. Define the sender channel on the second component's queue manager (`Sender(1)`). This sender channel transports 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')
   maxmsgl (104857600) replace
   ```

d. Define a receiver channel on the second component's queue manager (`Receiver(4)`). Messages sent by the first component to the second component are received by this receiver 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)
   maxmsgl (104857600) replace
   ```

3. Create and start a listener for each protocol in use. Create the listener in WebSphere MQ Explorer, or use the DEFINE LISTENER MQSC command. For more information see "Starting the WebSphere MQ channels and listeners" on page 214.
4. Start the sender channels (1) and (3) on the respective queue managers. You can set up channel initiators for these channels. Setting up receiver channels reduces processor load 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 queue manager that hosts the Configuration Manager to support all sender channels created for the brokers. Setting up a single receiver channel requires a single definition on the Configuration Manager and a single sender definition on each broker, the sender definitions on each broker 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, if you have created a User Name Server as a component of your broker domain.

All WebSphere MQ connections between WebSphere Message Broker components, and between clients and WebSphere Message 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).

Tuning the broker

Before you start:

Ensure that the following requirements are met:

- Your user ID has the correct authorizations to perform the task. Refer to "Security requirements for administrative tasks" on page 723.

This topic lists the various tasks that enable you to tune different aspects of the broker's performance; select the tasks that are relevant to your enterprise:

- "Setting the JVM heap size" on page 224
- "Increasing the stack size on Windows, Linux, and UNIX systems" on page 224

- "Increasing the stack size on z/OS"
- "Publish/subscribe performance tuning" on page 225
- "Setting configuration timeouts" on page 227

Setting the JVM heap size

When you start an execution group, it creates a Java virtual machine (JVM) for executing a Java user-defined node.

You can pass parameters to the JVM to set the minimum and maximum heap sizes; the default maximum heap size is 256 MB. To give more capacity to a message flow that is going to process large messages, reduce the minimum JVM heap size to allow the main memory heap to occupy more address space.

Increase the maximum heap size only if you use Java intensively with, for example, user-defined nodes.

Use caution when you set the maximum heap size, because the Java Runtime Environment takes the values for its initial, maximum, and current heap sizes to calculate how frequently it drives garbage collection. A large maximum heap size drives garbage collection less frequently. If garbage collection is driven less frequently, the heap size associated with the execution group continues to grow.

Use the information on JVM parameter values on the mqsichangeproperties command to set the heap size that you require.

Increasing the stack size on Windows, Linux, and UNIX systems

Increase the stack size on Windows, Linux, and UNIX systems by setting the **MQSI_THREAD_STACK_SIZE** environment variable to an appropriate value.

When you restart brokers that are running on the system, they use the new value.

The value of **MQSI_THREAD_STACK_SIZE** that you set is used for every thread that is created within a DataFlowEngine process. If the execution group has a large number of message flows assigned to it, and you set a large value for **MQSI_THREAD_STACK_SIZE**, the DataFlowEngine process needs a large amount of storage for the stack.

Increasing the stack size on z/OS

Change the stack size on z/OS by altering or adding the **LE_CEE_RUNOPTS** environment variable in the component profile.

Broker components on z/OS are compiled using the XPLINKage (extra performance linkage), which adds optimization to the runtime code. However, if the initial stack size is not large enough, stack extents are used. 128 KB is used in each extent. Ensure that you choose a large enough downward stack size because the performance of XPLINK can be adversely affected when stack extents are used.

To determine suitable stack sizes, you can use theLanguage Environment® Report Storage tool.

To use this tool, use the **RPTSTG** option with the **_CEE_RUNOPTS** environment variable to test a message flow. Set this option in the component profile (BIPBPROF for a broker) during the development and test of message flows that are intended for production; for example:

```
export _CEE_RUNOPTS=XPLINK\(ON\),RPTSTG(ON)
```

You can then override the default values for the stack sizes on z/OS by altering or adding the **LE_CEE_RUNOPTS** environment variable in the component profile.

To update the component profile, take the following steps:

1. Stop the component.
2. Make the necessary changes to the profile.
3. Submit BIPGEN to re-create the ENVFILE.
4. Restart the component.

For example, you can change the default values of 50 K and 512 K in the following line:

```
export _CEE_RUNOPTS=XPLINK(ON),THREADSTACK(ON,50K,15K,ANYWHERE,KEEP,512K,128K)
```

When you use the **RPTSTG** option, it increases the time that an application takes to run, so use it as an aid to the development of message flows only, and not in your final production environment. When you have determined the correct stack sizes needed, remove this option from the **_CEE_RUNOPTS** environment variable.

XPLINK stacks grow downward in virtual storage while the standard linkage grows upward. To avoid a performance impact caused by switching between downward stack space and upward stack space during run time, compile user-defined extensions using the XPLINK option where possible. If your message flow uses user-defined extensions that have been compiled with the standard linkage convention, set a suitable value for the upward stack size.

Publish/subscribe performance tuning

Tune your brokers, and the databases that they use, to handle a large number of subscriptions.

WebSphere Message Broker supports up to 25 000 subscriptions on a broker. The following sections describe some of the actions that you can take to tune your brokers and databases to handle these subscriptions efficiently.

Brokers

Change the broker property jvmMaxHeapSize. The default value for this property is 256 MB.

The value of this property must be large enough for all of the topics in the subscriptions. For example, if you have 10 000 subscriptions, each for a topic that uses 20 KB of storage, set the value of the jvmMaxHeapSize property to at least 200 MB.

Use the "JVM parameter values" on page 485 information within the mqsichangeproperties command to increase the value of the jvmMaxHeapSize property to 512 MB. You must specify the value in bytes, as shown in this example:

```
mqsichangeproperties brokername -o ComIbmJVMManager -n jvmMaxHeapSize -v 536870912
```

where `brokername` is the name of your broker.

The Configuration Manager uses the list of subscriptions which might be stored on your local hard disk:

- Windows On Windows systems, the directory is created at %ALLUSERSPROFILE%\Application Data\IBM\MQSI where %ALLUSERSPROFILE% is the environment variable that defines the system working directory. The default directory depends on the operating system:
  - On Windows XP and Windows Server 2003: C:\Documents and Settings\All Users\Application Data\IBM\MQSI
  - On Windows Vista and Windows Server 2008: C:\ProgramData\IBM\MQSI

  The actual value might be different on your computer.
- Linux UNIX On Linux and UNIX systems, the directory is created at /var/mqsi.

The directory must be at least twice the size of the topic space; that is, for 10 000 subscriptions that each use 20 KB, the size of the directory must be at least 512 MB.

Databases

The broker stores its subscription information in its database. You might need to tune your database to handle the maximum 25 000 subscriptions.

- Windows UNIX Linux Two limits are significant when using DB2. Both limits affect the ability to successfully restart the broker.
  - The first limit occurs when there are approximately 1000 subscriptions. The DB2 parameter **APP_CTL_HEAP_SZ** must be set to a high value to enable the broker to query its database at startup; a value of 8192 is typically large enough for 1000 subscriptions. You can change the value by starting a db2 command prompt, and issuing the command db2 update db cfg using APP_CTL_HEAP_SZ 8192. You might then need to end any connections to the database.
  - The second limit occurs at approximately 8000 subscriptions. When the broker attempts to start, the following error might be reported in the system log:

    ```
    Database error: SQL State '54028';
    Native Error Code '-429';
    Error Text '[IBM][CLI Driver][DB2/LINUX] SQL0429N  The maximum number of concurrent
    LOB locators has been exceeded.  SQLSTATE=54028 '.
    ```

    This error is caused by a limit to the number of LOB handles in DB2. To overcome this problem, you require a patch in DB2; you need to edit file db2cli.ini.

    Linux UNIX On Linux and UNIX systems, this file is located in {DB2InstanceHome}/sqllib/cfg/db2cli.ini..

    Windows On Windows 32-bit editions, this file is located in C:\Program Files\IBM\SQLLIB\db2cli.ini..

    Windows On Windows 64-bit editions, this file is located in C:\Program Files(x86)\IBM\SQLLIB\db2cli.ini.

    Add the following lines to the file:

    ```
    [{Database name}]
    PATCH2=50
     LobCacheSize=1048576
    ```

The PATCH line instructs DB2 to free up LOB locators after it has used them, and the `LobCacheSize` parameter adjusts the total memory that is available to LOB locators; in this case 1 GB. You might then need to restart the DB2 instance.

- z/OS On z/OS, if you are using DB2 Version 8, a database limit occurs at approximately 15 000 subscriptions. To overcome this, modify the value of **NUMLKUS**. A value of 20000 can support 25 000 subscriptions.

Collectives

When a subscription is made to a broker that is a member of a collective, or that is directly linked to another broker, all brokers that are connected to the broker create a proxy subscription. The total number of proxy subscriptions and direct subscriptions must be less than 25 000 for each of your brokers. This limit has implications on how you plan your broker topology.

For example, consider a collective of N brokers.

To maximize connectivity, you connect an instance of a client to each broker with each of those instances subscribing to the same unique topic. Therefore, for N brokers, each unique topic has N clients.

In this situation, each broker has a subscription to each client that is connected to it, and also a proxy subscription to each of the other brokers in the collective.

Therefore, each broker has N subscriptions for every unique topic (one for the client that is directly connected, and N-1 for the proxy subscriptions to all of the other brokers). If there are T unique topics, ensure that N*T <= 25 000. That is, if you have 1000 unique topics, restrict the size of your collective to a maximum of 25 brokers.

Setting configuration timeouts

Change timeouts that affect configuration tasks in the broker.

Before you start:

Read Deployment overview to understand the conditions under which these timeouts apply.

The broker processes configuration requests from the Configuration Manager:

- User requests. When you issue a command against the broker, for example mqsideploy.
- Internal requests.

Several factors affect the time that a broker takes to apply and respond to these requests. These include the load on the broker's computer, network delays between components, and the work that execution groups are performing at the time the request is received. The number of message flows in an execution group, and their complexity, and large message sets, also affect the time taken.

You can change the length of time that a broker can take to perform these actions using two parameters that you can set on the mqsicreatebroker and mqsichangebroker commands. The combined default value for these parameters is approximately six minutes (360 seconds).

During development and test of message flows and broker configurations, experiment with the values that you set for these timeout to determine appropriate values for your resources.

- **-g** *ConfigurationChangeTimeout*

  This value defines the maximum time (in seconds) that is allowed for a user configuration request to be processed, and defaults to five minutes (300 seconds). The value is affected by the system load (including processor use), and by each execution group's load. If the request is not completed in this time, the broker generates warning message BIP2066, but continues to implement the change. The broker records further diagnosis information in the system and event logs.

- **-k** *InternalConfigurationTimeout*

  This value defines the maximum time (in seconds) that is allowed for an internal configuration change to be processed and defaults to one minute (60 seconds). For example, it defines the length of time that is allowed for the broker to start an execution group before a response is required.

  The broker starts an internal process to start an execution group and to make all the message flows active. Part of this initialization is performed serially (one execution group at a time), therefore if the change affects more than one execution group the time required increases. If an execution group exceeds this timeout, the broker generates a warning message BIP2080. However, the initialization continues and the execution group is started. The broker records further diagnosis information in the system and event logs.

The sum of the *ConfigurationChangeTimeout* and the *InternalConfigurationTimeout* represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response. Check that typical configurations complete successfully within the time that you have specified, to minimize warning messages. Look for warning messages in the Broker Administration perspective in the Alerts view. When all messages disappear, the deployment has completed. If you start a deploy and record how long it takes for all messages to disappear from the Alerts view, you can use this time interval as the basis for setting these timeout values.

If the broker is on a production system, increase the values for both *ConfigurationChangeTimeout* and *InternalConfigurationTimeout* to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied. Also consider increasing the value if you have merged message flows into fewer execution groups that you are using for testing.

If the broker is on a development or test system, you might want to reduce timeouts (in particular, the *ConfigurationChange Timeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

Modifying the broker's publish/subscribe engine

Check or change the state of the broker's publish/subscribe engine.

Before you start:

- Start the broker by using the mqsistart command.

You can control the state of the broker's publish/subscribe engine by modifying the predefined configurable service.

- "Checking the status of the broker's publish/subscribe engine"
- "Disabling the broker's publish/subscribe engine"
- "Enabling the broker's publish/subscribe engine" on page 230

Checking the status of the broker's publish/subscribe engine

View the status of the broker's publish/subscribe engine to check if the broker or its queue manager is handling publish/subscribe application messages.

Before you start:

- Start the broker by using the mqsistart command.

To check the status of the configurable service for the broker's publish/subscribe engine:

1. Run the mqsireportproperties command to view the status of the PublishSubscribe configurable service:

   ```
   mqsireportproperties WBRK61_DEFAULT_BROKER -c PublishSubscribe -o AllReportableEntityNames -r
   ```

   The response to this command includes the status of the **psmode** parameter:

   - If **psmode** is set to enabled (the default value), the broker's publish/subscribe engine is active and is handling all publish/subscribe application messages; publications, subscriptions, registrations, and retained subscriptions.
   - If **psmode** is set to disabled, the broker's publish/subscribe engine is inactive; all publish/subscribe application messages are being handled by the broker's queue manager.

     You can disable the broker's publish/subscribe engine only if you have installed WebSphere MQ Version 7.0 on this computer.

   The setting for the broker's publish/subscribe engine always takes precedence, and the broker ensures that the state of the queue manager's publish/subscribe engine is compatible:

   - If you start a broker for which **psmode** is set to enabled, it sets the status of the queue manager's publish/subscribe engine to `DISABLED`.
   - If you start a broker for which **psmode** is set to disabled, it sets the status of the queue manager's publish/subscribe engine to `ENABLED`.
2. If the reported status is not correct for your broker domain, and you want to change the status of the broker's publish/subscribe engine, see "Disabling the broker's publish/subscribe engine" or "Enabling the broker's publish/subscribe engine" on page 230.

   If you change the status, you must restart the broker for that change to take effect.

Disabling the broker's publish/subscribe engine

Configure your broker to give control of publish/subscribe applications to the broker's queue manager.

Before you start:

- Check that you have installed WebSphere MQ Version 7.0 on this computer. You can disable the broker's publish/subscribe engine only if you have created or migrated the queue manager to Version 7.0.

- Check that the Command Server on this queue manager is started (this component must be active if the queue manager's publish/subscribe engine is to be used)
- Start the broker by using the mqsistart command.

To disable the broker's publish/subscribe engine:

1. Run the mqsichangeproperties command to change the status of the PublishSubscribe configurable service:

   ```
   mqsichangeproperties WBRK61_DEFAULT_BROKER -c PublishSubscribe -o Interface -n psmode
   -v disabled
   ```

2. Stop the broker by using the mqsistop command.
3. Start the broker by using the mqsistart command. The broker checks the status of its publish/subscribe engine. Now that it is disabled, it updates the status of the queue manager and hands over control of all publish/subscribe clients and applications.
4. If appropriate, you must ensure that all your publish/subscribe clients reregister their subscriptions with the WebSphere MQ Version 7.0 queue manager publish/subscribe engine.

When this task has completed successfully, all publish/subscribe application messages and operations are managed by the queue manager.

Enabling the broker's publish/subscribe engine

Configure your broker to gain control of publish/subscribe applications from the WebSphere MQ Version 7.0 queue manager.

Before you start:

- Check that you have installed WebSphere MQ Version 7.0 on this computer. If not, the broker's publish/subscribe engine is permanently enabled and you cannot change its state.
- Start the broker by using the mqsistart command.

To enable the broker's publish/subscribe engine:

1. Run the mqsichangeproperties command to change the status of the PublishSubscribe configurable service:

   ```
   mqsichangeproperties WBRK61_DEFAULT_BROKER -c PublishSubscribe -o Interface -n psmode
   -v enabled
   ```

2. Stop the broker by using the mqsistop command.
3. Start the broker by using the mqsistart command. The broker checks the status of its publish/subscribe engine. Now that it is enabled, it updates the status of the queue manager and takes control of all publish/subscribe application messages and operations.

When this task has completed successfully, all publish/subscribe application messages and operations are managed by the broker.

Preparing the environment for WebSphere Adapters nodes

Before you can use the WebSphere Adapters nodes, you must set up the broker runtime environment so that you can access the Enterprise Information System (EIS).

Before you start:

Read WebSphere Adapters nodes.

To enable the WebSphere Adapters nodes in the broker runtime environment, configure the broker with the location of the EIS provider JAR files and native libraries. (On Windows, the location of the JAR files cannot be a mapped network drive on a remote Windows computer; the directory must be local or on a Storage Area Network (SAN) disk.)

1. The WebSphere Adapters nodes require libraries from the EIS vendors. For more information on how to obtain and use these libraries, see Developing message flow applications using WebSphere Adapters.
2. Use the following commands to make the JAR files and shared libraries available to the WebSphere Adapters nodes.
   - To set up the dependencies, use the following command.

     ```
     mqsichangeproperties broker name -c EISProviders -o EIS type -n jarsURL -v jar directory
     mqsichangeproperties broker name -c EISProviders -o EIS type -n nativeLibs -v bin directory
     ```

     For example:

     ```
     mqsichangeproperties brk6 -c EISProviders -o SAP -n jarsURL -v c:\sapjco\jars
     mqsichangeproperties brk6 -c EISProviders -o SAP -n nativeLibs -v c:\sapjco\bin
     ```

     After you have run the mqsichangeproperties command, restart the broker so that the changes take effect.
   - To report the dependencies, use the following command.

     ```
     mqsireportproperties broker name -c EISProviders -o EIS type -r
     ```

     For example:

     ```
     mqsireportproperties brk6 -c EISProviders -o SAP -r
     ```
   - On z/OS, run this command by customizing and submitting BIPJADPR.

When you have set up the environment for the WebSphere Adapters nodes, you can perform the preparatory tasks that are listed in Developing message flow applications using WebSphere Adapters.

Preparing the environment for IMS nodes

Before you can use the IMS nodes, you must set up the broker runtime environment so that you can access the IMS system.

Before you start:

Read IBM Information Management System (IMS™).

Complete the following steps to ensure that WebSphere Message Broker can connect to the IMS system.

1. Ensure that IMS Connect is installed and started on the IMS system.
2. Define a configurable service for each IMS system to which you want to connect. For example, to create an IMSConnect configurable service for the IMS instance IMSA that is running on `test.ims.ibm.com`, port 9999, run the mqsicreateconfigurableservice command as shown:

   ```
   mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c IMSConnect -o myIMSConnectService
   -n Hostname,PortNumber,DataStoreName -v test.ims.ibm.com,9999,IMSA
   ```

For details about how to create, change, and report configurable services, see “Changing connection details for IMS nodes” on page 465. You can also configure connection properties on the IMS node.

3. Use the mqsisetdbparms command to set security details in the broker store. For example, to associate a user ID and password pair with an IMS Connect connection, run the mqsisetdbparms command as shown:

```
mqsisetdbparms WBRK61_DEFAULT_BROKER -n ims::mySecurityIdentity -u myuserid -p mypassword
```

You can also configure the security details on the node.

Modifying a broker

Modify a broker by using the command line on the system where the broker component is installed.

Before you start:

You must have completed the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task; see “Security requirements for administrative tasks” on page 723.
- Create a broker.
- On Windows, Linux, and UNIX systems, you must set up your command-line environment before performing this task by running the product profile or console; see Setting up a command environment.

Windows On Windows, you can also use the Command Assistant to complete this task.

The parameters you can change on the broker affect the physical broker that was created by using the command line.

You can also modify the broker reference in the workbench, where you can change broker properties, such as configuration timeouts.

Choose the appropriate task for your platform from the following links:

- “Modifying a broker on Windows, Linux, and UNIX systems”
- “Modifying a broker on z/OS” on page 234

Modifying a broker on Windows, Linux, and UNIX systems

Use the mqsichangebroker command on Windows, Linux, and UNIX to modify your broker.

Before you start:

You must have completed the following task for the appropriate platform:

- “Creating a broker on Linux and UNIX systems” on page 180
- “Creating a broker on Windows” on page 182

Windows On Windows, you can also use the Command Assistant to complete this task.

To modify a broker on Windows, Linux, and UNIX:

1. Stop the broker by using the mqsistop command.

2. Enter the mqsichangebroker command, specifying the broker name and one or more 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 ConfigurationDelayTimeout> <-v StatisticsMajorInterval>
<-P httpListenerPort> <-y ldapPrincipal> <-z ldapCredentials>
<-c ICUconverterpath> <-x userExitPath> <-e activeUserExits>
<-o operationMode> <-r UserLilPath64> <-b UserExitPath64>
<-f functionlevel>
```

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 service 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 not enabled for this broker.

-t Indicates that the broker runs as a WebSphere MQ trusted application.

-n Indicates that the broker must 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.

The broker starts this listener when a message flow that includes HTTP nodes or Web Services support is started

-y Is the user principal for access to an LDAP directory.

-z Is the user password for access to LDAP.

-c Is a delimited set of directories to search for additional code page converters.

-x Is the path that contains the location of all user exits to be loaded for 32-bit execution groups in this broker.

-e Is the list of active user exits.

-o Is the operation mode that the broker will use.

-r Indicates from where 64-bit LIL (loadable implementation libraries) files are loaded.

-b Is the path that contains the location of all user exits to be loaded for 64-bit execution groups in this broker.

-f Indicates the maximum function level of your broker that you want to enable.

For example, to change the user ID that is used to run the broker, enter the following command:

```
mqsichangebroker WBRK_BROKER -i wbrkuid -a wbrkpw
```

To change the configuration timeout, enter the following command:

```
mqsichangebroker WBRK_BROKER -g 500
```

For further information about these parameters, and more examples, see "mqsichangebroker command - Windows, Linux, and UNIX systems" on page 405.

3. Restart the broker by using the mqsistart command. The broker restarts with the new properties.

You cannot change all of the parameters with which you created a broker. If you cannot change a property by using mqsichangebroker, delete the broker and then create a new broker with the new properties.

Modifying a broker on z/OS

Use the mqsichangebroker command on z/OS to modify your broker.

Before you start:

You must have completed the following task:

- "Creating a broker on z/OS" on page 183

To modify a broker:

1. Ensure that the broker is running.
2. Stop the broker components by issuing the following command:
   ```
   /F BROKERNAME, PC
   ```
3. When the broker 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 of the parameters with which you created a broker. If you cannot modify a parameter that you want to change by using the changebroker command, then delete the broker and create a new one. By creating a new broker you can redefine all of the parameters.

You can change the following parameters:

g ConfigurationChangeTimeout

k InternalConfigurationTimeout

s UserNameServerQueueManagerName

l UserLilPath

v StatisticsMajorInterval

P HTTPListenerPort

j publish/subscribe access control is to be enabled.

d publish/subscribe access control is not enabled.

y LdapPrincipal

z LdapCredentials

c ICUConverterPath

x UserExitPath

e ActiveUserExits

f functionlevel

For further information about these parameters, see "mqsichangebroker command - z/OS" on page 411.

Viewing broker properties

You can view broker properties by using the mqsireportbroker command.

Before you start:

You must have completed the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task; see "Security requirements for administrative tasks" on page 723.
- Create a broker.
- On Windows, Linux, and UNIX systems, you must set up your command-line environment before performing this task by running the product profile or console; see Setting up a command environment.

Use the mqsireportbroker command or domain to view all the properties associated with a broker. The command shows both parameters entered on the command line and viewable in the workbench.

1. Run the mqsireportbroker command. For example, to view the properties of the broker SOAPBR, run the following command:

   ```
   mqsireportbroker SOAPBR
   ```

2. View responses of the mqsireportbroker command to check on current settings. Examples are given in "mqsireportbroker command" on page 616.
3. If you want to make any changes, run the mqsichangebroker command, specifying the required parameters.

Changing the operation mode of your broker

Change the operation mode in which your broker is working by using the mqsimode command.

You must change your broker configuration to ensure that your brokers are running in the operation mode for which you have purchased a license. You can change the mode to starter, adapter, or enterprise.

When you view the broker domain in the workbench, the current mode of each broker is displayed. If the resources deployed to a broker exceed the permitted amounts, the display indicates these violations. You cannot change the broker mode in the workbench; you must use the mqsimode command to make changes.

1. Open a command prompt.

- Linux On Linux, run the mqsiprofile command to initialize the command environment.
- Windows On Windows, click **Start** → **Programs** → **IBM WebSphere Message Brokers 6.1** → **Command Console** to open a command console.

2. Run the mqsimode command with the **-o** parameter to change the value of the run time, and the **-b** parameter to restrict the command to a single broker; see "mqsimode command" on page 602.
3. Check for error messages. If you attempt to re-configure the broker to a mode which is not sufficient for the deployed resources, the mqsimode command issues a warning indicating that changing the mode is not allowed. Resolve any violations, if required; see Resolving problems that occur during deployment of message flows.
4. (Optional) Run the mqsimode command again to confirm that there are no violations.

See further examples of changing the mode of your broker:

- "Example: Changing the operation mode of your broker" on page 606
- "Example: Changing the Trial Edition to the full edition" on page 606

Moving from Trial Edition

You want to convert from Trial Edition mode to an alternative edition.

Contact your IBM representative to upgrade your license.

To ensure that all brokers are no longer created in trial mode, you must install the full package. If you leave your installation in the Trial Edition all new brokers that you create will be in trial mode by default.

1. Stop all components by using the mqsistop command and close any workbench sessions on this computer; see "mqsistop command" on page 660.
2. Uninstall the trial package, you do not need to uninstall the Message Broker Toolkit; see Uninstalling. The resources that you have defined are retained when you uninstall and reinstall the product. Your Message Broker Toolkit data is not affected by changing the run time.
3. Install only the broker component of the full package; see Installation Guide.
4. Open a command prompt.
   - Linux On Linux, run the mqsiprofile command to initialize the command environment.
   - Windows On Windows, click **Start** → **Programs** → **IBM WebSphere Message Brokers 6.1** → **Command Console** to open a command console.
5. Ensure your brokers and configuration manager are started; see "Starting and stopping a broker" on page 323 and "Starting and stopping a Configuration Manager" on page 325.
6. Use the mqsimode command without the **-b** parameter to change the mode of all of your brokers. Specify the mode that you require for your brokers by setting the **-o** parameter (*Mode_Type*) to enterprise, starter, or adapter.

   ```
   mqsimode –i localhost –p 2414 –q WBRK61_DEFAULT_CONFIGURATION_MANAGER –o Mode_Type
   ```

   For example, to change all of your brokers in enterprise mode run the following command:

   ```
   mqsimode –i localhost –p 2414 –q WBRK61_DEFAULT_CONFIGURATION_MANAGER –o enterprise
   ```

All brokers that you create in the future are created in enterprise mode unless you specify otherwise.

Ensure that you upgrade other required products, for example, WebSphere MQ and DB2 if you have trial versions of those products.

You must change your broker configuration to ensure that your brokers are running in the operation mode for which you have purchased a license; see "Changing the operation mode of your broker" on page 235.

Checking the operation mode of your broker

Use the mqsimode command to find out the operation mode of your broker.

Run the mqsimode command specifying the **-b** parameter to report the mode being used by your broker. Running this command also reports any mode violations.

For example:

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER –b Broker_Name
```

If your broker is working in the starter mode and the name of your broker is *Broker_Name* this command displays the following messages:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1807: Discovering mode information from broker 'Broker_Name'...
BIP1802: Broker 'Broker_Name' is in starter mode.
BIP8071: Successful command completion.
```

Modifying a Configuration Manager

Modify a Configuration Manager by using the command line on the system where the Configuration Manager component is installed.

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. See "Security requirements for administrative tasks" on page 723
- "Creating a Configuration Manager" on page 192
- On Windows, Linux, and UNIX systems, you must set up your command-line environment before performing this task, by running the product profile or console; see Setting up a command environment

Modify a Configuration Manager using the command line on the system where the Configuration Manager component is installed. On Windows, you can also use the Command Assistant to complete this task.

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 Broker Toolkit or a Configuration Manager Proxy application, which might or might not be on the same workstation 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 239

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 240.

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 197

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 changeconfigmgr parameters that you want to change. Note that you can abbreviate changeconfigmgr to cc. For example:

   ```
   MODIFY <configurationmanagername>,changeconfigmgr 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 changeconfigmger 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:

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 415 for further information on these parameters.

Viewing Configuration Manager properties

You can view Configuration Manager properties by using the mqsireportconfigmgr command.

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. See "Security requirements for administrative tasks" on page 723
- "Creating a Configuration Manager" on page 192
- On Windows, Linux, and UNIX systems, you must set up your command-line environment before performing this task, by running the product profile or console; see Setting up a command environment

Use the mqsireportconfigmgr command or domain to view all the properties associated with a Configuration Manager. The command shows both parameters entered on the command line and viewable in the workbench.

1. Run the mqsireportconfigmgr command. For example, to view the properties of the Configuration Manager SOAPCM, run the following command:

   ```
   mqsireportconfigmgr SOAPCM
   ```

2. View responses of the mqsireportconfigmgr command to check on current settings. Examples are given in "mqsireportconfigmgr command" on page 621.
3. If you want to make any changes, run the mqsichangeconfigmgr command, specifying the required 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 Broker 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 on page 240, 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 on page 240, 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 on page 240 and the complete deployment in Step 7 on page 240, any messages that report a change in status will have been sent to the old Configuration Manager.

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 723
- "Creating a User Name Server" on page 202
- On Windows, UNIX systems, and Linux, you must set up your command-line environment before performing this task, by running the product profile or console; refer to Setting up a command environment

Modify a User Name Server using the command line on the system where the User Name Server component is installed. On Windows, you can also use the Command Assistant to complete this task.

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 242
- "Modifying a User Name Server on z/OS" on page 243

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 on System z, Linux on x86 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

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 206

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.

Moving from WebSphere Message Broker on a distributed platform to z/OS

Information on how to define resources for WebSphere Message Broker for z/OS.

Recommendations are given in the following topics:

- "z/OS customization overview" on page 153
- "Customizing the z/OS environment" on page 152
- "Creating a broker on z/OS" on page 183
- "Creating a User Name Server on z/OS" on page 206
- "Creating a Configuration Manager on z/OS" on page 197
- "Administration in z/OS" on page 667

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 Message Broker for z/OS broker. If you have extended WebSphere Message Broker in a distributed environment with user-defined parsers or message processing nodes, port them 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).

For information about message flow transactionality, see Message flow transactions.

Moving user applications

You can write your own applications to work with WebSphere Message Broker. If these applications use the common subset of functionality of all WebSphere Message Broker brokers, no migration is necessary. If you are using functionality that is available on some WebSphere Message Broker platforms only, for example message segmentation and WebSphere MQ message groups, be aware that WebSphere Message Broker for z/OS does not provide support for this migration.

Deleting an execution group from a broker using the command line

Use the command line as an alternative method to delete an execution group from the Message Broker Toolkit.

Before you start:

Complete the following task:

- "Adding an execution group to a broker using the command line" on page 191

Instead of employing the Message Broker 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 Broker Toolkit see "Deleting an execution group using the Message Broker Toolkit" on page 262

1. Open a command prompt that has the environment configured for this version of WebSphere Message Broker.
2. Enter the following command to delete the execution group:

   ```
   mqsideleteexecutiongroup -i host -p PORT  -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.

   PORT 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 a broker

Delete a broker using the command line on the system where the broker component is installed.

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 723
- Remove the broker from the broker domain in the workbench ("Removing a broker from a broker domain" on page 258).

On Windows, you can also use the Command Assistant to complete this task.

On Windows, Linux, and UNIX systems, you must set up your command-line environment before deleting a broker, by running the product profile or console; see Setting up a command environment.

You can remove the broker from the broker topology using the workbench, but the physical broker is not 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"
- "Deleting a broker on z/OS" on page 246

Deleting a broker on Linux and UNIX systems

Delete the physical broker component.

To delete a broker on Linux and UNIX systems:

1. Remove the broker from the broker domain, in the workbench. Refer to "Removing a broker from a broker domain" on page 258.
2. Stop the broker using the mqsistop command.
3. 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

Delete the physical broker component.

Windows 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 258.
2. Stop the broker using the mqsistop command.
3. 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

Delete the physical broker component.

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 258.
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, storage group and table spaces.

Note:

a. Not all files are deleted from the component directory in the file system.
b. When the BIPDLDB job drops the broker DB2 database, it also deletes any Image Copy references to itself that you currently have. If you restore the broker in future, you must also reinstate the Image Copy references.

Deleting 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 723

Delete a Configuration Manager using the command line on the system where the Configuration Manager component is installed. On Windows, you can also use the Command Assistant to complete this task.

On Windows, UNIX systems, and Linux, you must set up your command-line environment before deleting a Configuration Manager, by running the product profile or console; refer to Setting up a command environment.

Follow the link for the appropriate platform:

- "Deleting a Configuration Manager on Linux and UNIX systems" on page 247
- "Deleting a Configuration Manager on Windows" on page 247

• "Deleting a Configuration Manager on z/OS" on page 248

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:

1. Stop the Configuration Manager using the "mqsistop command" on page 660command.
2. Delete the Configuration Manager using the "mqsideleteconfigmgr command" on page 557 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 581 command.
- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the "mqsideleteconfigmgr command" on page 557 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.

Windows To delete a Configuration Manager:

1. Stop the Configuration Manager using the "mqsistop command" on page 660 command.
2. Delete the Configuration Manager using the "mqsideleteconfigmgr command" on page 557 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 581 command.

- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the “mqsideleteconfigmgr command” on page 557 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 723
- 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 248 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. On Windows, you can also use the Command Assistant to complete this task.

On Windows, UNIX systems, and Linux, you must set up your command-line environment before deleting a User Name Server, by running the product profile or console; refer to Setting up a command environment.

Follow the link for the appropriate platform.

- "Deleting a User Name Server on Linux and UNIX systems"
- "Deleting a User Name Server on Windows"
- "Deleting a User Name Server on z/OS"

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 on System z, Linux on x86 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

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

Create the resources for your broker domain in the workbench on Linux on x86 or Windows.

Before you start:

Created the physical broker domain components. Refer to "Configuring broker domain components" on page 177.

This task is the second part of the two-stage process to create and configure your broker domain. Use the workbench to configure and administer the broker domain components.

Launch the workbench in one of the following ways:

- Linux From the main menu:
  - On Red Hat, click **Programming → IBM WebSphere Message Broker Toolkit**.
  - On SUSE Linux, click **All Applications → WebSphere Message Broker Toolkit**.
- Windows Click **Start → IBM Software Development Platform → IBM WebSphere Message Brokers Toolkit → WebSphere Message Broker Toolkit**, or double-click the shortcut on your desktop labelled 'WebSphere Message Broker Toolkit'.
- Use the following commands in a command prompt from their location in the root directory for the package group:
  - Linux
    ```
    ./eclipse -product com.ibm.etools.msgbroker.tooling.ide
    ```
  - Windows
    ```
    eclipse.exe -product com.ibm.etools.msgbroker.tooling.ide
    ```

See the following tasks for instructions on how to configure a broker domain in the workbench:

- "Creating a domain connection" on page 251
- "Modifying domain connection properties" on page 253
- "Deleting a domain connection" on page 254
- "Adding a broker to a broker domain" on page 255
- "Modifying broker properties" on page 257
- "Removing a broker from a broker domain" on page 258
- "Removing deployed children from a broker" on page 259
- "Adding an execution group to a broker in the workbench" on page 259
- "Deleting an execution group using the Message Broker Toolkit" on page 262

- "Removing deployed children from an execution group" on page 262

When configuring your broker domain, you are prompted to deploy all changes to the Configuration Manager. You can set your user preferences to suppress the prompt to deploy after each change. See "Changing Broker Administration preferences" on page 308.

When you make changes to the broker domain, and deploy to the Configuration Manager, a short delay might occur before the workspace is updated and the Configuration Manager tells you that the deploy has worked. The delay depends on the network configuration, and the number of changes to make to the configuration of the broker domain during the deployment.

Creating a domain connection

Create a connection to a Configuration Manager so that you can manage the domain resources.

Before you start:

Complete the following tasks:

- "Creating a Configuration Manager" on page 192.
- Create and start a listener for the Configuration Manager. For details on how to create and start a listener, follow the instructions for listeners in the topic: "Starting the WebSphere MQ channels and listeners" on page 214.

This topic shows you how to:

- Create a domain connection in the workbench using the Create a Domain Connection wizard.
- Enter a set of parameters to create a `.configmgr` file.
- Use the parameters contained within the `.configmgr` file 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. Click **New** → **Domain Connection** to open the Domain Connection wizard.
3. In the Create a Domain Connection wizard, enter:
   a. The value for the WebSphere MQ **Queue Manager Name** 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 server-connection channel in the **SVRCONN Channel Name** field. The channel has a default name of `SYSTEM.BKR.CONFIG`.

      You can create more than one server-connection channel and define a different SSL certificate on each channel to enforce; for example, users with `view` access on to one channel and users with `deploy` access on to a different channel.

      You can then create WebSphere MQ exits on each channel to provide additional authentication of the WebSphere MQ message sent to the Configuration Manager.

You must create the server-connection channel manually on the Configuration Manager's queue manager by using one of the following options:

- The WebSphere MQ runmqsc command to create a channel with options `CHLTYPE(SVRCONN)` and `TRPTYPE(TCP)`.
- The WebSphere MQ Explorer to create a server-connection channel with the transmission protocol set to TCP.

For more information see your WebSphere MQ documentation.

The default name of `SYSTEM.BKR.CONFIG` is assumed if you do not change the name, or attempt to delete it. The name of the server-connection channel is changed only if you enter another name in place of `SYSTEM.BKR.CONFIG`.

e. Optional: The **Class** of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java class name, but you can leave this field blank if it does not apply to your domain connection.

f. Optional: The **JAR File Location** for the Security Exit required to connect to the WebSphere MQ queue manager. Click **Browse** to find the file location. You can leave this field blank if it does not apply to your domain connection. You must provide a **JAR File Location** if you enter a Security Exit **Class**.

g. Optional: The **Cipher Suite**, **Distinguished Names**, **CRL Name List**, **Key Store**, and **Trust Store** parameters are required to enable SSL. For more information, see "Implementing SSL authentication" on page 57. The **Cipher Suite** field displays available cipher suites. Click **More** to configure Custom SSL Cipher Suites in the Broker Administration Preferences window. If a **Cipher Suite** is not specified, all of the 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 Name**, **Host** name, or TCP **Port** number. An error message is displayed in the Create a Domain Connection wizard if you try to create a second broker domain using the same **Queue Manager Name**, **Host** name, and **Port** number.

4. Click **Next** to begin the domain connection to the Configuration Manager.
5. If you click **Cancel**, the Create a Domain Connection wizard closes, forcing disconnection from the domain.
6. After the domain connection has been made, enter:
   a. The name of your **Project**. The project is the container for your domain connection. If you have not already created a project, you can specify the name of a new project here. The project is created with the domain connection.
   b. The **Connection name**. The Connection name is the name that 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, in Domain Connections. The project holds the `.configmgr` domain connection file.

   The view of the broker domain is displayed in the Domains view.

Modifying domain connection properties

Change the properties of the domain connection to redefine the connection to the Configuration Manager.

Before you start:

To complete this task, you must have completed the following tasks:

- "Creating a domain connection" on page 251
- Disconnect from the broker domain. Refer to "Connecting to and disconnecting from the broker domain" on page 319.

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 perspective Navigator view, expand Domain Connections, and open your 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 server-connection channel. The channel has a default name of `SYSTEM.BKR.CONFIG`.

      You can create more than one server-connection channel and define a different SSL certificate on each channel to enforce, for example, users with `view` access on to one channel and users with `deploy` access on to a different channel.

      You can then create WebSphere MQ exits on each channel to provide additional authentication of the WebSphere MQ message sent to the Configuration Manager.

      You must create the server-connection channel manually on the queue manager of the Configuration Manager by using one of the following methods:

      - The runmqsc command, to create a channel with options `CHLTYPE(SVRCONN)` and `TRPTYPE(TCP)`.
      - WebSphere MQ Explorer, to create a server-connection channel with the transmission protocol set to TCP.

      For more information see your WebSphere MQ documentation.

      The default name of `SYSTEM.BKR.CONFIG` is assumed if you do not change the name, or attempt to delete it. The name of the server-connection channel is changed only if you enter another name in place of `SYSTEM.BKR.CONFIG`.
   e. 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.

f. 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. Leave this field blank if it does not apply to your domain connection.

   You must enter the location of the JAR file if you enter a Security Exit class.

g. 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 57 for more information). The cipher suite field displays available cipher suites. Click **More** to configure a custom cipher suite in the Broker Administration perspective preferences window. If you do not specify a cipher suite, 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** to open the Domain Connection editor. Follow steps 3 to 5 above to make the changes you require.

Deleting a domain connection

Delete a domain connection to remove the connection to the Configuration Manager, and the ability to manage a domain through that Configuration Manager.

Before you start:

You must complete the following task:

- "Creating a domain connection" on page 251

To delete a domain connection, delete the corresponding `.configmgr` file from your 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 perspective Navigator view:
   a. Expand Domain Connections
   b. Open the appropriate project.
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 project.
- The view of the broker domain and its hierarchy is removed from the Domains view.

Adding a broker to a broker domain

How to add a broker to a broker domain.

Before you start:

You must complete the following tasks:

- "Creating a broker" on page 178
- "Connecting to and disconnecting from the broker domain" on page 319

Adding a broker to the broker topology creates a broker reference in the configuration repository; it does not create the physical broker. When you add a broker, you must 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. In the Domains view, right-click the default configuration manager, and click **New → Broker Reference**.
3. In the Create a Broker Reference 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**. If you click **Cancel** at this prompt, the wizard remains open.
   b. Type the name of the broker.
   c. Type the WebSphere MQ **Queue Manager Name** that the Configuration Manager is using.

      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 221.
      2) You can associate a WebSphere MQ **Queue Manager Name** 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. For more information see Publish/subscribe topology deployment.

   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 broker archive (BAR) file is deployed. See Deploying a broker archive file.

You can set user preferences so that you are not prompted to deploy the publish/subscribe topology. See “Changing Broker Administration preferences” on page 308. 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 want 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. 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:

Add any further execution groups to the broker that you require. 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

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.

Before you start:

You must complete the following task:

- “Adding a broker to a broker domain” on page 255

The new broker inherits the same short and long description as the original broker, and the same execution groups are added; the message flows under the execution groups of the original broker are not inherited by the new broker’s execution groups. 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."
- To rename the broker's queue manager, refer to "Modifying broker properties."

Modifying broker properties

You can modify broker properties by adding a long or short description, and by customizing the broker and multicast properties.

Before you start:

To perform this task, you must have completed the following tasks:

- "Adding a broker to a broker domain" on page 255
- Start the broker. See "Starting and stopping a broker" on page 323.

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. For details of all of the broker properties, see "Broker properties" on page 367.
   b. The multicast properties.

      Expand **Multicast** in the left panel of the properties window.

      Click **Advanced** to access the advanced multicast properties.

      For details of all of the multicast properties, see "Setting up a multicast broker" on page 270.
   c. The description for the broker.

      Click **Description** in the left panel of the properties window. Enter either a short or a long description, or both, for the broker.
3. Click **OK** to save all of the modifications to the broker.

   An automatic broker configuration deployment is performed immediately to implement the changed broker properties, except for the broker queue manager name, and the short and long descriptions, for which deployment is not required to update the broker properties.

Renaming a broker

You might need to rename a broker (and possibly its queue manager) if your original attempt at creating a broker reference contained an error.

Before you start:

You must complete the following tasks:

- "Adding a broker to a broker domain" on page 255

For further information about rename the broker queue manager, see “Modifying broker properties” on page 257.

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

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, but it is marked as logically deleted from the configuration repository.

Before you start:

You must complete the following task:

- “Adding a broker to a broker domain” on page 255

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 244 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.

   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 244.

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 deployed children from a broker removes all message flows, message sets, and all other deployed objects from all execution groups on the selected broker. All execution groups are deleted. A new default execution is then created for the broker.

Although these objects are all removed from the broker, they are still available in the file system and workbench, and can be reused if required.

To remove the deployed children from a broker:

1. Switch to the Broker Administration perspective.
2. Expand the broker domain, in the Domains view, to reveal the broker you want to work with.
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 removes all message flows and message sets from all execution groups, and deletes all execution groups. An automatic broker configuration deployment 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 in the workbench

Use the workbench to add execution groups to a broker.

Before you start:

- Add the broker to the broker domain
- Connect to the broker domain

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly.

The mode that your broker is working in can affect the number of execution groups that you can use; see "Restrictions that apply in each operation mode" on page 368.

You can use one of three methods to complete this task:

- The workbench
- The mqsicreateexecutiongroup command
- The CMP API

This task describes the first method. If you prefer, you can create an execution group using the command line; see "Adding an execution group to a broker using the command line" on page 191. For information about the CMP API, see Developing applications that use the Configuration Manager Proxy API.

To add an execution group to a broker:

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the default configuration manager.
3. Right-click **Broker Topology**, and click **New → Execution Group**.
4. 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 box 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 **Execution Group name**.
   c. Select the **Processor Architecture** for this execution group to specify if the execution group process runs as a 32-bit or 64-bit application. Choose one of the following three options:
      - Default. The execution group is created to run in 32-bit mode for systems that support only 32-bit mode, and 64-bit mode for all other systems (64-bit mode only or both modes).
      - 32 bit. The execution group is created to run in 32-bit mode.
      - 64 bit. The execution group is created to run in 64-bit mode.

      Do not set a processor architecture that is not supported on the target broker computer. For details of the support on each operating system, see Support for 32-bit and 64-bit platforms.

      If you create a 64-bit execution group, the broker's queue manager must also operate in 64-bit mode. All WebSphere MQ Version 6 and Version 7 queue managers on 64-bit platforms run in 64-bit mode.

      To change the default value for this option, which is initially set to Default, click **Window → Preferences → Broker Administration** and update the setting for *Execution Group Platform Processor Architecture*.
5. Click **Next**.
6. Optional: Enter a short or long description for the execution group.
7. Click **Finish** to add the execution group to the broker.

In the Domains view, the execution group is added to the appropriate broker.

Copying an execution group

You can copy an existing execution group from a broker to another broker within the same broker domain.

Before you start:

You must complete the following tasks:

- "Adding an execution group to a broker in the workbench" on page 259
- "Adding a broker to a broker domain" on page 255

The new execution group inherits the same short and long description as the original execution group, and is automatically given a new name. The new execution group does not inherit any other properties of the original execution group. The message flows in the original execution group are not copied to the new execution group.

The mode that your broker is working in can affect the number of execution groups that you can use; see "Restrictions that apply in each operation mode" on page 368.

To copy an execution group:

1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the name of the execution group that you want to copy, then click **Copy**.
3. Right-click the name of the broker to which you want to copy the execution group, then click **Paste**. The broker must be in the same broker domain as the original execution group.

A copy of the execution group is created on the broker; the new execution group has a unique name.

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 in the workbench" on page 259

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**.
5. Deploy a BAR file to the execution group to apply the change to the execution group. You can use any valid BAR file to do this. For more information about deploying BAR files, see Deploying a broker archive file. If you do not deploy a BAR file to the execution group, the changes that you made in the Message Broker Toolkit are not applied to the runtime execution group.

The name of the execution group is updated in the Domains view.

Deleting an execution group using the Message Broker Toolkit

Before you start:

You must complete the following task:

- "Adding an execution group to a broker in the workbench" on page 259

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 244.

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, message sets and all other deployed objects. These are deleted from the execution group, although they are still available in the file system and toolkit.

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 a publish/subscribe topology

Create and configure the resources that are required to develop a topology for your publish/subscribe applications and brokers.

Follow the information in this section to create a publish/subscribe topology that is controlled by the broker. The broker publish/subscribe engine must be enabled for this configuration to be valid. If you have installed WebSphere MQ Version 7.0, and have decided to enable the queue manager's publish/subscribe engine, this information here is not appropriate. Refer to WebSphere MQ Version 7.0 documentation for configuration details.

To configure a publish/subscribe topology in the broker domain:

1. Design and configure your broker domain.

   For further information, refer to "Planning a broker domain" on page 97 and "Configuring broker domain components" on page 177
2. Define the topic trees that you require.

   For further information, refer to Topics and "Adding a new topic" on page 279.
3. Decide which security options to use.

   For further information, refer to "Publish/subscribe security" on page 78 and "Securing the publish/subscribe domain" on page 86.

Setting up the broker domain for publish/subscribe

Refer to the following topics:

"Creating a broker" on page 178

"Modifying a broker" on page 232

"Adding a broker to a broker domain" on page 255

"Configuring broker domain components" on page 177

Publish/subscribe topologies

A *publish/subscribe topology* consists of 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:

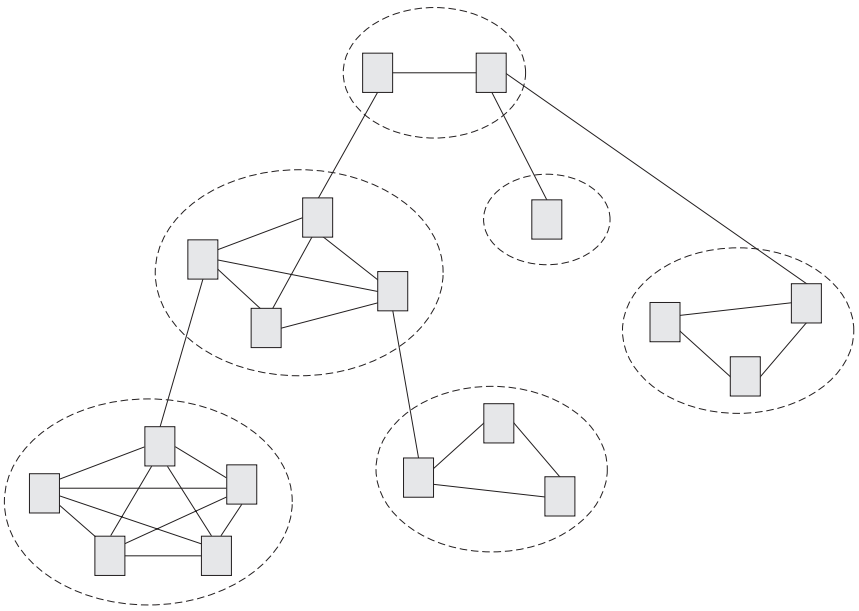
- Client applications can communicate with a nearby broker rather than with a distant broker, thereby getting better response times.
- By using more than one broker, more subscribers can be supported.

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.

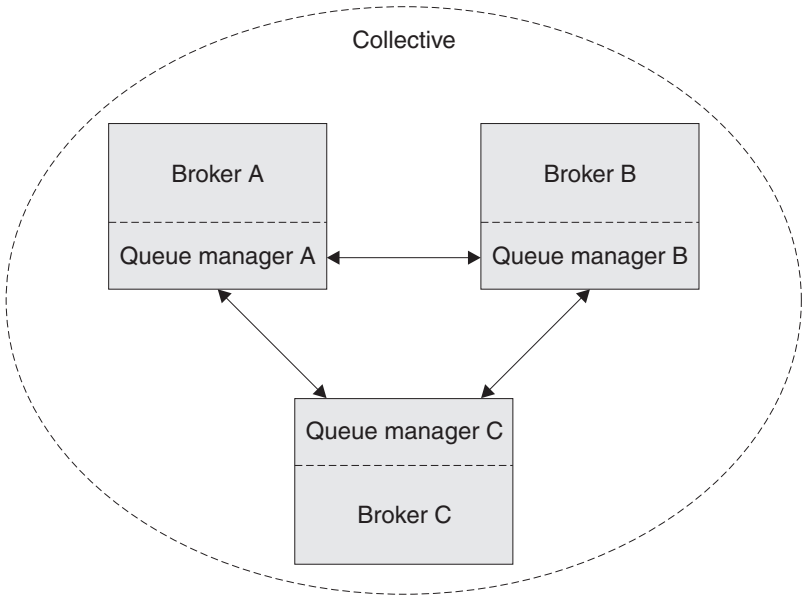


Collectives: A *collective* is a set of brokers that are fully interconnected and form part of a multi-broker network for publish/subscribe applications.

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 apply to specific topics, but some properties apply to all Multicast messages that are 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 things:

- 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.

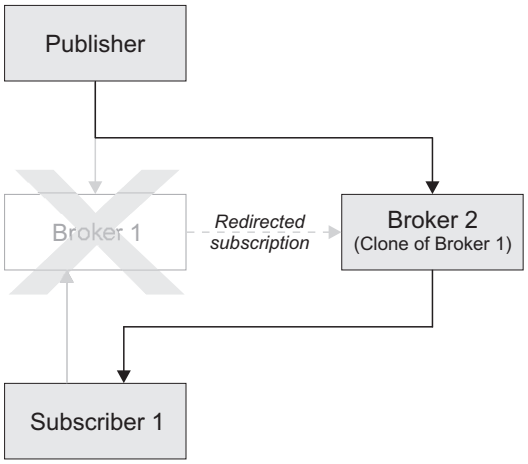
Cloned brokers: A *cloned broker* is a broker for which you have defined one or more clones; the subscription table of a cloned broker is replicated to all other brokers with which it is cloned.

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 instead of Broker 1.



If two brokers are clones within a collective, duplicate messages might be sent to subscribers that are registered with brokers inside that collective.

Use the mqsichangeproperties command to define cloned brokers; the property *clonedPubSubBrokerList* is used to do this.

Migrated topologies:

If you have a WebSphere MQ Publish/Subscribe broker network, you can continue to use this network unchanged.

The introduction of WebSphere Message 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 do anything. You can retain your existing WebSphere MQ Publish/Subscribe network, and install and configure a WebSphere Message 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 Message Broker product.

With the WebSphere Message Broker product, there are two ways in which a broker can be joined to the WebSphere MQ Publish/Subscribe network; it can be joined 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 the properties of the Broker Topology editor.

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 to not display a background image.
6. Optional: On the **Description** page, you can 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.

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 using either the Message Broker Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Broker 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 following steps show you how 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; the brokers 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 using either the Message Broker Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Broker 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 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 that is to be deleted and select Delete, or select the collective that is to be deleted and press the Delete key, or select Delete from the 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 using either the Message Broker Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Broker 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 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 the 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 using either the Message Broker Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Broker 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 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

Set up a multicast broker either by using the workbench or by using the Configuration Manager Proxy Java API. This topic describes how to use the workbench.

Before you can use multicast, you must define the topics that are capable of being multicast. See "Making topics multicast" on page 276.

For information about how to use the Configuration Manager Proxy (CMP), see Developing applications that use the Configuration Manager Proxy API and Class com.ibm.broker.config.proxy.BrokerProxy.MulticastParameterSet.

To enable a broker to handle multicast requests:

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, right-click the broker that you want to modify, and select **Properties**.
5. In the left pane of the properties window, select **Multicast**.
6. Select **Multicast Enabled**.
7. Select **Multicast IPv6 Support Enabled** if you want to use IPv6. If you do not select **Multicast IPv6 Support Enabled**, the IPv6 properties are not available.
8. Optional: Modify the following properties; any properties that are not modified take the default value.

   IPv4 Minimum Address
   The lowest IPv4 address that the broker can use for its multicast transmissions.

   This address must be in the range 224.0.0.0 through 239.255.255.255

   The default value is 239.255.0.0

   IPv4 Maximum Address
   The highest IPv4 address that the broker can use for its multicast transmissions.

   This address must be in the range 224.0.0.0 through 239.255.255.255 and must not be lower than the value of the Minimum Address.

   The default value is 239.255.255.255

   IPv4 Broker Network Interface
   The name of the network interface over which multicast packets are transmitted. This name is relevant only when the broker is running on a host with more than one network interface.

   This name can be a host name or an IPv4 address.

   The default value is None. If you select the default value, the network interface used is operating system dependent.

   IPv6 Minimum Address
   The lowest IPv6 address that the broker can use for its multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:0:1 through ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff

The default value is ff02:0:0:0:0:0:0:1

IPv6 Maximum Address
The highest IPv6 address that the broker can use for its multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:0:1 through ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff and must not be lower than the value of the Minimum Address.

The default value is ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff

IPv6 Broker Network Interface
The name of the network interface over which multicast packets are transmitted. This name is relevant only when the broker is running on a host with more than one network interface.

This name can be a host name or an IPv6 address.

The default value is None. If you select the default value, the network interface used is operating system dependent.

Protocol Type
The multicast protocol type.

Valid values are PTL, PGM/IP, and UDP encapsulated PGM.

The default value is PTL.

For more information, see "Multicast protocol types" on page 275.

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 size must 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 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 property, 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, indicating that the multicast packet must remain local to its originator and does not pass through any routers. The maximum value is 255.

Do not use a value of 0. In some operating systems using a value of 0 might prevent messages from being received, but in other operating systems (for example, Windows 2003, Windows XP, and Linux), a value of 0 does not have this effect.

Overlapping Multicast Topic Behavior
Valid values are Accept, Reject, or Revert.

The default value is Accept.
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 child topics, *xxx* that is enabled for multicast, and *yyy* that is not enabled for multicast.
The three possible settings are:

Accept
The default value. A matching multicast subscription is accepted and all publications matching the topic, except those that are specifically excluded, are multicast. In the preceding example, a multicast subscription to *multicast/#* receives messages published on *xxx* over multicast, but does not receive any messages published on *yyy*.

Reject A multicast subscription to a topic with child topics 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 child topics that are disabled for multicast, result in unicast transmission. A multicast subscription to *multicast/#* receives messages published on *xxx* and *yyy*, but the messages are sent unicast rather than multicast.

Maximum Key Age
The maximum age, in minutes, of a topic encryption key before it must be redefined.

The default value is 360.

9. Optional: In the left pane of the properties window, expand Multicast and click **Advanced**. You can now modify the following additional properties:

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. Select one of the following values from the menu:

Disabled
The default value. Multicast data is transmitted as fast as possible. If the rate at which messages are submitted to be multicast exceeds the server 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*.

If you select Static, you can also select a value for the property *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. However, the rate never exceeds the *Broker Transmission Rate Limit Value*.

Broker Transmission Rate Limit Value
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 property must not exceed the capabilities of the server or network.

This value must be in the range 10 through 1000000.

Client NACK Back Off Time
The maximum time, in milliseconds, that a client listens for another clients NACKs before sending its own NACK.

This value must 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 must 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; The default value of 500 buffers uses approximately 15 MB of main memory.

If memory use is important, try using different values for this property and look at the effect on the overall performance of your application when transmission rates are high.

This value must 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 must 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 must be in the range 1 through 20.

The default value is 7.

This property is not used in Version 6.0.

Broker Minimal History Size
The minimum size, in kilobytes, of a buffer that is allocated to archive all transmitted packets. This buffer is shared by all reliable topics, and can be used to recover lost packets.

This value must be in the range 1000 through 1000000.

The default value is 60000.

Broker NACK Accumulation Time
The time, in milliseconds, that NACKs are aggregated in the broker before recovered packets are sent.

This value must 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 property is applicable only to PGM multicast protocols.

The default value is 262144 which represents 256 MB.

Important: Be aware that by increasing the value of a property, for example *Broker Minimal History Size*, you increase the amount of memory that is required by the Java Virtual Machine (JVM). This increase might cause a `JVM Out of Memory` error when a subscription to the broker is attempted for the first time after this change. If this error occurs, either increase your JVM heap size, or reduce the value of the property (for example, *Broker Minimal History Size*) that you have just increased.

10. Click **OK**.
11. Restart the broker for the changes that you have made to take effect.

The preferred way to change the broker's multicast configuration is to use the workbench. However, you can also use the command mqsichangeproperties to change the broker's properties.

Warning: Any changes to the broker configuration that you make on the mqsichangeproperties are overwritten with the configuration that is held in the Configuration Manager whenever the broker configuration is deployed.

The following table relates the preceding properties to the corresponding names of the parameters on the mqsichangeproperties command that support multicast. For full details of this command, see the "mqsichangeproperties command" on page 437.

Property name	mqsichangeproperties parameter
Multicast Enabled	multicastEnabled
IPv4 Minimum Address	multicastMinimumIPv4Address
IPv4 Maximum Address	multicastMaximumIPv4Address

Property name	mqsichangeproperties parameter
IPv4 Broker Network Interface	multicastMulticastInterface
Multicast IPv6 Support Enabled	Not required
IPv6 Minimum Address	multicastMinimumIPv6Address
IPv6 Maximum Address	multicastMaximumIPv6Address
IPv6 Broker Network Interface	multicastMulticastInterface
Protocol Type	multicastProtocolType
Data Port	multicastDataPort
Broker Packet Size	multicastPacketSizeBytes
Broker Heartbeat Timeout	multicastHeartbeatTimeoutSec
Broker Multicast TTL	multicastMCastSocketTTL
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
Client NACK Check Period	multicastNackCheckPeriodMillis
Client Packet Buffer Number	multicastPacketBuffers
Client Socket Buffer Size	multicastSocketBufferSizeKbytes
Broker History Cleaning Time (deprecated in V6)	Not applicable
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 command 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.

Use commas to separate the properties that are being changed, and their values.

For the changes to be effective, restart the broker.

Multicast protocol types: WebSphere Message Broker supports two different types of multicast protocol:

- PTL (Packet Transfer Layer)
- PGM (PGM/IP and PGM UDP encapsulated)

PTL provides compatibility with WebSphere Business Integration Message Broker Version 5.0, where it is the only multicast protocol that is supported. For new multicast deployments, use one of the two PGM multicast protocols.

The broker supports two implementations of the PGM multicast protocol, PGM/IP and PGM UDP encapsulated. The complexity of your network topology affects which option you choose:

- 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, do not run any other work 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.

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 Topics 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 for Topics window, select **Multicast**.
6. Select **Enabled** for the topic root, or child topic root, in the *Multicast* property that you want to enable:

   For the topic root, the options are either Enabled or Disabled. The default is Disabled.

   For a child topic root, the options are Inherit, Enabled, or Disabled. The default is Inherit.
7. **Automatic Multicast IPv4 Address** is selected by default. If you clear **Automatic Multicast IPv4 Address** you must type in the name of the *IPv4 MC Group Address*. This property is mandatory.
8. **Automatic Multicast IPv6 Address** is cleared by default. You can type in the name of the *IPv6 MC Group Address*.
9. Optional: Select **Encrypted**.
10. Select the *Quality of Service* that you require. The options are Reliable or Unreliable. The default is Reliable.
11. 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. Using up all the IRLM storage might cause problems when you try to restart the broker.

The following actions might prevent you from using up all the IRLM storage and thereby avoid problems when you try to restart the broker:

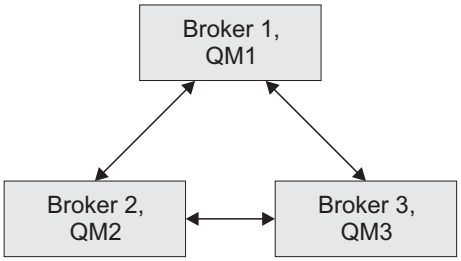
1. Tune the publish/subscribe topology:
   a. Balance execution groups across more brokers; by balancing execution groups across more brokers, you cause fewer execution groups to require to start at the same time and to have concurrent locks for the same DB2 subsystem.
   b. Put the brokers in publish/subscribe collectives; by putting the brokers in collectives, you reduce the number of subscriptions in a single broker table, and also reduce the amount of concurrent access to DB2. See "Publish/subscribe topologies" on page 263 for more information.
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 require.
   b. If you are unable to choose a value of MAXCSA that is 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. By using this option, you cause the IRLM to place in its private address space the control block structures that relate to locking. See the DB2 information center (z/OS) (Versions 8 and 9) for more information.

Note: There might be a slight (approximately 1 to 2 percent) performance degradation when you use the `PC=YES` option on the START irlmproc command.

Setting up cloned brokers

Each broker that is to be cloned with other brokers must be told which brokers are to be its clones.

- Windows Linux UNIX 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 -n clonedPubSubBrokerList -v \"broker2,QM2,broker3,QM3\"`
  2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"broker1,QM1,broker3,QM3\"`
  3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"broker1,QM1,broker2,QM2\"`
- z/OS 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 -n clonedPubSubBrokerList -v "broker2,QM2,broker3,QM3"`
  2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "broker1,QM1,broker3,QM3"`
  3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -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.

- Windows Linux UNIX 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 -n clonedPubSubBrokerList -v \"broker1,QM1,broker2,QM2,broker3,QM3\"`
  2. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"+broker4,QM4\"`
  3. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"+broker4,QM4\"`
  4. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"+broker4,QM4\"`

- z/OS 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 -n clonedPubSubBrokerList -v "broker1,QM1,broker2,QM2,broker3,QM3"`
  2. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "+broker4,QM4"`
  3. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "+broker4,QM4"`
  4. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "+broker4,QM4"`

Deleting a cloned broker

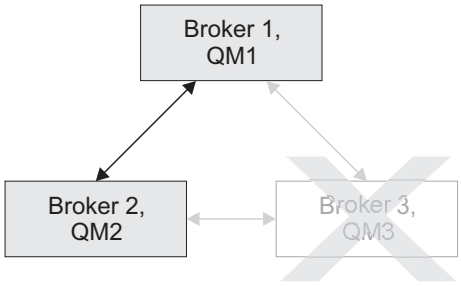
To delete a broker from a set of cloned brokers, use the **mqsichangeproperties** command to delete the brokers that were its clones, and to tell each of the other brokers that one of its clones has been deleted.

- Windows Linux UNIX 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 -n clonedPubSubBrokerList -v \"-broker3\"`

2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"-broker3\"`
3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v \"\"`

- z/OS 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 -n clonedPubSubBrokerList -v "-broker3"`
  2. `mqsichangeproperties broker2 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "-broker3"`
  3. `mqsichangeproperties broker3 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -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 280
- "Querying subscriptions" on page 280

Adding a new topic

You can define a new topic explicitly by using either the Message Broker Toolkit or the Configuration Manager Proxy Java API.

This topic describes how to use the Message Broker 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, do the following:

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 that is 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 using either the Message Broker Toolkit or the Configuration Manager Proxy Java API.

This topic describes how to use the Message Broker 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 subscription by using either the Message Broker Toolkit or the Configuration Manager Proxy Java API.

This topic describes how to use the Message Broker 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.

Configuring global coordination of transactions (two-phase commit)

Globally coordinate message flow transactions with a WebSphere MQ queue manager to ensure the data integrity of transactions.

Before you start:

Complete the following tasks:

- Create and configure databases, see "Configuring broker and user databases" on page 109
- Create a broker, see "Creating a broker" on page 178

On distributed platforms, the default behavior of the broker is to manage all message flow transactions by using a one-phase commit approach. In many contexts this approach is sufficient, but if your business requires assured data integrity (for example, for audit reasons or for financial transactions), configure the broker's WebSphere MQ queue manager to manage the message flow transactions in a two-stage commit approach by using the XA protocol standard. For more information about global coordination of transactions, see "The Transactional model" on page 282.

z/OS On z/OS, all transactions are globally coordinated by Resource Recovery Service (RRS), therefore the instructions in this topic do not apply. RRS must, however, be available; see "Resource Recovery Service planning on z/OS" on page 169.

To configure your system for global coordination of transactions:

1. Ensure that the databases are configured for global coordination. For information about how to perform this configuration, see "Configuring databases for global coordination of transactions" on page 124.
2. Configure the broker environment so that the broker's queue manager coordinates transactions. The steps to configure the broker environment depend on the database manager that you are using and, if your databases are connected with ODBC, whether the broker's queue manager and the execution group are 32-bit or 64-bit. (JDBC connections are not dependent on 32-bit or 64-bit.)

   If you are using shared memory to connect directly to a 64-bit database instance, you must use a 64-bit queue manager to globally coordinate transactions (all WebSphere MQ Version 6 queue managers on 64-bit platforms are 64-bit). A 32-bit queue manager cannot connect directly to a 64-bit database instance.

- "Configuring global coordination with DB2 using a 32-bit queue manager" on page 285
- "Configuring global coordination with DB2 using a 64-bit queue manager" on page 287
- "Configuring global coordination with Informix using a 32-bit queue manager" on page 290
- "Configuring global coordination with Informix using a 64-bit queue manager" on page 292
- "Configuring global coordination with Oracle using a 32-bit queue manager" on page 295
- "Configuring global coordination with Oracle using a 64-bit queue manager" on page 300
- "Configuring global coordination with Sybase" on page 303

3. Configure the message flow for global coordination. For information about how to perform this configuration, see Configuring globally coordinated message flows.

When you have completed these steps, your message flows are processed by using global coordination, which is managed by the queue manager.

You must complete all of the steps correctly; if you do not, global coordination will not work.

For an example of how you can use WebSphere MQ to globally coordinate transactions, look at the following sample:

- Error Handler sample

You can view samples only when you use the information center that is integrated with the Message Broker Toolkit.

The Transactional model

If you configure message flow resources to participate in coordinated transactions, the actions taken to update those resources conform to the transactional model to ensure consistency and integrity of data

A message flow consists of the following constituent parts:

- An input queue
- The message flow or logic, which is defined by a sequence of nodes
- None or more database tables
- None or more output queues

The following steps represent a typical sequence of events when a message flow processes a message:

1. A message is taken from the input queue.
2. Data is read from or written to the database tables and queues.
3. The system is inactive and awaits the next input message.

This sequence of events makes no distinction between accessing tables and writing output messages. Although a flow often produces some sort of output message, no real distinction is made between producing an output message and updating a database table; in both cases, the state of the data in the system changes.

Consider the following diagram:

```
=====x=========x===x=======x=============x====x=====
     1         2   3       4             5    6
```

The line represents the data in the system as time passes. At time 1, a message arrives and is taken from the input queue. At times 2, 3, 4, and 5, data is used to update database tables, or it is written to queues. Changes in the state of the data are indicated in the diagram by the *x* symbol. At time 6, the output messages are sent and the system is inactive. Between these events, the state of the data is not changed; this state is indicated in the diagram by the = symbol.

If a failure occurs on the system (for example, a loss of power to the computer on which the broker is running), the changes to the state of tables and queues that were made before the failure have been implemented, but no more changes take place after the failure. This situation is unacceptable in certain circumstances; for example, if a system failure occurs when making a payment from a current account to a mortgage account, the payment might be taken from the current account, but it is not added to the mortgage account.

Transactions

To avoid the problem that is described previously, the broker, queuing systems, and databases have a *transactional model*. As processing proceeds, additional data is recorded that allows the original state to be restored in the event of a failure. The following diagram illustrates the state of this extra data:

```
-----x=========x===x=======x=============x====x-----
     1         2   3       4             5    6
```

The line in the diagram represents the extra data in the system as time passes. At time 1, a message arrives and is taken from the input queue. Before time 1, no extra data exists in the system; this state is indicated in the diagram by the - symbol. After time 1, the state represents the fact that a message has been taken from the queue so that it can be put back on the queue if necessary. At times 2, 3, 4, and 5, data is used to update database tables or it is written to queues. Again, the state of the extra data changes so that the changes to tables and queues can be undone if necessary. At time 6, the output messages are sent, the system is inactive, and extra data in the system no longer exists.

Between these events, the state of the extra data does not change; this state is indicated by the = symbol. If a failure occurs at any time between time 1 and time 6, the extra data is used to restore the original state of the system's data, therefore, effectively, no data has been written to the output queues, none of the tables have been updated, and the input message has not been taken from the input queue. If no failure occurs, the changes become permanent at time 6 (an undo operation that follows a subsequent failure will not undo the changes).

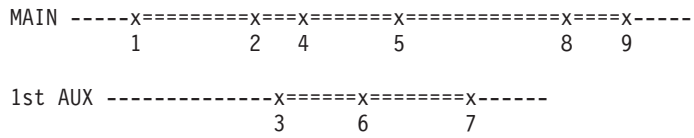
This mode of operation is known as *coordinated transaction mode*. The successful completion of a transaction is known as its *commit*. Unsuccessful completion is known as *rollback*.

Uncoordinated Auxiliary Transactions

The key feature of the coordinated transaction mode of operation is that, regardless of where or when the failure appears, either all of the changes to queues and tables that are associated with one input message are made, or none of the changes are made. However, this behavior is not always suitable, as the following examples illustrate:

- You want to create an audit log of all attempts at processing. The log entries need to be committed even when updates to the main tables and queues are rolled back.
- You want to send an acknowledgment or non-acknowledgment message back to the originator of the messages that you are processing, according to whether the message processing succeeds or fails. These messages must be sent even when the updates to the main tables and queues are rolled back.

To satisfy such requirements, you can configure message flows to change queues and tables in a separate, or auxiliary, transaction. This behavior is illustrated in the following diagram:



The *MAIN* line represents the main transaction, which includes the extra data that is recorded to restore the original state if necessary. The *1st AUX* line represents an auxiliary transaction. At time 3, an update to a table or queue is made, and another update is made at time 6. At time 7, the message flow determines that all the changes that must be made under the auxiliary transaction are complete, and it commits the changes.

If the message flow fails before time 7, the state of the system would be unchanged because both transactions would be rolled back. If failure occurs after time 7 but before time 9, the auxiliary transaction would already have been committed but the main transaction would be rolled back. If a failure has not occurred by time 9, both transactions are committed.

Database Auxiliary Transactions

You can use more than one auxiliary transaction, and make a number of updates to database tables that can be committed or rolled back. You can then make additional changes to the same database tables, or to different tables, and then commit or rollback these changes.

Each database that you use has its own auxiliary transaction; therefore, if the message flow updates tables that belong to different database instances (different data source names), an auxiliary transaction exists for each database. You must commit or roll back these transactions individually. Updates that have not been committed or rolled back when the operation completes (at time 9 in the example shown previously) are committed or rolled back automatically by the broker, according to whether the processing succeeded or failed.

Some databases types, such as DB2 on AIX, do not allow both coordinated and uncoordinated transactions in the same database instance. In these cases, you must create separate database instances. Configure one database instance for coordinated transactions, and configure the second instance for uncoordinated transactions.

Use the ESQL COMMIT and ROLLBACK statements to commit and roll back auxiliary database transactions. Obtain operations outside the main transaction by specifying the UNCOORDINATED keyword on the individual database statements (for example, the INSERT and UPDATE statements).

Queue Auxiliary Transactions

Not all queuing systems have the database capability that is described in the previous section. With WebSphere MQ, each individual uncoordinated read or write operation to a queue has an implied commit action, therefore you cannot put two messages and then decide to commit both or roll back both. The COMMIT and ROLLBACK statements therefore operate only on databases.

Nodes

The previous sections refer to message flows, but not to nodes. The way in which a message flow is divided into nodes has no effect on transactions. For operations on databases, an unlimited number of nodes can make updates to the main transaction, and to an unlimited number of auxiliary transactions, without restriction.

Configuring global coordination with DB2 using a 32-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in DB2 databases under the control of a 32-bit queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions; see "Configuring databases for global coordination of transactions" on page 124.

All WebSphere MQ Version 6 queue managers on 32-bit platforms run in 32-bit mode. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate message flows that are deployed only to 32-bit execution groups.

To configure your broker environment for global coordination using a 32-bit queue manager as the transaction manager:

1. Decide whether the broker will connect to databases using TCP/IP or using shared memory.

   For more information about TCP/IP connections, see the example in the section about message SQL1224N in Resolving problems when using databases.

   To enable shared memory:

   a. Stop the broker by running the following command, where *broker* is the name of your broker:

      ```
      mqsistop broker
      ```

   b. Run the following command to ensure that the broker is run in an environment with the extended memory variable exported:

      ```
      export EXTSHM=ON
      ```

   c. Restart the broker by running the following command, where *broker* is the name of your broker:

      ```
      mqsistart broker
      ```

   d. On the DB2 server, ensure that shared memory support is turned on. For more information, see "Configuring databases for global coordination of transactions" on page 124.

2. Linux On Linux on x86, run the mqsimanagexalinks command.
3. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or

contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

Linux On Linux on x86:

a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/queue_manager_name/qm.ini`, where *queue_manager_name* is the name of the queue manager that is associated with the broker.

b. At the end of the `qm.ini` file, paste the following stanza:

```
XAResourceManager:
Name=DB2
SwitchFile=install_dir/sample/xatm/db2swit
XAOpenString=db=MyDataSource,uid=MyUserId,pwd=MyPassword,toc=t
XACloseString=
ThreadOfControl=THREAD
```

c. On the **SwitchFile** line of the stanza, replace *install_dir* with the full path to the installation location of the WebSphere Message Broker instance.

The switch file is supplied by WebSphere Message Broker.

d. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

- *MyDataSource* is the name of the data source to which you want to connect.
- *MyUserId* must be the user name that the broker uses to connect to the database.

  You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

  1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
  2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
  3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command
- *MyPassword* is the password that is associated with the user name.

e. Accept the default values for all the other lines in the stanza. For example:

```
XAResourceManager:
Name=DB2
SwitchFile=/opt/mqsi/sample/xatm/db2swit
XAOpenString=db=MYDB,uid=wbrkuid,pwd=wbrkpw,toc=t
XACloseString=
ThreadOfControl=THREAD
```

Windows On Windows:

a. From the **Start** menu, open WebSphere MQ Explorer.

b. Open the queue manager's Properties dialog box, then open **XA resource managers**.

c. In the **SwitchFile** field, enter the full path to the switch file, as shown in the following example where *install_dir* is the location in which the broker is installed:

```
install_dir\sample\xatm\db2swit.dll
```

d. In the **XAOpenString** field, paste the following string:

```
db=MyDataSource,uid=MyUserId,pwd=MyPassword,toc=t
```

e. In the **XAOpenString** field, replace the values with values that are appropriate for your configuration:
   - *MyDataSource* is the name of the data source to which you want to connect.
   - *MyUserId* must be the user name that the broker uses to connect to the database.

     You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

     1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
     2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
     3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command
   - *MyPassword* is the password that is associated with the user name.

   For example:

   ```
   db=MYDB,uid=wbrkuid,pwd=wbrkpw,toc=t
   ```

f. Accept the default values for all the other fields on the page.

4. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

DB2 is now configured for global coordination with a 32-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with DB2 using a 64-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in DB2 databases under the control of a 64-bit queue manager.

Before you start:
- Ensure that the databases are configured for global coordination of transactions, see "Configuring databases for global coordination of transactions" on page 124.

All WebSphere MQ Version 6 queue managers on 64-bit platforms run in 64-bit mode. 64-bit queue managers can coordinate transactions only in 64-bit mode. If the broker uses a 64-bit queue manager, you can globally coordinate message flows

that are deployed to either 64-bit or 32-bit execution groups, but if you are using 32-bit execution groups, you must define the data source name of the user database in both `odbc32.ini` and `odbc64.ini`. If the broker uses a 64-bit queue manager, or has a 64-bit execution group, the databases to which the broker connects must also be 64-bit mode.

To configure your broker environment for global coordination using a 64-bit queue manager as the transaction manager:

Follow the instructions appropriate to your execution groups:

- "32-bit execution groups"
- "64-bit execution groups" on page 289

32-bit execution groups

To configure the broker's queue manager to coordinate message flows that are deployed to a 32-bit execution group:

1. Linux UNIX On Linux (except Linux on x86) and UNIX systems, run the mqsimanagexalinks command.
2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX On Linux (except Linux on x86) and UNIX:

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/`*`queue_manager_name`*`/qm.ini`, where *queue_manager_name* is the name of the queue manager that is associated with the broker.

   b. At the end of the `qm.ini` file, paste the following stanza:

   ```
   XAResourceManager:
   Name=DB2
   SwitchFile=db2swit
   XAOpenString=db=MyDataSource,uid=MyUserId,pwd=MyPassword,toc=t
   XACloseString=
   ThreadOfControl=THREAD
   ```

   The switch file is supplied by WebSphere Message Broker.

   c. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

   - *MyDataSource* is the name of the data source to which you want to connect.
   - *MyUserId* must be the user name that the broker uses to connect to the database.
   - *MyPassword* is the password that is associated with the user name.

   d. Accept the default values for all the other lines in the stanza. For example:

   ```
   XAResourceManager:
   Name=DB2
   SwitchFile=db2swit
   XAOpenString=db=MYDB,uid=wbrkuid,pwd=wbrkpw,toc=t
   XACloseString=
   ThreadOfControl=THREAD
   ```

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

```
endmqm queue_manager_name
strmqm queue_manager_name
```

When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/queue_manager_name/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

DB2 is now configured for global coordination with a 64-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

64-bit execution groups

To configure the broker's queue manager to coordinate message flows that are deployed to a 64-bit execution group:

1. Linux UNIX On Linux (except Linux on x86) and UNIX, run the mqsimanagexalinks command.
2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX On Linux (except Linux on x86) and UNIX:

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/queue_manager_name/qm.ini`, where *queue_manager_name* is the name of the broker that is associated with the queue manager.

   b. At the end of the `qm.ini` file, paste the following stanza:

   ```
   XAResourceManager:
   Name=DB2
   SwitchFile=db2swit
   XAOpenString=db=MyDataSource,uid=MyUserId,pwd=MyPassword,toc=t
   XACloseString=
   ThreadOfControl=THREAD
   ```

   The switch file is supplied by WebSphere Message Broker.

   c. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

   - *MyDataSource* is the name of the data source to which you want to connect.
   - *MyUserId* must be the user name that the broker uses to connect to the database.

     You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

     1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.

2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

- *MyPassword* is the password that is associated with the user name.

d. Accept the default values for all the other lines in the stanza. For example:

```
XAResourceManager:
Name=DB2
SwitchFile=db2swit
XAOpenString=db=MYDB,uid=wbrkuid,pwd=wbrkpw,toc=t
XACloseString=
ThreadOfControl=THREAD
```

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

DB2 is now configured for global coordination with a 64-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Informix using a 32-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in Informix databases under the control of a 32-bit queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions, see "Configuring databases for global coordination of transactions" on page 124.

All WebSphere MQ Version 6 queue managers on 32-bit platforms run in 32-bit mode. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate message flows that are deployed only to 32-bit execution groups.

To configure your broker environment for global coordination using a 32-bit queue manager as the transaction manager:

1. Linux UNIX On Linux on x86, Linux on x86-64, and UNIX, run the mqsimanagexalinks command.
2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

Linux UNIX **On Linux on x86, Linux on x86-64, and UNIX::**

a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/`*queue_manager_name*`/qm.ini`, where *queue_manager_name* is the name of the queue manager that is associated with the broker.

b. At the end of the `qm.ini` file, paste the following stanza:

```
XAResourceManager:
Name=INFORMIX
SwitchFile=infswit
XAOpenString=DB=MyDatabase\;RM=MyResourceManager\
   USER=MyUserId\;PASSWD=MyPassword
ThreadOfControl=PROCESS
```

c. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

- *MyDatabase* is the name of the database to which you want to connect.
- *MyResourceManager* is the name of the Informix IDS Server.
- *MyUserId* must be the user name that the broker uses to connect to the database.

  You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

  1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
  2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
  3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command
- *MyPassword* is the password that is associated with the user name.

d. Accept the default values for all the other lines in the stanza. For example:

```
XAResourceManager:
Name=INFORMIX
SwitchFile=infswit
XAOpenString=DB=MYDB\;RM=idsserver\;
   USER=wbrkuid\;PASSWD=wbrkpw
ThreadOfControl=PROCESS
```

Windows **On Windows:**

a. From the **Start** menu, open WebSphere MQ Explorer.

b. Open the queue manager's Properties dialog box, then open **XA resource managers**.

c. In the **SwitchFile** field, enter the full path to the switch file, as shown in the following example, where *install_dir* is the location in which the broker is installed:

```
install_dir\sample\xatm\infswit.dll
```

d. In the **XAOpenString** field, paste the following string:

```
DB=MyDatabase;RM=MyResourceManager;
   USER=MyUserId;PASSWD=MyPassword
```

e. In the **XAOpenString** field, replace the values with values that are appropriate for your configuration:

- *MyDatabase* is the name of the database to which you want to connect.
- *MyResourceManager* is the name of the Informix IDS Server.

- *MyUserId* must be the user name that the broker uses to connect to the database.
- *MyPassword* is the password that is associated with the user name.

For example:

```
DB=MYDB;RM=idsserver;USER=wbrkuid;PASSWD=wbrkpw
```

f. Accept the default values for all the other fields on the page.

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Informix is now configured for global coordination with a 32-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Informix using a 64-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in Informix databases under the control of a 64-bit queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions, see "Configuring databases for global coordination of transactions" on page 124.

All WebSphere MQ Version 6 queue managers on 64-bit platforms run in 64-bit mode. 64-bit queue managers can coordinate transactions only in 64-bit mode. If the broker uses a 64-bit queue manager, you can globally coordinate message flows that are deployed to either 64-bit or 32-bit execution groups, but if you are using 32-bit execution groups, you must define the data source name of the user database in both `odbc32.ini` and `odbc64.ini`. If the broker uses a 64-bit queue manager, or has a 64-bit execution group, the databases to which the broker connects must also be 64-bit mode.

Follow the appropriate instructions for your execution groups:

- "32-bit execution groups"
- "64-bit execution groups" on page 294

32-bit execution groups

To configure the broker's queue manager to coordinate message flows that are deployed to a 32-bit execution group:

1. Linux UNIX On all Linux platforms except Linux on x86, and on UNIX systems, run the mqsimanagexalinks command.

2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX On all Linux platforms except Linux on x86, and on UNIX:

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/`*queue_manager_name*`/qm.ini`, where *queue_manager_name* is the name of the broker that is associated with the queue manager.

   b. At the end of the `qm.ini` file, paste the following stanza:

      ```
      XAResourceManager:
      Name=INFORMIX
      SwitchFile=infswit
      XAOpenString=db=MyDatabase\;RM=MyResourceManager\;
         USER=MyUserId\;PASSWD=MyPassword
      ThreadOfControl=PROCESS
      ```

      The switch file is supplied by WebSphere Message Broker.

   c. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

      - *MyDatabase* is the name of the database to which you want to connect.
      - *MyResourceManager* is the name of the Informix IDS Server.
      - *MyUserId* must be the user name that the broker uses to connect to the database.

        You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

        1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
        2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
        3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

      - *MyPassword* is the password that is associated with the user name.

   d. Accept the default values for all the other lines in the stanza. For example:

      ```
      XAResourceManager:
      Name=INFORMIX
      SwitchFile=infswit
      XAOpenString=db=MYDB\;RM=idsserver\;
         USER=wbrkuid\;PASSWD=wbrkpw
      ThreadOfControl=PROCESS
      ```

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the

name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Informix is now configured for global coordination with a 64-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

64-bit execution groups

To configure the broker's queue manager to coordinate message flows that are deployed to a 64-bit execution group:

1. Linux UNIX On all Linux platforms except Linux on x86, and on UNIX systems, run the mqsimanagexalinks command.
2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX On all Linux platforms except Linux on x86, and on UNIX:

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/`*`queue_manager_name`*`/qm.ini`, where *queue_manager_name* is the name of the broker that is associated with the queue manager.

   b. At the end of the `qm.ini` file, paste the following stanza:

   ```
   XAResourceManager:
   Name=INFORMIX
   SwitchFile=infswit
   XAOpenString=db=MyDatabase\;RM=MyResourceManager\;
      USER=MyUserId\;PASSWD=MyPassword
   ThreadOfControl=PROCESS
   ```

   The switch file is supplied by WebSphere Message Broker.

   c. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

   - *MyDatabase* is the name of the database to which you want to connect.
   - *MyResourceManager* is the name of the Informix IDS Server.
   - *MyUserId* must be the user name that the broker uses to connect to the database.

     You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

     1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
     2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
     3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command
   - *MyPassword* is the password that is associated with the user name.

   d. Accept the default values for all the other lines in the stanza. For example:

```
XAResourceManager:
Name=INFORMIX
SwitchFile=infswit
XAOpenString=db=MYDB\;RM=idsserver\;
   USER=wbrkuid,pwd=wbrkpw
ThreadOfControl=PROCESS
```

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Informix is now configured for global coordination with a 64-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Oracle using a 32-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in Oracle databases under the control of a 32-bit queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions; see "Configuring databases for global coordination of transactions" on page 124.

Complete this task to configure the broker environment when the broker uses a 32-bit queue manager.

All WebSphere MQ Version 6 queue managers on 32-bit platforms run in 32-bit mode. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate message flows that are deployed only to 32-bit execution groups.

To configure your broker environment for global coordination using a 32-bit queue manager as the transaction managerwith the DataDirect drivers:

1. Linux UNIX On Linux on x86, Linux on x86-64, and UNIX, run the mqsimanagexalinks command.
2. Linux On Linux on x86 and Linux on x86-64, run the mqsi_setupdatabase command.
3. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX **On Linux on x86, Linux on x86-64, and UNIX:**

a. Open the queue manager's qm.ini file in a text editor. The qm.ini file is located at /var/mqm/qmgrs/*queue_manager_name*/qm.ini, where *queue_manager_name* is the name of the broker that is associated with the queue manager.

b. Add one of the following stanzas to the end of the qm.ini:

   - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA+SQLNET=MyNetServiceName
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +Sid=MySID
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```

   - If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA
   +SQLNET=MyNetServiceNameForCluster
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +ServiceName=MyServiceName
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```

c. On the **SwitchFile** line of the stanza, replace *SwitchFileName* with the name of the switch file for your operating system. The following table shows the switch file for each operating system.

Operating system	Switch file
HP-UX on PA-RISC	UKor8dtc20.sl
All other platforms	UKor8dtc23.so

   The switch file is supplied by WebSphere Message Broker.

d. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

   - *MyHostName* is the name of the TCP/IP host that hosts the Oracle database.
   - *MyPortNumber* is the TCP/IP port on which the Oracle database is listening.
   - *MySID* is the Oracle System Identifier (SID) of the database.
   - *MyUserId* must be the user name that the broker uses to connect to the database.

     You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

- *MyPassword* is the password that is associated with the user name.
- *MyDataSourceName* is the ODBC data source name for the database.
- *MyNetServiceName* is the Net Service Name defined for the connect descriptor to your data source.
- *MyNetServiceNameForCluster* is the Net Service Name defined for the connect descriptor to your cluster. This value is "WMBSERVICE" in the tnsnames example entry shown.
- *MyServiceName* is the value set for the Service Name in the stanza referenced by *MyNetServiceNameForCluster*. This value is "racxa.test.com" in the tnsnames example entry shown.

An example entry in `tnsnames.ora` might contain the following content:

```
WMBSERVICE =
      (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST=harac1)(PORT=1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST=harac2)(PORT=1521))
           (CONNECT_DATA =
                    (SERVER = DEDICATED)
                    (SERVICE_NAME = racxa.test.com)
            )
       )
```

e. Accept the default values for all the other lines in the stanza. For example:

- On AIX:
  - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc23.so
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+Sid=diaz
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

  - If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc23.so
XAOpenString=ORACLE_XA+SQLNET=WMBSERVICE
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+ServiceName=racxa.test.com
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

- On HP-UX on PA-RISC:
  - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc20.sl
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+Sid=diaz
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

– If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc20.sl
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521
   +ServiceName=racxa.test.com
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

Windows **On Windows:**

a. From the **Start** menu, open WebSphere MQ Explorer.

b. Open the queue manager's Properties dialog box, then open **XA resource managers**.

c. In the **SwitchFile** field, enter the full path to the switch file, as shown in the following example where *install_dir* is the location in which the broker is installed:

```
install_dir\bin\UKor8dtc23.dll
```

d. In the **XAOpenString** field, paste the following string:

- If you are not using Oracle Real Application Clusters:

```
ORACLE_XA+SQLNET=MyNetServiceName
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +Sid=MySID
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
```

- If you are using Oracle Real Application Clusters:

```
ORACLE_XA+SQLNET=MyNetServiceNameForCluster
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +ServiceName=MyServiceName
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
```

e. In the **XAOpenString** field, replace the values with values that are appropriate for your configuration:

- *MyHostName* is the name of the TCP/IP host that hosts the Oracle database.
- *MyPortNumber* is the TCP/IP port on which the Oracle database is listening.
- *MySID* is the Oracle System Identifier (SID) of the database.

- *MyUserId* must be the user name that the broker uses to connect to the database.

  You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

  1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
  2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.
  3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command
- *MyPassword* is the password that is associated with the user name.
- *MyDataSourceName* is the ODBC data source name for the database.
- *MyNetServiceName* is the Net Service Name defined for the connect descriptor to your data source.
- *MyNetServiceNameForCluster* is the Net Service Name defined for the connect descriptor to your cluster. This value is "WMBSERVICE" in the tnsnames example entry shown.
- *MyServiceName* is the value set for the Service Name in the stanza referenced by *MyNetServiceNameForCluster*. This value is "racxa.test.com" in the tnsnames example entry shown.

An example entry in `tnsnames.ora` might have the following content:

```
WMBSERVICE =
      (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST=harac1)(PORT=1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST=harac2)(PORT=1521))
           (CONNECT_DATA =
                    (SERVER = DEDICATED)
                    (SERVICE_NAME = racxa.test.com)
            )
       )
```

For example:

- If you are not using Oracle Real Application Clusters:

```
ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+Sid=diaz
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
```

- If you are using Oracle Real Application Clusters:

```
ORACLE_XA+SQLNET=WMBSERVICE
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521
   +ServiceName=racxa.test.com
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
```

f. Accept the default values for all the other fields on the page.

4. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

```
endmqm queue_manager_name
strmqm queue_manager_name
```

When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/queue_manager_name/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Oracle is now configured for global coordination with a 32-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Oracle using a 64-bit queue manager

Configure your broker environment to globally coordinate message flow transactions with updates in Oracle databases under the control of a 64-bit queue manager.

Before you start:

- Ensure that you have configured the databases for global coordination of transactions.

All WebSphere MQ Version 6 queue managers on 64-bit platforms run in 64-bit mode. 64-bit queue managers can coordinate transactions only in 64-bit mode. If the broker uses a 64-bit queue manager, you can globally coordinate message flows that are deployed to either 64-bit or 32-bit execution groups, but if you are using 32-bit execution groups, you must define the data source name of the user database in both `odbc32.ini` and `odbc64.ini`. If the broker uses a 64-bit queue manager, or has a 64-bit execution group, the databases to which the broker connects must also be 64-bit mode.

To configure your broker environment for global coordination using a 64-bit queue manager as the transaction manager with the DataDirect drivers:

1. Linux UNIX On Linux and UNIX, run the mqsimanagexalinks command.
2. Linux UNIX On Linux on x86-64, Solaris on SPARC, HP-UX on PA-RISC, and AIX, run the mqsi_setupdatabase command.
3. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX **On Linux (except Linux on x86) and UNIX:**

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/queue_manager_name/qm.ini`, where *queue_manager_name* is the name of the broker that is associated with the queue manager.
   b. Add one of the following stanzas to the end of the `qm.ini` file:
      - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA
   +SQLNET=MyNetServiceName
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +Sid=MySID
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```

- If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA
   +SQLNET=MyNetServiceNameForCluster
   +HostName=MyHostName
   +PortNumber=MyPortNumber
   +ServiceName=MyServiceName
   +ACC=P/MyUserId/MyPassword
   +sestm=100+threads=TRUE
   +DataSource=MyDataSourceName
   +DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```

c. On the **SwitchFile** line of the stanza, replace *SwitchFileName* with the name of the switch file for your operating system. The following table shows the name of the switch file for each operating system.

Operating system	Switch file
HP-UX on PA-RISC	UKor8dtc20.sl
All other platforms	UKoradtc23.so

The switch file is supplied by WebSphere Message Broker.

d. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

- *MyHostName* is the name of the TCP/IP host that hosts the Oracle database.
- *MyPortNumber* is the TCP/IP port on which the Oracle database is listening.
- *MySID* is the Oracle System Identifier (SID) of the database.
- *MyUserId* must be the user name that the broker uses to connect to the database.

  You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

  1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.
  2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.

3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

- *MyPassword* is the password that is associated with the user name.
- *MyDataSourceName* is the ODBC data source name for the database.
- *MyNetServiceName* is the Net Service Name defined for the connect descriptor to your data source.
- *MyNetServiceNameForCluster* is the name for the connect descriptor to your cluster. This value is "WMBSERVICE" in the tnsnames example entry shown.
- *MyServiceName* is value set for the Service Name in the stanza referenced by *MyNetServiceNameForCluster*. This value is "racxa.test.com" in the tnsnames example entry shown.

An example entry in `tnsnames.ora` might have the following content:

```
WMBSERVICE =
      (DESCRIPTION =
         (ADDRESS = (PROTOCOL = TCP)(HOST=harac1)(PORT=1521))
      (ADDRESS = (PROTOCOL = TCP)(HOST=harac2)(PORT=1521))
           (CONNECT_DATA =
                    (SERVER = DEDICATED)
                    (SERVICE_NAME = racxa.test.com)
            )
       )
```

e. Accept the default values for all the other lines in the stanza. For example:

- On AIX:
  - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc23.so
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+Sid=diaz
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

  - If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc23.so
XAOpenString=ORACLE_XA
   +SQLNET=WMBSERVICE
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521
   +ServiceName=racxa.test.com
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

- On HP-UX on PA-RISC:
  - If you are not using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc20.sl
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521+Sid=diaz
```

```
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

– If you are using Oracle Real Application Clusters:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc20.sl
XAOpenString=ORACLE_XA+SQLNET=diaz
   +HostName=diaz.hursley.ibm.com
   +PortNumber=1521
   +ServiceName=racxa.test.com
   +ACC=P/wbrkuid/wbrkpw
   +sestm=100+threads=TRUE
   +DataSource=MYDB+DB=MYDB+K=2+
XACloseString=
ThreadOfControl=THREAD
```

4. AIX On AIX (for 64-bit execution groups only), set the environment variable `DDTEK_XA_DYNAMIC_REGISTRATION=1` if you want to enable Oracle data sources for use in global coordination from a queue manager and broker to perform dynamic XA registration. Set the environment variable before starting WebSphere MQ or the broker.
5. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Oracle is now configured for global coordination with a 64-bit queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Sybase

Configure your broker environment to globally coordinate message flow transactions with updates in Sybase databases under the control of a queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions.

All WebSphere MQ Version 6 queue managers on 32-bit platforms run in 32-bit mode. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate message flows that are deployed only to 32-bit execution groups.

All WebSphere MQ Version 6 queue managers on 64-bit platforms run in 64-bit mode. 64-bit queue managers can coordinate transactions only in 64-bit mode. If the broker uses a 64-bit queue manager, you can globally coordinate message flows that are deployed to either 64-bit or 32-bit execution groups, but if you are using

32-bit execution groups, you must define the data source name of the user database in both `odbc32.ini` and `odbc64.ini`. If the broker uses a 64-bit queue manager, or has a 64-bit execution group, the databases to which the broker connects must also be 64-bit mode.

To configure your broker environment for global coordination using a WebSphere MQ queue manager as the transaction manager with the DataDirect drivers:

1. Linux UNIX On Linux and UNIX, run the mqsimanagexalinks command.
2. Configure the broker's queue manager with XA resource manager information for each database that is involved in the transaction that the queue manager will globally coordinate. If the message flows reference message dictionaries, or contain Publication nodes, you must use the same method to define XA resource manager information for the broker database and for the user databases.

   Linux UNIX **On Linux and UNIX:**

   a. Open the queue manager's `qm.ini` file in a text editor. The `qm.ini` file is located at `/var/mqm/qmgrs/`*`queue_manager_name`*`/qm.ini`, where *queue_manager_name* is the name of the broker that is associated with the queue manager.

   b. At the end of the `qm.ini` file, paste the following stanza:

   ```
   XAResourceManager:
       Name=SYBASEXA
       SwitchFile=SwitchFileName
       XAOpenString=-NSYBASEDB -AMyServerName,MyPortNumber -Uuid -Ppwd -K2
       XACloseString=
       ThreadOfControl=THREAD
   ```

   c. The switch file is supplied by WebSphere Message Broker and varies by operating system. The following table shows the name of the switch file for each operating system.

Operating system	Switch file path
HP-UX on PA-RISC	`UKasedtc20.sl`
All other UNIX and Linux platforms (except Linux on System z)	`UKasedtc23.so`

   d. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:
      - *MyServerName* is the name of the TCP/IP host that hosts the Sybase ASE server.
      - *MyPortNumber* is the TCP/IP port on which the Sybase ASE server is listening.
      - *uid* must be the user name that the broker uses to connect to the database.

        You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

        1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.

2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.

3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

- *pwd* is the password that is associated with the user name.

e. Accept the default values for all the other lines in the stanza. For example:

- On AIX:

```
XAResourceManager:
    Name=SYBASEXA
    SwitchFile=UKasedtc23.so
    XAOpenString=-NSYBASEDB -Adiaz,1521 -Uwbrkuid -Pwbrkpw -K2
    XACloseString=
    ThreadOfControl=THREAD
```

- On HP-UX on PA-RISC:

```
XAResourceManager:
Name=SybaseXA
SwitchFile=UKasedtc20.sl
XAOpenString=-NSYBASEDB -Adiaz,1521 -Uwbrkuid -Pwbrkpw -K2
XACloseString=
ThreadOfControl=THREAD
```

On Windows:

a. From the **Start** menu, open WebSphere MQ Explorer.

b. Open the queue manager's Properties dialog box, then open **XA resource managers**.

c. In the **SwitchFile** field, enter the full path to the switch file, as shown in the following example where *install_dir* is the location in which the broker is installed:

```
install_dir\bin\ukase23.dll
```

d. In the **XAOpenString** field, paste the following string:

```
-NSYBASEDB -AMyServerName,MyPortNumber -WWinsock -Uuid -Ppwd -K2
```

e. In the **XAOpenString** field, replace the values with values that are appropriate for your configuration:

- *install_dir* is the location in which the broker is installed.
- *MyServerName* is the name of the TCP/IP host that hosts the Sybase ASE server.
- *MyPortNumber* is the TCP/IP port on which the Sybase ASE server is listening.
- *uid* must be the user name that the broker uses to connect to the database.

  You can define the user name that the broker uses in a number of ways; make sure you specify the correct name in this file. The broker determines the user name by checking the following conditions in the order listed:

  1) A specific user name and password for this data source name (DSN), that you have defined by running the mqsisetdbparms command.

  2) A default user name and password for all DSNs, that you have defined by specifying the **-u** parameter on the mqsicreatebroker command.

  3) The broker's service user name, which you define with the **-i** parameter on the mqsicreatebroker command

- *pwd* is the password that is associated with the user name.

For example:

```
-NSYBASEDB -Adiaz,1521 -WWinsock -Uwbrkuid -Pwbrkpw -K2
```

f. Accept the default values for all the other fields on the page.

3. Stop then restart the queue manager to apply the changes, because `qm.ini` is read only while the queue manager is running.

   To stop then restart the queue manager, enter the following commands, where *queue_manager_name* is the name of the queue manager:

   ```
   endmqm queue_manager_name
   strmqm queue_manager_name
   ```

   When the queue manager restarts, check the queue manager's log for any warnings that are associated with the restart. The log files are located in `/var/mqm/qmgrs/`*queue_manager_name*`/errors`, where *queue_manager_name* is the name of the queue manager that you restarted. When the queue manager restarts successfully, the changes that you made to `qm.ini` are applied.

Sybase is now configured for global coordination with your queue manager coordinating transactions.

Next: you can deploy globally coordinated message flows to the broker.

Configuring the workbench

You can configure a variety of settings in the workbench to suit your requirements and your working environment.

The following topics show you how to configure aspects of the workbench:

- "Changing workbench preferences"
- "Changing workbench capabilities" on page 307
- "Changing Broker Administration preferences" on page 308
- "Configuring CVS to run with the Message Broker Toolkit" on page 308
- "Configuring the Message Broker Toolkit to run Rational ClearCase" on page 309
- "Displaying selected projects in working sets" on page 309

A minimum display resolution of at least 1024 x 768 is required for some dialog boxes, such as the Preferences dialog box.

Changing workbench preferences

The workbench has a large number of preferences that you can change to suit your requirements. Some of these are specific to the product plug-ins that you have installed within the workbench, including those for WebSphere Message Broker. Others control more general options, such as the colors and fonts in which information is displayed.

To access the workbench preferences:

1. Select **Window** → **Preferences**.
2. Click the plus sign associated with **Workbench**, typically the first entry in the left pane. An expanded list of options appears, Select the aspect of the workbench that you want to modify. These options might be of interest:

   Startup and shutdown
   Switch on, or off, the prompt at toolkit startup that asks you to confirm the workspace location. Typically you switch this prompt off, so that it

does not appear, but you can force it to appear next time you start the workbench if you want to specify a different location.

You can also choose to display, or not display, the dialog that asks you to confirm shutdown of the workbench.

Colors and fonts
Change the default fonts and colors that appear in the workbench.

Perspectives
On this dialog, your choices include the option to open a new perspective in a new window.

3. When you have made your changes, click **OK** to close the Preferences dialog.

Below the Workbench category in the Preference dialog are items that refer specifically to WebSphere Message Broker resources, such as message flows. Review the following topics for information about setting preferences and other values that are specific to your use of these resources:

- Message flow preferences
- Changing ESQL preferences
- Configuring message set preferences
- Setting flow debugger preferences
- Changing trace settings

Changing workbench capabilities

You can configure the workbench to disable access to some of the functional capabilities of WebSphere Message Broker.

Capabilities is an Eclipse concept that allows you to enable or disable the components of a product. By default, all components of the workbench are enabled.

To access the workbench capabilities:

1. Select **Window** → **Preferences**.
2. Click the plus sign associated with **General**. An expanded list of options appears.
3. Click **Capabilities**. You can use the capabilities that are listed to enable or disable various product components; the capabilities are grouped according to a set of predefined categories.
4. Select **Message Broker Toolkit** from the list of capabilities that is displayed, and select the **Advanced** button. A window opens that has a check box for each of the predefined categories.
5. Select the check boxes for the categories that you want to either enable or disable; click either the **Enable All** or **Disable All** button and click **OK**. A pane describes the functionality that is enabled following this action.

The predefined categories for the workbench are listed below, together with a reference to more information about the relevant functional area of WebSphere Message Broker:

- **Message Broker Toolkit - Administration**. See "Administering the broker domain" on page 319.
- **Message Broker Toolkit - Core**. See Message Broker Toolkit.
- **Message Broker Toolkit - Development**. See Developing applications.

Changing Broker Administration preferences

Change preferences in the Broker Administration perspective.

You can change the following preferences to tailor how you work in the Broker Administration perspective:

- Show empty projects in the Navigator views.
- Warn before deleting alerts.
- Show ACL restricted objects in the Navigator views.
- 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 perspective 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**.

Configuring CVS to run with the Message Broker 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 Broker 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 Message 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 the Message Broker Toolkit to run Rational ClearCase

To use Rational® ClearCase with the Message Broker Toolkit, enable the capability in the Preferences page.

To enable Rational ClearCase in the Message Broker Toolkit:

1. Click **Window** → **Preferences** to open the Preferences window.
2. Expand **General** in the left pane, and click **Capabilities**.
3. In the Capabilities pane, click **Advanced**. The Advanced window opens.
4. Expand **Team**, and ensure that **ClearCase SCM Adapter** is selected.
5. Click **OK** to close the Advanced window.
6. Click **OK** or **Apply** to apply your changes.

After you enable the ClearCase capability, the ClearCase menu is displayed in the Broker Application Development perspective.

To work with ClearCase:

1. Click **ClearCase** → **Connect to Rational ClearCase**.
2. Right-click your project and click **Team** → **Add to Version Control** to add your projects to the ClearCase source control.
3. After you have added your projects to the ClearCase source control, you can perform ClearCase operations.

Displaying selected projects in working sets

Before you start:

Read the information about working sets in the concept topic about Resources.

A *working set* is a logical collection of application projects, that you can use to limit the number of resources that are displayed in the Broker Development view. Creating and using a working set allows you to reduce the visual complexity of what is displayed in the Broker Development view, making it easier to manage and work with your application projects.

To create a new working set:

1. Click on the down arrow to the right of the **Active working set** field in the Broker development view. By default this field contains `<all resources>`. A list is displayed containing existing working sets (if there are any) and options for editing and deleting existing working sets, and for creating a new working set.
2. Click **New Working Set**. The New Working Set window is displayed.

3. Type the name that you want to give to your new working set in the **Working set name** field.
4. Select from the displayed list each of the application projects that you want to include in this working set. You can also include all of the projects that are dependent on your selected application projects, by selecting **Automatically include dependent projects in this working set**.
5. Click **Finish**.

The new working set and its associated resources are displayed in the Broker Development view.

In addition to creating new working sets, you can also select, edit, and delete existing working sets using the options in the Broker Development view menu.

Changing locales

You can change the locale for the system on which a runtime component is installed.

The way in which you change the locale depends on the operating system:
- "Changing your locale on UNIX and Linux systems"
- "Changing your locale on Windows" on page 312
- "Changing your locale on z/OS" on page 313

WebSphere Message Broker uses code page converters to support character sets from different environments. "Code page converters" on page 313 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.

You can set environment variables to control the system locale. These can be defined in your environment to be system-wide, 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, which 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 catalogs

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 Message 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
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.

AIX If AIX Version 5.3 is installed on the computer on which you have created the broker, the TZ (timezone) environment variable might not work correctly in all locales. To overcome this situation, see the latest information available in techdoc AIX Version 5.3 timezone support (you must be connected to the Internet to access this document).

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 Message Broker you must add it by "Generating a new code page converter" on page 314.

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 Message 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 184 and "Installation information - Configuration Manager" on page 198 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 Message Broker, install that locale's message catalogs from the original install media.

You can set WebSphere Message 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 Message 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

Generate a code page converter to handle conversions of data that belongs to a code page that is not in the default set of code pages provided by WebSphere Message Broker.

Before you start:

- Read "Code page converters" on page 313, which provides information about what a code page converter is, and about the code pages that WebSphere Message Broker supports.

To generate a new code page converter:

1. Create or find a mapping data file with the file extension `.ucm` for the converter that you require. You can download `.ucm` files from the ICU Character set mapping files archive. These mapping data files are available and can be modified 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 that you choose to identify the code page. Make sure that this number is not already used in one of the Supported code pages. For example, you could rename `ibm-1284_P100-1996.ucm` to `ibm-1284.ucm`.
3. Go to ICU downloads and download the binary distribution for your system. An exact match is not important provided that the binary files are compatible. If you have problems building the converter, see the ICU user guide.
4. Extract the files from the binary distribution archive into 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. One of the extracted files is `makeconv.exe`; use this makeconv tool to convert the mapping data file (`.ucm` files) into a binary converter file (`.cnv` file), by entering the following command:

   ```
   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 values:
     - l for little-endian ASCII platforms
     - b for big-endian ASCII platforms
     - e for EBCDIC 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 that you need, into a directory that WebSphere Message Broker can access; for example, on UNIX: `/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 that is associated with the converter, include the **-c** parameter on the mqsicreatebroker command.
   - To alter an existing broker to recognize the converter, include the **-c** parameter on the mqsichangebroker command.
   - To affect all the products and the broker command-line tools that are using ICU, add the *directory* to the **ICU_DATA** environment variable. If you have already used either the mqsicreatebroker command or the mqsichangebroker command to specify the code page converter to be used, the broker ignores the **ICU_DATA** value.

   If you are using a converter that matches one of the built-in converters that are provided with Version 6.0, and that converter is the local code page for the broker, do not use the mqsicreatebroker command with the **-c** parameter to set the converter path. Use the **ICU_DATA** environment variable instead.

Using converters from a previous level of the product

If you have applications that need a code page that is not in the default set of code pages that WebSphere Message Broker Version 6.1 supports, you can use a code page from an earlier version of WebSphere Message Broker.

Before you start:

- Read "Code page converters" on page 313, which provides information about what a code page converter is, and about the code pages that WebSphere Message Broker supports.

The changes in converters between WebSphere Business Integration Message Broker Version 5.0 and WebSphere Message Broker Version 6.1 are significant, therefore the set of converters from the previous level has been included with WebSphere Message Broker Version 6.1.

To use one of the converters from the previous version:

1. Extract the list of WebSphere Business Integration Message Broker Version 5.0 code page converters from your installation directory to a temporary directory:
   - On Windows: `extract` *`install_dir`*`\sample\converters\mqsiconverters-v5.zip`
   - On Linux: `extract` *`install_dir`*`/sample/converters/mqsiconverters-v5.tar.bz2`
   - On UNIX: `extract` *`install_dir`*`/sample/converters/mqsiconverters-v5.tar.gz`

   where *install_dir* is the directory you have specified for your WebSphere Message Broker installation.
2. Find the `.cnv` file for the code page that you want in the temporary directory, and copy it to a directory that is accessible by the broker. Give the file a unique name to make sure that the copied file does not conflict with an existing converter; do not use a number that is already used in one of the supported code pages (Supported code pages); for example:
   - Linux UNIX On Linux and UNIX systems, copy the file to `/var/mqsi/converters`.

- Windows On Windows systems, copy the file to `%ALLUSERSPROFILE%\Application Data\IBM\MQSI\converters`, where `%ALLUSERSPROFILE%` is the environment variable that defines the system working directory. The default directory depends on the operating system:
  - On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\MQSI\converters`
  - On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\MQSI\converters`

  The value might be different on your computer.

If the converter is used by ESQL, the converter must be of the form `ibm-<ccsid>`, because converters are referenced through their numeric CCSID, not their name.

3. Associate the broker with the code page converter by entering the name of the directory where you have stored the converter.

   If you are using a converter that matches one of the built-in converters that are provided with Version 6.1, and that converter is the local code page for the broker, do not use the mqsicreatebroker or mqsichangebroker command with the **-c** parameter to set the converter path. Use the **ICU_DATA** environment variable instead.

   - To create a new broker that is associated with a converter, include the **-c** parameter on the mqsicreatebroker command.
   - To alter an existing broker to recognize the converter, include the **-c** parameter on the mqsichangebroker command; for example:

     ```
     mqsichangebroker broker_name -c directory
     ```
   - To affect all the products and the broker command-line tools that use international components for Unicode (ICU), add the directory to the **ICU_DATA** environment variable.

   If you have already used either the mqsicreatebroker or mqsichangebroker command to specify the code page converter to be used, the broker ignores the **ICU_DATA** value.

   To reproduce the behavior of the previous level of the product, copy the entire set of converters (`*.cnv`) and aliases (`*.icu`).

Part 3. Administering the broker domain

Administering the broker domain

Administering the broker domain includes the tasks that you operate frequently to activate and run your broker domain. Choose the method you prefer to administer your broker domain.

Administration of a broker domain includes the following tasks:

- "Connecting to and disconnecting from the broker domain"
- "Connecting to and disconnecting from the broker domain on z/OS" on page 320
- "Starting and stopping message flows" on page 322
- "Starting and stopping a broker" on page 323
- "Starting and stopping a Configuration Manager" on page 325
- "Starting and stopping the User Name Server" on page 327
- "Starting a WebSphere MQ queue manager as a Windows service" on page 329
- "Stopping a WebSphere MQ queue manager when you stop a component" on page 330
- "Viewing broker domain log information" on page 330
- "Refreshing broker domain log information" on page 331
- "Filtering broker domain log information" on page 331
- "Saving broker domain log information" on page 332
- "Clearing broker domain log information" on page 333
- "Changing the location of the work path" on page 333
- "Changing Event Log editor preferences" on page 334
- "Backing up resources" on page 335

These tasks can be performed by using one, or more, of the administrative techniques supported by WebSphere Message Broker:

- The Message Broker Toolkit
- The WebSphere Message Broker commands
- The Configuration Manager Proxy Java API

For each task, the administrative techniques that you can use are identified.

Connecting to and disconnecting from the broker domain

You can connect to, and disconnect from, the broker domain by using either the Message Broker Toolkit or the Configuration Manager Proxy Java API.

Before you start:

You must complete the following task:

- "Creating a domain connection" on page 251

This topic describes how to use the Message Broker Toolkit to connect to the broker domain. 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`.

      Note: If you click **Cancel** while the connection is being attempted, the connection that is in progress stops and the domain returns to its initial unconnected state.
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.

Connecting to and disconnecting from the broker domain on z/OS

How to connect to, and disconnect from, the broker domain on z/OS.

Before you start:

You must complete the following tasks:

- "Creating a Configuration Manager on z/OS" on page 197.
- Create and start a listener for the Configuration Manager. For details on how to create and start a listener, follow the instructions for listeners in the topic: "Starting the WebSphere MQ channels and listeners" on page 214.

Ensure that your Message Broker Toolkit computer and user ID have the appropriate authorization on the z/OS Configuration Manager.

In SDSF, grant access to your user ID.

1. For this to work on all machines, enter:

   ```
   '/F <started task name> CA U=<userID>,A=YES,P=YES,X=F'
   ```

   or to grant access to your user ID for a specific machine, enter:

   ```
   '/F <started task name> CA U=<userID>,A=YES,M=<machine name>,P=YES,X=F'
   ```
2. Verify the previous step by entering:

```
/F <configmgrname>,LA
```

This topic shows you how to:

- Create a domain connection in the workbench by using the Create a Domain Connection wizard.
- Enter a set of parameters to create a `.configmgr` file.
- Use the parameters contained within the `.configmgr` file 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. In the Domains view, right-click the default Configuration Manager.
3. Click **New** → **Domain Connection** to open the Create a Domain Connection wizard.
4. In the Create a Domain Connection wizard, enter:
   a. The value for the **Queue Manager Name** 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 **Class** of the Security Exit required to connect to the WebSphere MQ queue manager. This property must be a valid Java class name, but you can leave this field empty if it does not apply to your domain connection. See "Using security exits" on page 71.
   e. Optional: The **JAR File Location** for the Security Exit required to connect to the WebSphere MQ queue manager. Click **Browse** to find the file location. You can leave this field empty if it does not apply to your domain connection. You must specify a **JAR File Location** if you have specified a Security Exit **Class**.
   f. Optional: The **Cipher Suite**, **Distinguished Names**, **CRL Name List**, **Key Store**, and **Trust Store** parameters are required when enabling SSL. See "Implementing SSL authentication" on page 57. The **Cipher Suite** field displays available cipher suites. Click **More** to configure Custom SSL Cipher Suites in the Broker Administration Preferences window. If you do not specify a **Cipher Suite**, 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 Name**, **Host** name, or TCP **Port** number. An error message is displayed in the Create a Domain Connection wizard if you try to create a second broker domain that has the same **Queue Manager Name**, **Host** name, and **Port** number.
5. Click **Next** to begin the domain connection to the Configuration Manager.
6. If you click **Cancel**, the Create a Domain Connection wizard closes, forcing disconnection from the domain.
7. After the domain connection has been made, enter the following values:
   a. The name of your **Project**. The Project is the container for your domain connection. If you have not already created a project, you can specify the name of a new project here. The project is created with the domain connection.

b. The **Connection name**. The Connection name is the name you give to the `.configmgr` file that contains the parameters to connect to the Configuration Manager.

8. Click **Finish** to create the domain connection.

   The new domain connection is added to the Broker Administration perspective Navigator view, under Domain Connections. The project holds the `.configmgr` domain connection file.

   The view of the broker domain is displayed in the Domains view.

Starting and stopping message flows

Use the Message Broker Toolkit to start and stop message flows.

You can start and stop a message flow by using either the Message Broker Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Broker 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

Run the appropriate command to start or stop 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 723.
- On Linux, UNIX, and Windows systems, you must set up your command-line environment before performing this task, by running the product profile or console; refer to Setting up a command environment.

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"
- "Starting and stopping a broker on z/OS" on page 324

Starting and stopping a broker on Linux and UNIX systems

Run the appropriate command to start or 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 Message Broker commands.
2. To start a broker, 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 system log to ensure that the broker has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If any verification test is unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.
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 330.

Starting and stopping a broker on Windows

Run the appropriate command to start or stop a broker.

1. Open the WebSphere Message Broker command console. When you open the console, it sets up the environment that you need to run the WebSphere Message Broker commands.

   If you prefer, you can run the *install_dir*/bin/mqsiprofile command to set up the environment.
2. To start a broker, enter the following command on the command line:

   ```
   mqsistart WBRK_BROKER
   ```

   Substitute your own broker name for WBRK_BROKER.

   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 329.

   The broker and its associated queue manager are started. The command initiates the startup of the broker's Windows service.

   Check the Application Log in the Event Viewer to ensure that the broker has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If any verification test is unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.
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 330.

Starting and stopping a broker on z/OS

Run the appropriate command from SDSF to start or stop a broker.

1. Start the component by using the command /S <broker name>. This command produces the following output, where MQP1BRK is the name of the broker:

```
+BIP9141I MQP1BRK 0 The component was started
```

   Substitute your own broker name for MQP1BRK.

   The verification step runs, followed by starting the control process and any DataFlowEngine address spaces.

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process and DataFlowEngine address spaces are not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.
2. Alternatively, start the control process only by using the command:

   ```
   /S broker_name,STRTP=MAN
   ```

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process is not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

   No DataFlowEngine address spaces are started automatically when specifying STRTP=MAN. If the verification step is successful and the control process starts successfully, fully start the broker by issuing the console command:

   ```
   /F <broker name>, SC
   ```
3. To stop a broker, run the following command:

   ```
   /P <broker name>
   ```

Starting and stopping a Configuration Manager

Run the appropriate command to start or stop 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 723.
- On Linux, UNIX, and Windows systems, you must set up your command-line environment before performing this task, by running the product profile or console; refer to Setting up a command environment.

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"
- "Starting and stopping a Configuration Manager on Windows"
- "Starting and stopping a Configuration Manager on z/OS" on page 326

Starting and stopping a Configuration Manager on Linux and UNIX systems

Run the appropriate command to start or stop a Configuration Manager.

Use the mqsistart and mqsistop commands from the command line.

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 Message Broker commands.
2. To start aConfiguration Manager, enter the following command on the command line:

   ```
   mqsistart CMGR01
   ```

   Substitute your own Configuration Manager name for CMGR01. The Configuration Manager and its associated queue manager are started.

   Check the system log to ensure that the Configuration Manager has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If any verification test is unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.
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 queue manager that hosts the Configuration Manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 330.

Starting and stopping a Configuration Manager on Windows

Run the appropriate command to start or stop a Configuration Manager.

Use the mqsistart and mqsistop commands from the command line.

1. Open the WebSphere Message Brokercommand console. When you open the console, it sets up the environment that you need to run any of the WebSphere Message Broker commands.

   If you prefer, you can run the *install_dir*/bin/mqsiprofile command to set up the environment.

2. To start a Configuration Manager, enter the following command on the command line:

   ```
   mqsistart 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 queue manager associated with the Configuration Manager is started as a Windows service. See "Starting a WebSphere MQ queue manager as a Windows service" on page 329. The Configuration Manager and its associated queue manager are started. The command initiates the startup of the Windows service for the Configuration Manager.

   Check the Application Log in the Event Viewer to ensure that the Configuration Manager has initialized successfully. The log contains messages about verification procedures; if all tests are successful, only an initial start message is recorded. If any verification test is unsuccessful, the log also includes messages that provide details of the tests that have failed. If errors have been reported, review the messages and take the suggested actions to resolve these problems.

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 name `configmgr` is used.

   You can also request that the queue manager that hosts this Configuration Manager is stopped by this command. See "Stopping a WebSphere MQ queue manager when you stop a component" on page 330.

Starting and stopping a Configuration Manager on z/OS

Run the appropriate command from SDSF to start or stop a Configuration Manager.

1. Start the component by using the command `/S <Configuration Manager name>`. This command produces the following output where CMGR01 is the name of the Configuration Manager.

```
+BIP9141I CMGR01 0 The component was started
```

   Substitute your own Configuration Manager name for CMGR01.

   The verification step runs, followed by starting the control process and the Configuration Manager process.

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process and Configuration Manager process are not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

2. Alternatively, start the control process only by using the command:

   ```
   /S configmgr_name,STRTP=MAN
   ```

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process is not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

The Configuration Manager process is not started automatically when specifying STRTP=MAN. If the verification step is successful and the control process starts successfully, fully start the Configuration Manager by issuing the console command:

```
/F <Configuration
Manager name>, SC
```

3. To stop a Configuration Manager, run the following command:

   ```
   /P CMGR01
   ```

   Substitute your own Configuration Manager name for CMGR01.

Starting and stopping the User Name Server

Run the appropriate command to start or stop 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 723.
- On Windows, UNIX systems, and Linux, you must set up your command-line environment before performing this task, by running the product profile or console; refer to Setting up a command environment.

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" on page 328
- "Starting and stopping the User Name Server on z/OS" on page 328

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 Message 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. If errors have been reported, review the messages and take the suggested actions to resolve these problems.
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 330.

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 Message 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. If errors have been reported, review the messages and take the suggested actions to resolve these problems.

      You can also request that the queue manager that hosts the User Name Server is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service" on page 329.
3. To stop a User Name Server enter the following command on the command line:

   ```
   mqsistop usernameserver
   ```

   You can also request that the queue manager that hosts the User Name Server is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 330.

Starting and stopping the User Name Server on z/OS

Run the appropriate command from SDSF to start or stop a User Name Server.

1. Start the component by using the command /S <User Name Server name>. This command produces the following output where MQP1UNS is the name of the User Name Server:

```
+BIP9141I  MQP1UNS 0 The component was started
```

   Substitute your own User Name Server name for MQP1UNS.

   The verification step runs, followed by starting the control process and the User Name Server process.

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process and User Name Server process are not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.
2. Alternatively, start the control process only by using the command:

   ```
   /S usernameserver_name,STRTP=MAN
   ```

   If the verification step fails for any reason, the errors are reported to the STDOUT stream in the JOBLOG; the control process is not started. Review the messages to see what errors have been reported, and take the suggested actions to resolve these problems.

   The User Name Server process is not started automatically when specifying STRTP=MAN. If the verification step is successful and the control process starts successfully, fully start the User Name Server by issuing the console command:

   ```
   /F <User Name Server name>, SC
   ```
3. To stop a User Name Server, use the command:

   ```
   /P <User Name Server name>
   ```

Starting a WebSphere MQ queue manager as a Windows service

On Windows, you can start a WebSphere MQ queue manager as a Windows service to ensure the queue manager starts when you start your other components.

Before you start:

You must complete the following task:

- Stop the queue manager for the WebSphere Message Broker component, using the endmqm command. If you prefer, you can use WebSphere MQ Version 5 Services or WebSphere MQ Version 6 Explorer.

When you start a WebSphere Message Broker component (broker, Configuration Manager, or User Name Server), the "mqsistart command" on page 654 starts the associated queue manager if it is not already running.

When you start any of these components on Windows, the component starts as a service on Windows, but the associated queue manager does not.

You can change the properties of the queue manager service to set the 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 Message Broker component.

To start a WebSphere MQ Version 5 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*. This setting ensures that the queue manager is started whenever the WebSphere MQ Service (a Windows service) is started.
4. (Optional) 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 and isolates the operation of the WebSphere MQ Services service from any logged on user.
5. Restart the queue manager for the WebSphere Message Broker component using the strmqm command or WebSphere MQ Services. The changes to the queue manager's startup type take effect when you restart Windows.
6. Start the component using the "mqsistart command" on page 654.

To start a WebSphere MQ Version 6 queue manager as a Windows service:

1. Click **Start** → **Programs** → **IBM WebSphere MQ** → **WebSphere MQ Explorer**.
2. In the left pane, right-click the queue manager and select **Properties**. The Properties dialog opens. The General properties are displayed.
3. Find the Startup property and set it to *Automatic*.
4. Click **OK**. The Properties dialog closes and the change is applied.
5. Restart the queue manager for the WebSphere Message Broker component using the strmqm command or WebSphere MQ Explorer. The changes to the queue manager's startup type take effect when you restart Windows.
6. Start the WebSphere Message Broker component using the "mqsistart command" on page 654.

Stopping a WebSphere MQ queue manager when you stop a component

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 Message Broker component.

If you are using a single queue manager to support more than one WebSphere Message 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), 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.

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 broker.
-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 using either the Broker Administration perspective or the CMP API.

Follow the instructions in this topic to use the Broker Administration perspective. If you prefer to use the CMP API, see Developing applications that use the Configuration Manager Proxy API and Configuration Manager Proxy API.

Broker domain log information is written to the Event Log editor in the Broker Administration perspective.

The Event Log editor 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. Time stamps 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.

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 show 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," a note appears next to the view label to indicate that a filter has been applied.

Refreshing broker domain log information

Refresh the log that records the events that occur in the broker domain and are shown in the Event Log editor.

New messages display automatically in the Event Log. However, you can refresh the Event Log at any time:

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain to display its components.
3. Double-click on **Event Log** to open the Event Log editor in the editor area.
4. Right-click in the **Logs** pane and click **Revert**. **Revert** hides only existing messages; this option does not remove or overwrite existing messages in the Event Log editor, or in the log in the Configuration Manager.

Filtering broker domain log information

Filter records in the log to view information about specific resources in the broker domain.

The Event Log, which you can view in the workbench, contains information about events that occur in a broker domain. For each event, the Event log records the following details:

- Event type (whether the event is information, an error, or a warning)
- Event source (what caused the event, and where it originates within the broker domain, for example the Configuration Manager, and specified brokers
- Event time stamp (the date and time when the event occurred)

You can filter on the type, the source and the time stamp of the message, to restrict the number of log entries that are displayed in the Event Log editor. You can also filter events, to view a particular set of events. The filter settings that you define are kept for your next session. A note also appears next to the view label to indicate that a filter has been applied.

To filter entries, use the Filter Event Log dialog:

1. Switch to the Broker Administration perspective.
2. In the Domains view, open the Event Log for the appropriate broker domain.

3. From the Event Log editor menu, click **Event Log editor> Filter Log** The Filter Log dialog opens.
4. In the Filter Log dialog, use the following controls to apply the required filter:
   - Filter by message type (information, error, or warning).

     Select one or more of the three message types.
   - Filter by message source.

     Select one or more of the possible message sources for all known entries so far. If you click **Deselect All**, all sources are cleared.

     To view all sources click **Select All**. All events with the selected sources are included in the filter dialog table of entries.
   - Filter by time stamp.

     Click **Hide Events generated before** and select from the time stamps of all entries generated to date. The log hides messages that were generated before the specified time stamp.
   - Select one or more log entries from those displayed in the table; use the vertical scroll bar to access additional entries.
   - Below the table, click **Select All** to select all events, or **Deselect All** to deselect all events from this table.
   - Click **Restore Defaults** to reset all the options shown to their default values.

   In the default view of the Log Filter, all message types are selected, indicating that messages of all three types are displayed. All message sources are also selected, indicating that messages of all sources are displayed. The default time stamp is the one for the oldest known event so far. All events are selected in the table.

   You can combine filtering options (type, source, time stamp). The combinations that you select are identified in the table; entries are automatically selected or cleared to indicate the choices that you make. The editor is also updated based on the tables; selected entries are displayed, cleared entries are hidden.

   For incoming new events, the filtering options based on type, source, and time stamp are applied automatically.
5. To save the current filter settings, click **OK**. Event filtering is applied, based on your new settings, and the editor is refreshed.
6. To discard all changes before you save them, click **Cancel**.

Saving broker domain log information

Save the broker domain log information that is written to the Event Log Editor in the workbench.

Event Log information is deleted automatically from the Configuration Manager after 72 hours. You can save the log contents to file if you want to retain them:

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. Enter an appropriate directory in which to save the log information.

   The default file name is `log.txt`. However, you can change the name of this text file.

   You can also save the file in 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 saved log, open the `log.txt` file in an appropriate text editor.

Clearing broker domain log information

Clear broker domain log information to reduce the size of the log by using either the workbench or the CMP API.

Follow the instructions in this topic to use the workbench to clear the log. If you prefer to use the CMP API, see Developing applications that use the Configuration Manager Proxy API and Configuration Manager Proxy API.

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 the location of the work path

The work path directory is the location where a component stores internal data, such as installation logs, component details, and trace output. The shared-classes directory is also located in the work path directory and is used for deployed Java code. If the work path directory does not have enough capacity, redirect the directory to another file system that has enough capacity.

The work path is fixed at installation time so that WebSphere Message Broker can always find the information that it needs, and always knows where to store new information.

If you need to change the location (for example, if you do not have enough capacity on the automatically-designated file system), do not change the path to the directory; instead, redirect the old work path directory to a new location.

Changing the location of the work path on Windows systems

When you change the location of the work path, you mount the new partition at the location of the old work path directory.

To change the location of the work path on Windows:

1. Shut down all WebSphere Message Broker services and processes.
2. Create a new partition on the system. The new partition can be on the same drive as the old work path, or on a different drive.
3. Locate the work path directory for your installation on the local system by running the following command:

   ```
   echo %MQSI_WORKPATH%
   ```

4. Copy the contents of the work path directory to the new partition.
5. Delete the contents of the old work path directory.
6. Open the Computer Management dialog: click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Computer Management**; the Computer Management dialog opens.
7. In the left pane of the Computer Management dialog, click **Disk Management**. The new partition that you added, and any existing partitions, are listed in the right pane.
8. Right-click the new partition, then click **Change Drive Letter and Paths**. The Change Drive Letter and Paths dialog opens.
9. Click **Add**. The Add Drive Letter or Path dialog opens.
10. Ensure that **Mount in the following empty NTFS folder** is selected, then browse to the old work path location.
11. Click **OK**, then click **OK** again.

Any files that WebSphere Message Broker creates in the work path location are stored on the new partition.

Changing the location of the work path on Linux and UNIX systems

When you change the location of the work path, you can either mount the new partition at the location of the old work path directory, or you can replace the old work path directory with a soft link that points to the new work path directory.

To change the location of the work path on UNIX and Linux:

1. Shut down all WebSphere Message Broker services and processes.
2. Create a new directory on a suitable file system.
3. Locate the work path directory for your installation on the local system by running the following command:
   ```
   echo $MQSI_WORKPATH
   ```
4. Copy the contents of the work path directory to the new partition.
5. Delete the contents of the old work path directory.
6. Perform one of the following tasks so that the WebSphere Message Broker installation uses the new work path location:
   - Use the mount command to mount the new work path directory at the location of the old work path directory.
   - Delete the old work path directory and replace it with a soft link. Give the soft link the same name as the old work path directory and point the link to the new work path directory.

Any files that WebSphere Message Broker creates in the work path location are stored in the new location.

Changing Event Log editor preferences

You can change preferences for the Event Log editor using the Event Log editor preferences page.

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

Establish a backup process to preserve the integrity and consistency of your broker domain, and provide a mechanism to restore your components and 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.

It is important that you maintain regular backups of your broker domain and associated databases. You should refer to the information supplied with the database that you are using for details of the relevant database backup procedures.

Consult your database administrator and agree upon the following questions:

- Frequency of backups
- Quiesce backup points to take

Depending upon your workflow, these actions can be hourly, daily, or weekly.

You should plan to be always in a position to recover to a specific point in time, whatever happens. For example, take a backup of the broker domain, and quiesce of the broker databases, before you install a new application.

The following topics tell you how to back up and restore brokers, the Configuration Manager and the Message Broker Toolkit workspace:

- "Backing up the broker domain on distributed systems"
- "Backing up the broker domain on z/OS" on page 337
- "Backing up the Message Broker Toolkit workspace" on page 339

Backing up the broker domain on distributed systems

You can back up a broker domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. Back up broker domain resources, and plan for restoration of every broker that is deployed to by the Configuration Manager.

For more information about carrying out these steps, see the links at the end of this topic.

Backing up components

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.

   For example, for a DB2 broker database use the Backup wizard in the DB2 Control Center, or a command similar to:

   ```
   DB2 BACKUP DATABASE <broker db> TO "<backup directory>"
   ```

5. Back up the system work path.

   The work path is platform-specific:

   - Windows On Windows, the directory is `%ALLUSERSPROFILE%\Application Data\IBM\MQSI` where `%ALLUSERSPROFILE%` is the environment variable that defines the system working directory. The default directory depends on the operating system:
     - On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\MQSI`
     - On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\MQSI`

     The actual value might be different on your computer.
   - Linux UNIX On other distributed platforms, the directory is:

     `/var/mqsi`

   Also, back up broker-specific work paths that you have specified with the **-w** flag on the mqsicreatebroker command.

Restoring components

To restore the components:

1. **Stop and remove the existing components in the Configuration Manager domain.**
   a. Disconnect from the domain in the Message Broker Toolkit.
   b. Stop each broker.
   c. Stop the Configuration Manager.
   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 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.
2. **Recreate the components.**
   a. Create the Configuration Manager.
   b. Create each broker.
3. **Restore the components.**
   a. Restore any work paths.

b. If you are restoring a Configuration Manager that was backed up on z/OS, restore the Configuration Manager repository using the mqsirestoreconfigmgr command.

   Replace the previously backed-up `service.properties` file.

c. Restore each broker database.

   For example, for a DB2 broker database use the Restore wizard in the DB2 Control Center, or a command similar to:

   ```
   DB2 RESTORE DATABASE <broker db> FROM "<backup directory>" TAKEN AT <datetime>
   ```

d. Start the Configuration Manager.

e. Start each broker.

f. Connect to the Configuration Manager in the Message Broker Toolkit. This action re-imports the broker topology, excluding execution groups and flows, from the Configuration Manager.

g. Deploy the topology configuration in the Message Broker Toolkit. This deployment causes the Configuration Manager to give the UUIDs to the brokers. Note that if you are working on a platform other than Windows, this step is unnecessary.

Backing up the broker domain on z/OS

You can back up a broker domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. Back up broker domain resources, and plan for restoration, on every system on which you have installed and created a broker or other broker domain component.

For more information about carrying out these steps, see the links at the end of this topic.

Backing up components

To back up the components:

1. Stop each broker.
2. Record the BrokerUUID value from the following file: `<broker directory>/registry/<broker name>/CurrentVersion/BrokerUUID`.
3. Stop the Configuration Manager.
4. If you plan to restore the Configuration Manager data repository on distributed systems, take a copy of the file:

data directory/components/*Configuration Manager name*/<directory name>/service.properties

The *data directory* is platform-specific:

- Windows On Windows, the directory is `%ALLUSERSPROFILE%\Application Data\IBM\MQSI` where `%ALLUSERSPROFILE%` is the environment variable that defines the system working directory. The default directory depends on the operating system:
  - On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\MQSI`
  - On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\MQSI`

  The actual value might be different on your computer.
- Linux UNIX On other distributed platforms the directory is:

  `/var/mqsi`

Keep your saved copy of this file with the `.zip` file produced by the mqsibackupconfigmgr command, and must be copied to the equivalent place in the restored Configuration Manager data repository, after running the mqsirestoreconfigmgr command.

5. Back up the Configuration Manager data repository using the mqsibackupconfigmgr command, or JCL job `BIPBUCM`.
6. Back up each broker database by using the JCL job BIPBUDB.

   The BIPBUDB job contains the following variables:

Variable	Description
++BACKUPHLQ++	The high level qualifier of the template for the data set name.
++TEMPLATENAME++	The name of the data set allocation template.

For more information, see the *DB2 UDB for z/OS Utility Guide and Reference*, SC18-7427.

Restoring components

To restore the components:

1. **Stop and remove the existing components in the Configuration Manager domain.**
   a. Disconnect from the domain in the Message Broker Toolkit.
   b. Stop each broker.
   c. Stop the Configuration Manager.
   d. Delete each broker.
   e. Delete the Configuration Manager.
2. **Recreate the components.**
   a. Create the Configuration Manager repository.
   b. Create each broker.
3. **Restore the components.**
   a. Restore the Configuration Manager repository using the mqsirestoreconfigmgr command, or JCL job `BIPRSCM`.
   b. Restore each broker database using the JCL job BIPRSDB.

      The BIPRSDB job contains the following variable:

Variable	Description
++TOLOGPOINTVALUE++	This variable specifies a point on the log to which RECOVER is to recover. Specify either an RBA or an LRSN value.

      For more information, see the *DB2 UDB for z/OS Utility Guide and Reference*, SC18-7427.

   c. Set the BrokerUUID by editing the following file: `<broker directory>/registry/<broker name>/CurrentVersion/BrokerUUID`.
   d. Start the Configuration Manager.
   e. Start each broker.
   f. Connect to the Configuration Manager in the Message Broker Toolkit. This action re-imports the broker topology, excluding execution groups and flows, from the Configuration Manager.

g. Deploy the topology configuration in the Message Broker Toolkit. This deployment causes the Configuration Manager to give the UUIDs to the brokers.

Backing up the Message Broker Toolkit workspace

The Message Broker Toolkit workspace contains your personal settings and data, such as message flow and message set resources. You can have multiple workspaces in different locations and you can also have references to projects that are in other locations, therefore consider all of these locations when you back up your resources.

- Windows On Windows XP and Windows Server 2003, the default workspace directory is created at `C:\Documents and Settings\user_ID\IBM\wmbt61\workspace`.
- Windows On Windows Vista and Windows Server 2008, the default workspace directory is created at `C:\Users\user_ID\IBM\wmbt61\workspace`.
- Linux On Linux, the default workspace directory is created at `/home/user_ID/IBM/wmqi61/workspace`.

where *user_ID* is the user name with which you are logged on.

The workspace directory contains a directory called `.metadata`, which contains your personal settings and preferences for the workbench. If the `.metadata` directory gets corrupted, you lose these settings, and the workbench reverts to the default layout and preferences. If you have not backed up the `.metadata` directory, you must manually set any preferences again and import any projects, such as message flow projects, that were displayed in the Broker Development view. To back up the `.metadata` directory, take a copy of the directory.

In the workspace, a directory exists for each project (for example, a message flow project) that you have created in the workbench. These directories contain your data, which you must back up.

Use one of the following methods to back up the data in your workspace:

- Export the projects from within the workbench. You can export the projects directly as a compressed file. For instructions, see Exporting in the Eclipse Workbench User Guide.
- Copy the project directories from the workspace directory to another location.
- When you add resources to a broker archive (BAR) file that is ready for deployment, select **Include source files** which adds the message flow and message set source files, as well as the compiled files, to the BAR file. Take copies of the BAR file to back up its contents.

If you want to restore the resources, copy the directories back into your workspace directory and import the projects. For instructions, see Importing in the Eclipse Workbench User Guide.

Part 4. Reference

Databases

Sample ODBC definition files are supplied to help you define an ODBC connection to a database from your brokers or applications.

For copies of the sample ODBC definition files that are supplied with WebSphere Message Broker, see the following topics:

- "odbc32.ini sample file"
- "odbc64.ini sample file" on page 351

odbc32.ini sample file

A copy of the sample 32-bit ODBC definition file that is supplied with WebSphere Message Broker.

Purpose

Configure the `odbc32.ini` file when defining an ODBC connection to a database from a broker or from a 32-bit application. Follow the instructions in "Connecting to a database from Linux and UNIX systems: 32-bit considerations" on page 132.

AIX

```
;###############################################################################
;########## 32 bit ODBC database driver manager initialization file ##############
;###############################################################################
;# It is recommended that you take a copy of this file and then edit the copy.     #
;#                                                                                 #
;#                                                                                 #
;# 1. Complete the 'Mandatory information stanza' section at the end of the file.  #
;#                                                                                 #
;#                                                                                 #
;# 2. For each data source, add the name of the data source into the 'List of      #
;#    data sources stanza' section.                                                #
;#                                                                                 #
;#                                                                                 #
;# 3. For each data source, create a stanza in the 'Individual datasource stanzas' #
;#    section.                                                                     #
;###############################################################################

;########################################
;###### List of data sources stanza #######
;########################################
[ODBC Data Sources]
DB2V8DB=IBM DB2 Version 8 ODBC Driver
DB2V9DB=IBM DB2 Version 9 ODBC Driver
ORACLEDB=DataDirect 5.3 Oracle Driver
ORACLERACDB=DataDirect 5.3 Oracle Driver (Real Application Clusters)
SYBASEDB=DataDirect 5.3 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;########################################################
;########## Individual data source stanzas ################
;########################################################
```

```
;# DB2 version 8 stanza
[DB2V8DB]
Driver=<Your DB2 v8 install directory>/lib/libdb2.a
Description=DB2V8DB DB2 ODBC Database
Database=DB2V8DB

;# DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.a
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name defined for the RAC>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQL Stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
```

```
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1

;# Informix stanza
[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.so
Description=IBM Informix ODBC Driver
ServerName=<YourServerName>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

HP-UX on PA-RISC

```
;###############################################################################
;########## 32 bit ODBC database driver manager initialization file ##############
;###############################################################################
;# It is recommended that you take a copy of this file and then edit the copy.     #
;#                                                                                 #
;#                                                                                 #
;# 1. Complete the 'Mandatory information stanza' section at the end of the file.  #
;#                                                                                 #
;#                                                                                 #
;# 2. For each data source, add the name of the data source into the 'List of      #
;#    data sources stanza' section.                                                #
;#                                                                                 #
;#                                                                                 #
;# 3. For each data source, create a stanza in the 'Individual datasource stanzas' #
;#    section.                                                                     #
;###############################################################################

;########################################
;###### List of data sources stanza #######
;########################################
[ODBC Data Sources]
DB2V8DB=IBM DB2 version 8 ODBC Driver
DB2V9DB=OBM DB2 version 9 ODBC Driver
ORACLEDB=DataDirect 5.3 Oracle Driver
ORACLERACDB=DataDirect 5.3 Oracle Driver (Real Application Clusters)
SYBASEDB=DataDirect 5.3 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ################
;#########################################################

;# DB2 version 8 stanza
[DB2V8DB]
Driver=<Your DB2 v8 install directory>/lib/libdb2.sl
Description=DB2V8DB DB2 ODBC Database
Database=DB2V8DB
```

```
;# DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.sl
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.sl
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Server Name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.sl
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name defined for the RAC>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 Sybase Driver
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 Sybase Driver
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Server Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1
```

```
;# Informix Stanza
[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.sl
Description=IBM Informix ODBC Driver
ServerName=<Your Informix ServerName>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.3/lib/odbctrac.sl
InstallDir=<Your Broker install directory>/ODBC32/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Linux on x86 and Linux on x86-64

```
;#############################################################################
;########## 32 bit ODBC database driver manager initialization file ###############
;#############################################################################
;# It is recommended that you take a copy of this file and then edit the copy.     #
;#                                                                                 #
;#                                                                                 #
;# 1. Complete the 'Mandatory information stanza' section at the end of the file.  #
;#                                                                                 #
;#                                                                                 #
;# 2. For each data source, add the name of the data source into the 'List of      #
;#    data sources stanza' section.                                                #
;#                                                                                 #
;#                                                                                 #
;# 3. For each data source, create a stanza in the 'Individual datasource stanzas' #
;#    section.                                                                     #
;#############################################################################

;########################################
;###### List of data sources stanza ######
;########################################

[ODBC Data Sources]
DB2V8DB=IBM DB2 version 8 ODBC Driver
DB2V9DB=IBM DB2 version 9 ODBC Driver
ORACLEDB=DataDirect 5.3 Oracle Driver
ORACLERACDB=DataDirect 5.3 Oracle Driver (Real Application Clusters)
SYBASEDB=DataDirect 5.3 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 version 8 stanza
[DB2V8DB]
Driver=<Your DB2 v8 install directory>/lib/libdb2.so
Description=DB2V8DB DB2 ODBC Database
Database=DB2V8DB

;# DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.so
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB
```

```
;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Cluster stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name defined for the RAC>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;#Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;#Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Server Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1

;# Informix stanza
[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.sl
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;#######################################
;###### Mandatory information stanza #####
;#######################################
```

```
[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Solaris on SPARC

```
;###############################################################################
;########## 32 bit ODBC database driver manager initialization file ##############
;###############################################################################
;# It is recommended that you take a copy of this file and then edit the copy.     #
;#                                                                                 #
;#                                                                                 #
;# 1. Complete the 'Mandatory information stanza' section at the end of the file.  #
;#                                                                                 #
;#                                                                                 #
;# 2. For each data source, add the name of the data source into the 'List of      #
;#    data sources stanza' section.                                                #
;#                                                                                 #
;#                                                                                 #
;# 3. For each data source, create a stanza in the 'Individual datasource stanzas' #
;#    section.                                                                     #
;###############################################################################

;########################################
;###### List of data sources stanza #######
;########################################
[ODBC Data Sources]
DB2V8DB=IBM DB2 version 8 ODBC Driver
DB2V9DB=OBM DB2 version 9 ODBC Driver
ORACLEDB=DataDirect 5.3 Oracle Driver
ORACLERACDB=DataDirect 5.3 Oracle Driver (Real Application Clusters)
SYBASEDB=DataDirect 5.3 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 SQL Server Wire Protocol
INFORMIXDB= IBM Informix ODBC driver

;#########################################################
;########## Individual data source stanzas ################
;#########################################################

;# DB2 version 8 stanza
[DB2V8DB]
Driver=<Your DB2 v8 install directory>/lib/libdb2.so
Description=DB2V8DB DB2 ODBC Database
Database=DB2V8DB

;# DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.so
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

```
;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKor823.so
Description=DataDirect 5.3 Oracle Driver
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Net Service Name defined for the RAC>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKase23.so
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer Stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

odbc64.ini sample file

A copy of the sample 64-bit ODBC definition file that is supplied with WebSphere Message Broker.

Purpose

Configure the `odbc64.ini` file when you define an ODBC connection to a database from a 64-bit application. Follow the instructions in "Connecting to a database from Linux and UNIX systems: 64-bit considerations" on page 137.

AIX

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;####################################
;#### List of data sources stanza ######
;####################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 64bit SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ################
;#########################################################

;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

```
;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 64bit SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
AnsiNPW=Yes
Database=db
QuotedId=No
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza ######
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
```

```
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

HP-UX on PA-RISC

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;#####################################
;#### List of data sources stanza ######
;#####################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.sl
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.sl
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.sl
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
```

```
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.sl
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza ######
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.sl
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

HP-UX on Itanium

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
```

```
;# the 'List of data sources stanza' section.                          #
;#                                                                     #
;# 3. For each data source, create a stanza in the                     #
;# 'Individual data source stanzas' section.                           #
;######################################################################

;########################################
;###### List of data sources stanza #######
;########################################
[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 64bit SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;########################################################
;########## Individual data source stanzas ################
;########################################################

;# DB2 stanza
[DB2DB]
Driver=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
```

```
SelectMethod=0
NetworkAddress=<YourServerName>,<YourPortNumber>
SelectUserName=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 64bit SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
AnsiNPW=Yes
Database=db
QuotedId=No
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.sl
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
```

Linux on POWER

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################
```

```
;####################################
;#### List of data sources stanza ######
;####################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
```

```
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Linux on x86-64

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;########################################
;#### List of data sources stanza ######
;########################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect V5.3 64bit SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################
```

```
;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Server Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKmsss23.so
```

```
Description=DataDirect 5.3 64bit SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<your SQLServer Port Number>
AnsiNPW=Yes
Database=db
QuotedId=No
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Linux on System z

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;####################################
;#### List of data sources stanza ######
;####################################

[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
```

```
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Solaris on SPARC

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;########################################
;#### List of data sources stanza ######
;########################################
```

```
[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.3 64bit SQL Server Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
```

```
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8

;# UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKmsss23.so
Description=DataDirect 5.3 64bit SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<your SQLServer Port Number>
AnsiNPW=Yes
Database=db
QuotedId=No
ColumnSizeAsCharacter=1

;# Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

;########################################
;###### Mandatory information stanza #####
;########################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

Solaris on x86-64

```
;#########################################################################
;# 64 bit ODBC database driver manager initialisation file.              #
;#########################################################################
;# It is recommended that you take a copy of this file and then edit the  #
;# copy.                                                                 #
;#                                                                       #
;# 1. Complete the 'Mandatory information stanza' section                #
;# at the end of the file.                                               #
;#                                                                       #
;# 2. For each data source, add the name of the data source into         #
;# the 'List of data sources stanza' section.                            #
;#                                                                       #
;# 3. For each data source, create a stanza in the                       #
;# 'Individual data source stanzas' section.                             #
;#########################################################################

;########################################
;###### List of data sources stanza #######
;########################################
[ODBC Data Sources]
DB2DB=IBM DB2 ODBC Driver
ORACLEDB=DataDirect 5.3 64bit Oracle Wire Protocol
ORACLERACDB=DataDirect 5.3 64bit Oracle Wire Protocol (Real Application Clusters)
SYBASEDB=DataDirect 5.3 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.3 64bit Sybase UTF8 Wire Protocol
```

```
;#########################################################
;########## Individual data source stanzas ###############
;#########################################################

;# DB2 stanza
[DB2DB]
Driver=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB

;# Oracle stanza
[ORACLEDB]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Oracle Real Application Clusters stanza
[ORACLERACDB]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKora23.so
Description=DataDirect 5.3 64bit Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
ServiceName=<Your Oracle Real Application Cluster Service Name>
CatalogOptions=0
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

;# Sybase Stanza
[SYBASEDB]
Driver=<Your install directory>/ODBC64/V5.3/lib/UKase23.sl
Description=DataDirect 5.3 Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<YourServerName>,<YourPortNumber>
SelectUserName=1
EnableSPColumnTypes=2

;# Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.3/lib/UKase23.so
Description=DataDirect 5.3 64bit Sybase Wire Protocol
Database=<Your Database Name>
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
EnableSPColumnTypes=2
Charset=UTF8
```

```
;######################################
;###### Mandatory information stanza #####
;######################################

[ODBC]
;# To turn on ODBC trace set Trace=1
Trace=0
TraceFile=<A Directory with plenty of free space to hold trace ouput>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.3/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.3
UseCursorLib=0
IANAAppCodePage=4
```

Operations

To configure and administer the broker domain you must configure the broker and other components for your environment. You can configure and administer your broker domain and other components by running commands depending on your operating system.

Follow the links below for more information:
- "Broker properties"
- "Commands" on page 369
- "z/OS specific information" on page 667

Broker properties

View reference information for broker properties.

Broker properties are described in the following table.

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 keyring file. This name 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 Message Broker Toolkit, if you have installed these components on different computers.
SSL Password File Name	The SSL keyring 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 Message Broker Toolkit, if you have installed these components on different computers.
SSL HTTP Listener	Use the "mqsicreateusernameserver command" on page 540 or "mqsichangeusernameserver command" on page 499 to enable the SSL HTTP Listener authentication options.

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 437. See "Implementing quality of protection" on page 72 for details on setting these for temporary topics.

Restrictions that apply in each operation mode

The operation mode in which your broker is working defines how many execution groups and message flows you can use, and which nodes are available.

Trial Edition mode

All features are enabled, but you can use the product for only 90 days after installation.

Enterprise mode

All features are enabled and no restrictions or limits are imposed.

Starter Edition mode

All features are enabled, but the number of execution groups that you can create, and the number of message flows that you can deploy, are limited. You can create one execution group, and you can deploy a maximum of 10 message flows to that execution group.

Not all samples work in this mode. If you want to run samples to explore and understand the features of the product, you can install them on your development and unit test computers; see Development and unit test.

Remote Adapter Deployment mode

Only adapter-related features are enabled, and the types of node that you can use, and the number of execution groups that you can create, are limited. You can create up to two execution groups, with no limit on the number of deployed message flows in each of these execution groups.

Not all samples work in this mode. If you want to run samples to explore and understand the features of the product, you can install them on your development and unit test computers; see Development and unit test.

The following nodes cannot be deployed to a broker running in Remote Adapter Deployment mode:

WebSphere MQ		
MQGet node	MQHeader node	MQOptimizedFlow node
MQReply node	MQeInput node	MQeOutput node

JMS		
JMSHeader node	JMSMQTransform node	JMSReply node
MQJMSTransform node		

HTTP		
HTTPHeader node	HTTPRequest node	

Web services		
SOAPAsyncRequest node	SOAPAsyncResponse node	SOAPEnvelope node
SOAPExtract node		

WebSphere Adapters		
TwineballInput node	TwineballRequest node	

Routing		
AggregateControl node	AggregateReply node	AggregateRequest node
Collector node	Filter node	Label node
Publication node	Route node	RouteToLabel node

Transformation		
Compute node	Mapping node	XSLTransform node

Construction		
FlowOrder node	Passthrough node	ResetContentDescriptor node
Throw node	Trace node	TryCatch node

Database		
Database node	DatabaseRetrieve node	DatabaseRoute node
DataDelete node	DataInsert node	DataUpdate node
Extract node	Warehouse node	

Email		
EmailOutput node		

Validation		
Check node	Validate node	

Additional protocols		
SCADAInput node	SCADAOutput node	

Commands

All Message Broker Toolkit and runtime commands that are provided on distributed systems are listed, grouped by function, with references to command details.

For information about the equivalent commands on WebSphere Message Broker for z/OS, see "Summary of commands on Linux, UNIX, Windows, and z/OS platforms" on page 372.

WebSphere Message Broker Toolkit commands

mqsiapplybaroverride	"mqsiapplybaroverride command" on page 382
mqsicreatebar	"mqsicreatebar command" on page 386
mqsicreatemsgdefs	"mqsicreatemsgdefs command" on page 389
mqsicreatemsgdefsfromwsdl	"mqsicreatemsgdefsfromwsdl command" on page 398
mqsimigratemfmaps	"mqsimigratemfmaps command" on page 400
mqsireadbar	"mqsireadbar command" on page 385

WebSphere Message Broker commands

Broker commands

mqsiapplybaroverride	"mqsiapplybaroverride command" on page 382
mqsichangebroker	"mqsichangebroker command" on page 404
mqsicreatebroker	"mqsicreatebroker command" on page 513
mqsicreateexecutiongroup	"mqsicreateexecutiongroup command" on page 538
mqsideletebroker	"mqsideletebroker command" on page 555
mqsideleteexecutiongroup	"mqsideleteexecutiongroup command" on page 564
mqsimode	"mqsimode command" on page 602
mqsireadbar	"mqsireadbar command" on page 385
mqsireload	"mqsireload command" on page 611
mqsireportbroker	"mqsireportbroker command" on page 616

Database commands

mqsichangedbimgr	"mqsichangedbimgr command" on page 420
mqsicreatedb	"mqsicreatedb command" on page 536
mqsideletedb	"mqsideletedb command" on page 563
mqsimanagexalinks	"mqsimanagexalinks command" on page 593
mqsisetdbparms	"mqsisetdbparms command" on page 646
mqsi_setupdatabase	"mqsi_setupdatabase command" on page 653

Security commands

mqsireloadsecurity	"mqsireloadsecurity command" on page 613
mqsisetdbparms	"mqsisetdbparms command" on page 646
mqsisetsecurity	"mqsisetsecurity command" on page 652

Configuration Manager commands

mqsibackupconfigmgr	"mqsibackupconfigmgr command" on page 403
mqsichangeconfigmgr	"mqsichangeconfigmgr command" on page 415
mqsicreateconfigmgr	"mqsicreateconfigmgr command" on page 525
mqsideleteconfigmgr	"mqsideleteconfigmgr command" on page 557
mqsireportconfigmgr	"mqsireportconfigmgr command" on page 621

Summary of commands on Linux, UNIX, Windows, and z/OS platforms

The following table summarizes the runtime commands that are available on Linux, UNIX, and Windows platforms, 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
mqsiapplybaroverride	Utility JCL	`BIPOBAR`	"Contents of the broker PDSE" on page 679
mqsibackupconfigmgr	Utility JCL	`BIPBUCM`	"Contents of the Configuration Manager PDSE" on page 682
mqsichangebroker	1. Console command: modify 2. Utility JCL	1. `changebroker` 2. `BIPCHBK`	1. "mqsichangebroker command" on page 404 2. "Contents of the broker PDSE" on page 679
mqsichangeconfigmgr	1. Console command: modify 2. Utility JCL	1. `changeconfigmgr` 2. `BIPCHCM`	1. "mqsichangeconfigmgr command" on page 415 2. "Contents of the Configuration Manager PDSE" on page 682
mqsichangedbimgr	Not applicable		
mqsichangeflowmonitoring	1. Console command: modify 2. Utility JCL	1. `changeflowmonitoring` 2. `BIPCHME`	1. "mqsichangeflowmonitoring command" on page 421 2. "Contents of the broker PDSE" on page 679
mqsichangeflowstats	1. Console command: modify 2. Utility JCL	1. `changeflowstats` 2. `BIPCHMS`	1. "mqsichangeflowstats command" on page 426 2. "Contents of the broker PDSE" on page 679
mqsichangeflowuserexits	1. UNIX System Services (USS) command 2. Utility JCL	1. mqsichangeflowuserexits 2. BIPCHUE	1. "mqsichangeflowuserexits command" on page 433 2. "Contents of the broker PDSE" on page 679

mqsichangeproperties	Utility JCL	BIPCHPR	"Contents of the broker PDSE" on page 679
mqsichangetrace	Console command: modify	changetrace	"mqsichangetrace command" on page 492
mqsichangeusernameserver	1. Console command: modify 2. Utility JCL	1. changeusernameserver 2. BIPCHUN	1. "mqsichangeusernameserver command" on page 499 2. "Contents of the User Name Server PDSE" on page 681
mqsiclearmqpubsub	Utility JCL	BIPCLMP	"Contents of the broker PDSE" on page 679
mqsicreateaclentry	1. Console command: modify 2. Utility JCL	1. createaclentry 2. BIPCRACL	1. "mqsicreateaclentry command" on page 504 2. "Contents of the Configuration Manager PDSE" on page 682
mqsicreatebroker	USS command	mqsicreatebroker	"mqsicreatebroker command" on page 513
mqsicreateconfigmgr	Utility JCL	BIPCRCM	"Contents of the Configuration Manager PDSE" on page 682
mqsicreateconfigurableservice	Utility JCL	BIPJADPR	"Contents of the broker PDSE" on page 679
mqsicreatedb	Not applicable		
mqsicreateexecutiongroup	Utility JCL	BIPCREG	"Contents of the Configuration Manager PDSE" on page 682
mqsicreateusernameserver	USS command	mqsicreateusernameserver	"mqsicreateusernameserver command" on page 540
mqsicvp	Not applicable		
mqsideleteaclentry	1. Console command: modify 2. Utility JCL	1. deleteaclentry 2. BIPDLACL	1. "mqsideleteaclentry command" on page 548 2. "Contents of the Configuration Manager PDSE" on page 682
mqsideletebroker	Utility JCL	BIPDLBK	"Contents of the broker PDSE" on page 679
mqsideleteconfigmgr	Utility JCL	BIPDLCM	"Contents of the Configuration Manager PDSE" on page 682
mqsideleteconfigurableservice	Utility JCL	BIPJADPR	"Contents of the broker PDSE" on page 679
mqsideletedb	Not applicable		
mqsideleteexecutiongroup	Utility JCL	BIPDLEG	"Contents of the Configuration Manager PDSE" on page 682
mqsideleteusernameserver	Utility JCL	BIPDLUN	"Contents of the User Name Server PDSE" on page 681

mqsideploy	1. Console command: modify 2. Utility JCL	1. `deploy` 2. `BIPDPLY`	1. "mqsideploy command" on page 568 2. "Contents of the broker PDSE" on page 679
mqsiformatlog	Utility JCL	`BIPFMLG`	"Contents of the broker PDSE" on page 679
mqsijoinmqpubsub	Utility JCL	`BIPJNMP`	"Contents of the broker PDSE" on page 679
mqsilist	1. Console command: modify 2. Utility JCL	1. `list` 2. `BIPLIST`	1. "mqsilist (list resources) command" on page 581 2. "Contents of the Configuration Manager PDSE" on page 682
mqsilistaclentry	1. Console command: modify 2. Utility JCL	1. `listaclentry` 2. `BIPLIACL`	1. "mqsilistaclentry command" on page 583 2. "Contents of the Configuration Manager PDSE" on page 682
mqsilistmqpubsub	Utility JCL	`BIPLSMP`	"Contents of the broker PDSE" on page 679
mqsimanagexalinks	Not applicable		
mqsimigratecomponents	Utility JCL	`BIPMGCMP`	"mqsimigratecomponents command" on page 597
mqsimode	Not applicable		
mqsireadbar	Utility JCL	`BIPRBAR`	"Contents of the broker PDSE" on page 679
mqsireadlog	Utility JCL	`BIPRELG`	"Contents of the broker PDSE" on page 679
mqsireload	Console command: modify	`reload`	"mqsireload command" on page 611
mqsireloadsecurity	Utility JCL	`BIPRLSEC`	"Contents of the Configuration Manager PDSE" on page 682
mqsireportbroker	1. Console command: modify 2. Utility JCL	1. `reportbroker` 2. `BIPRPBK`	"Contents of the broker PDSE" on page 679
mqsireportconfigmgr	1. Console command: modify 2. Utility JCL	1. `reportconfigmgr` 2. `BIPRPCM`	"Contents of the Configuration Manager PDSE" on page 682
mqsireportflowmonitoring	1. Console command: modify 2. Utility JCL	1. `reportflowmonitoring` 2. `BIPRPME`	1. "mqsireportflowmonitoring command" on page 624 2. "Contents of the broker PDSE" on page 679

mqsireportflowstats	1. Console command: modify 2. Utility JCL	1. `reportflowstats` 2. `BIPRPMS`	1. "mqsireportflowstats command" on page 629 2. "Contents of the broker PDSE" on page 679
mqsireportflowuserexits	1. USS command 2. Utility JCL	1. mqsireportflowuserexits 2. BIPRPUE	1. "mqsireportflowuserexits command" on page 633 2. "Contents of the broker PDSE" on page 679
mqsireportproperties	Utility JCL	`BIPRPPR`	"Contents of the broker PDSE" on page 679
mqsireporttrace	Console command: modify	`reporttrace`	"mqsireporttrace command" on page 640
mqsirestoreconfigmgr	Utility JCL	`BIPRSCM`	"Contents of the Configuration Manager PDSE" on page 682
mqsisetdbparms	Utility JCL	`BIPSDBP`	"mqsisetdbparms command" on page 646
mqsisetsecurity	Not applicable		
mqsi_setupdatabase	Not applicable		
mqsistart	1. Console command: start 2. Console command: modify	1. Standard MVS start command 2. `startcomponent`	1. - 2. "mqsistart command" on page 654
mqsistartmsgflow	Utility JCL	`BIPSTMF`	"Contents of the Configuration Manager PDSE" on page 682
mqsistop	1. Console command: stop 2. Console command: modify	1. Standard MVS stop command 2. `'p' stopcomponent`	1. - 2. "mqsistop command" on page 660
mqsistopmsgflow	Utility JCL	`BIPSPMF`	"Contents of the Configuration Manager PDSE" on page 682

Syntax diagrams: available types

The syntax for commands and ESQL statements and functions is presented in the form of a diagram. The diagram tells you what you can do with the command, statement, or function 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:

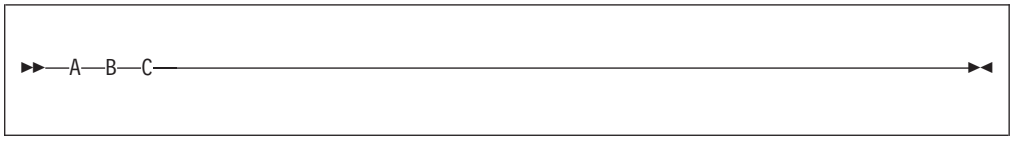
- "How to read railroad diagrams"
- "How to read dotted decimal diagrams" on page 378

How to read railroad diagrams

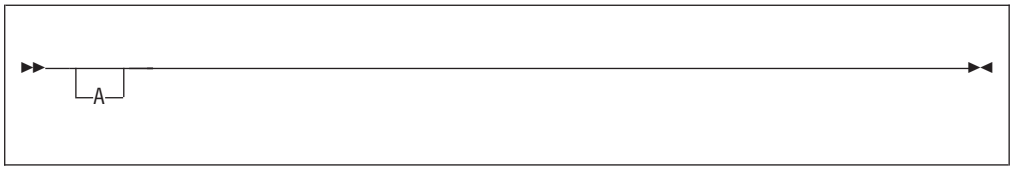
Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

The following examples show other conventions used in railroad diagrams.

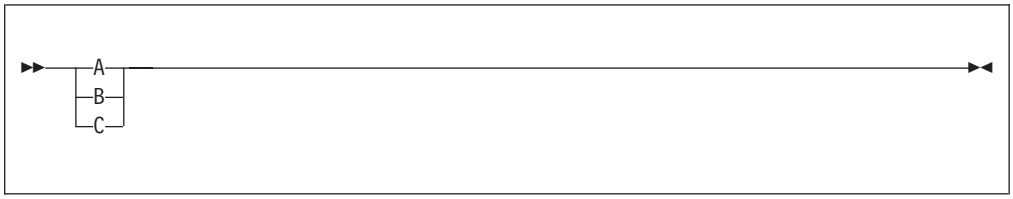
This example shows that you must specify values `A`, `B`, and `C`. Required values are shown on the main line of a railroad diagram:



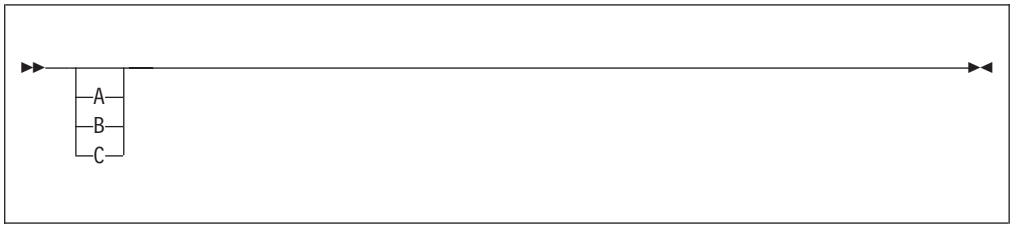
This shows that you can specify value `A`. Optional values are shown below the main line of a railroad diagram:



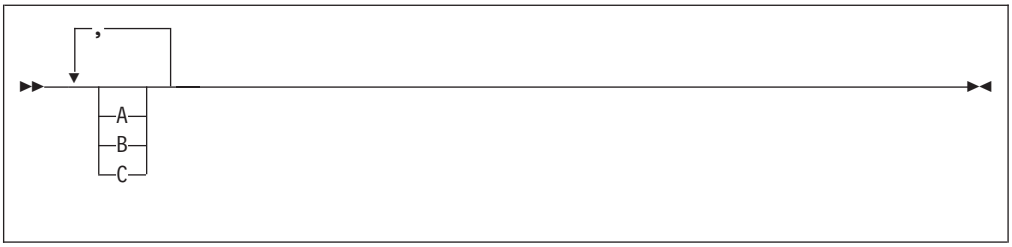
The next example specifies that values `A`, `B`, and `C` are options, one of which you must specify:



Values `A`, `B`, and `C` are options in this example, one of which you can specify:

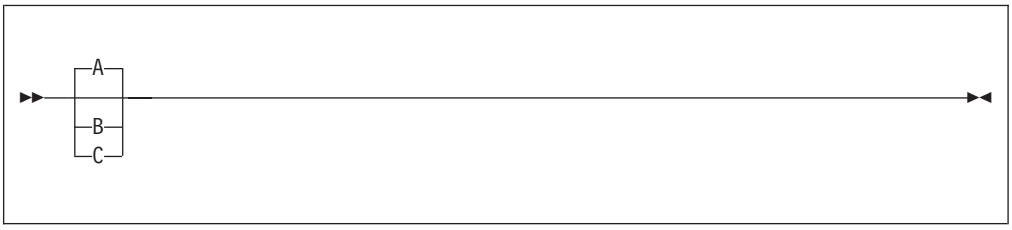


The next example shows that you can specify one or more of the values `A`, `B`, and `C`. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow:

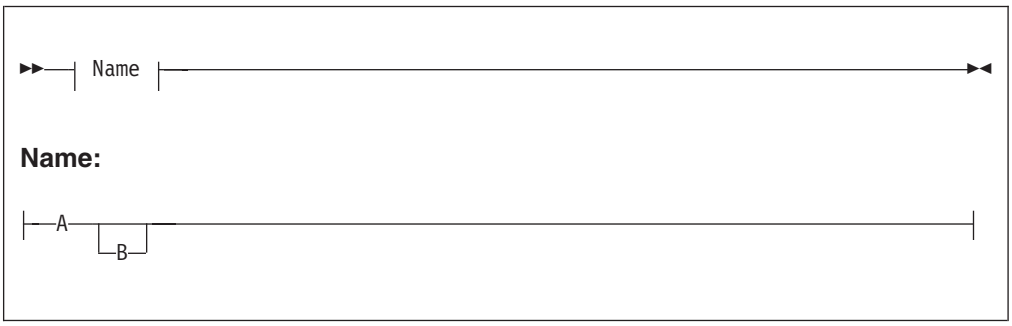


In this example, you can specify value A multiple times. The separator in this example is optional:

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:



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.

Characters allowed in commands

You must adhere to a few rules when you provide names or identifiers for the components and resources in your broker domain.

The character set that you can use to name brokers, Configuration Managers, execution groups, and message identifiers is as follows:

- Uppercase alphabetic characters A-Z
- Lowercase alphabetic characters a-z
- Numeric characters 0-9
- Any special characters supported by the underlying file system:
  - The following special characters are accepted on Windows platforms:

$ % ′ (apostrophe) ″ (quotation mark)

- (dash)	_ (underscore)	@	~ (tilde)
!	(	)	{
}	[	]	&
#	&	+	, (comma)
;	=	(space)	

– The following special characters, with the exception of a space, are accepted on Linux and 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 (message sets, message flows, User Name Server, and topics), any characters that are supported by the database configuration are supported.

On Windows platforms, broker names, Configuration Manager names, and fixed names (`UserNameServer`) are not case sensitive. For example, broker names `Broker1` and `BROKER1` refer to the same broker.

On Linux and UNIX systems, broker names and Configuration Manager names are case sensitive, and the examples above would refer to different brokers.

You must use `UserNameServer` as shown.

On z/OS systems, you must enclose mixed-case names in quotation marks.

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 Message 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 369 .

- 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 Message 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

Runtime and workbench commands (common)

Some commands are common to both runtime and workbench environments.

The following commands are available on workstations that have either the runtime, the workbench, or both components installed:

- "mqsiapplybaroverride command"
- "mqsireadbar command" on page 385

If either of these commands is run on a workstation that has both the runtime and workbench components installed, the version of the command that is used is determined by the relative locations of the workbench and runtime directories in the PATH variable of the workstation. The directory that appears earlier in the PATH takes precedence.

In most cases the workbench and runtime versions of the command are identical. However, if different levels of service pack have been applied to the workbench and runtime components, the updated commands might differ. For this reason, when either command is run, the first line of the output describes which version of the command is being used. For example:

```
BIP1138I: Overriding BAR File using runtime mqsiapplybaroverride.
```

mqsiapplybaroverride command

Use the mqsiapplybaroverride command to replace configurable values in the broker archive (BAR) deployment descriptor with new values that you specify in a properties file.

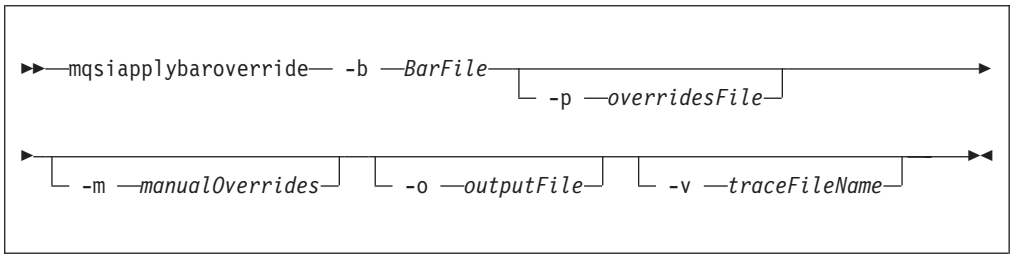
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPOBAR.

Purpose:

Write scripts to create broker archive files and apply different override values in the broker deployment descriptor archive file by using the mqsiapplybaroverride command, together with the mqsicreatebar command.

Syntax:



Parameters:

-b *BarFile*
(Required) The path to the broker archive file (in compressed format) to which the override values apply. The path can be absolute or relative to the executable command.

-p *overridesFile*
(Optional) The path to one of the following resources:

- A broker archive containing the deployment descriptor that is used to apply overrides to the BAR file.
- A properties file in which each line contains a `property-name=override` or `current-property-value=new-property-value` pair.
- A deployment descriptor that is used to apply overrides to the BAR file.

-m *manualOverrides*
(Optional) A list of the `property-name=override` pairs, `current-property-value=override` pairs, or a combination of them, to be applied to the BAR file. The pairs in the list are separated by commas (,). On Windows, you must enclose the list in double quotes (" "). If used in conjunction with the overridesFile (**–p**) parameter, overrides specified by the manualOverrides (**–m**) parameter are performed after any overrides specified the **–p** parameter have been made.

-o *outputFile*
(Optional) The name of the output BAR file to which the BAR file changes are to be made. If an output file is not specified, the input file is overwritten.

-v *traceFileName*
(Optional) Specifies that the internal trace is to be sent to the named file.

In all cases, any existing deployment descriptor in the BAR file is renamed to `META-INF\broker.xml.old`, replacing any existing file of that name.

Each override that is specified in a **–p** overrides file or a **–m** overrides list must conform to one of the following syntaxes:

- `FlowName#NodeName.PropertyName=NewPropertyValue` (or `FlowName#PropertyName=NewPropertyValue` for message flow properties) where:
  - *FlowName* is the name of the message flow without any .cmf extension (for example, Flow1).
  - *NodeName* is the optional name of the node whose property is overridden (for example, InputNode).
  - *PropertyName* is the name of the property being overridden (for example, queueName).
  - *NewPropertyValue* is the value to assign to that property (for example, PRODUCTION_QUEUE_NAME).
- `OldPropertyValue=NewPropertyValue`. This syntax does a global search and replace on the property value `OldPropertyValue`. It overrides the value fields of `OldPropertyValue` in the deployment descriptor with `NewPropertyValue`.
- `FlowName#NodeName.PropertyName` (or `FlowName#PropertyName` for message flow properties). This syntax removes any override applied to the property of the supplied name.

When the mqsiapplybaroverride command runs, it displays the version of the command that is being used (either runtime or toolkit) before it does anything else. For example:

```
BIP1138I: Overriding BAR File using runtime mqsiapplybaroverride
```

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all operating systems, the user ID used to invoke this command must have write authority to the BAR file on the local system.

Responses:

This command returns the following responses:

0 (Success) The request completed successfully.

2 (Failure) The command failed.

99 The supplied parameters are not valid.

Windows This command is supplied as a batch file. If you run the command in an automation system or from a script, use the Windows CALL command, to ensure that the correct ERRORLEVEL is returned:

```
...
CALL mqsiapplybaroverride
...
```

Examples:

Open the BAR file `myflow.bar`, and replace configurable values in its deployment descriptor (typically `broker.xml`) with those specified in the properties file `mychanges.properties`:

```
mqsiapplybaroverride -b myflow.bar -p mychanges.properties
```

Override the deployment descriptor in `c:\test.bar` by using the key=value pairs specified in `c:\my.properties`:

```
mqsiapplybaroverride -b c:\test.bar -p c:\my.properties
```

Override the deployment descriptor in `c:\test.bar` by using the deployment descriptor contained in `c:\previous.bar`:

```
mqsiapplybaroverride –b c:\test.bar –p c:\previous.bar
```

Override the deployment descriptor in `c:\test.bar` by using the deployment descriptor contained in `c:\broker.xml`:

```
mqsiapplybaroverride –b c:\test.bar –p c:\broker.xml
```

Override any properties with values set to OLDA and OLDB in `c:\test.bar` with the values NEWA and NEWB respectively:

```
mqsiapplybaroverride –b c:\test.bar –o OLDA=NEWA:OLDB=NEWB
```

Override the value of the property name **sampleFlow#MQInput.queueName** to NEWC:

```
mqsiapplybaroverride –b c:\test.bar –o sampleFlow#MQInput.queueName=NEWC
```

For an example of the details that are contained in a properties file, see Editing configurable properties.

mqsireadbar command

Use the mqsireadbar command to read a deployable BAR file and identify its defined keywords.

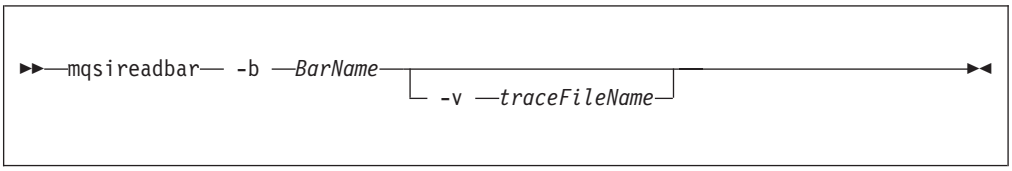
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPRBAR.

Purpose:

The mqsireadbar command returns the keywords defined for each deployable file within a deployable broker archive file.

Syntax:



Parameters:

-b *BarName*
(Required) The name of the BAR archive file to be read. The BAR file is in compressed format.

-v *traceFileName*
(Optional) The name of the file to which the command trace is sent. This option activates an internal debug trace; specify this option only at the request of an IBM service representative.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID that is used to start this command must have the authority to read the BAR file on the local system.

Responses:

This command returns the following responses:

0 (Success) Indicates that the request completed successfully.

2 (Failure) Indicates that the command failed.

99 Indicates that the supplied parameters are not valid.

The command displays the version of the command that is being run (either workbench or runtime environment), before all other response data:

```
BIP1052I: Reading BAR File using runtime mqsireadbar
```

The command then displays a list of the deployable files, together with their keywords. For example:

```
C:\test.bar
     BAR Entry: simpleflow.cmf (07/10/07 10:43:44)
          Author = Matt
          VERSION = v1.1
          Information = This flow simply removes messages from IN.Q
```

It also displays deployment descriptors and the list of any properties in the BAR file that can be overridden, together with the new values of any overrides that are currently applied. For example:

```
Deployment descriptor:
    simpleflow#additionalInstances
    simpleflow#commitCount
    simpleflow#commitInterval
    simpleflow#coordinatedTransaction
    simpleflow#MQInput.topicProperty
    simpleflow#MQInput.validateMaster
    simpleflow#MQInput.queueName = OVERRIDDEN.Q
    simpleflow#MQInput.serializationToken
```

Windows This command is supplied as a batch file. If you run the command in an automation system or from a script, use the Windows CALL command to ensure that the correct ERRORLEVEL is returned:

```
...
CALL mqsireadbar
...
```

Examples:

The following example reads the file `my_bar_file.bar` and returns defined keywords within the given file:

```
mqsireadbar -b my_bar_file.bar
```

WebSphere Message Broker workbench commands

This topic is a container for the commands that are part of the WebSphere Message Broker workbench.

These commands are available only on a machine that has the workbench installed.

See “Commands” on page 369 for a list of all the WebSphere Message Broker commands.

mqsicreatebar command

Use the mqsicreatebar command to create deployable broker archive files containing message flows and dictionaries.

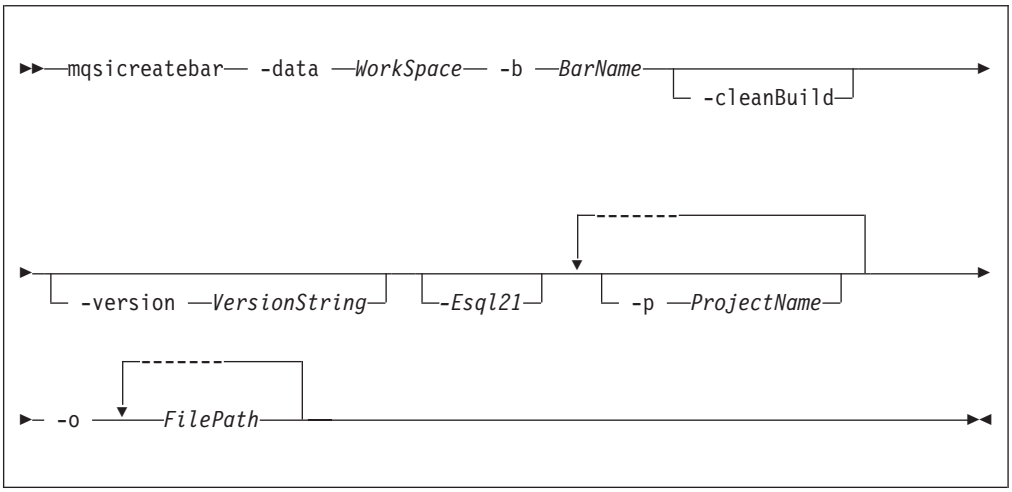
Supported platforms:

- Windows
- Linux on x86

Purpose:

If you choose to use a repository to store your message flows and dictionaries, you can write scripts to deploy the message flow applications using the mqsicreatebar command and the repository's command line tools.

Syntax:



Parameters:

-data *WorkSpace*
(Required) The path of the workspace in which your projects are created.

The workspace location is also the default location for projects. Relative paths are interpreted as being relative to the directory from which the command was started.

-b *BarName*
(Required) The name of the BAR (compressed file format) archive file where the result is stored. The BAR file is replaced if it already exists and the `META-INF/broker.xml` file is created.

-cleanBuild
(Optional) Refreshes the projects in the workspace and then invokes a clean build before new items are added to the broker archive.

Use the **-cleanBuild** parameter to refresh all the projects in the broker archive and invoke a clean build if amendments have been made to broker-archive resources using external tools.

-version *VersionString*
(Optional) Appends the _ (underscore) character and the value of *VersionString* to the names of the objects added to the BAR file, before the file extension.

-Esql21
(Optional) Compile ESQL for brokers at Version 2.1 of the product.

-p *ProjectName*
(Optional) Projects containing files to include in the BAR file. You can specify

multiple projects, which can include a message flow project, a message set project, or a message flow user-defined node project.

If a project that you specify is not currently part of your workspace, the command links the project to the workspace so that the files in the project can be included in the BAR file. The command does not copy the files into your workspace directory.

If a project that you specify is part of your workspace but is currently closed, the command opens and builds the project so that the files in the project can be included in 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 file.

You can add more than one deployable file to this command by using the following format: `-o FilePath1 FilePath2 .... FilePath'n'`

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux on x86, the user ID must have write access to the **-data** (workspace) and **-b** (BAR file location) directories.

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:

You must run the command from the eclipse directory. The default location of the eclipse directory is C:\Program Files\IBM\WMBT610\eclipse on Windows 32–bit editions or C:\Program Files(x86)\IBM\WMBT610\eclipse on Windows 64–bit editions: on Linux on x86 it is /opt/IBM/WMBT610/eclipse.

The following example creates a BAR file called myflow.bar in the workspace at C:\Workspace. The Test.msgflow message flow from the TestFlowProject is added to the BAR file:

```
mqsicreatebar -data C:\Workspace -b myflow.bar -p TestFlowProject -o TestFlowProject\TestFlow\Test.msgflow
```

The following example creates a BAR file called mySet.bar in the workspace at C:\Workspace. The messageSet.mset message set from the TestSetProject is added to the BAR file:

```
mqsicreatebar -data C:\Workspace -b mySet.bar -o TestSetProject\TestSet\messageSet.mset
```

The following example creates a BAR file called mySet.bar in the workspace at C:\Workspace. The messageSet.mset message set from the TestSetProject and Test.msgflow message flow from the TestFlowProject are added to the BAR file:

```
mqsicreatebar -data C:\Workspace -b mySet.bar -o TestFlowProject\TestFlow\Test.msgflow
TestSetProject\TestSet\messageSet.mset
```

mqsicreatemsgdefs command

Use the mqsicreatemsgdefs command to create message definition files.

Supported platforms:

- Windows
- Linux on x86

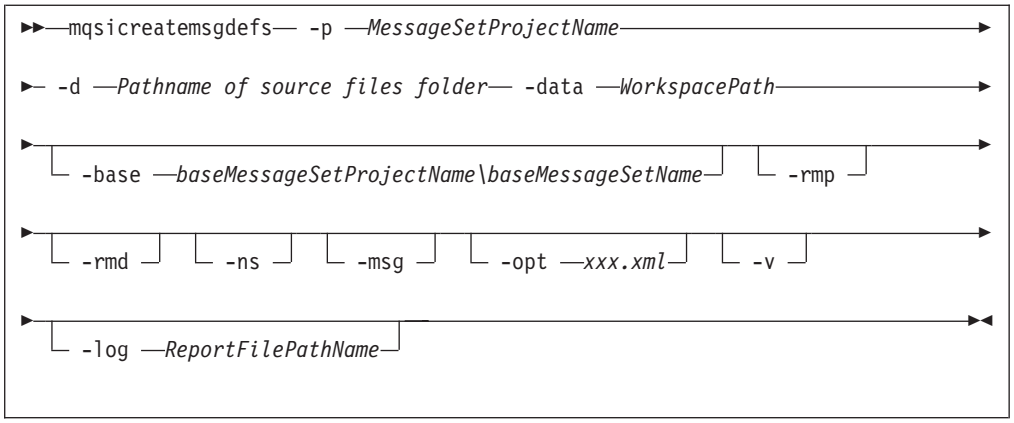
Purpose:

The mqsicreatemsgdefs command generates message definition files (*.mxsd), according to a set of import options that are specified in an option file. The generated files are placed in the specified message set folder.

The command takes as a parameter a directory where source files of various types, for example, C and COBOL source files, are located (in addition to various other parameters), and starts the appropriate operation based on the extensions to the files.

1. Ensure that only the files that are required for the command to run exist in the directory and subdirectory structure that you specify. One of the actions that the mqsicreatemsgdefs command performs is to copy all the files in the directory and subdirectories into the workspace, before creating the message definition. Files that are not related to the message definitions that you are trying to create might also be copied.
2. You must specify the **-data** *WorkspacePath* parameter to specify the target workspace.

Syntax:



Parameters:

-p *MessageSetProjectName*
(Required) The name of the message set project. If the project does not exist, a new message set project is created.

-d *Pathname of source files folder*
(Required) The absolute or relative path name of the directory of definition files (source files).

All relevant files that are located in any sub-folders under the source files folder are scanned and imported.

-data *WorkspacePath*
(Required) The path of the workspace in which your projects are created.

The workspace location is also the default location for projects. Relative paths are interpreted as being relative to the directory from which the command was started.

-base *baseMessageSetProjectName\baseMessageSetName*
(Optional) If a new message set is to be created, specifies the existing message set project and message set, on which it is based.

-rmp
(Optional) Replaces the existing project of the same name.

-rmd
(Optional) Replaces an existing message definition file of the same name.

1. If you omit this flag, and a message definition file of the same name exists, a warning is returned.
2. The location of the generated message definition file in the message set is determined by the target namespace.

-ns
(Optional) If a new message set is to be created, the message set is enabled for namespace support.

-msg
(Optional) Creates messages from complex imported structures.

-opt *xxx.xml*
(Optional) The absolute or relative path name of the options file. The options file can be one of the following types:

- **C language** - "C options file for the mqsicreatemsgdefs command" on page 391

- **COBOL language** - "COBOL options file for the mqsicreatemsgdefs command" on page 393
- **XSD_NO_NS** - "XSD options file for the mqsicreatemsgdefs command" on page 394

If you do not specify an option, the default options file (`mqsicreatemsgdefs.xml`) is used; see "Default options file for the mqsicreatemsgdefs command" on page 396.

You can copy the default options file, and customize it, to create an options file for your environment.

-v (Optional) Verbose report.

-log *ReportFilePathname*
(Optional) Absolute or relative path name of the report file. If you omit this option, the report is written to the default log file (`mqsicreatemsgdefs.report.txt`) in the Eclipse current directory.

If you specify *-log* without the report file path name, or with a path name that is not valid, the command issues an error message and stops.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

Examples:

The following example creates or uses the message set project `newproject` in the source file `c:\myproject\source` and replaces the existing message project and message definition files of the same name.

```
mqsicreatemsgdefs -p newproject -d c:\myproject\source -rmp -rmd
```

C options file for the mqsicreatemsgdefs command:

Specify the options for the mqsicreatemsgdefs command when you import a C header file.

The following table lists the elements in the C language section of the options file. The following restrictions apply:

- You must specify, in an XML file, a valid value for the options listed, unless otherwise specified. See "Default options file for the mqsicreatemsgdefs command" on page 396 for details of the syntax.
- Values of options are case-sensitive.

- If you do not specify the *-opt* parameter on the mqsicreatemsgdefs command, the default options file called `mqsicreatemsgdefs.xml` is used; see "Default options file for the mqsicreatemsgdefs command" on page 396.

For further information on using these options, see Importing from C.

<C> element	**Possible values**
COMPILER_NAME[1]	• Msvc (Default) • Icc • AIXgcc • AIXxlc • OS390
CODEPAGE	• SO8859-1 • Cp037 • Cp1252 (Default)
FLOATING_POINT_FORMAT	• IEEE Extended INTEL (Default) • IEEE Extended AIX • IEEE Extended OS/390 • IEEE Non-Extended • IBM 390 Hexadecimal
INCLUDE_PATH[2]	Absolute path names of other header files, or an empty string (this is the default value).
BYTE_ORDER	• Little Endian (Default) • Big Endian
ADDRESS_SIZE	• 32 (Default) • 64
SIZE_OF_LONG_DOUBLE	• 64 (Default) • 128 (Not supported)
PACK_LEVEL[1]	• 1 • 2 • 4 • 8 (Default) • 16
SIZE_OF_ENUM	• 1 • 2 • 4 • 5 (Default)
PRESERVE_CASE_IN_VARIABLE_NAMES	• True (Default) • False
STRING_ENCODING	• SPACE - Fixed-length strings (Default) • NULL - Null-terminated strings

<C> element	Possible values
STRING_PADDING_CHARACTER	• SPACE (Default) • NUL • 'c' • "c" • 0xYY • YY • U+xxxx
SCHEMA_TARGET_NAMESPACE_URI	A valid namespace URI, or empty (default)
MESSAGE_PREFIX[3]	A string with which to prefix created messages, or empty. Default is `msg_`.
PRE_PROCESSING_OPTION	• none (default) • ale_idoc • file_idoc

Notes:

1. If COMPILER_NAME is set to AIXxlc, the value of PACK_LEVEL is not used.
2. In INCLUDE_PATH, separate paths by the system-dependent path separator character.
3. MESSAGE_PREFIX is ignored if PRE_PROCESSING_OPTION is ale_idoc or file_idoc.

COBOL options file for the mqsicreatemsgdefs command:

Specify the options for the mqsicreatemsgdefs command when you import a COBOL copybook.

The following table lists the elements in the COBOL language section of the options file. The following restrictions apply:

- You must specify in an XML file a valid value for the options listed, unless otherwise specified. See "Default options file for the mqsicreatemsgdefs command" on page 396 for details of the syntax.
- Options values are case-sensitive.
- If you do not specify the *-opt* parameter on the mqsicreatemsgdefs command, the default options file (`mqsicreatemsgdefs.xml`) is used; see "Default options file for the mqsicreatemsgdefs command" on page 396.

For further information about using these options, see Importing from COBOL copybooks.

<COBOL> element	Possible values
PLATFORM_SELECTION	• 0 (Win32) (Default) • 1 (AIX) • 2 (z/OS)
CODEPAGE	• ISO8859_1 (Default) • 037

<COBOL> element	Possible values
FLOATING_POINT_FORMAT	• IEEE Non-Extended (Default) • IBM 390 Hexadecimal
ENDIAN	• Big • Little (Default)
EXT_DECIMAL_SIGN	• ASCII (Default) • EBCDIC • EBCDIC Custom
TRUNC	• STD (Default) • OPT • BIN
NSYMBOL	• DBCS • NATIONAL (Default)
QUOTE	• SINGLE • DOUBLE (Default)
CREATE_DEFAULT_VALUES FROM_INITIAL_VALUES	• True • False (Default)
CREATE_FACETS_FROM LEVEL_88_VALUE_CLAUSES	• True • False (Default)
PRESERVE_CASE_IN VARIABLE_NAMES	• True (Default) • False
CREATE_NULL_VALUES_FOR_FIELDS	• True • False (Default)
NULL_CHARACTER	• SPACE (Default) • NUL • 'c' • "c" • 0xYY • YY • U+xxxx
STRING_PADDING_CHARACTER	• SPACE (Default) • NUL • 'c' • "c" • 0xYY • YY • U+xxxx
SCHEMA_TARGET_NAMESPACE_URI	A valid namespace URI, or empty (default)
MESSAGE_PREFIX	A string with which to prefix created messages, or empty. Default is `msg_`.

XSD options file for the mqsicreatemsgdefs command:

Specify the options for the mqsicreatemsgdefs command when you import an XML Schema file.

The following tables list the elements in the XML Schema sections of the options file.

The first table applies to all message sets, but the second table applies only to message sets that do not support namespaces. The following restrictions apply:

- You must specify in an XML file a valid value for each of the options listed, unless otherwise specified.
- Options values are case-sensitive.
- If you do not specify the *-opt* parameter on the **mqsicreatemsgdefs** command, the default options file (`mqsicreatemsgdefs.xml`) is used; see "Default options file for the mqsicreatemsgdefs command" on page 396.

For further information about using these options, see Importing from XML Schema.

<XSD> element	**Possible values**
MSG	• elements (default) • types • both

<XSD_NO_NS> element	**Possible values**
IMPORT	• modify (default) • reject
REDEFINE	• modify (default) • reject • accept
LIST	• modify (default) • reject • accept
UNION	• modify (default) • reject • accept
ABSTRACT_CT	• modify (default) • reject • accept
ABSTRACT_ELEMENT	• modify (default) • reject • accept
XSD_PREFIX	• xsi (default) • <any other prefix>
URI_PREFIX_PAIRS **Note:** You can specify zero, or more URI_PREFIX_PAIRS elements.	Attribute pair • uri=<uri_value> • prefix=<prefix_value>

Default options file for the mqsicreatemsgdefs command:

Options for the mqsicreatemsgdefs command take default values if you do not specify an options file.

The following text lists the supplied default options file used with the mqsicreatemsgdefs command.

If you want to make any changes to the default file contents, the file is stored in the package group directory structure at `%PACKAGE_GROUP_LOCATION%\plugins\com.ibm.etools.msg.importer.cmdline_%VERSION%\mqsicreatemsgdefs.xml`. For example, if you have installed into the default location on Windows XP, the file is stored at `C:\Program Files\IBM\SDP70Shared\plugins\com.ibm.etools.msg.importer.cmdline_`*`build_version`*`\mqsicreatemsgdefs.xml`. *build_version* is the exact build version of the installed component; for example, `6.1.100.200803031447`.

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTIONS>
<!-- Message Definition File Import Options -->
<!-- Import Options for C -->
  <C>
    <!-- COMPILER_NAME = (Msvc|icc|AIXgcc|AIXxlc|OS390) -->
    <COMPILER_NAME>Msvc</COMPILER_NAME>
    <!-- CODEPAGE = (ISO8859-1|Cp037|Cp1252) -->
    <CODEPAGE>Cp1252</CODEPAGE>
    <!-- FLOATING_POINT_FORMAT = (IEEE Extended INTEL|
         IEEE Extended AIX|IEEE Extended OS/390|
         IEEE Non-Extended|IBM 390 Hexadecimal) -->
    <FLOATING_POINT_FORMAT>IEEE Extended INTEL</FLOATING_POINT_FORMAT>
    <!-- BYTE_ORDER = (Little Endian|Big Endian) -->
    <BYTE_ORDER>Little Endian</BYTE_ORDER>
    <!-- ADDRESS_SIZE = (32|64) -->
    <ADDRESS_SIZE>32</ADDRESS_SIZE>
    <!-- SIZE_OF_LONG_DOUBLE = (64|128) -->
    <!-- NOTE: 128 is not supported; therefore 64 is always the value -->
    <SIZE_OF_LONG_DOUBLE>64</SIZE_OF_LONG_DOUBLE>
    <!-- PACK_LEVEL = (1|2|4|8|16) -->
    <PACK_LEVEL>8</PACK_LEVEL>
    <!-- SIZE_OF_ENUM = (1|2|4|5) -->
    <SIZE_OF_ENUM>5</SIZE_OF_ENUM>
    <!-- STRING ENCODING = SPACE | NULL) -->
    <!-- NOTE: SPACE = Fixed length strings, NULL = Null terminated strings -->
    <STRING_ENCODING>SPACE</STRING_ENCODING>
    <!-- STRING PADDING CHARACTER  = (SPACE|NUL|'c'|"c"|0xYY|YY|U+xxxx)-->
    <!-- Note: Only used for Fixed Length strings -->
    <STRING_PADDING_CHARACTER>SPACE</STRING_PADDING_CHARACTER>
    <!-- PRESERVE_CASE_IN_VARIABLE_NAMES = (true|false) -->
    <PRESERVE_CASE_IN_VARIABLE_NAMES>true</PRESERVE_CASE_IN_VARIABLE_NAMES>
    <!-- INCLUDE_PATH = absolute paths to other include files -->
    <!--  Paths should be separated by the system-dependent path-separator character.
          On UNIX systems, this character is ':'; on Win32 systems it is ';' -->
    <INCLUDE_PATH />
    <!-- SCHEMA TARGET NAMESPACE URI = (... any valid namespace URI or empty) -->
    <SCHEMA_TARGET_NAMESPACE_URI/>
    <!-- MESSAGE_PREFIX = (msg_ ... any string including empty string) -->
    <MESSAGE_PREFIX>msg_</MESSAGE_PREFIX>
    <!-- PRE_PROCESSING_OPTION = (none|ale_idoc|file_idoc) -->
    <PRE_PROCESSING_OPTION>none</PRE_PROCESSING_OPTION>
  </C>
<!-- Import Options for COBOL -->
  <COBOL>
    <!-- PLATFORM_SELECTION = (0:"Win32"|1:"AIX"|2:"z/OS") -->
    <PLATFORM_SELECTION>Win32</PLATFORM_SELECTION>
    <!-- CODEPAGE = (ISO8859_1|037) -->
```

```
    <CODEPAGE>ISO8859_1</CODEPAGE>
    <!-- FLOATING_POINT_FORMAT = (IEEE Non-Extended|IBM 390 Hexadecimal) -->
    <FLOATING_POINT_FORMAT>IEEE Non-Extended</FLOATING_POINT_FORMAT>
    <!-- ENDIAN = (Big|Little) -->
    <ENDIAN>Little</ENDIAN>
    <!-- EXT_DECIMAL_SIGN = (ASCII|EBCDIC|EBCDIC Custom) -->
    <EXT_DECIMAL_SIGN>ASCII</EXT_DECIMAL_SIGN>
    <!-- TRUNC = (STD|OPT|BIN) -->
    <TRUNC>STD</TRUNC>
    <!-- NSYMBOL = (DBCS|NATIONAL) -->
    <NSYMBOL>NATIONAL</NSYMBOL>
    <!-- QUOTE = (SINGLE|DOUBLE) -->
    <QUOTE>DOUBLE</QUOTE>
    <!-- CREATE_DEFAULT_VALUES_FROM_INITIAL_VALUES = (true|false) -->
    <CREATE_DEFAULT_VALUES_FROM_INITIAL_VALUES>false</CREATE_DEFAULT_
            VALUES_FROM_INITIAL_VALUES>
    <!-- CREATE FACETS FROM LEVEL 88 VALUE CLAUSES = (true|false) -->
    <CREATE_FACETS_FROM_LEVEL_88_VALUE_CLAUSES>false</CREATE_FACETS_
            FROM_LEVEL_88_VALUE_CLAUSES>
    <!-- PRESERVE_CASE_IN_VARIABLE_NAMES = (true|false) -->
    <PRESERVE_CASE_IN_VARIABLE_NAMES>true</PRESERVE_CASE_IN_VARIABLE_NAMES>
    <!--  CREATE NULL VALUES FOR FIELDS = (true|false) -->
    <CREATE_NULL_VALUES_FOR_FIELDS>false</CREATE_NULL_VALUES_FOR_FIELDS>
    <!-- NULL CHARACTER  = (SPACE|NUL|'c'|"c"|0xYY|YY|U+xxxx)-->
    <NULL_CHARACTER>SPACE</NULL_CHARACTER>
    <!-- STRING PADDING CHARACTER  = (SPACE|NUL|'c'|"c"|0xYY|YY|U+xxxx)-->
    <!-- Note: Only used for Fixed Length strings -->
    <STRING_PADDING_CHARACTER>SPACE</STRING_PADDING_CHARACTER>
    <!-- SCHEMA TARGET NAMESPACE URI = (... any valid namespace URI or empty) -->
    <SCHEMA_TARGET_NAMESPACE_URI/>
    <!-- MESSAGE_PREFIX = (msg_ ... any string including empty string) -->
    <MESSAGE_PREFIX>msg_</MESSAGE_PREFIX>
  </COBOL>
<!-- Import Options for XML Schema in general -->
  <XSD>
    <!-- MSG = (elements|types|both) -->
    <!-- Create messages from imported complex global elements, -->
    <!-- or from imported global complex types, or both -->
    <MSG>elements</MSG>
  </XSD>
<!-- Import Options for XML Schema when importing into a message set
       that does NOT support namespaces -->
  <XSD_NO_NS>
    <!-- IMPORT = (modify|reject|accept) -->
    <IMPORT>modify</IMPORT>
    <!-- REDEFINE = (modify|reject|accept) -->
    <REDEFINE>modify</REDEFINE>
    <!-- LIST = (modify|reject|accept) -->
    <LIST>modify</LIST>
    <!-- UNION = (modify|reject|accept) -->
    <UNION>modify</UNION>
    <!-- ABSTRACT_CT = (modify|reject|accept) -->
    <ABSTRACT_CT>modify</ABSTRACT_CT>
    <!-- ABSTRACT_ELEMENT = (modify|reject|accept) -->
    <ABSTRACT_ELEMENT>modify</ABSTRACT_ELEMENT>
    <!-- XSD_PREFIX = (xsi|... any other prefix) -->
    <XSD_PREFIX>xsi</XSD_PREFIX>
    <!-- This is where you list the additional uri/prefix pairs. -->
    <!-- URI prefix pairs can be listed as follows:  -->
    <!-- <URI_PREFIX_PAIRS uri="http://www.ibm.com" prefix="ibm" /> -->
    <!-- <URI_PREFIX_PAIRS uri="http://www.eclipse.org" prefix="eclipse"/> -->
  </XSD_NO_NS>
</OPTIONS>
```

mqsicreatemsgdefsfromwsdl command

Use the mqsicreatemsgdefsfromwsdl command to import a single WSDL definition.

Supported platforms:

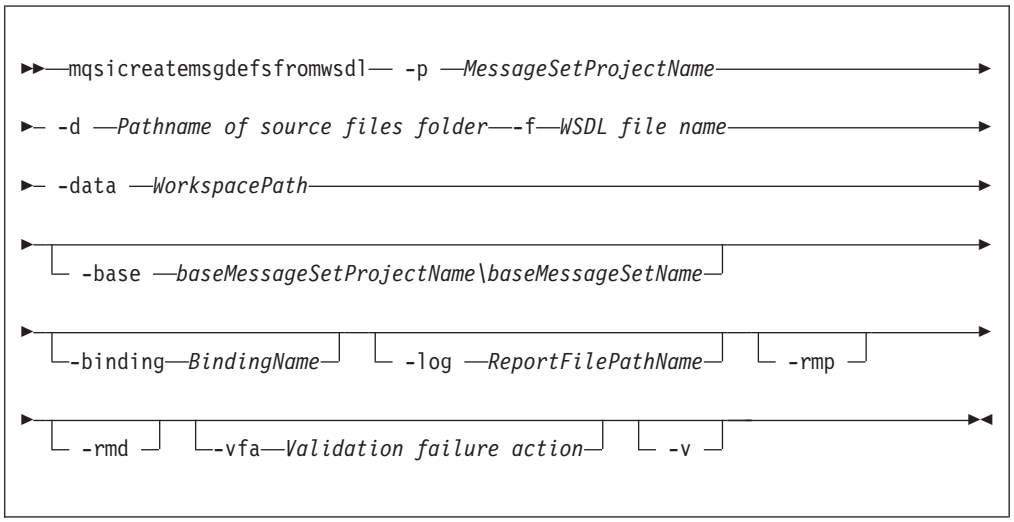
- Windows
- Linux on x86

Purpose:

If the WSDL is split into multiple files then the file specified must contain the WSDL service definition or binding definition. The WS-I validator can be run automatically on the imported WSDL under the control of the -vfa flag.

1. Ensure that only the files that are required for the WSDL definition you are importing exist in the directory and subdirectory structure. One of the actions the mqsicreatemsgdefsfromwsdl command performs is to copy all the files in the directory and subdirectories into the workspace prior to creating the message definition. Files that are not associated to that WSDL definition but exist in the directory are also copied.
2. If the WSDL definition uses a relative path that includes files outside of the directory or subdirectory structure specified, you must import these files into the workspace before you run the command. Make sure that the relative paths are still valid after importing these files into the workspace.
3. Message sets that are created are namespace enabled.
4. Existing message sets must be namespace enabled and have an XML physical format.
5. If you are creating a new message set for runtime parsing, you should base it on an existing message set which has an XML physical format.

Syntax:



Parameters:

-p *MessageSetProjectName*
(Required) The name of the message set project. If the project exists, it must be namespace-enabled. If the project does not exist, a new namespace-enabled project is created.

-base *baseMessageSetProjectName\baseMessageSetName*
(Optional) If a new message set is to be created, specify the existing message set project and message set on which it is based

-binding *BindingName*
(Optional) The name of a binding to be imported. This parameter is mandatory if the WSDL definition includes more than one binding, but optional if the WSDL definition includes a single binding

-d *Pathname of source files folder*
(Required) The absolute or relative path name of the directory where the top-level WSDL file is located. The top-level WSDL file may contain the entire WSDL definition, or it may be the top of a hierarchy of files, each of which may import further files via import elements. An import element specifies the location of the resource to import with a location attribute

The importer attempts to resolve all relative import locations relative to the specified directory; the importer also attempts to resolve any absolute import locations that it encounters. However, avoid using absolute import locations, because any further imports in the hierarchy must use absolute locations after the first time you specify an absolute location.

-data *WorkspacePath*
(Required) The path of the workspace in which your projects are created.

The workspace location is also the default location for projects. Relative paths are interpreted as being relative to the directory from which the command was started.

-f *<WSDL file name>*
(Required) The file name of the top-level WSDL file to be imported.

Where a path is required to fully identify the filename, the path should be specified using the **-d** parameter.

-log *ReportFilePathName*
(Optional) Absolute or relative path name of the report file; if omitted, the report is written to the default log file and is named *Wsdl-file-name*`.wsdl.report.txt`. *Wsdl-file-name* is the name of the WSDL definition you are importing and it is placed in the directory from which the command is invoked.

-rmd
(Optional) Replaces an existing message definition file of the same name.

Note:

1. If you omit this flag, and a message definition file of the same name exists, a warning is returned.
2. The location of the generated message definition file in the message set is determined by the target namespace.

-rmp
(Optional) Replaces the existing project of the same name.

-v (Optional) Verbose report.

-vfa
(Optional) Validation failure action. Specifies the required action if WS-I compliance checking detects a problem in the WSDL to be imported. The default is set to fail. Select from:

- fail: If the WSDL definition is not WS-I compliant, the import process stops, and errors are written to the log file.
- warn: If the WSDL definition is not WS-I compliant, the import process writes warning errors to the log file.
- ignore: If the WSDL definition is not WS-I compliant, the import process ignores them and informational messages of how this WSDL definition is not compliant to the WS-I profile are written to the logfile.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

Examples:

In the following example, the WSDL document `service.wsdl` which exists in the directory `wsdlfiles`, is to be imported into the project myProject and overwrite the project if it exists.

```
mqsicreatemsgdefsfromwsdl  -p myProject -d .\wsdlfiles -f service.wsdl -rmd -data .\wsdlfilewspc
```

In the following example, the WSDL document `service.wsdl` which exists in the directory `wsdlfiles`, is to be imported to create a new message set project (newProj) based on an existing project (existingProj).

```
mqsicreatemsgdefsfromwsdl  -p newProj -base existingProj -d .\wsdlfiles -f service.wsdl -data .\wsdlfilewspc
```

mqsimigratemfmaps command

Use the mqsimigratemfmaps command to migrate resources to WebSphere Message Broker Version 6.1 from WebSphere Business Integration Message Broker Version 5.0.

Supported platforms:

- Windows
- Linux on x86

Purpose:

The mqsimigratemfmaps command migrates mapping and database definitions to Version 6.1:

- Create `.msgmap` files in Version 6.1 format, based on existing Version 5.0 `.mfmap` files.
- Create `.dbm` files in Version 6.1 format, based on existing Version 5.0 `.xmi` files.

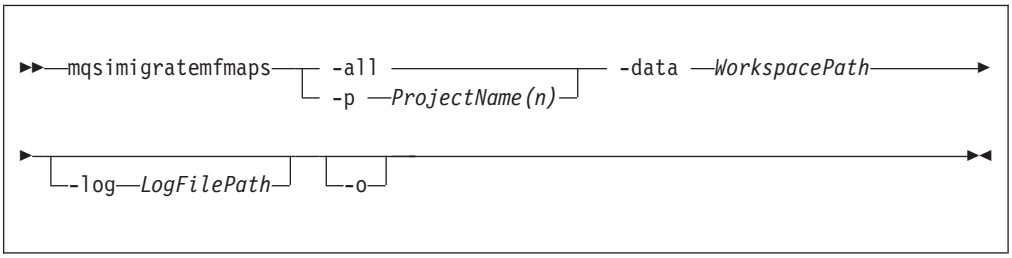
This command is valid only for migration of Version 5.0 resources. You do not need to migrate Version 6.0 maps, which are compatible with Version 6.1. If you have data definitions in your Version 6.0 projects, you must migrate these in the workbench; you cannot use this command.

When the command has completed:

1. The `.xmi` files are deleted from the workspace. If you want to retain these files, you must back up these resources before you migrate them.
2. The `.mfmap` files are left in the workspace, therefore you can run mapping migration again.
3. The `.mfmap` files are not included in builds and are not recognized as valid development resources. They are present in the workspace as simple text files and you can delete them at your own discretion. The Version 5.0 map editor is no longer available; you can view the contents of the `.mfmap` files only with a text editor.
4. When the `.mfmap` files have been migrated, you can edit the `.msgmap` files in the mapping editor.

The command executable file is located in the package group installation directory; the default location is `/opt/IBM/WMBT610` on Linux on x86, `C:\Program Files\IBM\WMBT610` on Windows 32–bit editions, and `C:\Program Files(x86)\IBM\WMBT610` on Windows 64–bit editions.

Syntax:



Parameters:

-all
(Required) All project names in the workspace are checked and migrated.

You must specify only one of all or p.

-p *ProjectName(n)*
(Required) The specific project name, or names, in the workspace to be checked and migrated.

You must specify only one of all or p.

-data *WorkspacePath*
(Required) The path of the workspace in which your projects are created.

The workspace location is also the default location for projects. Relative paths are interpreted as being relative to the directory from which the command was started.

-log *Logfilepath*
(Optional) Alternative report file path.

If you do not specify an alternative path, this command generates a detailed migration log in the folder from which you ran the command. The log file has

the default name of `mqsimigratemfmaps.report.txt`. If the file already exists, new information is appended to the existing content.

-o (Optional) Overwrite files that already exist.

This parameter is valid only for mapping files. It is ignored for data definition files.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

Examples:

The following example migrates the `sample` project in the `c:\wmqi\test` directory.

```
mqsimigratemfmaps -p sample -data c:\wmqi\test
```

The following example migrates all the mapping definitions that are defined in the project MyFlowProject. It also migrates the database definitions in the project and creates a new data design project named MyFlowProject_DDP. If you have defined duplicate database definitions in this project, each duplicate definition is created in a new data design project named MyFlowProject_DDP2, MyFlowProject_DDP3, and so on. The migration process is recorded in the log file `c:\temp\migration.log`.

```
mqsimigratemfmaps -p MyFlowProject -data c:\mbtk\workspace -log c:\temp\migration.log
```

The following example migrates the mappings that you have defined in all the projects in the directory `c:\mbtk\workspace`. If any of the projects contain data definitions, these are also migrated. New data design projects are created with names that are based on the project name, as shown in the previous example. The **-log** parameter is not specified, therefore the log file `mqsimigratemfmaps.report.txt` is created in the directory from which the command is started.

```
mqsimigratemfmaps -all -data c:\mbtk\workspace
```

Runtime commands

This topic is a container for the runtime commands of WebSphere Message Broker.

These commands are available on a machine where any of the runtime components are installed

See “Commands” on page 369 for a list of all the WebSphere Message Broker commands.

mqsibackupconfigmgr command

Use the mqsibackupconfigmgr command to back up the Configuration Manager repository.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS using BIPBUCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

This command backs up the Configuration Manager repository. You ca use the backed up data to restore the repository for migration or recovery purposes.

Usage notes:

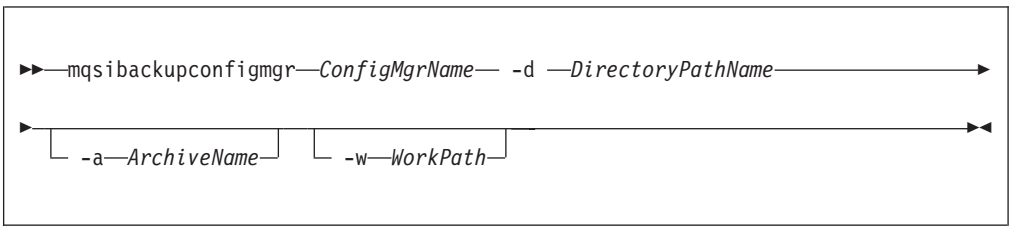
- Before you run this command you must stop the Configuration Manager.
- If you plan to restore the Configuration Manager data repository on z/OS, and the platform from which it was originally backed up was not z/OS, or the other way round, you must take a copy of the file:

```
<Configuration
Manager directory>/components/<component name>/<directory name>/service.properties
```

  You must keep this file with the compressed `.zip` file that is produced by the mqsibackupconfigmgr command. After you have run the mqsirestoreconfigmgr command, copy the file to the equivalent place in the restored Configuration Manager data repository.
- If you specified the **-w** (workpath location) option on the mqsicreateconfigmgr command when you created the Configuration Manager, you must specify the **-w** (workpath location) option on the mqsibackupconfigmgr command.

  If, however, you did not specify the -w option on the mqsicreateconfigmgr command, the Configuration Manager uses the default workpath location; in which case, the **-w** option on the mqsibackupconfigmgr command is optional.

Syntax:



Parameters:

ConfigMgrName
(Required) The name of the Configuration Manager.

-d *DirectoryPathName*
(Required) The directory where the backup 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`.

-w *WorkPath*
(See the Usage notes section above for information about when this option is required.) Specifies the path for the Configuration Manager repository. If the wrong work path location is specified, the following error message is returned:

```
BIP1077S when taking a backup on configmgr using the command - mqsibackupconfigmgr
```

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID must be a member of mqbrkrs.

On all platforms, the user ID must have the authority to read, at least, the directory specified by `%MQSI_WORKPATH%`, and to write to the directory in which the backup is to be placed.

Examples:

```
mqsibackupconfigmgr ConfigMgr -d c:\mqsitest\db –a archive1
```

mqsichangebroker command

Use the mqsichangebroker command to change one or more of the configuration parameters of the broker.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command either as a console command, or by customizing and submitting BIPCHBK; see “Contents of the broker PDSE” on page 679

Purpose:

You can modify the value of many, but not all, of the parameters that you set when you created the broker. For example, after you change a password, you must run the mqsichangebroker command. You can also use this command to set the **UserExitPath** property.

You must stop the broker before you issue this command, and restart the broker for the changes to take effect:

- On Windows, Linux, and UNIX systems, use the mqsistop and mqsistart commands to stop and restart the broker.

- On z/OS, you must have started the original broker control process by using the `/S` option. You must stop the broker components using the `/F broker, PC` option and start the broker components again using the `/F broker, SC` option.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsichangebroker command - Windows, Linux, and UNIX systems"
- "mqsichangebroker command - z/OS" on page 411

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

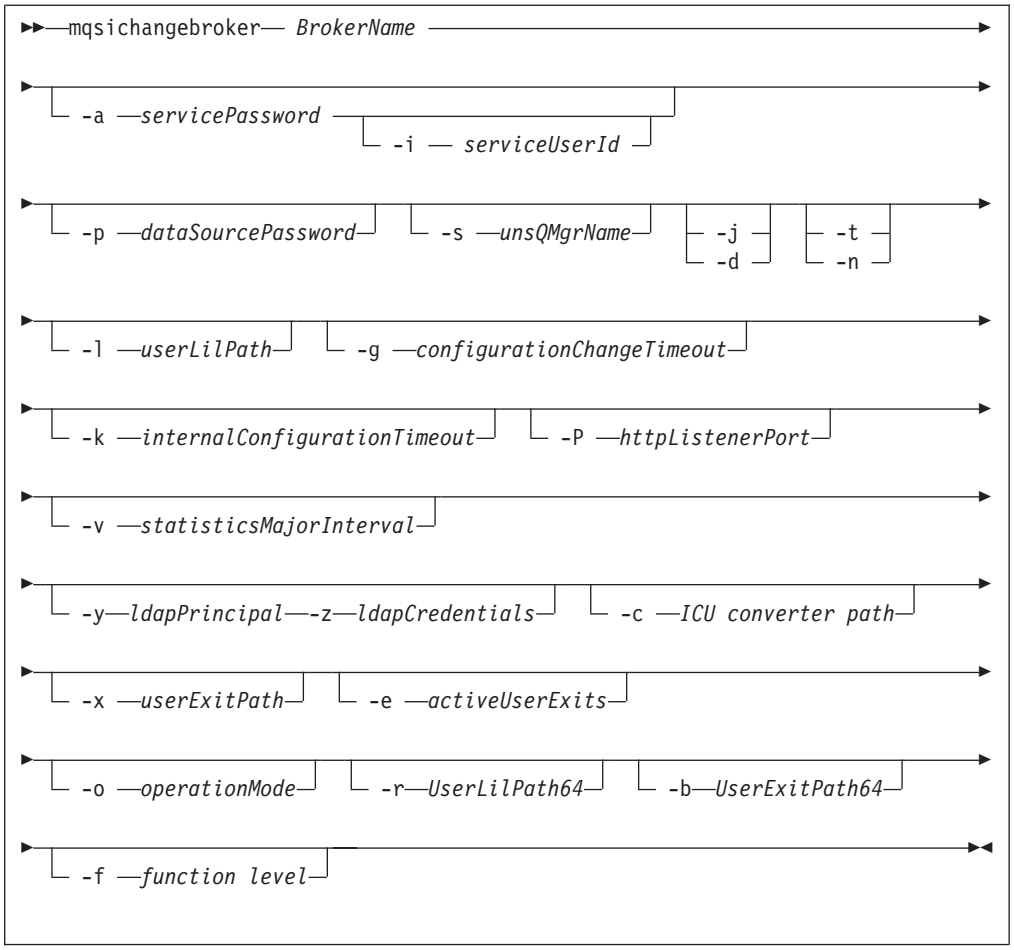
On z/OS systems, the user ID used to run this command must be a member of a group that has READ and WRITE access to the component directory and access to WebSphere MQ.

Using LDAP: Ensure that the registry is appropriately secured to prevent unauthorized access. The setting of **LdapPrincipal** and **LdapCredentials** parameters on mqsichangebroker is not required for correct operation of the broker. The password is not stored in clear text in the file system.

mqsichangebroker command - Windows, Linux, and UNIX systems:

Use the mqsichangebroker command on Windows, Linux, and UNIX systems to modify your broker.

Syntax:



Parameters:

BrokerName

(Required) This parameter must be the first parameter. Specify the name of the broker to modify.

-a *servicePassword*

(Optional) The password for the *serviceUserId*.

For compatibility with existing systems, you can 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 compatibility with Windows systems, but is not used in relation to *serviceUserId*. **-a** is used as a default only if **-p** is not specified. See the **-p** parameter description for further details.

If you have created your broker to use the *serviceUserId* and *servicePassword* for database access, ensure that you update both instances of the password on this command by specifying the **-p** *dataSourcePassword* parameter. Specify the **-p** *dataSourcePassword* parameter in the following circumstances:

- You omitted the **-u** *dataSourceUserId* and **-p** *dataSourcePassword* parameters.
- You included the **-u** *dataSourceUserId* and **-p** *dataSourcePassword* parameters, but provided the same user ID and password for the service user ID using **-a** *servicePassword* and **-i** *serviceUserId*.

To complete a password change successfully:

- Stop the broker.
- Change the password using the appropriate operating system function.
- Use the mqsichangebroker command to update all parameters that reference this same password.
- Restart the broker.

-i *serviceUserId*

(Optional) The user ID under which the broker runs. You must also change the password (**-a**) if you change this value.

You can specify the *serviceUserId* in any valid user name syntax. On Windows systems, valid formats are:

- `\\server\username`
- `.\username`
- `username`

On Linux and UNIX systems, only the last format, `username`, is valid.
Do not use a domain name as part of the *serviceUserId* parameter.
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* that you specify must be a member of the mqbrkrs local group. On Windows systems, 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 Message Broker has been installed), and the working directory (if specified by the **-w** parameter).
On Windows systems, if you specify that the broker is to run as a WebSphere MQ trusted application (**-t** parameter), you must also add the service user ID to the mqm group. On Linux and UNIX systems, specify the *serviceUserId* as `mqm` if you set the **-t** parameter.
The security requirements for the *serviceUserId* are described in "Security requirements for Windows platforms" on page 725 and in "Security requirements for Linux and UNIX platforms" on page 724.

-p *dataSourcePassword*

(Optional) The password of the user ID with which the databases that contain broker tables and user data are to be accessed.

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, you can specify **-p** as an empty string (two double quotation marks, ""). In this case, DB2 grants WebSphere Message 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 Message Broker, creating the most secure configuration.

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 and its database access, update both instances at the same time. (See the description of the **-a** parameter for further details.)

-s *unsQMgrName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server.

To remove topic-based security, specify an empty string (two quotation marks, "").

-j (Optional) Publish/subscribe access is enabled for the broker. This parameter is valid only with the **-s** parameter.

-d (Optional) Publish/subscribe access is not enabled for the broker.

-t (Optional) The broker runs as a WebSphere MQ trusted application.

For more details about using WebSphere MQ trusted applications, see the *Intercommunication* section of the WebSphere MQ Version 6 information center online or WebSphere MQ Version 7 information center online.

-n

(Optional) The broker ceases to run as a WebSphere MQ trusted application.

-l *userLilPath*
(Optional) A list of paths (directories) from which the broker loads 32-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Use the **-l** flag for 32–bit LIL files.

On Linux and UNIX systems, directory names are case sensitive, and you must include the names in single quotation marks if they contain mixed case characters.
Do not include environment variables in the path; the broker ignores them.

Create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Message Broker installation directory.
If you specify more than one additional directory, each directory must be separated by the default platform-specific path separator.

The **-l** parameter sets the LIL path for 32-bit execution groups only.
For 64-bit execution groups, which is the default in WebSphere Message Broker Version 6.1, you must append the corresponding library path to the environment variable `MQSI_LILPATH64`.

-g *configurationChangeTimeout*
(Optional) The maximum time (in seconds) that is allowed for a user configuration request to be processed. It defines the length of time taken within the broker to apply to an execution group a configuration change that you have initiated. For example, if you deploy a configuration from the workbench, the broker must respond to the Configuration Manager within this time.

A message flow cannot respond to a configuration change while it is processing an application message. An execution group that has been requested to change its configuration returns a negative response to the deployed configuration message if any one of its message flows does not finish processing an application message and apply the configuration change within this timeout.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-k *internalConfigurationTimeout*
(Optional) The maximum time (in seconds) that is allowed for an internal

configuration change to be processed. For example, it defines the length of time taken within the broker to start an execution group.

The response time of each execution group differs according to system load and the load of its own processes. The value must reflect the longest response time that any execution group takes to respond. If the value is too low, the broker returns a negative response, and might issue error messages to the local error log.
Specify the value in seconds, in the range 10 to 3600. The default is 60.
For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-P *httpListenerPort*
(Optional) Enter the number of the port on which the Web services support is listening.

The broker starts this listener when a message flow that includes HTTP nodes or Web services support is started; the default is 7080.

Ensure that the port that you specify has not been specified for any other purpose.

-v *statisticsMajorInterval*
(Optional) The time interval (in minutes) at which statistics and accounting archive records are written. For internal accounting, the valid range is from 10 to 14400 minutes; the default value is 60

An interval of zero minutes indicates that the operating system has an external method of notification (the ENF timer) and is not using an internal timer within WebSphere Message Broker.

-y *ldapPrincipal*
(Optional, 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, but mandatory when *ldapPrincipal* is provided.) The user password for access to LDAP.

-c *ICU converter path*
(Optional) A delimited set of directories to search for additional code page converters. On Windows systems, the delimiter is a semicolon (;). On UNIX and Linux systems, the delimiter is a colon (:).

The code page converters must be either of the form `icudt32_codepagename.cnv`, or in an ICU data package called `icudt32.dat`.

Do not use this parameter to set the converter path if you are using a converter that matches one of the built-in converters that are provided with Version 6.0, and that converter is the local code page for the broker. Use the **ICU_DATA** environment variable instead.

-x *userExitPath*
(Optional) The path that contains the location of all user exits to be loaded for 32-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

-e *activeUserExits*
(Optional) Active user exits. By default, user exits are inactive. Adding a *userExit* name to this colon-separated list changes its default state to active for

this broker. You can use the mqsichangeflowuserexits command to override the default state at the execution group or message flow level. If you specify a user exit name, and no library is found to provide that user exit when the execution group starts, a BIP2314 message is written to the system log, and the execution group fails to start.

-o *operationMode*
(Optional) Use this parameter to set the mode of your broker; see Operation modes. Valid values that you can set are enterprise (the full package, enterprise mode), starter (Starter Edition mode), and adapter (Remote Adapter Deployment mode). The default is enterprise unless you have the Trial Edition, when the default is trial. If no **-o** parameter is set, the default is set automatically.

-r *UserLilPath64*
(Optional) A list of paths (directories) from which the broker loads 64-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Because you can define 32-bit or 64-bit LIL files on the following platforms:

- AIX
- HP-UX on PA-RISC
- Linux on x86-64
- Solaris on SPARC

use the following flags:

- **-r** to set the path where the 64-bit LIL files are stored.
- **-l** to set the path where the 32-bit LIL files are stored.

On Linux and UNIX systems, directory names are case sensitive, and you must include the names in single quotation marks if they contain mixed case characters.
Do not include environment variables in the path; the broker ignores them.

Create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Message Broker installation directory.
If you specify more than one directory, separate directories by a semicolon (`;`) on Windows systems, or a colon (`:`) on Linux and UNIX systems.

-b *UserExitPath64*
(Optional) The path that contains the location of all user exits to be loaded for 64-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

Because you can run 32-bit or 64-bit execution groups on the following platforms:

- AIX
- HP-UX on PA-RISC
- Linux on x86-64
- Solaris on SPARC

use the following flags:

- **-b** to set the path where the 64-bit user exits are stored.
- **-x** to set the path where the 32-bit user exits are stored.

-f *function level*
(Optional) Use this parameter to enable function that becomes available in

WebSphere Message Broker fix packs. Valid values that you can set are all, which enables all functions, or a version string of the form V.R.M.F (for example, 6.1.0.3), which indicates the maximum level of feature function to enable.

This command does not disable all new features, and it is not possible to use this flag to run the broker at a different major version.

To change other broker properties, first delete and re-create the broker, and then use the workbench to redeploy the broker's configuration. To change the user ID that is used for database access, see "Authorizing access to broker and user databases" on page 122.

Examples:

Windows, Linux, and UNIX systems:

Change the User Name Server that is associated with the broker by specifying a different queue manager:

```
mqsichangebroker WBRK_BROKER -s WBRK_UNS_QM
```

Remove topic-based security from the broker by removing the associated User Name Server:

```
mqsichangebroker WBRK_BROKER -s ""
```

Define the path to user-defined exits:

```
mqsichangebroker WBRK_BROKER -x /opt/3rdparty/wmbexit
```

Enable the new function that is supplied in WebSphere Message Broker V6.1.0.3:

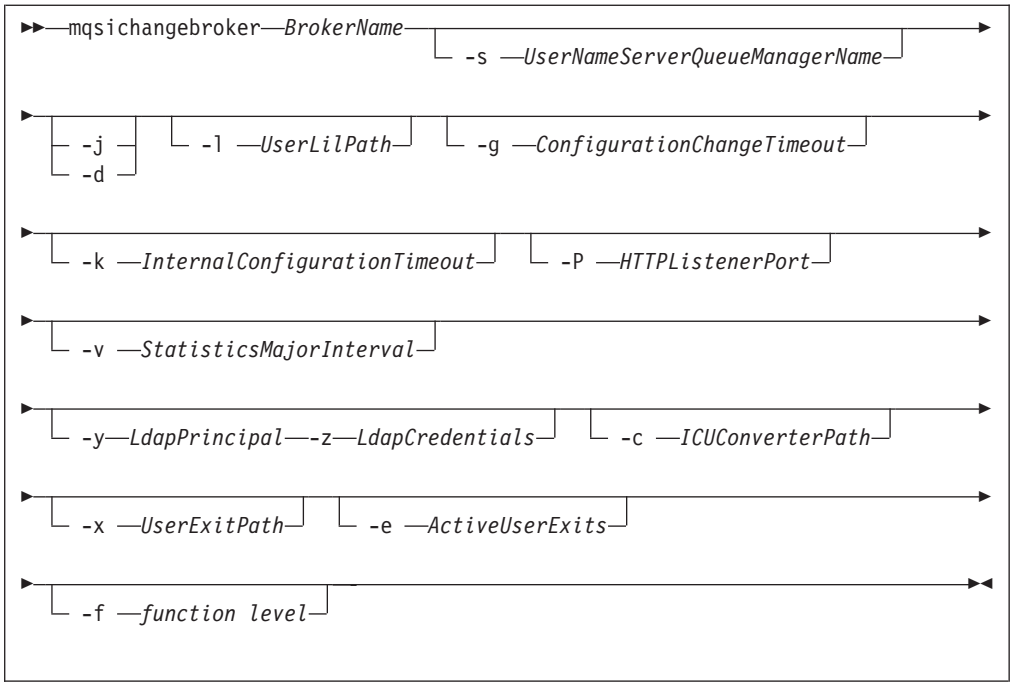
```
mqsichangebroker WBRK_BROKER -f 6.1.0.3
```

mqsichangebroker command - z/OS:

Use the mqsichangebroker command on z/OS to modify your broker.

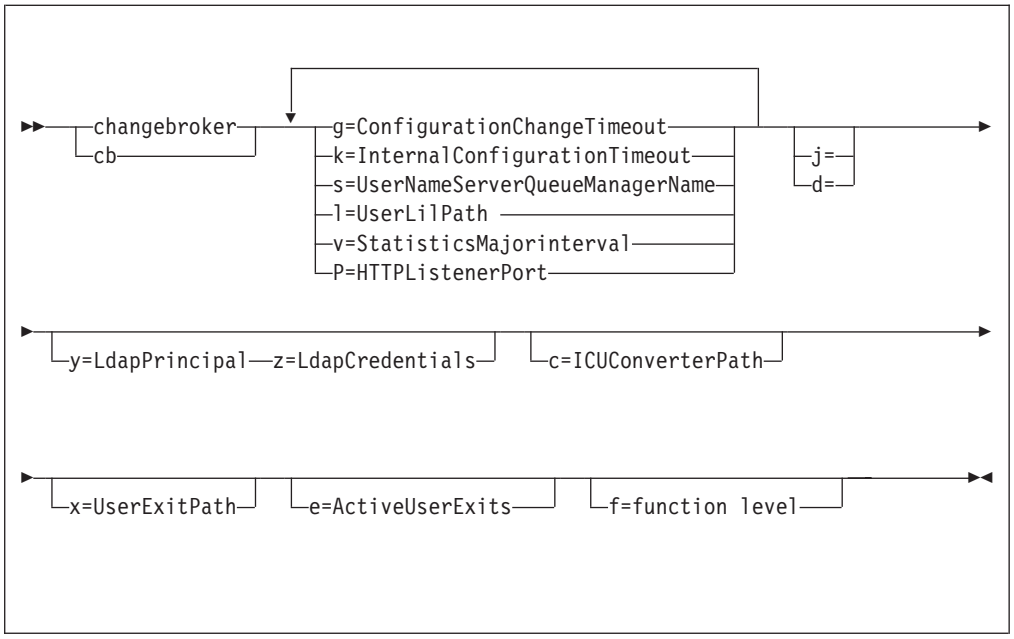
Syntax:

z/OS command - BIPCHBK:



z/OS console command:

Synonym: cb



Parameters:

BrokerName
(Required) This parameter must be the first parameter. Specify the name of the broker to modify.

This parameter is implied in the console form of the command.

-s *UserNameServerQueueManagerName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server.

To remove topic-based security, specify an empty string (two quotation marks, ″″).

This name is case sensitive; enclose the names in single quotation marks if they are in mixed case.

-j (Optional) Publish/subscribe access is enabled for the broker. This parameter is valid only with the **-s** parameter.

-d (Optional) Publish/subscribe access is not enabled for the broker.

-l *UserLilPath*
(Optional) A list of paths (directories) from which the broker loads 32-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Use the **-l** flag for 32–bit LIL files.

This name is case sensitive; enclose the names in single quotation marks if they are in mixed case.
Do not include environment variables in this path; WebSphere Message Broker ignores them.

Create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Message Broker installation directory.
If you specify more than one additional directory, each directory must be separated by the default platform-specific path separator.

-g *ConfigurationChangeTimeout*
(Optional) The maximum time (in seconds) that is allowed for a user configuration request to be processed. It defines the length of time taken within the broker to apply to an execution group a configuration change that you have initiated. For example, if you deploy a configuration from the workbench, the broker must respond to the Configuration Manager within this time.

A message flow cannot respond to a configuration change while it is processing an application message. An execution group that has been requested to change its configuration returns a negative response to the deployed configuration message if any one of its message flows does not finish processing an application message and apply the configuration change within this timeout.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

For information about how to set the value for this timeout, see “Setting configuration timeouts” on page 227.

-k *InternalConfigurationTimeout*
(Optional) The maximum time (in seconds) that is allowed for an internal configuration change to be processed. For example, it defines the length of time taken within the broker to start an execution group.

The response time of each execution group differs according to system load and the load of its own processes. The value must reflect the longest response time that any execution group takes to respond. If the value is too low, the broker returns a negative response, and might issue error messages to the local error log.

Specify the value in seconds, in the range 10 to 3600. The default is 60. For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-P *HTTPListenerPort*
(Optional) Enter the number of the port on which the Web services support is listening.

The broker starts this listener when a message flow that includes HTTP nodes or Web services support is started; the default is 7080.

Ensure that the port that you specify has not been specified for any other purpose.

-v *StatisticsMajorInterval*
(Optional) Specify the interval (in minutes) at which statistics and accounting archive records are to be written. The valid range is from 10 to 14400 minutes; the default value is 60.

An interval of zero minutes indicates that the operating system has an external method of notification (the ENF timer) and is not using an internal timer within WebSphere Message Broker.

-y *LdapPrincipal*
(Optional, 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, but mandatory when *ldapPrincipal* is provided.) The user password for access to LDAP.

-c *ICUConverterPath*
(Optional) A delimited set of directories to search for additional code page converters; the delimiter is a period (.).

The code page converters must be either of the form `icudt32_codepagename.cnv`, or in an ICU data package called `icudt32.dat`.

Do not use this parameter to set the converter path if you are using a converter that matches one of the built-in converters that are provided with Version 6.0, and that converter is the local code page for the broker. Use the **ICU_DATA** environment variable instead.

-x *UserExitPath*
(Optional) The path that contains the location of all user exits to be loaded for 32-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

-e *ActiveUserExits*
(Optional) Active user exits. By default, user exits are inactive. Adding a *userExit* name to this colon-separated list changes its default state to active for this broker. You can use the mqsichangeflowuserexits command to override the default state at the execution group or message flow level. If you specify a user exit name, and no library is found to provide that user exit when the execution group starts, a BIP2314 message is written to the system log, and the execution group fails to start.

-f *function level*
(Optional) Use this parameter to enable function that becomes available in WebSphere Message Broker fix packs. Valid values that you can set are all,

which enables all functions, or a version string of the form V.R.M.F (for example, 6.1.0.3), which indicates the maximum level of feature function to enable.

This command does not disable all new features, and it is not possible to use this flag to run the broker at a different major version.

To change other broker properties, first delete and re-create the broker, and then use the workbench to redeploy the broker's configuration. To change the user ID that is used for database access, see "Authorizing access to broker and user databases" on page 122.

Examples:

z/OS: You must use a comma between each command option, as shown in these examples.

Change the configuration timeout parameters:

```
F MQP1BRK,cb g=100,k=200
```

Specify both the **x** and **e** parameters to set the user exit path and set an exit to active state:

```
/f MA05BRK,cb x='/u/test/wbi/MsgFlowTrackingUserExit/zOS',e='MqsiStrUserExit02:MqsiStrUserExit03'
```

Enable the new function that is supplied in WebSphere Message Broker V6.1.0.3:

```
F MQP1BRK,cb f=6.1.0.3
```

mqsichangeconfigmgr command

Use the mqsichangeconfigmgr command to changes one or more of the properties of the Configuration Manager.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways; as a console command, or by customizing and submitting BIPCHCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

You can modify most of the parameters that you set when you created the Configuration Manager; you cannot change the queue manager name or the parameters that are related to migration.

You must stop the Configuration Manager before you can issue this command, and restart the Configuration Manager for the changes to take effect:

- On Windows, Linux, and UNIX systems, use the mqsistop and mqsistart commands.
- On z/OS, you must have started the original Configuration Manager control process using the `/S` option. You must stop the Configuration Manager components using the `/F Configuration Manager, PC` option and start the Configuration Manager components again using the `/F Configuration Manager, SC` option.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsichangeconfigmgr command - Windows"
- "mqsichangeconfigmgr command - Linux and UNIX systems" on page 418
- "mqsichangeconfigmgr command - z/OS" on page 419

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

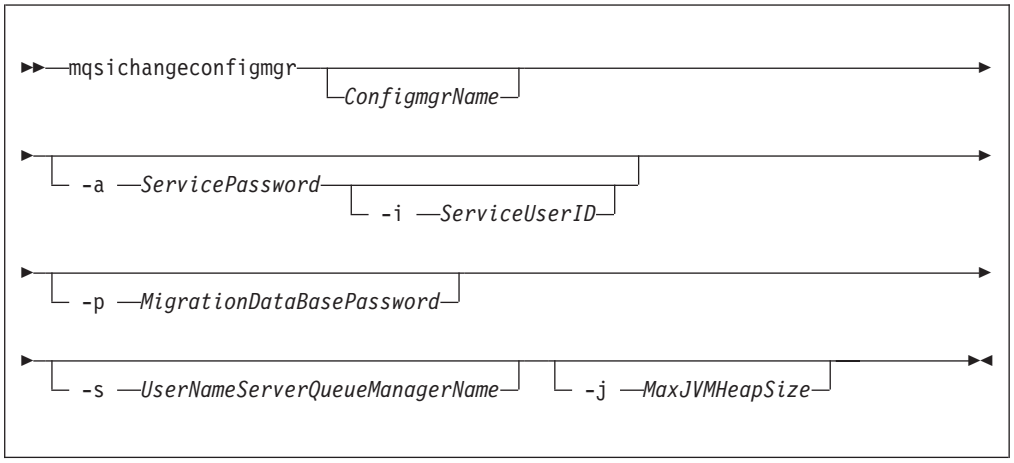
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsichangeconfigmgr command - Windows:

Syntax:



Parameters:

ConfigmgrName
(Optional) The name, which is not case sensitive, of the Configuration Manager that you want to change.

The default name on Windows, if this parameter is not specified, is 'ConfigMgr'.

-i *ServiceUserID*
(Optional) 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 user name 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 (either direct or indirect) of the local group **mqbrkrs**. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Message 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 725.

-a *ServicePassword*
(Optional) 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, you 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.

-p *MigrationDataBasePassword*
(Optional) 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 256 MB is used.

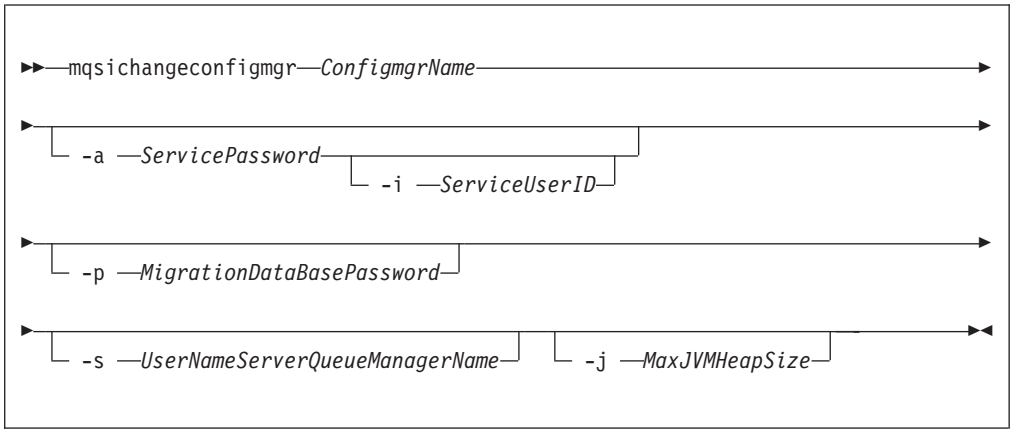
If you want to change other properties, you must delete and recreate the Configuration Manager.

Examples:

```
mqsichangeconfigmgr CMGR01 -i ADMIN
mqsichangeconfigmgr CMGR01 -s ""
```

mqsichangeconfigmgr command - Linux and UNIX systems:

Syntax:



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager that you want to change.

This must be the first parameter specified and the name is case-sensitive.

-i *ServiceUserID*

(Optional) This can be specified in any valid user name syntax for the platform.

The `ServiceUserID` specified must be a member (either direct or indirect) of the local group **mqbrkrs**. The `ServiceUserID` must also be authorized to access the home directory (where WebSphere Message 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 Linux and UNIX platforms" on page 724.

-a *ServicePassword*
(Optional) 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, you 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.

-p *MigrationDataBasePassword*
(Optional) 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 256 MB is used.

If you want to change other properties, you must delete and recreate the Configuration Manager.

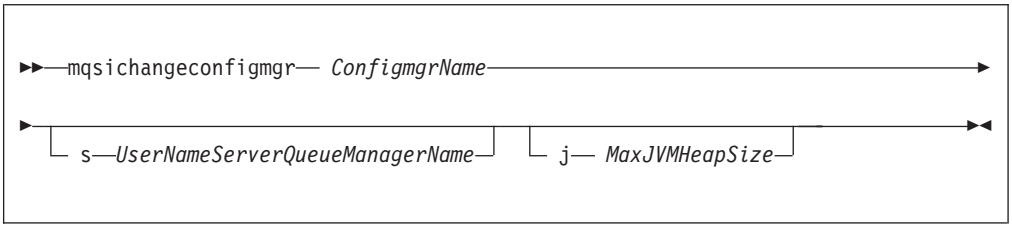
Examples:

```
mqsichangeconfigmgr CMGR01 -i ADMIN
mqsichangeconfigmgr CMGR01 -s ""
```

mqsichangeconfigmgr command - z/OS:

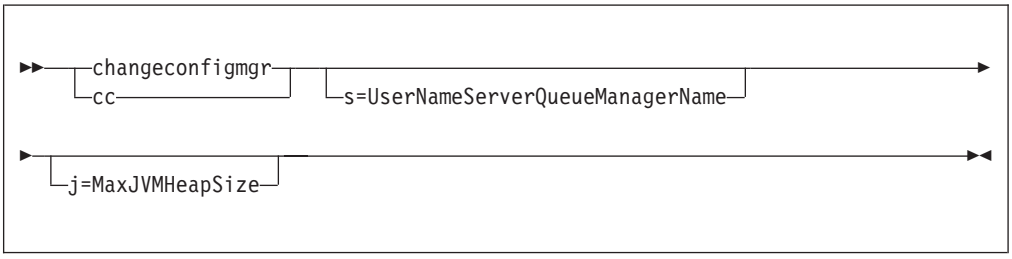
Syntax:

z/OS command - BIPCHCM:



z/OS console command:

Synonym: cc



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager that you want to change. This parameter is implied when using the console command.

This must be the first parameter specified and the name is case-sensitive.

-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 256 MB is used.

If you want to change other properties, you must delete and recreate the Configuration Manager.

Examples:

```
mqsichangeconfigmgr CMGR01 -s ""
```

mqsichangedbimgr command

Use the mqsichangedbimgr command to update the Derby Network Server.

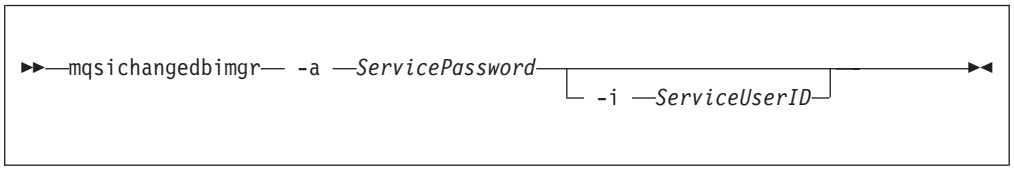
Supported platforms:

- Windows XP and Windows Server 2003

Purpose:

The mqsichangedbimgr command updates the Windows service that runs the Derby Network Server, with a new user password, a new user ID, or both.

Syntax:



Parameters:

-a *ServicePassword*
(Required) The new password for the ServiceUserID. To complete a password change successfully, you must:

- Stop the broker by using the mqsistop command.
- Change the password by using the appropriate Windows facilities.
- Use the appropriate mqsichange commands to update all the components that reference this same ServiceUserID and password.
- Restart the broker by using the mqsistart command.

-i *ServiceUserID*
(Optional) The user ID under which the default database runs. Specify this parameter in any valid user name 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. It can be a direct or indirect member of the group. The ServiceUserID must also be authorized to access the home directory (where you have installed WebSphere Message Broker).

The security requirements for the ServiceUserID are detailed in "Security requirements for Windows platforms" on page 725.

Authorization:

The user ID used to run this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Examples:

The following example changes the password used for the `DatabaseInstanceMgr` service:

```
mqsichangedbimgr -a wbrkpw1
```

mqsichangeflowmonitoring command

Use the mqsichangeflowmonitoring command to enable monitoring of message flows.

Supported operating systems:

- Windows
- Linux and UNIX systems
- z/OS: Run this command either as a console command, or by customizing and submitting BIPCHME; see "Contents of the broker PDSE" on page 679.

Purpose:

Use the mqsichangeflowmonitoring command to carry out the following tasks:

- Activate or deactivate monitoring for a message flow
- Enable or disable an event source within a message flow
- Associate a configurable service with a message flow by setting its `monitoringProfile` property.

This facility can be used to change the state (enabled or disabled) of any event in the message flow, whether it was configured using the node's monitoring properties, or a monitoring profile configurable service. The new state takes effect immediately (but safely).

The state change persists, and so it survives a restart of the message flow or execution group. For an event configured using the node's monitoring properties, the state change is lost if the message flow is redeployed, unless the node property is also changed.

Select the appropriate link for details of the mqsichangeflowmonitoring command on the platform that your enterprise uses:

- "mqsichangeflowmonitoring command - Windows, Linux and UNIX systems"
- "mqsichangeflowmonitoring command - z/OS" on page 424

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

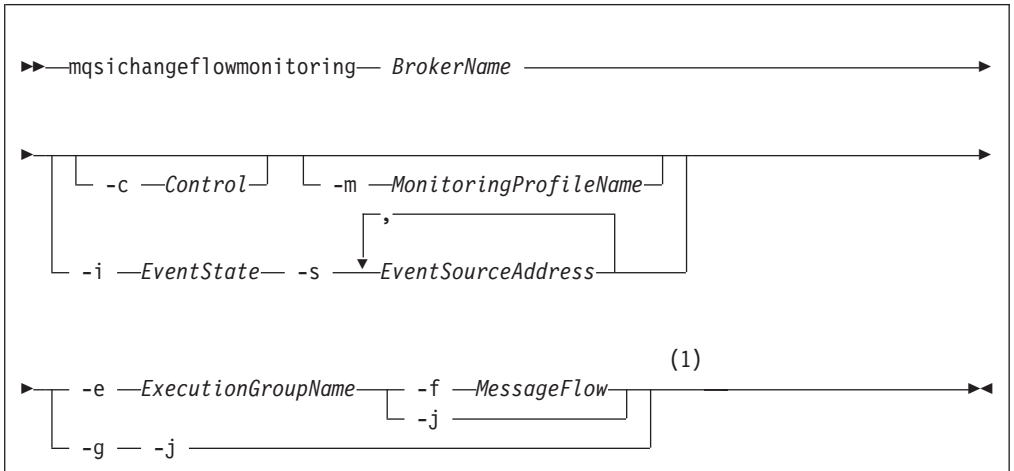
If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

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.

mqsichangeflowmonitoring command - Windows, Linux and UNIX systems:

Syntax:



Notes:

1 See the parameter descriptions for information on valid combinations of the e, f, g, and j parameters.

Parameters:

BrokerName
(Required) Specify the label of the broker to which the message flows that you want to be monitored are deployed.

-c *Control*
(Optional) Specify the string value that controls monitoring for the specified message flows. Possible values are:

active - activate monitoring

inactive - deactivate monitoring

-m *MonitoringProfileName*
(Optional) Specify the name of the monitoring profile which the specified message flows should use.

If there is no monitoring profile with the specified name on the specified broker, the command completes successfully, and the message flows attempt to use the specified monitoring profile. Each message flow logs a warning in the User Trace to indicate that it was instructed to use a nonexistent monitoring profile. No event message is created. If a monitoring profile with the specified name is later deployed to the broker, the message flows do not immediately begin to use it. A refresh of the monitoring state can be triggered by issuing the command again with the –c option to activate or reactivate monitoring.

-i EventState
(Optional) Specify the string value that controls monitoring for the specified event source. Valid only when used with the -e and -f parameters. Possible values are:

enable - enable monitoring for the specified event sources.

disable - disable monitoring for the specified event sources.

-s EventSourceAddress
(Optional) Comma-separated list of the event sources to be enabled or disabled. Valid only when used with the -e and -f parameters. This value takes the form <*node name*>.<*event source*>, where <*event source*> is one of the following values:

'terminal.<*terminal name*>'

'transaction.Start'

'transaction.End'

'transaction.Rollback'

If a message flow contains two or more nodes with identical names, the event sources on those nodes cannot be accurately addressed. If this is attempted, behavior is undefined.

Note: <*node name*> is the label of the node as known by the broker runtime components. If the node is in a subflow the label reflects this. For example, flow A contains an instance of flow B as a subflow labeled 'myB'; flow B contains an instance of a Compute node labeled 'myCompute'. The <*node name*> for the Compute node is 'myB.myCompute'.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group to which the message flows that you want to be monitored are deployed.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

-f *MessageFlow*
(Required) Specify the label for the message flow, for which the monitoring options are to be activated or updated.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

-g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

-j (Required) Specifies that the command applies to **all** message flows that belong to the execution group.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

Examples:

Assign *monitoringProfile1* to *messageFlow1* in execution group *default*:

```
mqsichangeflowmonitoring WBRK_BROKER -e default
     -f messageFlow1 -m monitoringProfile1
```

Activate monitoring for all message flows in all execution groups:

```
mqsichangeflowmonitoring WBRK_BROKER -c active -g -j
```

Enable individual event sources:

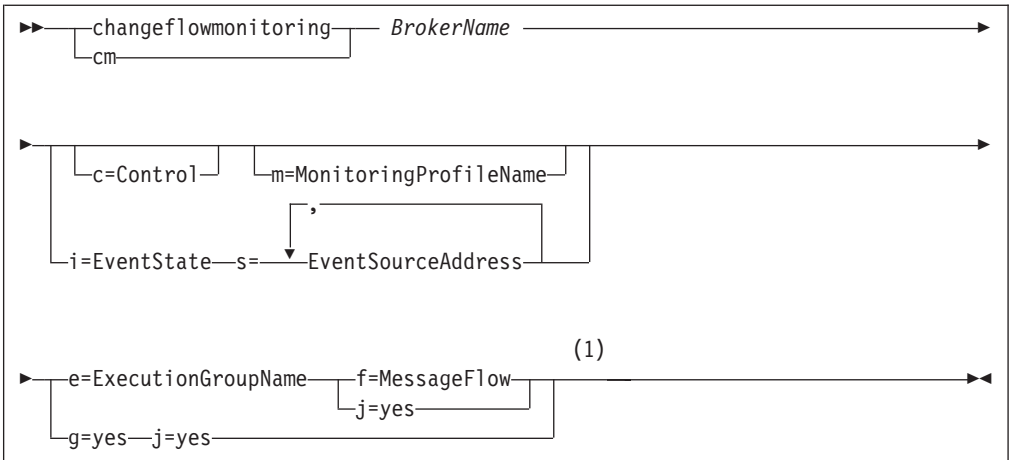
```
mqsichangeflowmonitoring WBRK_BROKER
     -e default
     -f myMessageFlow
     -s "SOAP Input1.terminal.out,MQOutput1.terminal.in"
     -i enabled
```

mqsichangeflowmonitoring command - z/OS:

Syntax:

z/OS console command:

Synonym: cm



Notes:

1 See the parameter descriptions for information on valid combinations of the e, f, g, and j parameters.

Parameters:

BrokerName
(Required) Specify the label of the broker to which the message flows that you want to be monitored are deployed.

This parameter is implied on the console form of the command.

c=Control
(Optional) Specify the string value that controls monitoring for the specified message flows. Possible values are:

active - activate monitoring

inactive - deactivate monitoring

m=MonitoringProfileName
(Optional) Specify the name of the monitoring profile which the specified message flows should use.

If there is no monitoring profile with the specified name on the specified broker, the command completes successfully, and the message flows attempt to use the specified monitoring profile. Each message flow logs a warning in the User Trace to indicate that it was instructed to use a nonexistent monitoring profile. No event message is created. If a monitoring profile with the specified name is later deployed to the broker, the message flows do not immediately begin to use it. A refresh of the monitoring state can be triggered by issuing the command again with the –c option to activate or reactivate monitoring.

i=EventState
(Optional) Specify the string value that controls monitoring for the specified event source. Valid only when used with the -e and -f parameters. Possible values are:

enable - enable monitoring for the specified event sources.

disable - disable monitoring for the specified event sources.

s=EventSourceAddress
(Optional) Comma-separated list of the event sources to be enabled or

disabled. Valid only when used with the -e and -f parameters. This value takes the form <*node name*>.<*event source*>, where <*event source*> is one of the following values:

'terminal.<*terminal name*>'

'transaction.Start'

'transaction.End'

'transaction.Rollback'

If a message flow contains two or more nodes with identical names, the event sources on those nodes cannot be accurately addressed. If this is attempted, behavior is undefined.

Note: <*node name*> is the label of the node as known by the broker runtime components. If the node is in a subflow the label reflects this. For example, flow A contains an instance of flow B as a subflow labeled 'myB'; flow B contains an instance of a Compute node labeled 'myCompute'. The <*node name*> for the Compute node is 'myB.myCompute'.

e=ExecutionGroupName
(Required) Specify the name for the execution group to which the message flows that you want to be monitored are deployed.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

f=MessageFlow
(Required) Specify the label for the message flow, for which the monitoring options are to be activated or updated.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

j (Required) Specifies that the command applies to **all** message flows that belong to the execution group.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

Example:

The following example shows the command to activate monitoring on all message flows in all execution groups:

```
F MI10BRK,cm c=active,g=yes,j=yes
```

mqsichangeflowstats command

Use the mqsichangeflowstats command to control the accumulation of statistics about message flow operation.

Supported Platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPCHMS; see “Contents of the broker PDSE” on page 679

Purpose:

Use the mqsichangeflowstats command to:

- Turn on or off accounting and statistics snapshot publication, or archive record output.
- Specify that the command is 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.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsichangeflowstats command - Windows, Linux and UNIX systems”
- “mqsichangeflowstats command - z/OS” on page 430

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

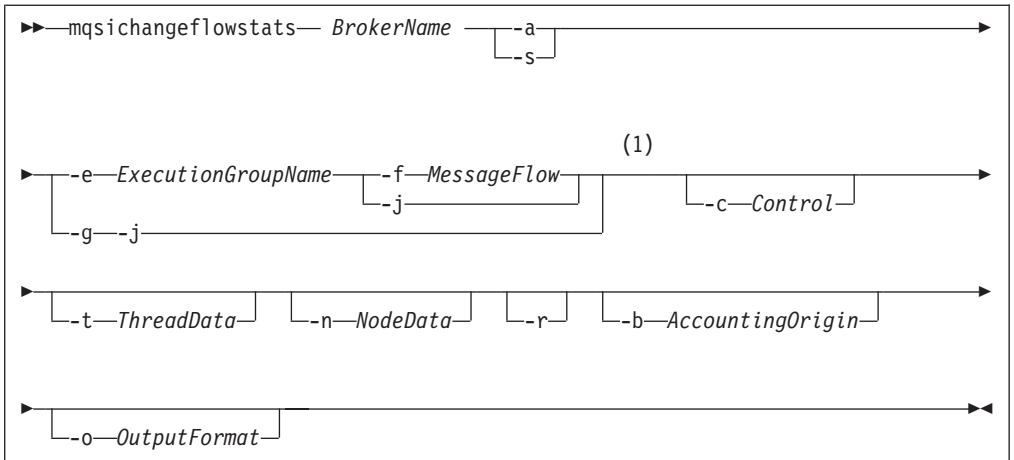
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsichangeflowstats command - Windows, Linux and UNIX systems:

Syntax:



Notes:

1 See the parameter descriptions for information on valid combinations of the e,f, g, and j parameters.

Parameters:

BrokerName
(Required) 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.

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.

You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which accounting and statistics options are to be changed.

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.

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.

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.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

Note: 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) This parameter applies only to archive data and specifies that archive data is to be reset.

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 set only if the internal interval notification mechanism is being used, and not an external method.

-b *AccountingOrigin*
(Optional) Specifies that the environment tree path *Broker.Accounting.Origin* is used to partition the collected statistics into distinct outputs. Possible values are:

- none - do not partition statistics according to accounting origin data
- basic - partition statistics according to accounting origin data

-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.

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
```

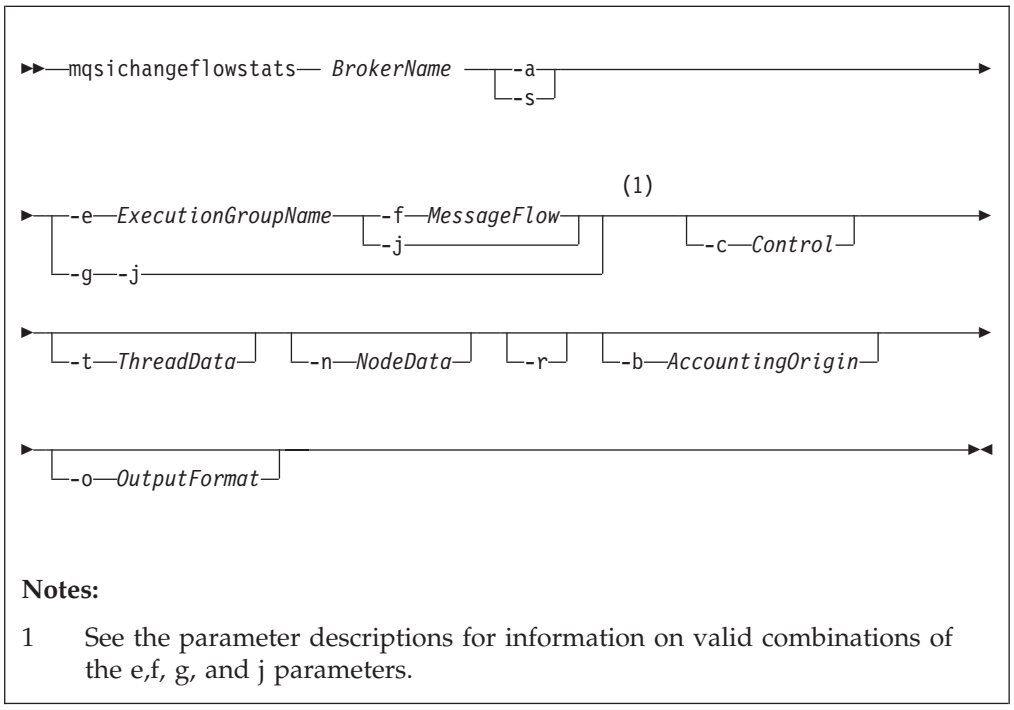
Turn off snapshot data for all message flows in all execution groups for Broker A.

```
mqsichangeflowstats BrokerA -s -g -j -c inactive
```

mqsichangeflowstats command - z/OS:

Syntax:

z/OS command - BIPCHMS:

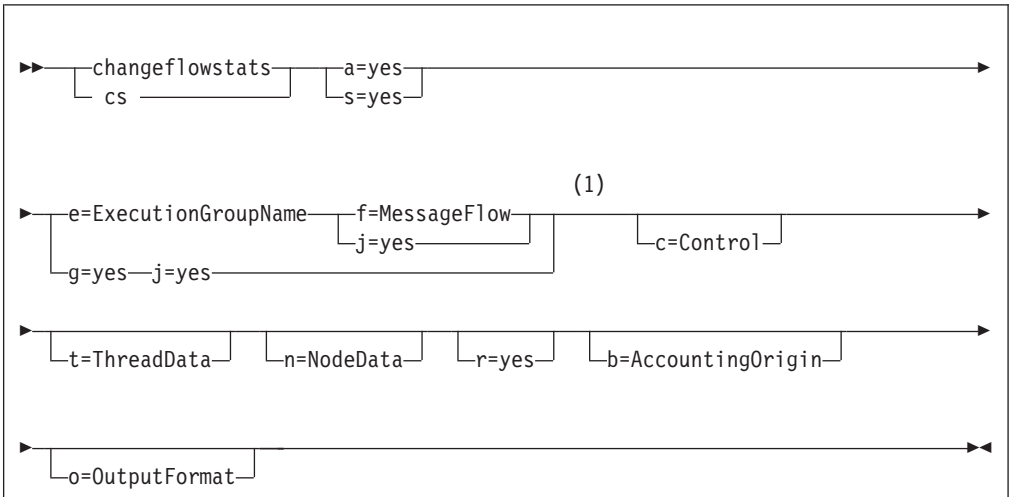


Notes:

1 See the parameter descriptions for information on valid combinations of the e,f, g, and j parameters.

z/OS console command:

Synonym: cs



Notes:

1 See the parameter descriptions for information on valid combinations of the e,f, g, and j parameters.

Parameters:

BrokerName
(Required) Specify the label of the broker for which accounting and statistics are to be changed.

This parameter is implied on the console form of the command.

-a (Required) Specify that the command modifies archive accounting and statistics collection.

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.

You must specify either **-a** or **-s**. If you do not specify one of these arguments you receive a warning message.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which accounting and statistics options are to be changed.

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.

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.

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.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

Note: 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) This parameter applies only to archive data and specifies that archive data is to be reset.

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 set only if the internal interval notification mechanism is being used, and not an external method.

-b *AccountingOrigin*
(Optional) Specifies that the environment tree path *Broker.Accounting.Origin* is used to partition the collected statistics into distinct outputs. Possible values are:

- none - do not partition statistics according to accounting origin data
- basic - partition statistics according to accounting origin data

-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 - statistics reports are output as SMF type 117 records.

Examples:

Using the command BIPCHMS:

- Turn on snapshot statistics for the message flow ″myFlow1″ in all execution groups 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″ , 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
  ```

The following example uses the console form of the command. Turn on archive accounting for all the message flows in all the execution groups that belong to the broker and output the report as SMF records.

```
F VCP2BRK,CS A=YES,G=YES,J=YES,C=ACTIVE,O=SMF
```

mqsichangeflowuserexits command

Use the mqsichangeflowuserexits command to set the list of active or inactive user exits. A list of active and a list of inactive user exits is maintained for each execution group and message flow. The effective state of user exits for a given flow is decided when the flow starts.

Supported operating systems:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting the BIPCHUE utility; see “Contents of the broker PDSE” on page 679

Purpose:

The order of precedence is message flow, execution group, and then broker default. The active list takes precedence over the inactive list in the message flow and execution group settings.

If the state for a given user exit is not set for the message flow, its state is taken from the execution group setting. If its state is not set for the message flow or execution group, it takes the default state which is implicitly inactive, or can be explicitly defined as active by the broker property *activeUserExits*, through the mqsichangebroker command.

If a particular user exit name is present in both the active and inactive lists for a message flow or execution group, then the active list takes precedence and the user

exit is active for that level. Therefore, if you want to change a user exit from active to inactive you must specify it as part of the inactive list, by using the **-i** flag and also remove it from the active list by re-specifying the new active list by using the **-a** flag.

When multiple exits are active for a given flow, they are invoked in a defined order. Those exits in the message flow's active list are invoked first in the order in which they were specified on the **-a** flag.

After those have been invoked, the exits in the execution group's active list (which were in neither the message flow's active nor inactive list) are invoked. These exits are invoked in the order in which they were specified on the **-a** flag.

All user exits that are not mentioned in the execution group's or message flow's active or inactive list, but are in the broker's active list, are invoked in the order in which they were specified when the broker property *activeUserExits* was set.

If any of the user exits specified in either the active or inactive list are not registered for the target execution group, the command fails with a BIP8858 error.

After successful command completion, if a user exit becomes invalid, the following action is taken, depending on which list the user exit appeared in:

- If the user exit was specified in the message flow's active or inactive list, the flow fails to start and a BIP2315 message is written to system log.
- If the user exit was specified in the execution group's active or inactive list, the execution group fails to start and a BIP2314 message is written to system log.

A user exit might become invalid for one of the following reasons:

- The broker or execution group is re-started after you change the MQSI_USER_EXIT_PATH variable by removing the directory containing the user exit library.
- The broker or execution group is re-started after you change the *userExitPath* broker property by removing the directory containing the user exit library.
- The user exit library (or one of its dependencies) is removed, or the broker is unable to load it.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsichangeflowuserexits command - Windows, Linux, and UNIX systems" on page 435
- "mqsichangeflowuserexits command - z/OS" on page 436

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

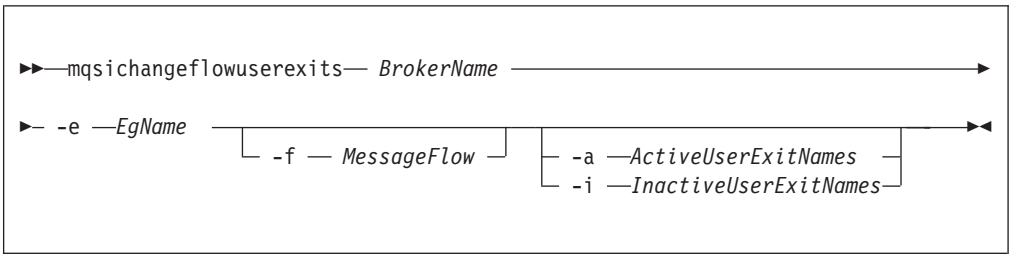
If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID that is used to run this command must be a member of the mqbrkrs group.

mqsichangeflowuserexits command - Windows, Linux, and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required). The name of the broker.

-e *EgName*
(Required). The name of the execution group.

-f *MessageFlow*
(Optional). The name of the message flow.

If you supply this value, the user exit is changed for that message flow; if you do not, the user exit is set at the execution group level.

-a *ActiveUserExitNames*
(Optional). A list of the names, separated by colons, of the active user exits. These are the names registered by the user exits when they were loaded. If any of the user exits listed are not registered for the target execution group, then the command fails with a BIP8858 error.

-i *InactiveUserExitNames*
(Optional). A list of the names, separated by colons, of the inactive user exits. These are the names registered by the user exits when they were loaded. If any of the user exits listed are not registered for the target execution group, then the command fails with a BIP8858 error.

Examples:

Setting active exits at flow level

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -a exit2
BIP8071I: Successful command completion.
```

Setting inactive exits at flow level

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -i exit1
BIP8071I: Successful command completion.
```

Setting active exits at execution group level

```
mqsichangeflowuserexits WBRK_BROKER -e default -a exit3,exit1
```

```
BIP8071I: Successful command completion.
```

Setting inactive exits at execution group level

```
mqsichangeflowuserexits WBRK_BROKER -e default -1 exit2
BIP8071I: Successful command completion.
```

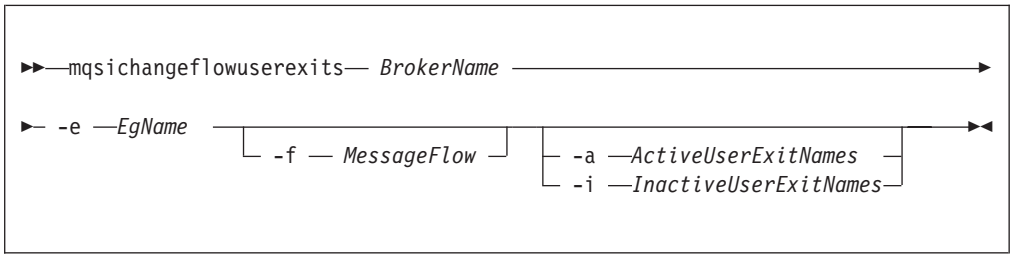
Changing exit1 to inactive and leaving exit2 active at flow level (A command had previously been issued with ″-a exit1:exit2″ to set them both active)

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -i exit1 -a exit2
BIP8071I: Successful command completion.
```

mqsichangeflowuserexits command - z/OS:

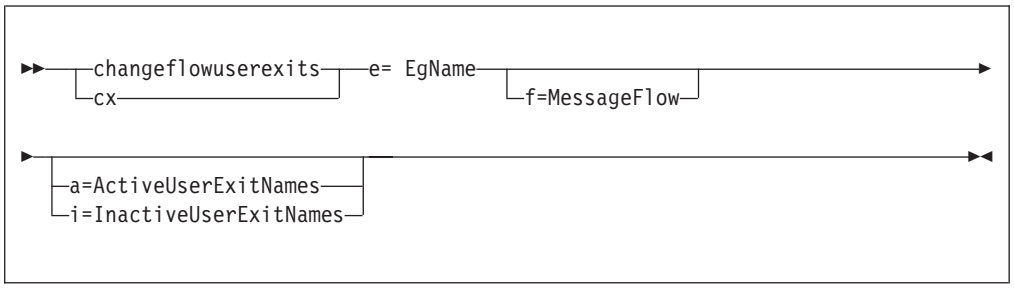
Syntax:

z/OS command - BIPCHUE:



z/OS console command:

Synonym: cx



Parameters:

BrokerName
(Required). The name of the broker.

-e *EgName*
(Required). The name of the execution group.

-f *MessageFlow*
(Optional). The name of the message flow.

If you supply this value, the user exit is changed for that message flow; if you do not, the user exit is set at the execution group level.

-a *ActiveUserExitNames*
(Optional). A list of the names, separated by colons, of the active user exits.

These are the names registered by the user exits when they were loaded. If any of the user exits listed are not registered for the target execution group, then the command fails with a BIP8858 error.

-i *InactiveUserExitNames*
(Optional). A list of the names, separated by colons, of the inactive user exits. These are the names registered by the user exits when they were loaded. If any of the user exits listed are not registered for the target execution group, then the command fails with a BIP8858 error.

Examples:

Setting active exits at flow level

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -a exit2
BIP8071I: Successful command completion.
```

Setting inactive exits at flow level

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -i exit1
BIP8071I: Successful command completion.
```

Setting active exits at execution group level

```
mqsichangeflowuserexits WBRK_BROKER -e default -a exit3,exit1
BIP8071I: Successful command completion.
```

Setting inactive exits at execution group level

```
mqsichangeflowuserexits WBRK_BROKER -e default -1 exit2
BIP8071I: Successful command completion.
```

Changing exit1 to inactive and leaving exit2 active at flow level (A command had previously been issued with ″-a exit1:exit2″ to set them both active)

```
mqsichangeflowuserexits WBRK_BROKER -e default -f myFlow -i exit1 -a exit2
BIP8071I: Successful command completion.
```

mqsichangeproperties command

Use the mqsichangeproperties command to modify broker properties and properties of broker resources.

Supported platforms:

- Windows systems.
- Linux and UNIX systems.
- z/OS. Run this command by customizing and submitting the BIPCHPR utility; see “Contents of the broker PDSE” on page 679.

Purpose:

Use the mqsichangeproperties command to change properties that are associated with a broker:

- Properties that affect the whole broker; for example, an HTTP listener
- Properties that affect one or more execution groups; for example, the broker registry
- Properties that affect a configurable service; for example, a JMSProvider, or the broker’s publish/subscribe engine

You can also use the Configuration Manager Proxy (CMP) to change properties.

Use the mqsireportproperties command to view properties that are associated with a broker.

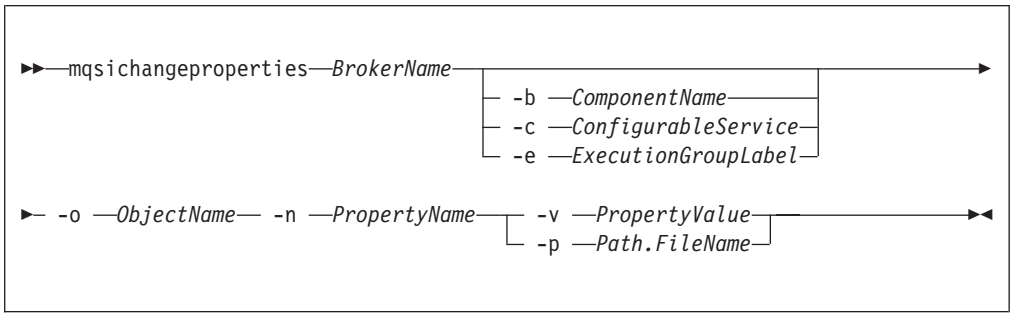
Usage notes:

Before you run the mqsichangeproperties command, ensure that the broker is running.

If you change any value, stop and restart the broker for the change to take effect.

When a message flow that includes HTTP nodes or Web Services support is started, the broker starts the HTTP listener.

Syntax:



Parameters:

BrokerName
(Required) The name of the broker to modify. This parameter must be the first parameter.

-b *ComponentName*
(Optional) The name of the component. Valid values are httplistener and securitycache.

-c *ConfigurableService*
(Optional) The type of configurable service. The type is predefined; for example, JMSProviders. You can define additional services of any defined type by using the mqsicreateconfigurableservice command.

The valid resource types are listed in "Configurable services properties" on page 444.

-e *ExecutionGroupLabel*
(Optional) The label of the execution group for which you want to change properties.

You can specify the special name `AllExecutionGroups` if you also specify either HTTPConnector or HTTPSConnector. This option changes broker-wide settings that affect SOAP nodes deployed to all execution groups for these objects.

-o *ObjectName*
(Required) The name of the configurable service object for which you want to change the properties.

You must also specify **-b**, **-e**, or **-c** with **-o**, except if you specify the object name BrokerRegistry, or the object name ComIbmJVMManager to change a property related to the heap size.

For compatibility with previous versions, you can also specify the value ComIbmXmlParserFactory for the **ObjectName**.

The valid object names are listed in "Configurable services properties" on page 444.

-n *PropertyName*
(Required) The name of the property to be changed. The available property names differ according to the component or configurable service that you have specified. All property names start with a lowercase character.

The property values are described in "Configurable services properties" on page 444.

-v *PropertyValue*
(Required) The value that is assigned to the property that is specified by the **-n** parameter.

You can specify more than one property name together with a corresponding value, using commas as separators, provided that you use a valid value for the corresponding property; for example, `-n Name1,Name2 -v Value1,Value2`.

Do not leave a space after each comma in the list of names and corresponding values.

Use `""` to specify an empty *PropertyValue* string.

If you set the **-c** parameter to EISProviders or JMSProviders, and the **-n** parameter to jarsURL, the expected value is a URL that specifies the file location of the EIS or JMS provider JAR files, but omits the `file://` part of the URL. (On Windows, the file location cannot be a mapped network drive on a remote Windows computer; the directory must be local or on a Storage Area Network (SAN) disk.)

UNIX On UNIX, if the **-v** parameter contains a semicolon (;), enclose the entire string in quotation marks, as shown in the following example:

```
mqsichangeproperties WBRK_BROKER -c JDBCProviders -o DB2EXTRA -n connectionUrlFormat
      -v "jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=
[password];"
```

-p *Path.FileName*
(Optional) The location and name of a file from which the command reads the property value. Use this command as an alternative to **-v** where the value of the property is complex and is defined in a file, such as an XML file.

The following conditions apply to the use of this parameter:

- **-p** can be used to set a single property only. Therefore, the **-n** parameter must have a single property name, not a comma-separated list.
- White space characters (including line feed, carriage return, and end of file characters) read from the file are preserved by the command.

Use this parameter for policy sets and bindings.

Use this parameter for monitoring profiles; the XML file must conform to the monitoring profile schema.

For detailed information about valid components, configurable services, object names, properties, and values, select the appropriate topic:

- "Broker registry object parameter values" on page 443
- "Configurable services properties" on page 444
- "HTTPConnector and HTTPSConnector parameter values (SOAP node connections)" on page 478
- "httplistener component parameter values" on page 480
- "Inter-broker communications parameter values (collectives)" on page 483
- "Inter-broker communications parameter values (clones)" on page 484
- "JVM parameter values" on page 485
- "Real-time nodes parameter values" on page 485
- "securitycache component parameter values" on page 491

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID that is used to run this command must be a member of the mqbrkrs group.

Additionally, the broker requires the following authority for the supported multicast protocols:

PGM/IP
The broker requires:

- Linux UNIX root authority on Linux and UNIX systems
- Windows Administrator authority on Windows systems
- z/OS 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.

Examples:

Always enter the command on a single line; in some examples, a line break has been added to enhance readability.

Changes to broker components

The following examples specify the **-b** parameter to identify a particular broker component.

- Enable the HTTPSConnector for the HTTP nodes deployed to the specified broker:

```
mqsichangeproperties TEST -b httplistener -o HTTPListener -n enableSSLConnector -v true
```

- Change the default SSL protocol from SSLv3 to TLS for the HTTP nodes that are deployed to the specified broker:

```
mqsichangeproperties TEST  -b httplistener -o HTTPSConnector -n sslProtocol -v TLS
```

- Change the securitycache timeout to 200 seconds:

```
mqsichangeproperties TEST -b securitycache -o SecurityCache -n cacheTimeout -v 200
```

Changes to properties associated with execution groups

The following examples include the **-e** parameter to specify the execution group to change.

- Change the clientPingInterval to 200 for a cloned broker:

```
mqsichangeproperties TEST -e default -o DynamicSubscriptionEngine -n clientPingInterval -v 200
```

- Enable multicast for a cloned broker:

```
mqsichangeproperties TEST -e default -o DynamicSubScriptionEngine -n multicastEnabled -v true
```

- Set the port number when changing properties for execution groups:

```
mqsichangeproperties TEST -e exgroup1 -o HTTPSConnector  -n explicitlySetPortNumber -v 7777
```

- Set the JVM port number to activate message flow debugging:

```
mqsichangeproperties TEST -e exgroup1 -o ComIbmJVMManager -n jvmDebugPort -v 8018
```

- Set the type of server keystore:

```
mqsichangeproperties TEST -e AddressSampleProvider -o ComIbmJVMManager -n keystoreType -v JKS
```

Changes to the BrokerRegistry object

- Set the HTTPConnector Port Range in the broker registry:

```
mqsichangeproperties BRKR1 -o BrokerRegistry -n httpConnectorPortRange -v 7777-8888
```

Changes to configurable services

The following examples include the **-c** parameter to specify the type of configurable service to change.

- Make the JAR files and shared libraries available to the WebSphere Adapter for SAP:

```
mqsichangeproperties WBRK_BROKER -c EISProviders -o SAP -n jarsURL,nativeLibs
      -v c:\sapjco\jars,c:\sapjco\bin
```

- Change all the nodes that are configured to use the *myIMSConnectService* configurable service. After you run this command, all nodes connect to the production system (`production.ims.ibm.com`) instead of the test system (`test.ims.ibm.com`).

  ```
  mqsichangeproperties WBRK_BROKER -c IMSConnect -o myIMSConnectService -n Hostname
  -v production.ims.ibm.com
  ```

- Update the security identity for the JDBCProvider service for Oracle:

```
mqsichangeproperties WBRK_BROKER -c JDBCProviders -o Oracle  -n securityIdentity
      -v OracleDSN
```

  OracleDSN is the DSN with which you have associated a user ID and password using the mqsisetdbparms command.

- Change the location of the JAR files for the IBM WebSphere MQ JMS client:

```
mqsichangeproperties WBRK_BROKER -c JMSProviders -o WebSphere_MQ
      -n jarsURL -v file://D:\SIBClient\Java
```

- Change the value of the properties proprietaryAPIAttr2 and proprietaryAPIAttr3 for a user-defined JMS provider configurable service definition called BEA_Weblogic, where the properties represent the URL of the BEA WebLogic bindings and the DNS name of the JMS Server:

```
mqsichangeproperties WBRK_BROKER -c JMSProviders -o BEA_Weblogic
      -n proprietaryAPIAttr2, proprietaryAPIAttr3
      -v t3://9.20.94.16:7001,BEAServerName
```

- Change the value of the jndiEnvironmentParms property in the definition of a user-defined JMS provider configurable service called *myJMSprovider*:

```
mqsichangeproperties WBRK_BROKER -c JMSProviders -o myJMSprovider
      -n jndiEnvironmentParms
      -v domainName=myDomain;timeout=6000
```

- Change all connections that are used by the adapter *myPeopleSoftAdapter.outadapter*. After you run this command, all adapters connect to the production system (`my.peoplesoft.production.com`) instead of the test system (`my.peoplesoft.qa.com`).

  ```
  mqsichangeproperties WBRK_BROKER -c PeopleSoftConnection -o myPeopleSoftAdapter.outadapter
  -n hostName -v "my.peoplesoft.production.com"
  ```

- Set the properties of monitoring profile 'mp1' by using the contents of file `mp1.xml`:

  ```
  mqsichangeproperties WBRK_BROKER -c MonitoringProfiles -o mp1 -n profileProperties -p mp1.xml
  ```

- When you create a policy set, its properties are always set to default values. Use this command to change the property values.

  Import a policy set to a broker from a file:

```
mqsichangeproperties WBRK_BROKER -c PolicySets -o Policy_2
      -n ws-security -p policyset.xml
```

  This command reads file `policyset.xml` and sets its contents as Policy_2 in broker WBRK_BROKER. The command is used to move policy sets between brokers, or to restore from a backup.

- Import a policy set bindings to a broker from a file:

```
mqsichangeproperties WBRK_BROKER -c PolicySetBindings -o Bindings_2
      -n ws-security -p bindings.xml
```

  This command reads file `bindings.xml` and sets its contents as Bindings_2 in broker WBRK_BROKER. The command is used to move policy set bindings between brokers, or to restore from a backup.

- Disable the broker's publish/subscribe engine so that the WebSphere MQ queue manager controls all publish/subscribe application messages and operations:

  ```
  mqsichangeproperties WBRK_BROKER -c PublishSubscribe -o Interface -n psmode -v disabled
  ```

  You can disable the broker's publish/subscribe engine only if you have installed WebSphere MQ Version 7.0 on this computer. If you run this command, and Version 6.0 is installed, the command completes, but has no effect.

- Enable the broker's publish/subscribe engine to handle all application messages and operations:

  ```
  mqsichangeproperties WBRK_BROKER -c PublishSubscribe -o Interface -n psmode -v enabled
  ```

- Change all connections that are used by the adapter *mySAPAdapter.outadapter*. After you run this command, all adapters connect to the production system (`production.sap.ibm.com`) instead of the test system (`test.sap.ibm.com`).

  ```
  mqsichangeproperties WBRK_BROKER -c SAPConnection -o mySAPAdapter -n applicationServerHost
  -v production.sap.ibm.com
  ```

- Enable TFIM identity mapping:

```
mqsichangeproperties WBRK_BROKER -c SecurityProfiles -o MyFirstSecurityProfile
      -n mapping,mappingConfig -v TFIM,http://tfimhost1.ibm.com
```

- Change the connection timeout for queries issued by the WebSphere Service Registry and Repository nodes to 180 seconds:

```
mqsichangeproperties WBRK_BROKER -c ServiceRegistries -o DefaultWSRR
      -n connectionTimeout -v 180
```

- Change the connection that is used by the adapter *mySiebelAdapter.outadapter*. After you run this command, all message flows in all execution groups that use this adapter connect to the production system (`my.siebel.production.com`) instead of the test system (`my.siebel.qa.com`). If you are using different adapters in the message flow, run the mqsichangeproperties command for each named adapter.

```
mqsichangeproperties WBRK_BROKER -c SiebelConnection -o mySiebelAdapter.outadapter
-n connectString -v "siebel://my.siebel.production.com/SBA_80/SSEObjMgr_enu"
```

- Update the TCPIPClient configurable service so that it does not make any client connections until they are required:

```
mqsichangeproperties WBRK_BROKER -c TCPIPClient -o ClientPort1452HostnameJsmith
      -n MinimumConnections -v 0
```

- Change the connection expiry time on TCPIPServer connections to 30 seconds:

```
mqsichangeproperties WBRK_BROKER -c TCPIPServer -o ServerPort1452
      -n ExpireConnectionSec -v 30
```

- For the FtpServer configurable service called TEST1, change the protocol to SFTP and change the server name to winlnx58:

```
mqsichangeproperties WBRK_BROKER -c FtpServer -o TEST1
-n protocol,serverName,scanDelay,remoteDirectory,securityIdentity,
cipher,compression,strictHostKeyChecking
-v  SFTP,winlnx58,30,.,chbatey,blowfish-cbc,9,no
```

Broker registry object parameter values:

Select the names of the properties and values that you want to change for the broker registry object.

To change these properties, you must specify the broker name and the **ObjectName** BrokerRegistry. You do not have to specify **-b**, **-c**, or **-e** to change properties associated with this object.

The following properties and values are valid:

-n brokerKeystorePass

The password used to access the server certificate from the specified keystore file.

Set the password by using mqsisetdbparms when the broker is stopped.

- Value type - string
- Initial value - brokerKeystore::password

-n brokerKeystoreFile

The path to the keystore file where the server certificate, which is to be loaded, has been stored. The HTTP listener expects a file with the default name `.keystore` in the home directory of the user who started the broker.

- Value type - string
- Initial value - *default value* (described previously)

-n brokerKeystoreType
The type of keystore file to be used for the server certificate.

- Value type - string
- Initial value - JKS

-n brokerTruststorePass
The password used to access the server certificate from the specified keystore file.

Set the password by using mqsisetdbparms when the broker is stopped.

- Value type - string
- Initial value - brokerKeystore::password

-n brokerTruststoreFile
The path to the keystore file where the server certificate, which is to be loaded, has been stored. The HTTP listener expects a file with the default name `.keystore` in the home directory of the user who started the broker.

- Value type - string
- Initial value - *default value* (described previously)

-n brokerTruststoreType
The type of keystore file to be used for the server certificate.

- Value type - string
- Initial value - JKS

-n httpConnectorPortRange
The numeric range of ports available to the HTTPConnector object that is associated with SOAP nodes.

- Value type - integer
- Initial value - 7800-7842

-n httpsConnectorPortRange
The numeric range of ports available to the HTTPSConnector object that is associated with SOAP nodes.

- Value type - integer
- Initial value - 7843-7884

See the "mqsichangeproperties command" on page 437 for examples of how to change BrokerRegistry parameters. Other examples are provided for particular tasks:

Viewing and setting keystore and truststore runtime properties at broker level

Accessing a secure WebSphere Service Registry and Repository

Configurable services properties:

The supplied configurable services, and the configurable services that you create, are defined by their names and properties.

You can use the supplied services, and you can create your own configurable services by using the mqsicreateconfigurableservice command. You can also delete services that you have created by using the mqsideleteconfigurableservice command.

If you want to change the properties for a configurable service, use the mqsichangeproperties command and specify the broker name and **-c** *ConfigurableService*. You must also set the **ObjectName** to the name of the service

for which you want to change properties; the name can be a predefined name (one of those shown for the service type in the following tables), or the name of a configurable service that you have created yourself. See the "mqsichangeproperties command" on page 437 for examples of its use.

To display one or more of the defined configurable services, use the mqsireportproperties command. The following example displays all configurable services that are available for a single broker:

```
mqsireportproperties brokerName -c AllTypes -o AllReportableEntityNames -r
```

Specify the appropriate parameters from those shown in the following tables on the mqsichangeproperties and mqsireportproperties commands:

- Specify **-c** to identify the configurable service type.
- Specify **-o** to identify the name of the configurable service object.
- Specify **-n** to identify the properties of the service.
- Specify **-v** to identify the values of the properties specified.

Follow the link to the configurable service that you want to use to view the available properties:

- "EISProviders configurable service" on page 446
- "FtpServer configurable service" on page 446
- "IMSConnect configurable service" on page 450
- "JDBCProviders configurable service" on page 451
- "JMSProviders configurable service" on page 452
- "MonitoringProfiles configurable service" on page 452
- "PeopleSoftConnection configurable service" on page 453
- "PolicySets configurable service" on page 453
- "PolicySet Bindings configurable service" on page 453
- "PublishSubscribe configurable service" on page 454
- "SAPConnection configurable service" on page 455
- "SecurityProfiles configurable service" on page 456
- "Service Registries configurable service" on page 456
- "SiebelConnection configurable service" on page 457
- "SMTP configurable service" on page 457
- "TCPIPClient configurable service" on page 458
- "TCPIPServer configurable service" on page 459

EISProviders configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
PeopleSoft SAP Siebel Twineball	jarsURL	A URL that specifies the file location of the EIS provider JAR files. Omit `file://` from the URL. On Windows, the file location cannot be a mapped network drive on a remote Windows computer; the directory must be local or on a Storage Area Network (SAN) disk. If you do not set the **-n** parameter on the mqsicreateconfigurableservice command, the default location for the EIS provider JAR files is the broker's shared-classes directory.
	nativeLibs	The file location of any libraries that the EIS provider owns. If you do not set the **-n** parameter on the mqsicreateconfigurableservice command, the default location for any libraries that the EIS provider owns, is the broker's LilPath.

FtpServer configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
None	accountInfo	Some FTP servers require an account name during the FTP logon procedure. If this property is specified, its value is the account name that is supplied when it is requested during FTP logon. If this property is not specified, and the server requests an account name, the FTP transfer fails.
	scanDelay	The period of time, in seconds, to wait after a scan of the directory results in no files being identified for processing. The default is 60 seconds. If it is set, this property overrides the Scan delay property on the **FTP** tab of the FileInput node that uses this service.
	protocol	The remote transfer protocol to use. Valid values are FTP or SFTP. If no protocol is specified in the configurable service, the value specified in the node is used.
	transferMode	The transfer mode of the FTP connection. Valid values are BINARY (the default) or ASCII. If it is set, this property overrides the Transfer mode property on the **FTP** tab of the FileInput or FileOutput node that uses this service. This property is valid only when FTP is specified as the protocol. If SFTP is specified, this property is ignored.
	connectionType	The FTP data socket connection. Valid values are ACTIVE or PASSIVE. This property is valid only when FTP is specified as the protocol. If SFTP is specified, this property is ignored.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	remoteDirectory	The relative or absolute directory name on the remote FTP server. If it is set, this property overrides the Server directory property on the **FTP** tab of the FileInput or FileOutput node that uses this service.

	securityIdentity	The name of a security identity that is defined using the mqsisetdbparms command. If it is set, this property overrides the Security identity property on the **FTP** tab of the FileInput or FileOutput node that uses this service. If the value of this property is secId, use the following command to define the security identity:
		• If you are using FTP:
		`mqsisetdbparms BrokerName -n ftp::secId -u userName -p password`
		• If you are using SFTP:
		`mqsisetdbparms BrokerName -n sftp::secId -u userName -p password`
		or
		`mqsisetdbparms BrokerName -n sftp::secId -u userName -i SSHIdentityFile -r Passphrase`

	serverName	The IP address and, optionally, port number for the remote FTP server. The syntax for the property is identical to that permitted for the FTP server and port property of the FileInput and FileOutput nodes (except it cannot be the name of an FtpServer configurable service).
	cipher	The cipher used for encryption. This property takes the form of one or more of the following values, separated by plus signs (+): • blowfish-cbc • 3des-cbc • aes128-cbc The cipher that you use for encryption depends on your SSH implementation. List the values in order of preference. This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored. If no value is specified, the following default is used: blowfish-cbc+3des-cbc+aes128-cbc

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	mac	Message Authentication Code. This property takes the form of one or more of the following values, separated by plus signs (+): • hmac-md5 • hmac-sha1 The MAC that you use depends on your SSH implementation. List the values in order of preference. This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored. If no value is specified, the following default is used: hmac-md5+ hmac-sha1
	compression	Specifies the level of compression to be used. Valid values are integers between 0 and 9, where 0 specifies no compression and 9 specifies maximum compression. This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored. If no value is specified, the default value of 0 is used.
	strictHostKeyChecking	Specifies how host keys are checked during the connection and authentication phase. Valid values are: **No** Specifies that the following action is performed: • If the connection is to a new host, connect and accept the host's key, and store it • If the connection is to a host that has been connected to previously and the host key has changed, issue an exception (in the FileOutput node). If you select No, a default known hosts file (managed by the broker) is used. **Yes** Connect only to known hosts with valid keys; otherwise issue an exception. If you select Yes, you must specify your own known hosts file using the knownHostsFile property. The default value is No. The host information is stored in a `known_hosts` file in the standard OpenSSH format. This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	knownHostsFile	The location of the known hosts file. The value must be the fully qualified path to a valid known hosts file. The host information is stored in a `known_hosts` file in the standard OpenSSH format. This property is mandatory if the strictHostKeyChecking property is set to Yes. If the strictHostKeyChecking property is set to No, this property is ignored. This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

IMSConnect configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
None	Hostname	The host name of the computer on which the IMS Connect instance is running. This property is mandatory; if you leave it blank, the node cannot connect to IMS Connect.
	PortNumber	The port number on which the IMS Connect instance is listening. This value must be a positive integer. This property is mandatory; if you leave it blank, the node cannot connect to IMS Connect.
	DataStoreName	The data store name against which the IMS Connect instance is running. This property is not mandatory, but if you do not set it, an exception is issued by any node that attempts to use this configurable service.
	SocketTimeoutSec	The socket timeout is the maximum amount of time that the IMS Connector for Java waits for a response from IMS Connect before it disconnects the socket and returns an exception to WebSphere Message Broker. If network problems or routing failures occur, this property prevents the client that is using the IMS resource adapter from waiting indefinitely for a response from IMS Connect. This property is based on the TCP/IP sockets that IMS Connect and the IMS resource adapter use to communicate; therefore, it is not applicable with the Local option. This property is not mandatory, and the default value is 0 (zero), indicating that the socket never times out. You can enter any valid integer in the range 0 to 2147483. You set the socket timeout independently of the execution timeout on the configurable service. The socket timeout is used to respond to network problems (such as a loss of connection), while the execution timeout is used to recover from a non-responsive IMS program. The socket timeout is typically longer than the execution timeout.
	ExecutionTimeoutSec	The execution timeout is the maximum amount of time that the IMS Connector for Java waits for a response from a transaction. This property is not mandatory, and the default value is 60, indicating that the IMS Connector for Java waits for 60 seconds. You can enter any valid integer in the range 1 to 2147483.

JDBCProviders configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
DB2 Informix Informix_With_Date_Format Microsoft_SQL_Server Oracle Sybase_JConnect6_05	connectionUrlFormat	A pattern that represents the connection URL definition, which is specific to a particular database type. For example, the pattern for DB2 is defined with the following fixed content: `jdbc:db2://[serverName]:[portNumber]/[databaseName] :user=[user];password=[password];` Do not use the mqsichangeproperties command to change the pattern itself; changes made to the pattern might cause unpredictable results.
	connectionUrlFormatAttr1 connectionUrlFormatAttr2 connectionUrlFormatAttr3 connectionUrlFormatAttr4 connectionUrlFormatAttr5	If the specified URL format contains non-standard JDBC data source properties, such as a server identifier, specify one of five general purpose connection attributes to define these additional properties. For example, if `connectionURLFormat = jdbc:oracle:thin:[user]/ [password]@[serverName]:[portNumber]: [connectionUrlFormatAttr1]` connectionUrlFormatAttr1 must contain an Oracle server identifier, which you must supply by defining the value for the property connectionUrlFormatAttr1 on the mqsicreateconfigurableservice or mqsichangeproperties command. The broker can then substitute all the required values into the required pattern.
	databaseName	The name of the database to which the data source entry enables connections; for example, employees.
	databaseType	The database type, for example, DB2.
	databaseVersion	The database version; for example, 9.1.
	description	An optional property to describe the data source definition.
	jarsURL	The local directory path on the system on which the broker is running, where the JAR file that contains the type 4 driver class is located.
	portNumber	The port number on which the database server is listening; for example, 50000.
	securityIdentity	A unique security key to perform a second broker registry lookup to find an entry under the broker's DSN entries, which store the encrypted password for the user on their associated host system; for example, `jdbc::mysecurityIdentity`. Use the mqsisetdbparms command to create a new DSN entry, as described in "Securing a JDBC type 4 connection" on page 147.
	serverName	The name of the server; for example, `host1`.
	type4DatasourceClassName	The name of the JDBC Type 4 data source class name that is used to establish a type 4 connection to a remote database and for coordinated transaction support; for example, `com.ibm.db2.jcc.DB2XADataSource`.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	type4DriverClassName	The name of the JDBC driver class name that is used to establish a connection; for example, `com.ibm.db2.jcc.DB2Driver`.

JMSProviders configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
ActiveMQ BEA_Weblogic FioranoMQ Generic_File Generic_LDAP JBoss JOnAS Joram OpenJMS Oracle_OEMS SeeBeyond SonicMQ SwiftMQ Tibco_EMS WebSphere_MQ WebSphere_WAS_Client	jarsURL	A URL that specifies the file location of the JMS provider JAR files. Omit `file://` from the URL. If you do not set the **-n** parameter on the mqsicreateconfigurableservice command, the default location for the JMS provider JAR files is the broker's shared-classes directory.
	jndiEnvironmentParms	This property is optional. A list of JNDI environment parameters expressed as name-value pairs separated by semicolons. Use these parameters in JMSInput, JMSOutput, and JMSReply nodes.
	nativeLibs	The file location of any libraries that the JMS provider owns. If you do not set the **-n** parameter on the mqsicreateconfigurableservice command, the default location for any libraries that the JMS provider owns, is the broker's LilPath.
	proprietaryAPIHandler	The name of the IBM supplied Java class to interface with a JMS provider's proprietary API.
	proprietaryAPIAttr1 proprietaryAPIAttr2 proprietaryAPIAttr3 proprietaryAPIAttr4 proprietaryAPIAttr5	These attributes are optional. If you configure any of these, they might be used on one or more method calls to the vendor proprietary API. The usage of these attributes is specific to a vendor interface and their meaning is determined by the IBM proprietary API Handler.

MonitoringProfiles configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
DefaultMonitoringProfile	profileProperties	The name of the monitoring profile. The monitoring options defined by the monitoring profile can be configured with an XML file that conforms to the monitoring profile schema.

PeopleSoftConnection configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
None	hostName	Identifies, either by name or IP address, the server that hosts PeopleSoft Enterprise. This property is mandatory. By default, the value of this property is an empty string, therefore you must set a valid host name or IP address.
	port	The port number that the adapter uses to access the PeopleSoft Enterprise server. This property is mandatory. By default, the value of this property is an empty string, therefore you must set a valid port number.

PolicySets configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
WSS10Default	config	Reserved for future use.
	ws-security	The content of the policy set. The content is edited with the Policy Sets and Policy Set Bindings editor and can be backed up and restored by using the **-p** parameter on the mqsireportproperties and mqsichangeproperties commands.

PolicySet Bindings configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
WSS10Default	associatedPolicySet	The name of the policy set configurable service with which this policy set binding is associated. This value is set by the Policy Set editor when an association is defined. If you are restoring a policy set and binding by using the mqsichangeproperties command, ensure that this command refers to the correct associated policy set.
	config	Reserved for future use.
	ws-security	The content of the policy set binding. The content is edited with the Policy Sets and Policy Set Bindings editor and can be backed up and restored by using the **-p** parameter on the mqsireportproperties and mqsichangeproperties commands.

PublishSubscribe configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
Interface	psmode	Indicates whether the broker's publish/subscribe engine is enabled or disabled. The initial value is set to enabled. Set this value to disabled if you want the broker's queue manager to handle all publish/subscribe application messages and operations.

SAPConnection configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
None	applicationServerHost	Specifies the IP address or the name of the application server host to which the adapter logs on. This property is mandatory; you must set the value to a valid SAP server host name or IP address.
	gatewayHost	The host name of the SAP gateway. This property is not mandatory. The default value is an empty string, which signifies that a SAP gateway is not used when the adapter connection is made.
	gatewayService	The identifier of the gateway on the gateway host that carries out the RFC services. This property is not mandatory. The default value is an empty string, which signifies that a SAP gateway service is not used when the adapter connection is made. Enter a valid SAP gateway server identifier.
	client	The client number of the SAP system to which the adapter connects. This property is not mandatory; if you do not specify a value for this property, the default value is 100, which matches the default SAP client. Set this property to the required SAP client number, which is a three-digit integer in the range 000 to 999.
	systemNumber	The system number of the SAP application server. This property is not mandatory; if you do not specify a value for this property, the default value is 00, which matches the default SAP system number. Set this property to the required SAP system number, which is a two-digit integer in the range 00 to 99.
	RFCTraceLevel	Specifies the global trace level. This property is not mandatory; if you do not specify a value for this property, the default value is 1, which is the standard trace level. • The default value is 1, which indicates that SAP JCo Java API logging occurs. • If you set this value to 3, SAP JCo JNI API logging occurs. • If you set this value to 5, error diagnostic logging occurs.
	RFCTracePath	Sets the fully qualified local path to the folder into which the RFC trace files are written. This property is mandatory only if the RFCTraceOn property is set to true. By default, the value of this property is an empty string, therefore, when RFCTraceOn is set to true, you must set a valid path for the RFCTracePath.
	RFCTraceOn	Specifies whether to generate a text file detailing the RFC activity for each event listener. This property is not mandatory and the default value is false. Set this property to true to enable RFC trace.
	rfcProgramID	The remote function call identifier under which the adapter registers in the SAP gateway. This value must match the Program ID that is registered in SAP (transaction SM59).

SecurityProfiles configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
Default_Propagation	authentication	NONE LDAP TFIM A user-defined value
	authenticationConfig	A provider-specific configuration string.
	authorization	NONE LDAP TFIM A user-defined value
	authorizationConfig	A provider-specific configuration string.
	mapping	NONE TFIM A user-defined value
	mappingConfig	A provider-specific configuration string.
	passwordValue	PLAIN MASK OBFUSCATE
	propagation	TRUE FALSE

Service Registries configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
DefaultWSRR	connectionFactoryName	The name of the WSRR WebSphere Application Server JMS provider JMS connection factory for Cache Notifications. The default value is jms/SRConnectionFactory.
	connectionTimeout	The WSRR connection timeout period in seconds. The default value is 180 seconds, which is 3 minutes.
	enableCacheNotification	The default value is False. Select True to enable WebSphere Message Broker WSRR Cache Notification.
	endpointAddress	The location or endpoint of the WSRR server. The default value is `http://host_name:9080/WSRR6_1/services/WSRRCoreSDOPort` For WSRR 6.2, the default value is `http://host_name:9080/WSRR6_2/services/WSRRCoreSDOPort`
	initialContextFactory	The name of the WSRR WebSphere Application Server JMS provider JMS context factory for Cache Notifications. The default value is `com.ibm.websphere.naming.WsnIitialContextFactory`
	locationJNDIBinding	The URL to the WebSphere Application Server JMS provider JNDI bindings. The default value is `iiop://host_name:2809/`
	needCache	The default value is True, indicating that the WebSphere Message Broker WSRR cache is enabled.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	predefinedCacheQueries	A list of semicolon-separated queries with which to initialize the WebSphere Message Broker WSRR cache. This query is defined by the WSRR query language.
	refreshQueriesAfter Notification	When a notification is received from WSRR, if refreshQueriesAfterNotification is set to True, the Cache is updated with the new version of the object immediately; if it is set to False, the Cache is updated on the next request. The default value is True.
	subscriptionTopic	The topic name that is used to receive WebSphere Application Server JMS provider Cache Notifications. The default value is jms/SuccessTopic.
	timeout	The cache expiry time in milliseconds. The default value is 10000000, which is approximately 166 minutes.

SiebelConnection configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties

None	connectString	The connection URL that is needed to connect to the Siebel server.
		
		This property is mandatory. By default, the value of this property is an empty string, therefore you must set a valid URL. The URL has the following format:
		`Protocol://machinename:port/enterprisename/object manager/server name`
		
		For example:
		• For Siebel 7.0.5 to 7.5x:
		`siebel://IP_ADDRESS/siebel/SSEObjMgr_ENU/sebldev1`
		• For Siebel 7.8:
		`siebel://IP_ADDRESS:2321/Sieb78/SSEObjMgr_enu`
		• For Siebel 8:
		`siebel://IP_ADDRESS:2321/SBA_80/SSEObjMgr_enu`
		The default port number is 2320, but in the examples above, for Siebel version 7.8 and 8, the port has been set to 2321.

SMTP configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
None	serverName	The name of the server; for example, host1.
	securityIdentity	The name of a security identity that is defined using the mqsisetdbparms command.

TCPIPClient configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
Default	Hostname	The host name of the remote system to which to connect with a client connection. A valid value is any IP address or computer name. You cannot change this value if there is already a configurable service with this name using the same port (unless the port is set to 0 (zero)).
	Port	The port number to be used for this configurable service. The default is 0 (zero), which means no port number. By default, the configurable service is disabled and the value of the port provided on the node is used instead. A port number can be assigned to only one configurable service at a time; if you try to assign a port number to more than one configurable service, an error occurs.
	MinimumConnections	The minimum number of client connections made by the broker. The broker attempts to establish this number of connections even if no flows are using the connections. The default value is 0 (zero), which means that the broker does not make any client connections until they are required.
	MaximumConnections	The maximum number of client connections that can be made on this port. The default value is 100, which means that, by default, the broker accepts up to 100 server connections.
	MaxReceiveRecordBytes	The maximum size a record can reach before an exception is thrown. The default value is 104,857,600, which means that, by default, the broker accepts messages with a maximum size of 100 MB. The record size is taken to be the size of the data including any delimiters.
	ExpireConnectionSec	The length of time (in seconds) that a connection is kept open without being used. The value can be any integer. A value of 0 (zero) causes the connection to be closed immediately, and a value of -1 causes the connection to remain open indefinitely (no expiry).
	SO_RCVBUF	The size (in bytes) of the SO_RCVBUF property on the socket. Valid values vary according to the operating system that you are using. This property is a standard TCP/IP property. The default value is 0 (zero), which sets the size of the SO_RCVBUF property to the operating system default.
	SO_SNDBUF	The size (in bytes) of the SO_SNDBUF property on the socket. Valid values vary according to the operating system that you are using. This property is a standard TCP/IP property. The default value is 0 (zero), which sets the size of the SO_SNDBUF property to the operating system default.
	TCP_NODELAY	The value of the TCP_NODELAY property on the socket. If the value is set to *True*, the socket sends data as soon as it is sent to its buffer. The default value is *False*.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	TrafficClass	The traffic class that is set on any connection that is established. Valid values are positive integers. The default value is -1, which leaves the TrafficClass set to the platform default.
	SO_LINGER	The SO_LINGER property on any connection that is established. This property is a standard TCP/IP property. The default value is *False*.
	SO_LINGER_TIMEOUT_SEC	The SO_LINGER_TIMEOUT_SEC property on any connection that is established. This property is a standard TCP/IP property. Valid values are positive integers. The default value is -1, which leaves the SO_LINGER_TIMEOUT_SEC value set to the operating system default.

TCPIPServer configurable service

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
Default	Port	The port number to be used for this configurable service. The default is 0 (zero), which means no port number. By default, the configurable service is disabled and the value of the port provided on the node is used instead. A port number can be assigned to only one configurable service at a time; if you try to assign a port number to more than one configurable service, an error occurs.
	MaximumConnections	The maximum number of server connections that can be made on this port. The default value is 100.
	MaxReceiveRecordBytes	The maximum size a record can reach before an exception is thrown. The default value is 104,857,600, which means that, by default, the broker accepts messages with a maximum size of 100 MB. The record size is taken to be the size of the data including any delimiters.
	ExpireConnectionSec	The length of time (in seconds) that a connection is kept open without being used. The value can be any integer. A value of 0 (zero) causes the connection to be closed immediately, and a value of -1 causes the connection to remain open indefinitely (no expiry).
	SO_RCVBUF	The size (in bytes) of the SO_RCVBUF property on the socket. Valid values vary according to the operating system that you are using. This property is a standard TCP/IP property. The default value is 0 (zero), which sets the size of the SO_RCVBUF property to the operating system default.

Supplied configurable services that are created for each broker	Properties for each configurable service that is defined	Description of properties
	SO_SNDBUF	The size (in bytes) of the SO_SNDBUF property on the socket. Valid values vary according to the operating system that you are using. This property is a standard TCP/IP property. The default value is 0 (zero), which sets the size of the SO_SNDBUF property to the operating system default.
	SO_KEEPALIVE	The value of the KEEPALIVE property on the socket. If the value is set to True, the socket checks that it is still connected after a specified time. The length of time depends on the TCP/IP implementation on the operating system, but is typically two hours. Keep alive processing works only if the underlying operating system supports SO_KEEPALIVE. This property is a standard TCP/IP property. The default value is False, which means that no keep alive processing is performed.
	TCP_NODELAY	The value of the TCP_NODELAY property on the socket. If the value is set to True, the socket sends data as soon as it is sent to its buffer. The default value is False.
	TrafficClass	The traffic class on any connection that is established. Valid values are positive integers. The default value is -1, which leaves the TrafficClass set to the platform default.
	SO_LINGER	The SO_LINGER property on any connection that is established. This property is a standard TCP/IP property. The default value is False.
	SO_LINGER_TIMEOUT_SEC	The SO_LINGER_TIMEOUT_SEC property on any connection that is established. This property is a standard TCP/IP property. Valid values are positive integers. The default value is -1, which leaves the SO_LINGER_TIMEOUT_SEC value set to the operating system default.

Preparing the environment for WebSphere Adapters nodes:

Before you can use the WebSphere Adapters nodes, you must set up the broker runtime environment so that you can access the Enterprise Information System (EIS).

Before you start:

Read WebSphere Adapters nodes.

To enable the WebSphere Adapters nodes in the broker runtime environment, configure the broker with the location of the EIS provider JAR files and native libraries. (On Windows, the location of the JAR files cannot be a mapped network drive on a remote Windows computer; the directory must be local or on a Storage Area Network (SAN) disk.)

1. The WebSphere Adapters nodes require libraries from the EIS vendors. For more information on how to obtain and use these libraries, see Developing message flow applications using WebSphere Adapters.

2. Use the following commands to make the JAR files and shared libraries available to the WebSphere Adapters nodes.
   - To set up the dependencies, use the following command.

     ```
     mqsichangeproperties broker name -c EISProviders -o EIS type -n jarsURL -v jar directory
     mqsichangeproperties broker name -c EISProviders -o EIS type -n nativeLibs -v bin directory
     ```

     For example:

     ```
     mqsichangeproperties brk6 -c EISProviders -o SAP -n jarsURL -v c:\sapjco\jars
     mqsichangeproperties brk6 -c EISProviders -o SAP -n nativeLibs -v c:\sapjco\bin
     ```

     After you have run the mqsichangeproperties command, restart the broker so that the changes take effect.
   - To report the dependencies, use the following command.

     ```
     mqsireportproperties broker name -c EISProviders -o EIS type -r
     ```

     For example:

     ```
     mqsireportproperties brk6 -c EISProviders -o SAP -r
     ```
   - On z/OS, run this command by customizing and submitting BIPJADPR.

When you have set up the environment for the WebSphere Adapters nodes, you can perform the preparatory tasks that are listed in Developing message flow applications using WebSphere Adapters.

FtpServer configurable service properties:

Select the properties and values that you want to change for an existing FtpServer configurable service or to create a new service.

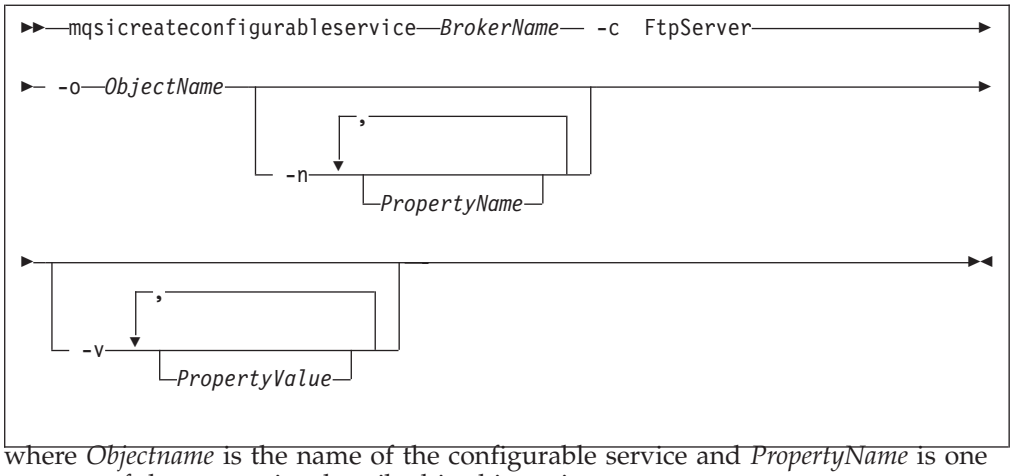
To change these properties, you must specify the broker name and **-c FtpServer**.

You must also set the **ObjectName** to the name of the configurable service that you have previously created.

See the "mqsichangeproperties command" on page 437 for examples of its use.

If you define an FtpServer configurable service by using the mqsicreateconfigurableservice command, you can then specify the name of this configurable service in the Remote server and port property on the **FTP** tab of the FileInput and FileOutput nodes.

To create an FtpServer configurable service, the command has the following syntax:



where *Objectname* is the name of the configurable service and *PropertyName* is one or more of the properties described in this topic.

If you define an FtpServer configurable service, you **must** specify a value for its serverName property. All the other properties are optional.

serverName
The internet address and, optionally, port number for the remote FTP or SFTP server. The syntax for the property is identical to that permitted for the Remote server and port property of the FileInput and FileOutput nodes (except that it cannot be the name of an FtpServer configurable service).

accountInfo
Some FTP servers require an account name during the FTP logon procedure. If this property is specified, its value is the account name supplied when requested during FTP logon. If this property is not specified and the server requests an account name, the FTP transfer will fail.

scanDelay
The period of time, in seconds, to wait after a scan of the directory has resulted in no files having been identified for processing. The default is 60 seconds. If set, this property overrides Scan delay on the **FTP** tab of the FileInput node which uses this service.

protocol
The remote transfer protocol to use. Valid values are FTP or SFTP. If no protocol is specified in the configurable service, the value specified in the node is used.

transferMode
The transfer mode of the FTP connection. This is either `BINARY` or `ASCII`. The default is `BINARY`. If set, this property overrides Transfer mode on the **FTP** tab of the FileInput or FileOutput node which uses this service.

This property is valid only when FTP is specified as the protocol. If SFTP is specified, this property is ignored.

connectionType
The FTP data socket connection. It is either `ACTIVE` or `PASSIVE`.

This property is valid only when FTP is specified as the protocol. If SFTP is specified, this property is ignored.

remoteDirectory
The relative or absolute directory name on the remote FTP server. If set, this property overrides Server directory on the **FTP** tab of the FileInput or FileOutput node which uses this service.

securityIdentity
The name of a security identity defined using the mqsisetdbparms command. If set, this property overrides Security identity on the **FTP** tab of the FileInput or FileOutput node that uses this service. If the value of this property is secId, define the security identity using the following command:

- If you are using FTP:

```
mqsisetdbparms WBRK61_DEFAULT_BROKER -n ftp::secId -u userName -p password
```

- If you are using SFTP:

```
mqsisetdbparms WBRK61_DEFAULT_BROKER -n sftp::secId -u userName -p password
```

or

```
mqsisetdbparms WBRK61_DEFAULT_BROKER -n sftp::secId -u userName -i SSHIdentityFile
-r Passphrase
```

cipher
The cipher used for encryption. This property takes the form of a list of one or more of the following values, separated by plus signs (+):

- blowfish-cbc
- 3des-cbc
- aes128-cbc

The cipher that you use for encryption depends on your SSH implementation. List the values in order of preference.

This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

If no value is specified, the following default is used: blowfish-cbc+3des-cbc+aes128-cbc

mac
Message Authentication Code. This property takes the form of a list of one or more of the following values, separated by plus signs (+):

- hmac-md5
- hmac-sha1

The MAC that you use depends on your SSH implementation. List the values in order of preference.

This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

If no value is specified, the following default is used: hmac-md5+hmac-sha1

compression
Specifies the level of compression to be used. Valid values are integers between 0 and 9, where 0 specifies no compression and 9 specifies maximum compression.

This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

If no value is specified, the default value of 0 is used.

strictHostKeyChecking
Specifies how host keys are checked during the connection and authentication phase. Valid values are:

No Specifies that the following action is performed:

- If the connection is to a new host, connect and accept the host's key, and store it

- If the connection is to a host that has been connected to previously and the host key has changed, issue an exception (FileOutput node).

If you select No, a default known hosts file (managed by the broker) is used.

Yes Connect only to known hosts with valid keys; otherwise issue an exception.

If you select Yes, you must specify your own known hosts file using the knownHostsFile property.

The default value is No.

The host information is stored in a `known_hosts` file in the standard OpenSSH format.

This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

knownHostsFile
The location of the known hosts file. The value must be the fully qualified path to a valid known hosts file.

The host information is stored in a `known_hosts` file in the standard OpenSSH format.

This property is mandatory if the strictHostKeyChecking property is set to Yes. If the strictHostKeyChecking property is set to No, this property is ignored.

This property is valid only when SFTP is specified as the protocol. If FTP is used, this property is ignored.

By default, none of these properties in the FtpServer configurable service definition is set. The only mandatory property when you define an FtpServer configurable service is serverName.

The following example of a mqsicreateconfigurableservice command shows how to create an FtpServer configurable service:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c FtpServer -o Server01
-n serverName,scanDelay,transferMode,connectionType,securityIdentity
-v one.hursley.abc.com:123,20,BINARY,ACTIVE,secId
```

With the exception of the accountInfo and connectionType properties, which you can set only on an FtpServer configurable service definition, you can set these properties either on an FtpServer configurable server definition or on the **FTP** tab of the FileInput and FileOutput nodes. Values set in properties in the FtpServer configurable service definition override the values set in the corresponding properties in the FileInput and FileOutput nodes.

If you set the accountInfo property, it is used during the login protocol when connecting to the FtpServer configurable service after supplying the user identifier and password. This information is sometimes required by FTP servers and requested as part of the login protocol. This setting allows the FileInput and FileOutput nodes to respond appropriately during login.

If you set the connectionType property, it alters the type of data socket that is used to transfer files to or from the FTP server. If you set this property to `ACTIVE`, this refers to a socket which is established by the remote server to the client (the broker message flow). If you set this property to `PASSIVE`, this refers to a socket which is established by the client to the remote server (as is the login or control socket). The

default is PASSIVE which is more likely to be tolerated by most types of firewall protection which already allows the client to login. You can set this property to ACTIVE if either the FTP server does not support PASSIVE connections, or there are special arrangements which your configuration must meet.

Changing connection details for IMS nodes:

You can configure IMS nodes to get connection details from a configurable service.

Before you start:

- Read IBM Information Management System (IMS) and Configurable services for background information.

Use the IMSConnect configurable service to change connection details for an IMS node. Each configurable service maps to a connection manager; therefore, nodes that use the same configurable service use the same manager. Two configurable services can connect to the same instance of IMS Connect. The properties of the IMS configurable services are described in "Configurable services properties" on page 444.

Creating, changing, reporting, and deleting configurable services

- To create a configurable service, run the mqsicreateconfigurableservice command, as shown in the following example. This example creates an IMSConnect configurable service for the IMS instance IMSA that is running on test.ims.ibm.com port 9999:

  ```
  mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c IMSConnect -o myIMSConnectService
  -n Hostname,PortNumber,DataStoreName -v test.ims.ibm.com,9999,IMSA
  ```

- To change a configurable service, run the mqsichangeproperties command, as shown in the following example. This example changes all the nodes that are configured to use the *myIMSConnectService* configurable service. After you run this command, all nodes connect to the production system (production.ims.ibm.com) instead of the test system (test.ims.ibm.com).

  ```
  mqsichangeproperties WBRK61_BROKER -c IMSConnect -o myIMSConnectService -n Hostname
  -v production.ims.ibm.com
  ```

- To display all IMSConnect configurable services, run the mqsireportproperties command, as shown in the following example:

  ```
  mqsireportproperties WBRK61_DEFAULT_BROKER -c IMSConnect -o AllReportableEntityNames -r
  ```

- You can delete a configurable service that you have created by running the mqsideleteconfigurableservice command, as shown in the following example:

  ```
  mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c IMSConnect -o myIMSconnectService
  ```

Setting up a JDBC provider for type 4 connections:

Use the mqsicreateconfigurableservice or the mqsichangeproperties command to configure a JDBC provider service.

Before you start:

- Create a broker
- Create your database following the database documentation.

When you include a DatabaseRetrieve, DatabaseRoute, JavaCompute, or Java user-defined node in a message flow, and interact with a database in that node, the broker must establish a connection with the database to fulfill the operations that

are performed by the node. You must define a JDBCProvider configurable service to provide the broker with the information that it needs to complete the connection.

A JDBCProvider configurable service supports connections to one database only; you must create a service for each database that your Java applications connect to.

1. Identify the type of database for which you require a JDBC provider. The following databases are supported with JDBC connections:
   - DB2
   - Informix
   - Oracle
   - SQL Server
   - Sybase

   Supported JDBC drivers are shown in Supported databases. JDBC type 4 connections to a SQLServer databases cannot participate in globally-coordinated (XA) transactions.
2. Run the mqsireportproperties command to view the list of available JDBC providers. Substitute the name of your broker in place of *broker_name*.

```
mqsireportproperties broker_name -c JDBCProviders -a -o AllReportableEntityNames
```

   This command displays a list of all the JDBC provider configurable services that are defined. If you have not created any new definitions, the following list of default supplied providers is shown:
   - DB2
   - Informix
   - Informix_With_Date_Format
   - Microsoft_SQL_Server
   - Oracle
   - Sybase_JConnect6_05

   If you are connecting to an Informix database:
   - Use Informix_With_Date_Format for compatibility with client applications that are dependent on the date format connection attribute that was used by earlier versions of Informix servers.
   - Use Informix for client applications that are not dependent on the date format attribute.
3. View the contents of the relevant definition. For example, run the following command to display the supplied Oracle definition:

```
mqsireportproperties broker_name -c JDBCProviders -o Oracle -r
```

   The command returns a list of all the properties for the Oracle definition. If you have not changed this definition, the properties are set to initial values, some of which you must change to create a viable definition. For example, the property **databaseName** is set to default_Database_Name, and you must change it to identify the specific database that you want to connect to.

   A JDBCProvider has the following properties:
   - **connectionUrlFormat**. A pattern that represents the connection URL definition, which is specific to a particular database type. For example, the pattern for DB2 is defined with the following content:

     ```
     jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];
     ```

     The pattern is used and completed by the broker at run time when it connects to the database. The values in brackets, for example `[serverName]`, are substituted by the broker into the pattern by using the values that you have specified on the mqsicreateconfigurableservice or mqsichangeproperties commands.

Do not use the mqsichangeproperties command to change the pattern itself; changes made to the pattern might cause unpredictable results.

- **connectionUrlFormat Attr1-5**. If the defined URL pattern for a database contains non-standard JDBC data source properties, such as a server identifier, specify these properties in addition to the standard attributes by using one of five general purpose connection URL attributes. For example:
  - If **connectionURLFormat** = jdbc:oracle:thin:[user]/[password]@[serverName]:[portNumber]:[connectionUrlFormatAttr1], **connectionUrlFormatAttr1** must contain an Oracle server identifier, which you must supply by defining the value for the property **connectionUrlFormatAttr1** on the mqsicreateconfigurableservice or mqsichangeproperties command. The broker can then substitute all the required values into the required pattern.
  - If **connectionURLFormat** = jdbc:informix-sqli://[serverName]:[portNumber]/[databaseName]:informixserver=[connectionUrlFormatAttr1];user=[user];password=[password], **connectionUrlFormatAttr1** must contain the name of the Informix instance on the server (typically specified by the INFORMIXSERVER environment variable). This value is case-sensitive.
- **databaseName**. The name of the database to which the data source entry enables connections; for example, employees.
- **databaseType**. The database type; for example, `DB2`.
- **databaseVersion**. The database version; for example, 9.1.
- **description**. An optional property to describe the data source definition.
- **jarsURL**. The local directory path, on the system on which the broker is running, where the JAR file that contains the type 4 driver class is located.

  In addition, a Storage Area Network disk can be used for the directory path, but a mapped network drive to a remote machine cannot be used.
- **portNumber**. The port number on which the database server is listening; for example, 50000.
- **securityIdentity**. A unique security key to perform a second broker registry lookup to find an entry under the broker's security identities, which store the encrypted password for the user on their associated host system; for example, mysecurityIdentity.

  Create a new security identity by using the mqsisetdbparms command, as described in "Securing a JDBC type 4 connection" on page 147. The value of **securityIdentity** (for example, mysecurityIdentity) must match the value that you specify following the prefix `jdbc::` for the parameter **-n** on that command.

  The security identity provides a user ID and password value pair, which are used to access the specified data source defined for a given JDBC provider entry. This property is ignored if the connection URL does not contain both a user ID and password pair, which require property values to be substituted for such inserts. The DataSourceUserId and DataSourcePassword properties under which the broker was created are used under the following conditions:
  - If the **securityIdentity** is blank, or if you have not changed it from the default value default_User@default_Server, but the identity is required for the connection URL pattern.
  - If you have entered a valid unique security identity key, but it cannot be found under the DSN key.

z/OS On z/OS, you must specify a user ID and password with appropriate authorization to access the database. Do not use the broker-started-task user ID for this purpose. (Assign a password to a started-task user ID only after considering all of the potential security implications.)

- **serverName**. The name of the server; for example, host1.
- **type4DatasourceClassName**. The name of the JDBC data source class name that is used to establish a type 4 connection to a remote database, and to coordinate transaction support. For example, specify com.ibm.db2.jcc.DB2XADataSource for DB2, or specify oracle.jdbc.xa.client.OracleXADataSource for Oracle. You must always specify the XA class name, even if you do not use coordinated transactions.
- **type4DriverClassName**. The name of the JDBC Type 4 driver class name that is used to establish a connection. For example, specify com.ibm.db2.jcc.DB2Driver for DB2, or specify oracle.jdbc.OracleDriver for Oracle.

4. If you want to use the provided definition, run the mqsichangeproperties command to replace default values with those specific to your database and environment. If you are in any doubt about the required values, consult your database administrator, or check the documentation that is provided with your chosen database. Some values depend on how and where you have installed the database product; for example, the property **jarsURL** identifies the location of the JAR files supplied and installed by the database provider.
5. If you want to create a new configurable service, perhaps because you want to retain the supplied service as a template for future definitions, run the mqsicreateconfigurableservice command to create the new definition.

```
mqsicreateconfigurableservice broker_name -c JDBCProviders -o provider_name
-n list of properties -v list of values
```

Enter the command on a single line; the example has been split to enhance readability.

Specify all the properties that are required by the database provider that you have chosen. To specify a list of properties and values, separate the items after each flag with a comma. For example, `-n databaseName,databaseType -v EmployeeDB,DB2`. If you do not specify all the properties on the mqsicreateconfigurableservice command, you can update them later with the mqsichangeproperties command.

6. When you have set up your JDBCProvider service, you must stop and restart the broker.

Next: if required, set up security for the JDBC connection, set up the environment to include the JDBCProvider service in globally-coordinated transactions, or both.

Configuring the broker to enable a JMS provider's proprietary API:

Some JMS providers provide an alternative interface to the standard JMS specification for particular JMS API calls. In these cases, IBM supplies a Java class to interface with that proprietary API.

For example, BEA WebLogic uses a component called a *Client Interposed Transaction Manager* to allow a JMS client to obtain a reference to the XAResource that is associated with a user transaction.

If the WebSphere Message Broker JMS nodes use BEA WebLogic as the JMS provider, and the nodes must participate in a globally coordinated message flow,

you must modify the configurable services properties that are associated with that vendor. The following table shows the properties that have been added to the configurable service for BEA WebLogic.

JMS provider	Property	Purpose	Default value
BEA_WebLogic	proprietaryAPIHandler	The name of the IBM supplied Java class to interface with a JMS provider's proprietary API.	`com.ibm.broker.apihandler. BEAWebLogicAPIHandler`
	proprietaryAPIAttr1	The Initial Context Factory class name for the vendor	`weblogic.jndi. WLInitialContextFactory`
	proprietaryAPIAttr2	The URL of the WebLogic bindings	*URL JNDI bindings*
	proprietaryAPIAttr3	The DNS name of the JMS server	*Server name*

In the list of JMS provider configurable services, the name of the IBM supplied Java class is set to the default value for the proprietaryAPIHandler property. Typically, you do not need to change this value, unless you are instructed to do so by an IBM Service team representative.

- Use the mqsichangeproperties command to modify values of the properties for this JMS provider.

  The following example shows how to change the values of the properties proprietaryAPIAttr2 and proprietaryAPIAttr3 for the JMS provider configurable service definition called BEA_Weblogic, where these properties represent the URL of the WebLogic bindings and the DNS Server name of the BEA WebLogic JMS Server:

  ```
  mqsichangeproperties WBRK61_DEFAULT_BROKER -c JMSProviders -o BEA_Weblogic
  -n proprietaryAPIAttr2,proprietaryAPIAttr3 -v t3://9.20.94.16:7001,BEAServerName
  ```

- Use the mqsireportproperties command to display the properties for a JMS provider.

  The following example shows how to display the properties for all the broker's JMS provider resources (the default JMS provider resources and those configurable services that are defined with the mqsicreateconfigurableservice command):

  ```
  mqsireportproperties WBRK61_DEFAULT_BROKER -c JMSProviders -o BEA_WebLogic –r
  ```

  The result of this command has the following format:

  ```
  ReportableEntityName=''
  JMSProviders
    BEA_Weblogic=''
      jarsURL='default_Path'
      nativeLibs='default_Path'
      proprietaryAPIAttr1='weblogic.jndi.WLInitialContextFactory'
      proprietaryAPIAttr2='t3://9.20.94.16:7001'
      proprietaryAPIAttr3='BEAServerName'
      proprietaryAPIAttr4='default_none'
      proprietaryAPIAttr5='default_none'
      proprietaryAPIHandler='com.ibm.broker.apihandler.BEAWebLogicAPIHandler'
  ```

  The default location for the JMS provider JAR files is the broker's shared-classes directory. You can specify an alternative location for the JAR files by using the mqsichangeproperties command, as shown in the following example:

  ```
  mqsichangeproperties WBRK61_DEFAULT_BROKER -c JMSProviders -o BEA_WebLogic -n jarsURL
  -v /var/mqsi/WebLogic
  ```

On Windows, the file location cannot be a mapped network drive on a remote Windows computer; the directory must be local or on a Storage Area Network (SAN) disk.

- Use the mqsicreateconfigurableservice command to add a JMS provider.

  The following example shows how to add a JMS provider called BEAV91 for broker WBRK61_DEFAULT_BROKER, specifying the name of an IBM supplied Java class called `com.ibm.broker.apihandler.BEAWebLogicAPIHandler` to handle vendor-specific API calls:

  ```
  mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c JMSProviders -o BEAV91
  -n proprietaryAPIHandler,proprietaryAPIAttr1,proprietaryAPIAttr2,proprietaryAPIAttr3
  -v com.ibm.broker.apihandler.BEAWebLogicAPIHandler,weblogic.jndi.WLInitialContextFactory,
  t3://9.20.94.16:7001,BEAServerName
  ```

- If you have defined a user-defined JMS provider configurable service, set the value for the proprietaryAPIHandler property manually.

Configuring monitoring event sources using a monitoring profile:

You can create a monitoring profile and use the mqsichangeflowmonitoring command to configure your message flows to emit monitoring events.

Before you start:

Read the following topics:

- Monitoring message flows
- Monitoring basics

You must have a message flow that contains a node to which you want to add a monitoring event.

Creating a monitoring profile:

First create a monitoring profile XML file. This is a file that lists the event sources in the message flow that will emit events, and defines the properties of each event.

Follow the guidance at Monitoring profile to create your monitoring profile XML file.

Applying a monitoring profile:

When you have created a monitoring profile XML file, follow these steps to apply it.

1. Use the mqsicreateconfigurableservice command to create a configurable service for the monitoring profile. In the following command example, replace *myBroker* with the name of your broker, and *myMonitoringProfile* with the name of your monitoring profile.

   ```
   mqsicreateconfigurableservice myBroker -c MonitoringProfiles
        -o myMonitoringProfile
   ```

2. Use the mqsichangeproperties command to associate your monitoring profile XML file with the configurable service. In the following command example, replace *myBroker* with the name of your broker, *myMonitoringProfile* with the name of your monitoring profile, and *myMonitoringProfile.xml* with the name of the monitoring profile XML file.

   ```
   mqsichangeproperties myBroker -c MonitoringProfiles -o myMonitoringProfile
        -n profileProperties -p myMonitoringProfile.xml
   ```

3. Use the mqsichangeflowmonitoring command to apply a monitoring profile configurable service to one or more message flows.
   - Apply a monitoring profile to a single message flow *messageflow1* in execution group *EG1*:
     ```
     mqsichangeflowmonitoring myBroker -e EG1
          -f messageflow1 -m myMonitoringProfile
     ```
   - Apply a monitoring profile to all message flows in all execution groups :
     ```
     mqsichangeflowmonitoring myBroker -g -j -m myMonitoringProfile
     ```

   Monitoring for the flow is inactive; applying the monitoring profile does not activate it.
4. Alternatively, use the broker archive editor to apply a monitoring profile configurable service to one or more message flows, by setting message flow property **Monitoring Profile Name**.
   a. In the workbench, switch to the Broker Application Development perspective.
   b. In the Broker Development view, right-click the BAR file and then click **Open with** > **Broker Archive Editor**.
   c. Click the **Manage and Configure** tab.
   d. Click the message flow on which you want to set the monitoring profile configurable service. The properties that you can configure for the message flow are displayed in the **Properties** view.
   e. In the **Monitoring Profile Name** field, enter the name of a monitoring profile.
   f. Save the BAR file.
   g. Deploy the BAR file.

   Monitoring for the flow is inactive; deploying the BAR file does not activate it.
5. Activate monitoring for the flow using the `mqsichangeflowmonitoring -c` command.

Updating a monitoring profile:

1. Follow the guidance at Monitoring profile to update your monitoring profile XML file.
2. Use the mqsichangeproperties command to update the configurable service to use the new XML file. For example:
   ```
   mqsichangeproperties myBroker -c MonitoringProfiles -o myMonitoringProfile
        -n profileProperties -p myMonitoringProfile.xml
   ```

Changing connection details for PeopleSoft adapters:

PeopleSoft nodes can get PeopleSoft connection details from either the adapter component or a configurable service. By using configurable services, you can change the connection details for adapters without the need to redeploy the adapters. To pick up new values when a configurable service is created or modified, you must reload the broker or execution group to which the adapter was deployed, by using the mqsistop and mqsistart commands, or the mqsireload command.

Before you start:

- Read WebSphere Adapters nodes and Configurable services for background information.

Use the PeopleSoftConnection configurable service to change connection details for a PeopleSoft adapter. The PeopleSoft node reads all connection properties from the adapter component that it is configured to use. If a configurable service exists that has the same name as the node's adapter component, the node uses the values that are defined in that configurable service to override the corresponding properties from the adapter. If a configurable service is being used, all properties that are exposed by the configurable service are taken from the configurable service. The only properties that are taken from the adapter are those that you cannot set on the configurable service. The properties of the PeopleSoft configurable service are described in "Configurable services properties" on page 444.

Creating, changing, reporting, and deleting configurable services

- To create a configurable service, run the mqsicreateconfigurableservice command, as shown in the following example. This example creates a PeopleSoftConnection configurable service for the PeopleSoft instance that is running on *my.peoplesoft.qa.com*:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c PeopleSoftConnection -o myPeopleSoftAdapter.outadapter
-n hostName,port -v "my.peoplesoft.qa.com",9000
```

  To pick up the new values in the configurable service, restart the execution group and message flow.
- To change a configurable service, run the mqsichangeproperties command, as shown in the following example. This example changes the connections that are used by the adapter *myPeopleSoftAdapter.outadapter*. After you run this command, all adapters connect to the production system (`my.peoplesoft.production.com`) instead of the test system (`my.peoplesoft.qa.com`):

```
mqsichangeproperties WBRK61_BROKER -c PeopleSoftConnection -o myPeopleSoftAdapter.outadapter -n hostName
-v "my.peoplesoft.production.com"
```

  To pick up the updated values in the configurable service, restart the execution group and message flow.
- To display all PeopleSoftConnection configurable services, run the mqsireportproperties command, as shown in the following example:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c PeopleSoftConnection -o AllReportableEntityNames -r
```

- You can delete a configurable service that you have created by running the mqsideleteconfigurableservice command, as shown in the following example:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c PeopleSoftConnection -o myPeopleSoftAdapter.outadapter
```

Importing a policy set and policy set binding:

Use the mqsichangeproperties command to import a policy set and associated binding.

This topic shows how to import policy set *myPolicySet* to broker *myBroker* from a file called *myPolicySet.xml*. The associated binding is *myPolicySetBinding*, which you import from *myPolicySetBinding.xml*.

1. Create a configurable service for the policy set, if one does not already exist.

   ```
   mqsicreateconfigurableservice myBroker -c PolicySets -o myPolicySet
   BIP8071I: Successful command completion.
   ```

2. Create a configurable service for the policy set binding, if one does not already exist.

   ```
   mqsicreateconfigurableservice myBroker -c PolicySetBindings -o myPolicySetBinding
   BIP8071I: Successful command completion.
   ```

3. Import the policy set.

```
mqsichangeproperties myBroker -c PolicySets -o myPolicySet -n ws-security -p myPolicySet.xml
```

4. Import the policy set binding.

```
mqsichangeproperties myBroker -c PolicySetBindings -o myPolicySetBinding
   -n ws-security -p myPolicySetBinding.xml
```

5. Change the value of the associatedPolicySet attribute. Set it to the name of the policy set with which this policy set binding was originally associated.

```
mqsichangeproperties myBroker -c PolicySetBindings -o myPolicySetBinding
   -n associatedPolicySet -v myPolicySet
```

PublishSubscribe configurable service properties:

Select the properties and values that you want to change for the PublishSubscribe configurable service.

To change these properties, you must specify the broker name and **-c PublishSubscribe**.

You must also set the **ObjectName** to Interface.

-n psmode
The state to which you want to set the broker's publish/subscribe engine. Valid values are enabled and disabled.

If you disable the broker's publish/subscribe engine, all application messages and operations are managed by the broker's queue manager. This option is valid only if you have installed WebSphere MQ Version 7.0 on this computer and created or migrated the queue manager to this version.

The default setting of **psmode** is enabled. If WebSphere MQ Version 7.0 is not installed on this computer, you can change this setting, but the option is ignored.

See the "mqsichangeproperties command" on page 437 for examples of how to disable and enable the broker's publish/subscribe engine.

Changing connection details for SAP adapters:

SAP nodes can get SAP connection details from either the adapter component or a configurable service. By using configurable services, you can change the connection details for adapters without the need to redeploy the adapters. To pick up new values when a configurable service is created or modified, you must reload the broker or execution group to which the adapter was deployed, by using the mqsistop and mqsistart commands, or the mqsireload command.

Before you start:

- Read WebSphere Adapters nodes and Configurable services for background information.

Use the SAPConnection configurable service to change connection details for an SAP adapter. The SAP node reads all connection properties from the adapter component that it is configured to use. If a configurable service exists that has the same name as the node's adapter component, the node uses the values that are defined in that configurable service to override the corresponding properties from the adapter. If any properties on the configurable service are set to an empty string, the values that are configured in the `.inadapter` or `.outadapter` file are used. The properties of the SAP configurable services are described in "Configurable services properties" on page 444.

Creating, changing, reporting, and deleting configurable services

- To create a configurable service, run the mqsicreateconfigurableservice command, as shown in the following example. This example creates a SAPConnection configurable service for the SAP adapter *mySAPAdapter.outadapter* that connects to the SAP host `test.sap.ibm.com`, and uses client *001* for the connections into that server:

  ```
  mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c SAPConnection
  -o mySAPAdapter.outadapter -n applicationServerHost,client -v test.sap.ibm.com,001
  ```

  To pick up the new values in the configurable service, restart the execution group and message flow.
- To change a configurable service, run the mqsichangeproperties command, as shown in the following example. This example changes the connections that are used by the adapter *mySAPAdapter.outadapter*. After you run this command, all adapters connect to the production system (`production.sap.ibm.com`) instead of the test system (`test.sap.ibm.com`):

  ```
  mqsichangeproperties WBRK61_BROKER -c SAPConnection -o mySAPAdapter.outadapter
  -n applicationServerHost -v production.sap.ibm.com
  ```

  To pick up the updated values in the configurable service, restart the execution group and message flow.
- To display all SAPConnection configurable services, run the mqsireportproperties command, as shown in the following example:

  ```
  mqsireportproperties WBRK61_DEFAULT_BROKER -c SAPConnection -o AllReportableEntityNames -r
  ```
- You can delete a configurable service that you have created by running the mqsideleteconfigurableservice command, as shown in the following example:

  ```
  mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c SAPConnection
  -o mySAPAdapter.outadapter
  ```

SecurityProfiles configurable service properties:

Select the objects and properties that you want to change for the SecurityProfiles configurable service.

To change these properties, you must specify the broker name and **-c SecurityProfiles**.

You must also set the **ObjectName** to either Default_Propagation or the name of a SecurityProfiles configurable service that you have defined by using the mqsicreateconfigurableservice. The properties and values are the same for all services.

This configurable service is independent of the securitycache component.

-n authentication

The type of authentication that is performed on the source identity.

- Value type - enum
- Initial value - NONE
- Other valid values:
  - LDAP
  - TFIM
  - A user-defined value

-n authenticationConfig
The information that the broker needs to connect to the provider, specific to the provider.

- Value type - string
- Initial value - None

-n authorization
The types of authorization checks that are performed on the mapped or source identity.

- Value type - enum
- Initial value - NONE
- Other valid values:
  - LDAP
  - TFIM
  - A user-defined value

-n authorizationConfig
How the broker connects to the provider, specific to the provider.

- Value type - string
- Initial value - None

-n mapping
The type of mapping that is performed.

- Value type - enum
- Initial value - NONE
- Other valid values:
  - TFIM
  - A user-defined value

-n mappingConfig
How the broker connects to the provider, specific to the provider.

- Value type - string
- Initial value - None

-n passwordValue
How passwords are treated when they enter a message flow.

- Value type - enum
- Initial value - PLAIN
- Other valid values:
  - MASK
  - OBFUSCATE

-n propagation
Indicates whether identity propagation is performed on output and request nodes.

- Value type - Boolean
- Initial value - TRUE

See the "mqsichangeproperties command" on page 437 for examples of its use.

Changing the configuration parameters for the WebSphere Service Registry and Repository nodes:

Use the mqsichangeproperties command to change the configuration parameters of the DefaultWSRR configurable service.

DefaultWSRR is a configurable service object that is supplied for each broker, it defines the WebSphere Service Registry and Repository (WSRR) configuration parameters. DefaultWSRR has a configurable service type of ServiceRegistries.

For details about configuration parameters that affect WSRR use, see Configuration parameters for the WebSphere Service Registry and Repository nodes.

To update the configuration parameters of the DefaultWSRR configurable service complete the following steps:

1. Ensure that the broker is running. If it is not, use the mqsistart command to start it.
2. Enter the following command to change the endpointAddress value and point to your WebSphere Service Registry and Repository server:

   ```
   mqsichangeproperties WBRK_BROKER -c ServiceRegistries -o DefaultWSRR
   -n endpointAddress
   -v http://localhost:9080/WSRR6_1/services/WSRRCoreSDOPort
   ```

   where:

   -c specifies the configurable service type
   (in this case, ServiceRegistries)
   -o specifies the name of the configurable service object
   (in this case, DefaultWSRR)
   -n specifies the names of the properties to be changed
   (in this case, endpointAddress)
   -v specifies the values of properties defined by the **-n** parameter
   (in this case,
   http://localhost:9080/WSRR6_1/services/WSRRCoreSDOPort)
   For WSRR 6.2 this becomes:
   http://localhost:9080/WSRR6_2/services/WSRRCoreSDOPort)
3. (Optional) Enter the following command to change the cache timeout value:

   ```
   mqsichangeproperties WBRK_BROKER -c ServiceRegistries -o DefaultWSRR
   -n timeout -v 3600000
   ```

   where:

   -c specifies the configurable service type
   (in this case, ServiceRegistries)
   -o specifies the name of the configurable service object
   (in this case, DefaultWSRR)
   -n specifies the names of the properties to be changed
   (in this case, timeout)
   -v specifies the values of properties defined by the **-n** parameter
   (in this case, 3600000 milliseconds to provide WSRR cache expiry timeout of 1 hour)
4. (Optional) Enter the following command to change the connectionTimeout value:

   ```
   mqsichangeproperties WBRK_BROKER -c ServiceRegistries -o DefaultWSRR
   -n connectionTimeout -v 240
   ```

   where:

-c specifies the configurable service type
(in this case, ServiceRegistries)
-o specifies the name of the configurable service object
(in this case, DefaultWSRR)
-n specifies the names of the properties to be changed
(in this case, connectionTimeout)
-v specifies the values of properties defined by the **-n** parameter
(in this case, 240 seconds to provide connection timeout for WSRR queries of 4 minutes)

5. (Optional) Enter the following command to preload the cache at broker startup with the results of specific queries:

```
mqsichangeproperties WBRK_BROKER -c ServiceRegistries -o DefaultWSRR
-n predefinedCacheQueries
-v "//*[@name=&apos;ConceptA1&apos;]"
```

where:

-c specifies the configurable service type
(in this case, ServiceRegistries)
-o specifies the name of the configurable service object
(in this case, DefaultWSRR)
-n specifies the names of the properties to be changed
(in this case, predefinedCacheQueries)
-v specifies the values of properties defined by the **-n** parameter
(in this case a simple full depth WSRR XPath query on the entity `ConceptA1`, `"//*[@name=&apos;ConceptA1&apos;]"`).
Note that single quotes in the WSRR query must be replaced by ')

Multiple queries can be specified, by delimiting them with ';'
For example, to perform a full depth query on the entities named `ConceptA1` and `ConceptB2` use:

```
-v "//*[@name=&apos;ConceptA1&apos;];//*[@name=&apos;ConceptB2&apos;]"
```

Individual queries can use the full power of the WSRR query language:

```
-v "/WSRR/WSDLService/ports[binding/portType
[@name=&apos;DemoCustomer&apos;
and @namespace=&apos;http://demo.sr.eis.ibm.com&apos;]]"
```

6. Restart the broker, by using the mqsistop command to stop the broker, followed by the mqsistart command to start it.

Changing connection details for Siebel adapters:

Siebel nodes can get Siebel connection details from either the adapter component or a configurable service. By using configurable services, you can change the connection details for adapters without the need to redeploy the adapters. To pick up new values when a configurable service is created or modified, you must reload the broker or execution group to which the adapter was deployed, by using the mqsistop and mqsistart commands, or the mqsireload command.

Before you start:

- Read WebSphere Adapters nodes and Configurable services for background information.

Use the SiebelConnection configurable service to change connection details for a Siebel adapter. The Siebel node reads all connection properties from the adapter

component that it is configured to use. If a configurable service exists that has the same name as the node's adapter component, the node uses the values that are defined in that configurable service to override the corresponding properties from the adapter. If a configurable service is being used, all properties that are exposed by the configurable service are taken from the configurable service. The only properties that are taken from the adapter are those that you cannot set on the configurable service. The properties of the Siebel configurable service are described in "Configurable services properties" on page 444.

Creating, changing, reporting, and deleting configurable services

- To create a configurable service, run the mqsicreateconfigurableservice command, as shown in the following example. This example creates a SiebelConnection configurable service for the Siebel instance that is running on *my.siebel.qa.com*:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c SiebelConnection -o mySiebelAdapter.outadapter
-n connectString -v "siebel://my.siebel.qa.com/SBA_80/SSEObjMgr_enu"
```

  To pick up the new values in the configurable service, restart the execution group and message flow.
- To change a configurable service, run the mqsichangeproperties command, as shown in the following example. This example changes the connections that are used by the adapter *mySiebelAdapter.outadapter*. After you run this command, all adapters connect to the production system (`my.siebel.production.com`) instead of the test system (`my.siebel.qa.com`):

```
mqsichangeproperties WBRK61_BROKER -c SiebelConnection -o mySiebelAdapter.outadapter -n connectString
-v "siebel://my.siebel.production.com/SBA_80/SSEObjMgr_enu"
```

  To pick up the updated values in the configurable service, restart the execution group and message flow.
- To display all SiebelConnection configurable services, run the mqsireportproperties command, as shown in the following example:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c SiebelConnection -o AllReportableEntityNames -r
```

- You can delete a configurable service that you have created by running the mqsideleteconfigurableservice command, as shown in the following example:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c SiebelConnection -o mySiebelAdapter.outadapter
```

HTTPConnector and HTTPSConnector parameter values (SOAP node connections):

Select the objects and properties associated with SOAP nodes that you want to change.

To change these properties, you must specify the broker name and **-e** with either the name of a single execution group, or the special value AllExecutionGroups. You can define all of these properties at the execution group level; therefore they apply to all SOAPInput and SOAPReply nodes that you deploy to the specified execution group.

If you want to change properties associated with HTTPInput and HTTPReply nodes, see "httplistener component parameter values" on page 480.

Choose the **ObjectName** from the following options:
- HTTPConnector for controlling HTTP communication.
- HTTPSConnector for controlling HTTPS communication.

The following combinations are valid:

-o HTTPConnector

The following properties and values are valid:

-n 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

-n explicitlySetPortNumber

The TCP/IP port number on which this Connector creates a server socket and awaits incoming connections.

Setting this value disconnects the automatic port-finding capability of the connector; this port is the only one allowed, and the connector fails to start if another program has already used this port.

- Value type - integer
- Initial value - 7800
- Other valid values - any integer in the range 0-65536.

-o HTTPSConnector

The properties listed for object name **HTTPConnector** are also valid for this object name. The following additional properties and values are valid:

-n algorithm

The certificate encoding algorithm to be used.

- Value type - string
- Initial value -
  - Solaris HP-UX SunX509 on Solaris and HP-UX
  - AIX z/OS Linux Windows IbmX509 on other systems (AIX, Linux, Windows, z/OS)

-n 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

-n 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 previously)

-n keystorePass

The password used to access the server certificate from the specified keystore file.

- Value type - string
- Initial value - changeit

-n keystoreType

The type of keystore file to be used for the server certificate.

- Value type - string
- Initial value - JKS

-n sslProtocol

The version of the SSL protocol to use.

- Value type - string
- Initial value - TLS

-n ciphers

A comma separated list of the encryption ciphers that can be used. If not specified (the default), then any available cipher can be used.

- Value type - string
- Initial value - null

The valid values for keystoreType, sslProtocol, and ciphers are JSSE-implementation specific, and these values are in the JSSE provider documentation.

See the "mqsichangeproperties command" on page 437 for examples of its use.

httplistener component parameter values:

Select the objects and properties associated with HTTP nodes that you want to change.

To change these properties, you must specify the broker name and **-b httplistener**. The httplistener component defines properties for the broker that are used for all the HTTPInput and HTTPReply nodes that you deploy to that broker, including a single listener for all HTTP nodes.

If you want to change properties associated with SOAPInput and SOAPReply nodes, see "HTTPConnector and HTTPSConnector parameter values (SOAP node connections)" on page 478.

Choose the **ObjectName** from the following options:

- HTTPListener for controlling the HTTPListener process

  You can use the following values with HTTPListener:

```
uuid='HTTPListener'
  enableSSLConnector='false'
  threadPoolSize=''
  traceOverrideLevel=''
  traceOverrideSize=''
  traceLevel='none'
  traceSize='4096 KB'
  javaDebugPort=''
```

- HTTPConnector for controlling HTTP communication.

  You can use the following values with HTTPConnector:

```
 uuid='HTTPConnector'
  address=''
  port='7080'
  allowTrace=''
  maxPostSize=''
  acceptCount=''
  bufferSize=''
  compressableMimeTypes=''
  compression=''
  connectionLinger=''
```

```
  connectionTimeout=''
  maxHttpHeaderSize=''
  maxKeepAliveRequests=''
  maxSpareThreads=''
  maxThreads='20'
  minSpareThreads=''
  noCompressionUserAgents=''
  restrictedUserAgents=''
  socketBuffer=''
  tcpNoDelay=''
  enableLookups='false'
```

- HTTPSConnector for controlling HTTPS communication.

  You can use the following values with HTTPSConnector:

```
uuid='HTTPSConnector'
  algorithm='Platform Default'
  clientAuth='Platform Default'
  keystoreFile='Platform Default'
  keystorePass=''
  keystoreType='Platform Default'
  sslProtocol='Platform Default'
  ciphers='Platform Default'
  address=''
  port=''
  allowTrace=''
  maxPostSize=''
  acceptCount=''
  bufferSize=''
  compressableMimeTypes=''
  compression=''
  connectionLinger=''
  connectionTimeout=''
  maxHttpHeaderSize=''
  maxKeepAliveRequests=''
  maxSpareThreads=''
  maxThreads=''
  minSpareThreads=''
  noCompressionUserAgents=''
  restrictedUserAgents=''
  socketBuffer=''
  tcpNoDelay=''
  enableLookups='false'
```

The following combinations are valid for the httplistener component:

-o HTTPListener

The following properties and values are valid:

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

-n maxKeepAliveRequests

The value is the maximum number of HTTP requests that can be pipelined until the connection is closed by the server. Setting the attribute to 1 disables HTTP/1.0 keep-alive, as well as HTTP/1.1 keep-alive and pipelining. Setting the value to -1 allows an unlimited amount of pipelined or keep-alive HTTP requests.

- Value type - integer
- Initial value -100

-n maxThreads

The value is the maximum number of request processing threads to be created by this Connector. This value, therefore, determines the maximum number of simultaneous requests that can be handled.

- Value type - integer
- Initial value - 200

enableSSLConnector

A Boolean value which can be used to enable or disable the HTTPS (SSL) connector. You must set this value to TRUE to start the HTTP listener listening for inbound SSL connections.

- Value type - Boolean
- Initial value - FALSE

-o HTTPConnector

The following properties and values are valid:

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 HTTPConnector creates a server socket and awaits incoming connections.

- Value type - integer
- Initial value - 7080

-o HTTPSConnector

The properties listed for object name **HTTPConnector** are also valid for this object name. The following additional properties and values are valid:

algorithm

The certificate encoding algorithm to be used.

- Value type - string
- Initial value -
  - Solaris HP-UX SunX509 on Solaris and HP-UX
  - AIX z/OS Linux Windows IbmX509 on other systems (AIX, Linux, Windows, z/OS)

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 expects a file called `.keystore` in the home directory of the user who started the broker.
- Value type - string
- Initial value - *default value* (described previously)

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 can be used. If not specified (the default), then any available cipher can be used.
- Value type - string
- Initial value - null

The properties listed for object name **HTTPConnector** are also valid for this object name. The valid values for keystoreType, sslProtocol, and ciphers are JSSE-implementation specific, and these values are in the JSSE provider documentation.

See the "mqsichangeproperties command" on page 437 for examples of how to change parameters for the httplistener component.

Other examples are provided for particular tasks:

"Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)" on page 63

Inter-broker communications parameter values (collectives):

Select the properties and values that you want to change for inter-broker communications for brokers that you have configured in collectives.

To change these properties, you must specify the broker name and **-e** with the name of an execution group. You must also specify DynamicSubscriptionEngine for the **ObjectName**.

The following properties and values are valid for execution groups for brokers that you have configured in collectives:

-n 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

-n 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. However, 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

See the "mqsichangeproperties command" on page 437 for examples of how to change parameters for brokers in collectives.

Inter-broker communications parameter values (clones):

Select the properties and values that you want to change for inter-broker communications for cloned brokers.

To change these properties, you must specify the broker name and **-e** with the name of an execution group. You must also specify DynamicSubscriptionEngine for the **ObjectName**.

The following properties and values are valid for execution groups for cloned brokers:

-n brokerPingInterval
The 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

-n clientPingInterval
The 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

-n clonedPubSubBrokerList
The list of brokers in which *brokername* registers to be a clone.
- Value type - string
- Initial value - none

See the "mqsichangeproperties command" on page 437 for examples of how to change parameters for cloned brokers. Other examples are provided for particular tasks:

"Setting up cloned brokers" on page 277

"Adding a cloned broker" on page 278

"Deleting a cloned broker" on page 278

JVM parameter values:

Select the objects and properties associated with the Java Virtual Machine (JVM) that you want to change.

To change these properties, you must specify the broker name, and set the **ObjectName** to ComIbmJVMManager. If you are setting a debug port number, you must also include **-e** and specify the name of the execution group which will use the port. If you are changing the heap size, omit **-e**; the size is effective for all execution groups created in the broker.

-n jvmMinHeapSize
The minimum size of the storage available to the JVM, specified in bytes.

- Value type - integer

-n jvmMaxHeapSize
The maximum size of the storage available to the JVM, specified in bytes. If you have configured a publish/subscribe domain, when execution groups retain publications, you might need to increase this value. If you have included the XSLTransform node in one or more message flows, and the node is processing very large XML messages, you might also need to change this parameter.

- Value type - integer

-n jvmDebugPort
The port on which the execution group is listening. You must set a port number to activate debug in the execution group.

- Value type - integer
- Initial value - 0

See the "mqsichangeproperties command" on page 437 for examples of how to change parameters for the JVM. Other examples are provided for particular tasks:

"Publish/subscribe performance tuning" on page 225

Using the Test Client in trace and debug mode

Attaching the flow debugger to an execution group for debugging

"Setting the JVM heap size" on page 224

Real-time nodes parameter values:

Select the properties and values associated with the Real-time nodes that you want to change.

To change these properties, you must specify the broker name and **-e** with the name of an execution group. You must also specify DynamicSubscriptionEngine for the **ObjectName**.

The following properties and values are valid for use with Real-timeInput and Real-timeOptimizedFlow nodes:

-n httpDispatchThreads
The number of threads that the broker dedicates to managing HTTP tunneling clients.

- Value type - integer
- Initial value -32

-n httpProtocolTimeout

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

-n 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

-n 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

-n interbrokerHost

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.

-n interbrokerPort

The port number on which the Broker listens for incoming inter-broker connections. If you want to run 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, you must stop and restart the broker, and redeploy the topology.

-n isysQopLevel

Applies to the system and allows brokers only to publish and subscribe.

- Value type - string
- Initial value - none

-n jvmMaxHeapSize

The size of the Java Virtual Machine (JVM) heap size used with the JVMManager for your Java user-defined nodes.

This value must be in the range 16 777 216 to 8 589 934 592.

- Value type - integer
- Initial value - 134 217 728

-n maxBrokerQueueSize

The maximum number of bytes that the broker can 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

-n maxClientQueueSize

The maximum number of bytes that the broker can 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.

-n maxConnections

The maximum number of concurrently-connected clients that 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

-n maxHopCount

The 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

-n maxMessageSize

The 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.

-n multicastMaximumIPv4Address

The highest IPv4 address that the broker can use for its multicast transmissions.

This address must be in the range 224.0.0.2 to 239.255.255.255.

- Value type - string
- Initial value - 239.255.255.255

-n multicastMaximumIPv6Address

The highest IPv6 address that the broker can use for its multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:0:1 to ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff.

- Value type - string
- Initial value - ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff

-n multicastMinimumIPv4Address
The lowest IPv4 address that the broker can use for its multicast transmissions.

This address must be in the range 224.0.0.2 to 239.255.255.255.
- Value type - string
- Initial value - 224.0.0.2

-n multicastMinimumIPv6Address
The lowest IPv6 address that the broker can use for its multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:0:1 to ff02:ffff:ffff:ffff:ffff:ffff:ffff:ffff.
- Value type - string
- Initial value - ff02:0:0:0:0:0:0:1

-n 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 to 1000.
- Value type - integer
- Initial value - 100

-n multicastDataPort
The UDP data port through which multicast packets are sent and received.
- Value type - integer
- Initial value - 34343

-n 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

-n multicastHeartbeatTimeoutSec
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

-n multicastLimitTransRate
Use this property in conjunction with the multicastTransRateLimitKbps property to control network congestion. Valid 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

-n multicastMaxKeyAge

The maximum age, in minutes, of a topic encryption key before it must be redefined.

- Value type - string
- Initial value - 360

-n multicastMaxMemoryAllowedKBytes

The 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.

-n 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

-n 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 must be in the range 1000 to 1000000.

- Value type - integer
- Initial value - 60000

-n multicastMulticastInterface

The 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. *IPv4 Broker Network Interface* and *IPv6 Broker Network Interface* use the same command; use a semicolon (;) to separate the IPv4 and IPv6 addresses.

- Value type - string
- Initial value - None

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.

-n multicastNACKAccumulationTimeMillis

The time, in milliseconds, that NACKs are aggregated in the broker, before recovered packets are sent. This value must be in the range 50 to 1000.

- Value type - integer
- Initial value - 300

-n multicastNACKCheckPeriodMillis

The time, in milliseconds, between periodic checks of reception status and sequence gap detection for NACK building. This value must be in the range 10 through 1000.

- Value type - integer
- Initial value - 500

-n 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. Valid 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

-n 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 33 KB and this value can be in the range 1 to 5000.

- Value type - integer
- Initial value - 500

-n multicastPacketSizeBytes
The size, in bytes, of multicast packets. This value must be in the range 500 to 32000.

- Value type - integer
- Initial value - 7000

-n multicastProtocolType
The protocol type. Valid values are:

- PTL
- PGM/IP
- PGM/UDP
- Value type - string
- Initial value - PTL

-n 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 to 10000.

- Value type - integer
- Initial value - 3000

-n multicastTransRateLimitKbps
Limits the overall transmission rate in Kb (kilobits) per second.

This property is effective only if the **multicastLimitTransRate** property is not disabled. Set the value of this property to be no higher than the maximum data transmission rate of the system or the network, and in the range 10 through 1000000

- Value type - integer
- Initial value - 9500

-n 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

-n pingTimeoutMultiple
The 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

-n statsInterval
The number of milliseconds between statistics publications. If set to 0, statistics publications are not generated. You do not have to restart the broker after changing this property; however, the broker might take up to a minute to start producing statistics after you have changed the value.

This value must be in the range 0 through 1000.

- Value type - integer
- Initial value - 0

This value refers to the publish/subscribe statistics interval only.

-n sysQopLevel
Applies to the system, and allows brokers only to publish.

- Value type - string
- Initial value - none

See the "mqsichangeproperties command" on page 437 for examples of how to change parameters for Real-time nodes. Other examples are provided for particular tasks:

Generating statistics reports

"Setting up a multicast broker" on page 270

securitycache component parameter values:

Select the objects and properties associated with the securitycache component that you want to change.

To change these properties, you must specify the broker name and **-b securitycache**. You must also set the **ObjectName** to SecurityCache.

-n cacheSweepInterval
The interval between each sweep of the security cache for removing invalidated entries. The time is specified in seconds.

- Value type - integer
- Initial value - 300
- Other valid values: any positive integer.

-n cacheTimeout
The timeout value for marking entries in the cache as not valid. The time is specified in seconds.

- Value type - integer
- Initial value - 60
- Other valid values: any positive integer.

See the “mqsichangeproperties command” on page 437 for examples of its use.

mqsichangetrace command

Use the mqsichangetrace command to set the tracing characteristics for a component.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS - as a console command

Purpose:

This command is valid for:

- User trace. Specify the **-u** option.
- 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.
- Trace nodes. Specify the **-n** option to switch Trace nodes on or off. You can significantly improve the performance of a flow by switching Trace nodes off.

You can initiate, modify, or end user tracing for a broker, or initiate, modify, or end service tracing for a broker, a Configuration Manager, or the User Name Server (identified by component name).

You can switch Trace nodes for a broker on and off, but you cannot use this command to initiate service tracing for 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 output for service and user trace generated by these commands is written to trace files in the `log` subdirectory. When you have completed the work you want to trace, use the mqsireadlog command to retrieve the log as an XML format file. Use either the mqsiformatlog command (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 that you are tracing. You must expect to see some impact on performance when trace is active.

If you want to trace the command processes 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.

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.

If you want to view the tracing options that are currently in effect, use the mqsireporttrace command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsichangetrace command - Windows, Linux, and UNIX systems"
- "mqsichangetrace command - z/OS" on page 496

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

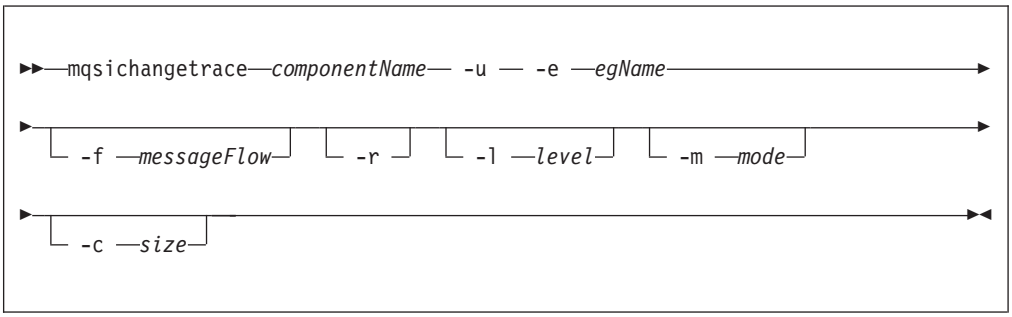
On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the group mqbrkrs.

mqsichangetrace command - Windows, Linux, and UNIX systems:

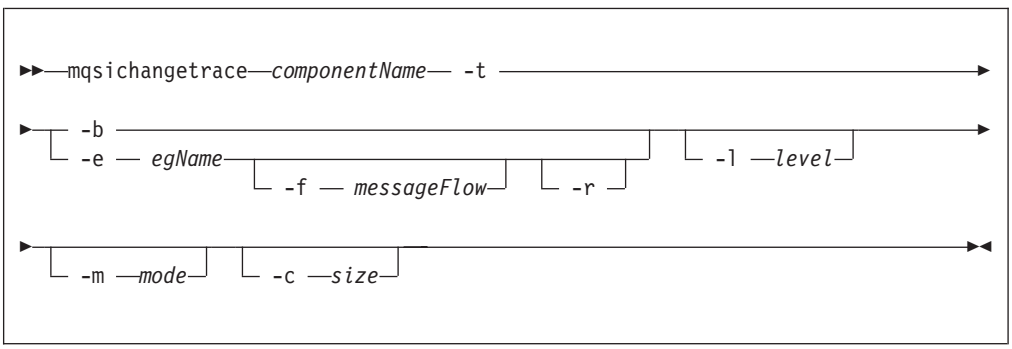
Use the mqsichangetrace command to set the tracing characteristics for a component.

Syntax:

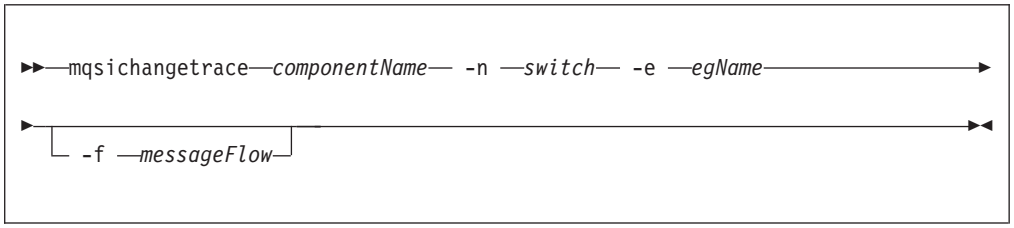
User trace:



Service trace:



Trace nodes:



Parameters:

componentName
(Required) Specify the name of the component that you want to trace; a broker, a Configuration Manager, or the fixed value, `UserNameServer`. All names are case sensitive on Linux and UNIX systems.

Key words `workbench` and `utility` are reserved, and must not be used as a component name.

-u (Required for user trace only if the component is a broker)

Specifies that user trace options are to be modified. This option is valid only if you have specified a broker name as the component name.

-e *egName*
(Required for user trace; optional for service trace)

Identifies the execution group for which trace options are to be modified (for example, started or stopped). This option is valid only for a broker.

-f *messageFlow*

(Optional) Identifies the message flow for which trace options are to be modified. This option is valid only if you have specified an execution group (flag **-e**).

-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 valid only if you have specified an execution group (flag **-e**).

-l *level*

(Optional) Set the level of the trace. The following options are supported:

- `normal`. This option provides a basic level of trace information.
- `none`. This option switches tracing off.
- `debug`. This option provides a more comprehensive trace.

This option is valid for all components. Each component is created with a default value of `none`. If you do not specify this parameter, the current value is unchanged. When you have successfully changed this value, it is persistent.

-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 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 parameter is valid only 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 second option is valid only for service trace and, when you have successfully changed 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, by using the mqsireadlog command;

- If you use this command with the **-f** option set, the log file is read directly from the file system. In this case, the maximum value that you can specify is 2097151, which allows a trace file up to 2 GB (gigabyte) to be created.
- If you use this command without setting the **-f** option, a WebSphere MQ message is sent to the broker to retrieve the log. In this case, do not allow the trace file to exceed 70 MB (megabytes). The maximum value that you can set is 70000.

On HP-UX, set the *size* value below 500 MB.

However you intend to retrieve the trace file, you might want to keep its size small, 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 parameter is valid only 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 second option is valid only for service trace.

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) Specifies that service trace options are to be modified.

-b

(Required) Specifies that service trace options for the `agent` subcomponent of the component specified are to be modified (for example, started or stopped). You can specify this flag only if **-t** is also specified.

Additional parameters exclusive to Trace nodes:

-n *switch*
(Required) Specifies the mode for trace flow. Valid values are `on` and `off`.

Examples:

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 -t -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
```

To switch off Trace nodes in the default execution group, use the command:

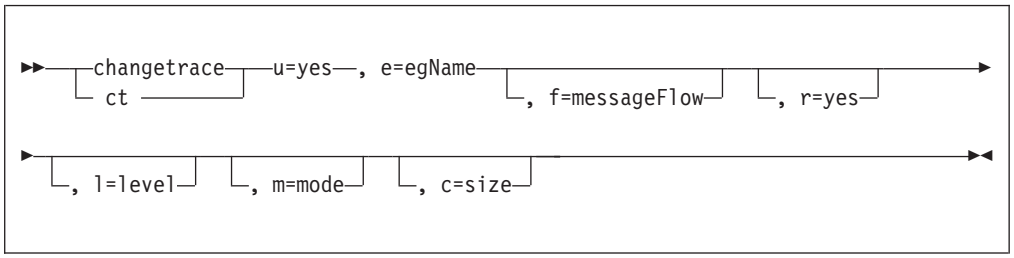
```
mqsichangetrace WBRK_BROKER –n off -e default
```

mqsichangetrace command - z/OS:

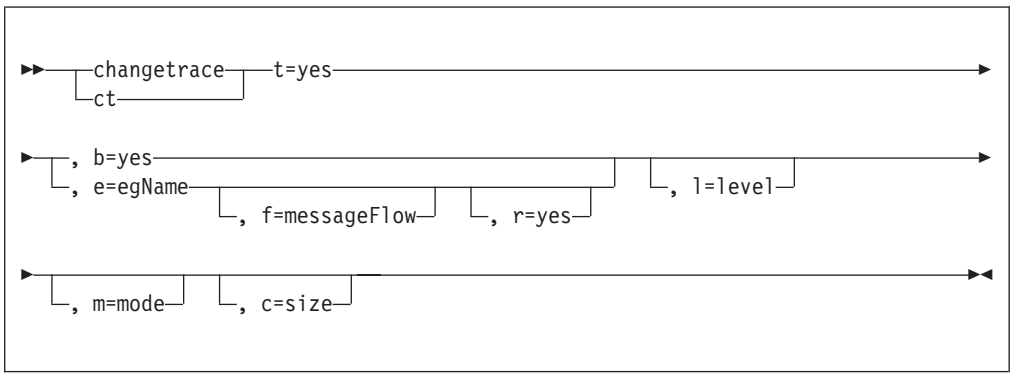
Use the mqsichangetrace command to set the tracing characteristics for a component.

Syntax:

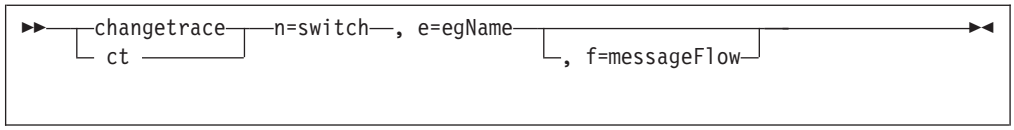
z/OS console command:
User trace



Service trace



Trace nodes:



Parameters:

-u (Required for user trace only if the component is a broker)

Specifies that user trace options are to be modified. This option is valid only if you have issued this command against a broker, that is, not a Configuration Manager or User Name Server.

-e *egName*

(Required for user trace; optional for service trace)

Identifies the execution group for which trace options are to be modified (for example, started or stopped). This option is valid only for a broker.

This name is case sensitive; you must include names in single quotes if they contains mixed case characters.

-f *messageFlow*

(Optional) Identifies the message flow for which trace options are to be modified. This option is valid only if you have specified an execution group (flag **-e**).

This name is case sensitive; you must include 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 valid only if you have specified an execution group (flag **-e**).

-l *level*

(Optional) Set the level of the trace. The following options are supported:

- `normal`. This option provides a basic level of trace information.
- `none`. This option switches tracing off.
- `debug`. This option provides a more comprehensive trace.

This option is valid for all components. Each component is created with a default value of `none`. If you do not specify this parameter, the current value is unchanged. When you have successfully changed this value, it is persistent.

-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 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 parameter is valid only 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 second option is valid only for service trace and, when you have successfully changed 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, by using the mqsireadlog command;

- If you use this command with the **-f** option set, the log file is read directly from the file system. In this case, the maximum value that you can specify is 2097151, which allows a trace file up to 2 GB (gigabyte) to be created.
- If you use this command without setting the **-f** option, a WebSphere MQ message is sent to the broker to retrieve the log. In this case, do not allow the trace file to exceed 70 MB (megabytes). The maximum value that you can set is 70000.

However you intend to retrieve the trace file, you might want to keep its size small, 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 parameter is valid only 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 second option is valid only for service trace.

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) Specifies that service trace options are to be modified.

-b

(Required) Specifies that service trace options for the `agent` subcomponent of the component specified are to be modified (for example, started or stopped). You can specify this flag only if **-t** is also specified.

Additional parameters exclusive to Trace nodes:

n=*switch*

(Required) Specifies the mode for trace flow. Valid values are `on` and `off`.

Examples:

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 BIPRELG, 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 BIPRELG, 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 BIPRELG, set the option for mqsireadlog to

```
T=, B=AGENT
```

To switch off Trace nodes for the default execution group, use the command:

```
F MQP1BRK,ct n='off', e='default'
```

mqsichangeusernameserver command

Use the mqsichangeusernameserver command to change one or more of the properties of the User Name Server.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPCHUN; see “Contents of the User Name Server PDSE” on page 681

Purpose:

On z/OS, you can change only the refresh interval and authentication properties of the User Name Server.

You must stop the User Name Server before you can issue this command, and restart the User Name Server for the changes to take effect:

- On Windows, Linux, and UNIX systems, use the mqsistop and mqsistart commands.
- On z/OS, you must have started the original User Name Server control process using the `/S` option. You must stop theUser Name Server components using the `/F User Name Server, PC` option and start the User Name Server components again using the `/S User Name Server, SC` option.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsichangeusernameserver command - Windows” on page 500
- “mqsichangeusernameserver command - Linux and UNIX systems” on page 501
- “mqsichangeusernameserver command - z/OS” on page 502

Authorization:

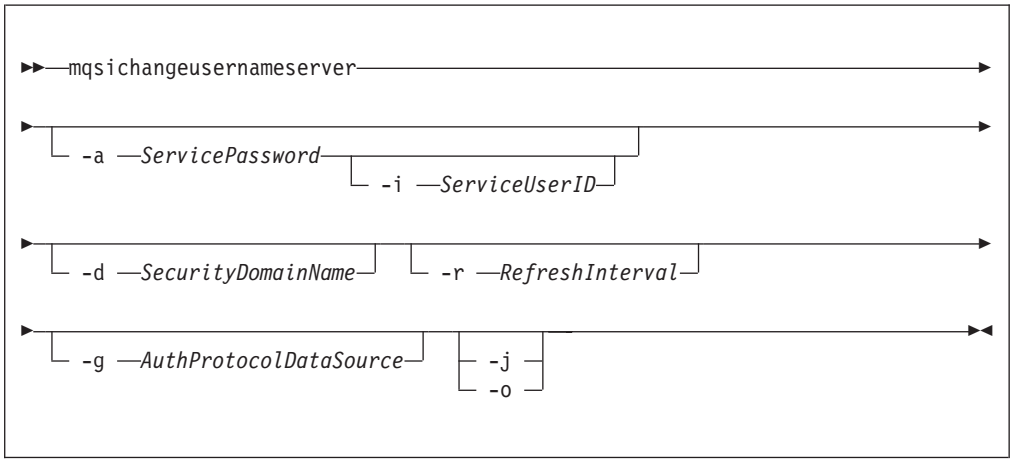
On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsichangeusernameserver command - Windows:



Parameters:

-i *ServiceUserID*
(Optional) The user ID under which the broker runs. You can only change this value if you also change the password.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 725.
`ServiceUserID` can be specified in any valid user name syntax. On Windows platforms, these are:

- `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 direct or indirect member of the local group **mqbrkrs**.
- A direct or indirect member of the local group **mqm**
- Authorized to access the home directory (where WebSphere Message Broker has been installed).

- Authorized to access the working directory (if specified by the **mqsicreateusernameserver -w** flag)

-a *ServicePassword*
(Optional) 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.

-d *SecurityDomainName*
(Optional) The name of the Windows system security domain.

-r *RefreshInterval*
(Optional) 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.

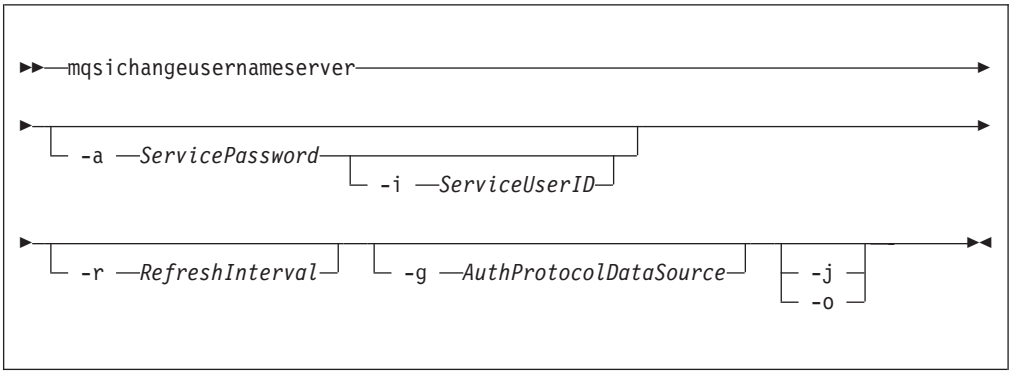
-o (Optional) Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.

Examples:

```
mqsichangeusernameserver -r 2000
```

mqsichangeusernameserver command - Linux and UNIX systems:

Syntax:



Parameters:

-i *ServiceUserID*
(Optional) The user ID under which the broker runs. You can only change this value if you also change the password.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Linux and UNIX platforms" on page 724. The `ServiceUserID` specified must be:

- Specified in the form `username`.
- A direct or indirect member of the local group **mqbrkrs**.
- A direct or indirect member of the local group **mqm**
- Authorized to access the home directory (where WebSphere Message Broker has been installed).
- Authorized to access the working directory (if specified by the **mqsicreateusernameserver -w** flag)

-a *ServicePassword*
(Optional) 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`.

-r *RefreshInterval*
(Optional) 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.

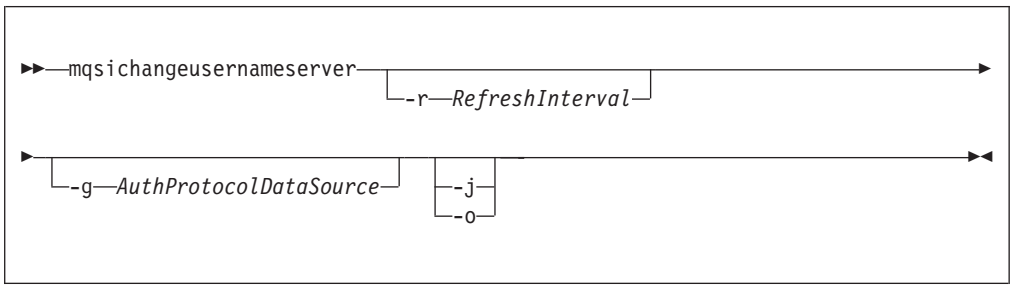
Examples:

```
mqsichangeusernameserver -r 2000
```

mqsichangeusernameserver command - z/OS:

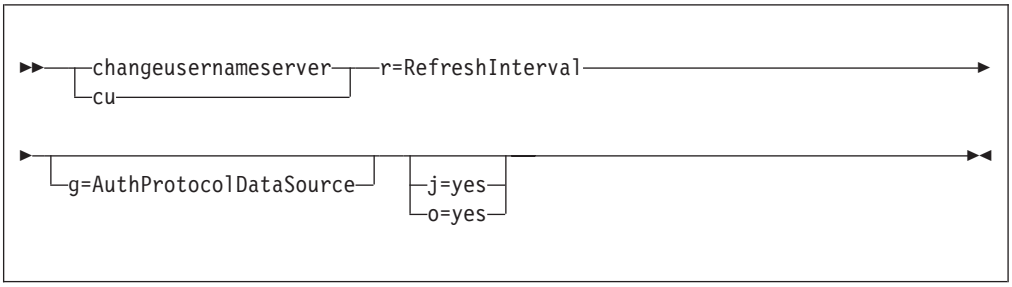
Syntax:

z/OS command - BIPCHUN:



z/OS console command:

Synonym: cu



Parameters:

-r *RefreshInterval*
(Required) 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.

Examples:

```
F MQP1UNS ,cu R=2000
```

mqsiclearmqpubsub command

Use the mqsiclearmqpubsub command to remove an WebSphere MQ Publish/Subscribe broker as a neighbor of this WebSphere Message Broker broker.

Supported platforms:

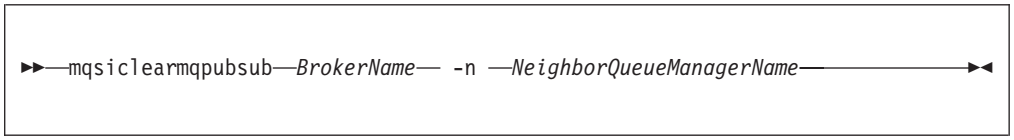
- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting the BIPCLMP utility; see “Contents of the broker PDSE” on page 679

Purpose:

The mqsiclearmqpubsub command removes knowledge of the WebSphere MQ Publish/Subscribe broker from the WebSphere Message Broker broker that you specify on this command. To complete this action, you must also run 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 Message Broker broker with an WebSphere MQ Publish/Subscribe broker network. Before you issue this command, you must ensure that the WebSphere Message 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:



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:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID must be a member of the mqbrkrs group.

On all platforms, the user ID used to run this command must have `put` and `inq` authority to the queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS.

Authorization:

Examples:

```
mqsiclearmqpubsub WBRK_BROKER -n MQBroker1
```

mqsicreateaclentry command

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.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPCRACL; see “Contents of the Configuration Manager PDSE” on page 682

Purpose:

If you create or modify an access control group, you must stop and restart the Configuration Manager for the change to take effect.

On z/OS, you must define an OMVS segment for user IDs and groups, in order for a Configuration Manager to obtain user ID and group information from the External Security Manager (ESM) database.

This command does not check for the existence of the specified component, therefore you can set up the access control list first.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsicreateaclentry command - Windows"
- "mqsicreateaclentry command - Linux and UNIX systems" on page 508
- "mqsicreateaclentry command - z/OS" on page 510

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

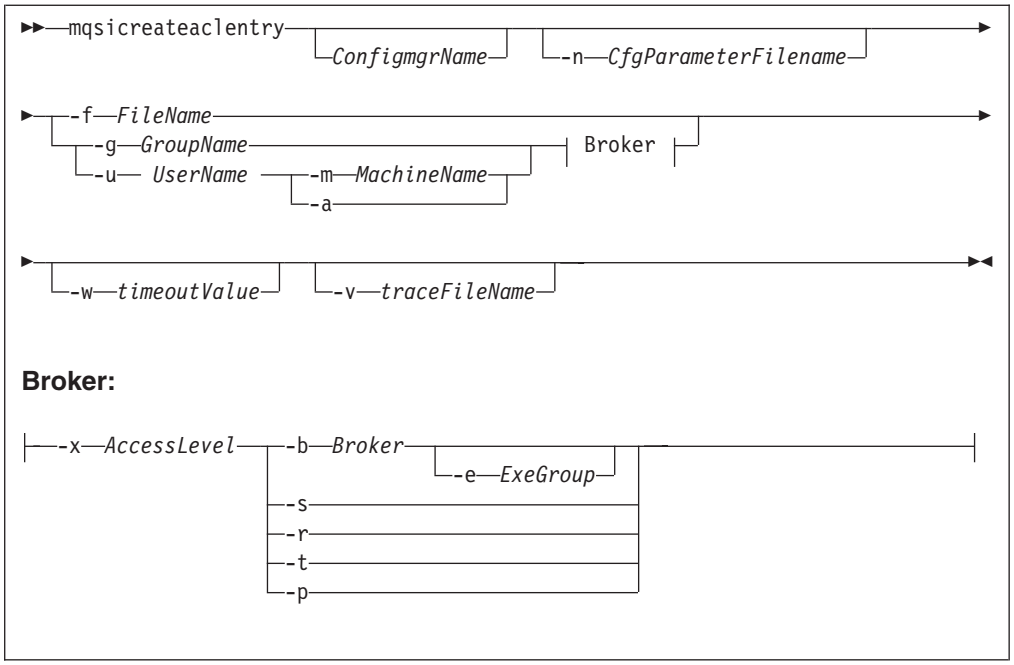
On Linux and UNIX, the user ID must be a member of mqbrkrs.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. Therefore the commands inherit a Full Control root ACL and you can carry out all operations. If you submit a console command to the Configuration Manager you can change all ACLs for that Configuration Manager.

On all platforms, the user ID used to run this command must have full control permissions for the object being changed.

mqsicreateaclentry command - Windows:

Syntax:



Parameters:

You must select:

- **-f**, or
- **-g** and **-x**, or **-u** and **-x**, and:
  - **-b**, or
  - **-s**, or
  - **-r**, or
  - **-t**, or
  - **-p**

ConfigmgrName
(Optional) The name of the Configuration Manager to which the access control lists are to be added.

The default name, if this parameter is not specified, is 'ConfigMgr'.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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.

If you select **-u**, you must select either **a** or **m**

-m *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.

If you select **-u**, you must select either **a** or **m**

-a (Optional) The specified user name can be on any machine. This option can not be used with **-m**.

If you select **-u**, you must select either **a** or **m**

-x *AccessLevel*
(Optional) The required access level given for this group. This option can be any one of the following letters:

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.

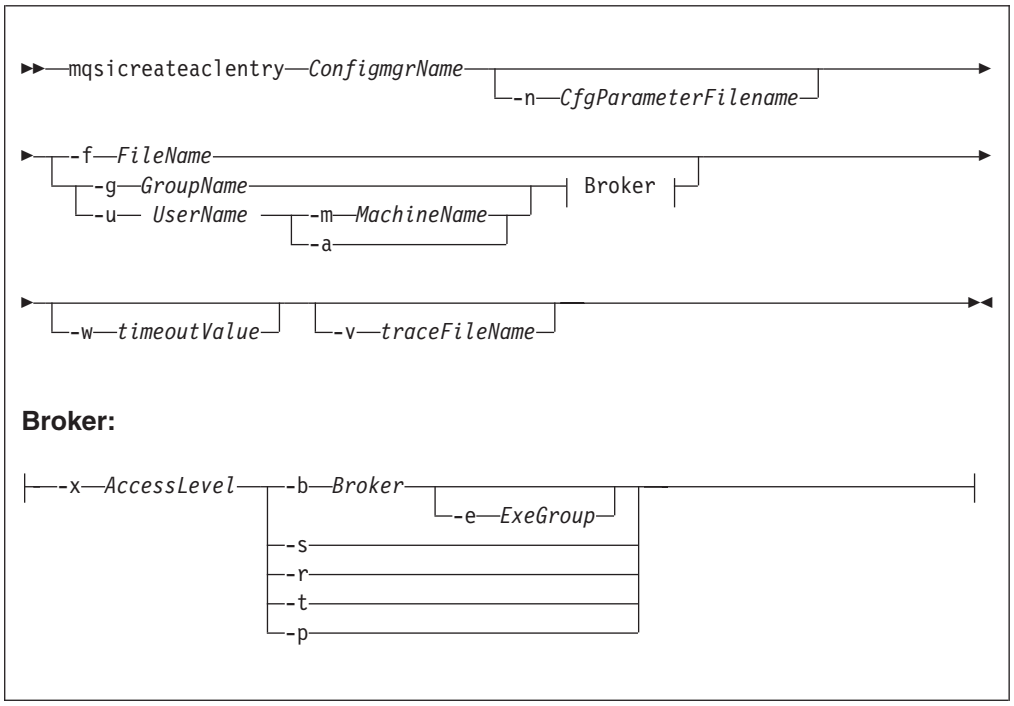
-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

```
mqsicreateaclentry CMGR01 -f c:\test\mylist
mqsicreateaclentry CMGR01 -g GROUPA -x F -b MYBROKER
```

mqsicreateaclentry command - Linux and UNIX systems:

Syntax:



Parameters:

You must select:

- **-f**, or
- **-g** and **-x**, or **-u** and **-x**, and:
  - **-b**, or
  - **-s**, or
  - **-r**, or
  - **-t**, or
  - **-p**

ConfigmgrName
(Required) The name of the Configuration Manager to which the access control lists are to be added. This parameter must be the first parameter specified and its name is case-sensitive.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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.

If you select **-u**, you must select either **a** or **m**

-m *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.

If you select **-u**, you must select either **a** or **m**

-a (Optional) The specified user name can be on any machine. This option can not be used with **-m**.

If you select **-u**, you must select either **a** or **m**

-x *AccessLevel*
(Optional) The required access level given for this group. This option can be any one of the following letters:

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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

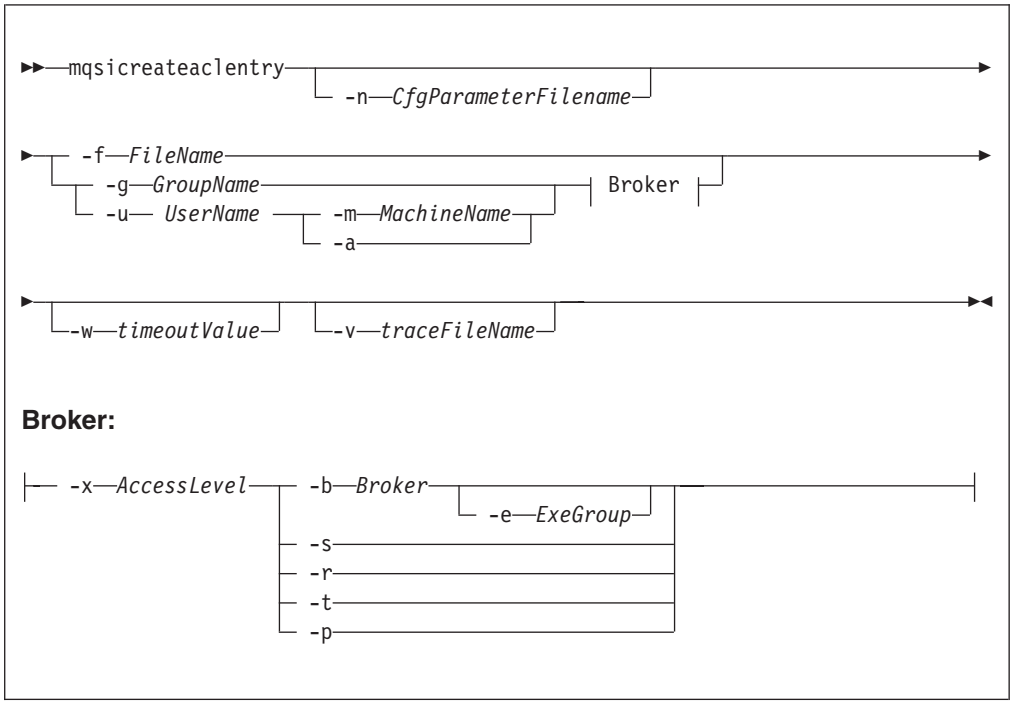
Examples:

```
mqsicreateaclentry CMGR01 -f c:\test\mylist
mqsicreateaclentry CMGR01 -g GROUPA -x F -b MYBROKER
```

mqsicreateaclentry command - z/OS:

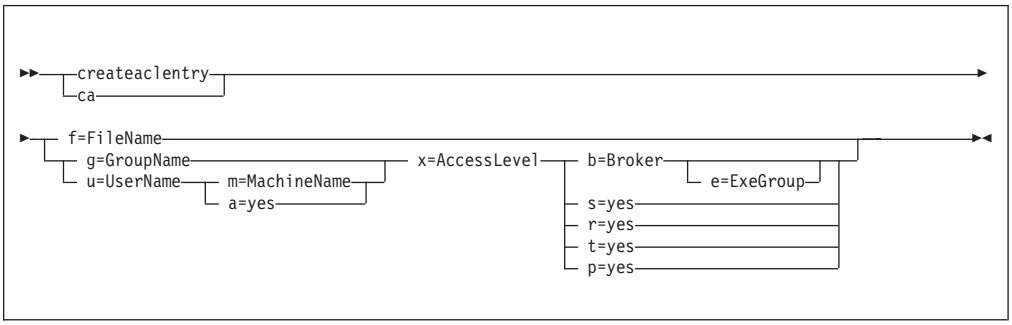
Syntax:

z/OS command - BIPCRACL:



z/OS console command:

Synonym: ca



Parameters:

You must select:

- **-f**, or
- **-g** and **-x**, or **-u** and **-x**, and:
  - **-b**, or
  - **-s**, or
  - **-r**, or
  - **-t**, or
  - **-p**

ConfigmgrName
This parameter is implicit because you specify the component that you want to MODIFY.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

Remove the statement `encoding="UTF-8"` from the first line of the `.configmgr` file, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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.

If you select **-u**, you must select either **a** or **m**

-m *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.

If you select **-u**, you must select either **a** or **m**

-a (Optional) The specified user name can be on any machine. This option can not be used with **-m**.

-x *AccessLevel*
(Optional) The required access level given for this group. This option can be any one of the following letters:

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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

On z/OS you must use a comma between each command option. The following example shows the z/OS version of the example listed in "mqsicreateaclentry command - Windows" on page 505:

```
/f CMGR01,ca g='GROUPA' ,x='F' ,b='MYBROKER'
```

mqsicreatebroker command

Use the mqsicreatebroker command to create a broker and its associated resources.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPCRBK; see "Contents of the broker PDSE" on page 679

Purpose:

The mqsicreatebroker command:

1. Checks whether the specified WebSphere MQ queue manager already exists:
   - If it does not exist:
     - If you run this command on z/OS, it reports an error and fails.
     - If you run this command on Linux, UNIX, or Windows, this command creates a queue manager.

       The queues that are created include a dead letter queue (DLQ), SYSTEM.DEAD.LETTER.QUEUE. The security settings are the same as those of other broker-specific WebSphere MQ queues.

       If a message received by a message flow cannot be processed, it is typically backed out onto the input queue. If it cannot be backed out, or the message flow is not configured to back out messages, or to perform alternative error processing, the broker puts the message to the DLQ.

       The mqsideletebroker command does not delete the default DLQ (unless the queue manager is deleted).
   - If it does exist, check that the queue manager has a DLQ defined; it is not created by this command on an existing queue manager, but is required because the broker puts messages that cannot be processed to the DLQ.

     If you use WebSphere MQ clusters in your domain, define the queue manager before you run this command, and configure the queue manager in the cluster to benefit from reduced administration and increased availability.
2. Starts the WebSphere MQ queue manager, if it not already running, except on z/OS.

   If the queue manager is created on Windows by this command, it is not started as a service. The queue manager stops if you log off. Therefore, 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.

3. Connects to the associated queue manager.
4. Creates the WebSphere MQ queues that are required by the broker, if they do not already exist.
5. Checks the version of WebSphere MQ that is installed. If WebSphere MQ Version 7.0 is installed, checks that the queue manager's publish/subscribe mode (attribute PSMODE) is not set to ENABLED. If it is, mqsicreatebroker sets the mode to COMPATIBILITY.
6. Creates database tables for the broker in the relevant schema in the specified database. If the schema, determined by user ID, or the tables, do not exist, they are created. If the tables already exist, the command adds rows specific to this broker to the existing tables.

   If the parameters that you specify on this command result in this broker sharing tables within a database schema with other brokers, you must ensure that all these brokers are at the same version of the product.
7. On Windows only, installs a service under which the broker runs.
8. Creates the broker in one of the available modes. If the full package is installed, the default mode is enterprise. If the trial package is installed, the default mode is trial. For more information, see Operation modes.
9. Creates a record for the component in the registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsicreatebroker command - Windows, Linux, and UNIX systems" on page 516
- "mqsicreatebroker command - z/OS" on page 522

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must be a member of both the mqbrkrs group and the mqm group.

On z/OS systems, the user ID used to run this command must be a member of a group that has both READ and WRITE access to the component directory. The user ID must also have access to WebSphere MQ resources, and DB2.

Using LDAP: Ensure that the registry is appropriately secured to prevent unauthorized access. You do not need to set the **LdapPrincipal** and

LdapCredentials parameters on mqsicreatebroker for correct operation of the broker. The password is not stored in clear text in the file system.

WebSphere MQ queues created:
- SYSTEM.BROKER.ADAPTER.FAILED
- SYSTEM.BROKER.ADAPTER.INPROGRESS
- SYSTEM.BROKER.ADAPTER.NEW
- SYSTEM.BROKER.ADAPTER.PROCESSED
- SYSTEM.BROKER.ADAPTER.UNKNOWN
- SYSTEM.BROKER.ADMIN.QUEUE
- SYSTEM.BROKER.AGGR.CONTROL
- SYSTEM.BROKER.AGGR.REPLY
- SYSTEM.BROKER.AGGR.REQUEST
- SYSTEM.BROKER.AGGR.TIMEOUT
- SYSTEM.BROKER.AGGR.UNKNOWN
- SYSTEM.BROKER.CONTROL.QUEUE
- SYSTEM.BROKER.EDA.COLLECTIONS
- SYSTEM.BROKER.EDA.EVENTS
- SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- SYSTEM.BROKER.INTERBROKER.MODEL.QUEUE
- SYSTEM.BROKER.INTERBROKER.QUEUE
- SYSTEM.BROKER.MODEL.QUEUE
- SYSTEM.BROKER.TIMEOUT.QUEUE
- SYSTEM.BROKER.WS.ACK
- SYSTEM.BROKER.WS.INPUT
- SYSTEM.BROKER.WS.REPLY

Access authority is granted for the WebSphere Message 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:

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 create the database using the DB2 Control Center, the bind is completed for you. If you use the command interface, the bind is not completed for you. For example, to create or re-create a bind for the database `MYDB` enter 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 Linux and UNIX platforms, binding is necessary for all databases. For example, to create binding for database `WBRKBKDB`, you must enter 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 do not use the default DB2 user ID and password (`db2admin`), you must replace the values in the `db2 connect` command with the correct values.

If you run the mqsicreatebroker command and it fails, resolve the problem that caused the failure:

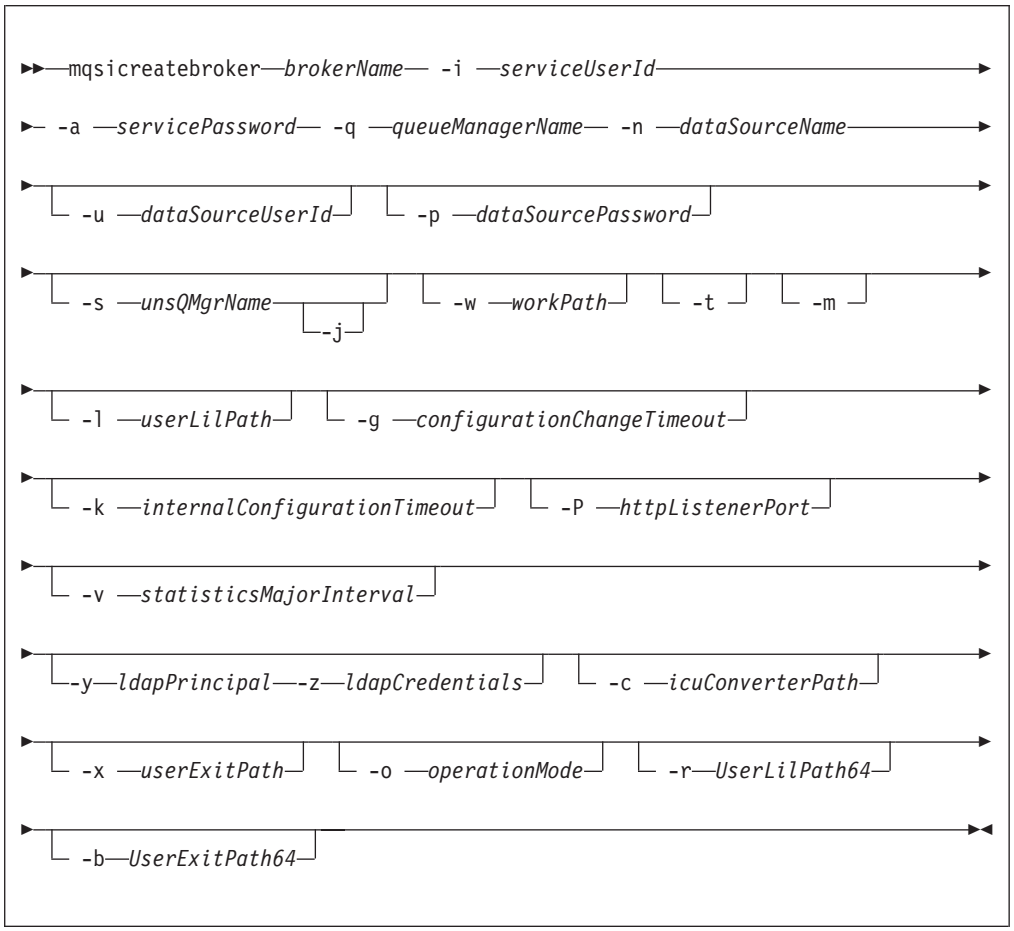
- Check responses; see "Responses" on page 515.
- Check the error logs; see Local error logs.
- Check the error messages in the error log; see Diagnostic messages.

When you run the same command again, you might receive a series of messages indicating items that cannot be created. Receiving these messages does not indicate a problem with the mqsicreatebroker command itself.

mqsicreatebroker command - Windows, Linux, and UNIX systems:

Use the mqsicreatebroker command to create a broker on a distributed system.

Syntax:



Parameters:

brokerName
(Required) The name of the broker that you are creating. This parameter must be the first parameter. It is case sensitive on Linux and UNIX systems.

For restrictions on the character set that you can use, see "Characters allowed in commands" on page 379.

-i *serviceUserId*
(Required) The user ID under which the broker runs.

You can specify the *serviceUserId* in any valid user name syntax. On Windows systems, valid formats are:

- `\\server\username`
- `.\username`
- `username`

On Linux and UNIX systems, only the last format, `username`, is valid.
Do not use a domain name as part of the *serviceUserId* parameter.
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* that you specify must be a member of the mqbrkrs local group. On Windows systems, 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 Message Broker has been installed), and the working directory (if specified by the **-w** parameter).
On Windows systems, if you specify that the broker is to run as a WebSphere MQ trusted application (**-t** parameter), you must also add the service user ID to the mqm group. On Linux and UNIX systems, specify the *serviceUserId* as `mqm` if you set the **-t** parameter.
The security requirements for the *serviceUserId* are described in "Security requirements for Windows platforms" on page 725 and in "Security requirements for Linux and UNIX platforms" on page 724.

-a *servicePassword*
(Required) The password for the *serviceUserId*.

For compatibility with existing systems, you can 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 compatibility with Windows systems, but is not used in relation to *serviceUserId*. **-a** is used as a default only if **-p** is not specified. See the **-p** parameter description for further details.

-q *queueManagerName*
(Required) The name of the queue manager that is 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 they are case sensitive.

Each broker *must* have its own unique queue manager. A broker cannot share a queue manager with another broker.

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, either create the queue manager before you issue this command, or change the settings of this queue manager to make it the default after it has been created. Use the WebSphere MQ Explorer to make this change.

The queue manager attribute MAXMSGLEN (the maximum length of messages that can be put to queues) is updated to 100 MB. This attribute is updated regardless of whether the queue manager is created by this command.

For restrictions on the character set that you can use, see "Characters allowed in commands" on page 379.

-n *dataSourceName*
(Required) The ODBC data source name (DSN) of the database in which the broker tables are created. If you have not used the same name for both the DSN and the database, this parameter must specify the DSN, not the name of the database.

This database must already exist. You must create a System DSN ODBC connection for this DSN, if you have not already done so.

Linux If you have a DB2 database on Linux, enter the appropriate DB database alias name; an ODBC DSN is not required.

-u *dataSourceUserId*
(Optional) The user ID with which databases that contain broker tables and user data are to be accessed. If you do not specify this ID, it defaults to the value that is specified by the **-i** parameter.

This user ID must have authority to create tables within this database, and read from and write to those tables.

Windows On Windows systems, if your broker database exists in DB2, and this user ID is not known to DB2, it is created for you within DB2.

Linux UNIX On Linux and UNIX systems, the service user must have been granted the correct privilege before entering this command. If your database is SQL Server, you must create this user ID as a SQL Server login ID and give it the correct access before you create the broker.

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 that run in this broker can access and manipulate the application data that is held within it, without having to specify explicit schema names. For further details see:

- "Security requirements for Windows platforms" on page 725
- "Security requirements for Linux and UNIX platforms" on page 724

-p *dataSourcePassword*
(Optional) The password of the user ID with which databases that contain broker tables and user data are to be accessed. If you do not specify this parameter, it defaults to the **servicePassword** that is specified by the **-a** parameter.

For compatibility with existing systems, you can 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.

Linux UNIX For DB2 on Linux and UNIX systems, you can specify **-u** and **-p** as empty strings (two quotation marks ""). In this case, you are using the **serviceUserId** and its **servicePassword** for the DB2 connection and that stores the password. If you specify **-a** as an empty string as well as **-u** and **-p**, WebSphere Message Broker stores no passwords; in this case the command prompts you to set a password and stores that.

-s *unsQMgrName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server.

Specify this parameter if you require either authentication services or publish/subscribe access control. If you do not specify this parameter, the broker assumes that no User Name Server is defined. To enable publish/subscribe access control, specify the **-s** and **-j** parameters.

-j (Optional) If you require publish/subscribe access control, specify this parameter. You must also specify the **-s** parameter.

-w *workPath*
(Optional) The directory in which working files for this broker are stored. If you do not specify this parameter, files are stored in the default work path, which was specified when the product was installed. If you specify this parameter, you must create this directory before you start the broker. On Windows systems, this directory cannot be on a networked drive.

This directory is also used for trace records that are created when tracing is active. These records are written to a subdirectory, `log`, which you must create before you start the broker.

Error logs that are written by the broker when a process ends abnormally are stored in this directory. On Windows systems, use this parameter 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 parameter using the mqsichangebroker command. To specify or change the work path, delete and recreate the broker.

Specifying this parameter creates a separate working directory for the broker. This working directory is a subset of the default working directory structure that contains fewer subdirectories and no `common\profiles` subdirectory.

-t (Optional) The broker runs as a WebSphere MQ trusted application.

If you specify this parameter on Windows systems, add the *serviceUserId* (identified by the **-i** parameter) to the mqm group.

If you specify this parameter on HP-UX and Solaris, specify the *serviceUserId* as `mqm`.

For more details about using WebSphere MQ trusted applications, see the *Intercommunication* section of the WebSphere MQ Version 6 information center online or WebSphere MQ Version 7 information center online.

-m (Optional) Migrate an existing WebSphere MQ Publish/Subscribe broker. If you specify this parameter, the queue manager that is identified by the **-q** parameter must be the queue manager that the WebSphere MQ Publish/Subscribe broker is using.

-l *userLilPath*
(Optional) A list of paths (directories) from which the broker loads 32-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Use the **-l** flag for 32–bit LIL files.

On Linux and UNIX systems, directory names are case sensitive, and you must include the names in single quotation marks if they contain mixed case characters.
Do not include environment variables in the path; the broker ignores them.

Create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Message Broker installation directory.
If you specify more than one directory, separate directories by a semicolon (`;`) on Windows systems, or a colon (`:`) on Linux and UNIX systems.

-g *configurationChangeTimeout*
(Optional) The maximum time (in seconds) that is allowed for a user configuration request to be processed. It defines the length of time taken within the broker to apply to an execution group a configuration change that you have initiated. For example, if you deploy a configuration from the workbench, the broker must respond to the Configuration Manager within this time.

A message flow cannot respond to a configuration change while it is processing an application message. An execution group that has been requested to change its configuration returns a negative response to the deployed configuration message if any one of its message flows does not finish processing an application message and apply the configuration change within this timeout.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-k *internalConfigurationTimeout*
(Optional) The maximum time (in seconds) that is allowed for an internal configuration change to be processed. For example, it defines the length of time taken within the broker to start an execution group.

The response time of each execution group differs according to system load and the load of its own processes. The value must reflect the longest response time that any execution group takes to respond. If the value is too low, the broker returns a negative response, and might issue error messages to the local error log.
Specify the value in seconds, in the range 10 to 3600. The default is 60.
For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-P *httpListenerPort*
(Optional) Enter the number of the port on which the Web services support is listening.

The broker starts this listener when a message flow that includes HTTP nodes or Web services support is started; the default is 7080.

Ensure that the port that you specify has not been specified for any other purpose.

-v *statisticsMajorInterval*
(Optional) Specify the interval (in minutes) at which statistics and accounting archive records are to be written. The valid range is from 10 to 14400 minutes; the default value is 60.

-y *ldapPrincipal*
(Optional, 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, but mandatory when *ldapPrincipal* is provided.) The user password for access to LDAP.

-c *icuConverterPath*
(Optional) A delimited set of directories to search for additional code page converters. On Windows systems, the delimiter is a semicolon (;). On UNIX and Linux systems, the delimiter is a colon (:).

Do not use this parameter to set the converter path if you are using a converter that matches one of the built-in converters that are provided, and that converter is the local code page for the broker. Use the **ICU_DATA** environment variable instead.

-x *userExitPath*
(Optional) The path that contains the location of all user exits to be loaded for 32-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

-o *operationMode*
(Optional) Use this parameter to set the mode of your broker; see Operation modes. Valid values that you can set are enterprise (the full package, enterprise mode), starter (Starter Edition mode), and adapter (Remote Adapter Deployment mode). The default is enterprise unless you have the Trial Edition, in which case the default is trial. If no **-o** parameter is set then the default is set automatically.

-r *UserLilPath64*
(Optional) A list of paths (directories) from which the broker loads 64-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Because you can define 32-bit or 64-bit LIL files on the following platforms:

- AIX
- HP-UX on PA-RISC
- Linux on x86-64
- Solaris on SPARC

use the following flags:

- **-r** to set the path where the 64-bit LIL files are stored.
- **-l** to set the path where the 32-bit LIL files are stored.

On Linux and UNIX systems, directory names are case sensitive, and you must include the names in single quotation marks if they contain mixed case characters.
Do not include environment variables in the path; the broker ignores them.

Create your own directory for storing your `.lil` or `.jar` files. Do not save them in the WebSphere Message Broker installation directory.
If you specify more than one directory, separate directories by a semicolon (`;`) on Windows systems, or a colon (`:`) on Linux and UNIX systems.

-b *UserExitPath64*
(Optional) The path that contains the location of all user exits to be loaded for 64-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

Because you can run 32-bit or 64-bit execution groups on the following platforms:

- AIX
- HP-UX on PA-RISC

- Linux on x86-64
- Solaris on SPARC

use the following flags:

- **-b** to set the path where the 64-bit user exits are stored.
- **-x** to set the path where the 32-bit user exits are stored.

Examples:

Create a broker that has topic-based security enabled:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
 -q WBRK_QM -s WBRK_UNS_QM -n WBRKBKDB
```

Create a broker to run as a trusted application:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
 -q WBRK_QM -n WBRKBKDB -t
```

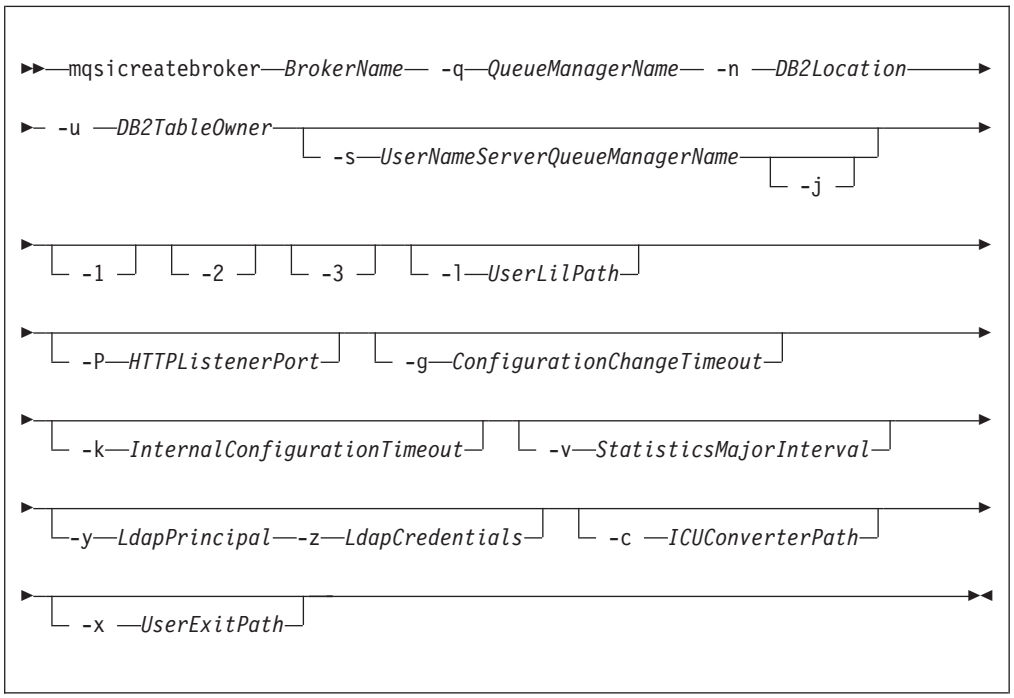
Create a broker that references user exits:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw
 -q WBRK_QM -n WBRKBKDB -x /opt/3rdparty/wmbexits
```

mqsicreatebroker command - z/OS:

Syntax:

z/OS command - BIPCRBK:



Parameters:

BrokerName
(Required) The name of the broker that you are creating. This parameter must be the first parameter, and if you create a broker with an uppercase name, the name must be specified in uppercase in the workbench.

For restrictions on the character set that you can use, see "Characters allowed in commands" on page 379.

-q *QueueManagerName*
(Required) The name of the queue manager that is 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 they are case sensitive.

Each broker *must* have its own unique queue manager. A broker cannot share a queue manager with another broker.

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, either create the queue manager before you issue this command, or change the settings of this queue manager to make it the default after it has been created. Use the WebSphere MQ Explorer to make this change.

The queue manager attribute `MAXMSGLEN` (the maximum length of messages that can be put to queues) is updated to 100 MB. This attribute is updated regardless of whether the queue manager is created by this command.

For restrictions on the character set that you can use, see "Characters allowed in commands" on page 379.

-n *DB2Location*
(Required) The location of the database in which the broker tables are created.

-u *DB2TableOwner*
(Required) The user ID with which databases that contain broker tables and user data are to be accessed.

This user ID must have authority to create tables within this database, and read from and write to those tables.

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 that run in this broker can access and manipulate the application data that is held within it, without having to specify explicit schema names.

-s *UserNameServerQueueManagerName*
(Optional) The name of the WebSphere MQ queue manager that is associated with the User Name Server.

Specify this parameter if you require either authentication services or publish/subscribe access control. If you do not specify this parameter, the broker assumes that no User Name Server is defined. To enable publish/subscribe access control, specify the **-s** and **-j** parameters.

-j (Optional) If you require publish/subscribe access control, specify this parameter. You must also specify the **-s** parameter.

-l *UserLilPath*
(Optional) A list of paths (directories) from which the broker loads 32-bit loadable implementation libraries (LIL files) for user-defined message processing nodes. Use the **-l** flag for 32–bit LIL files.

This name is case sensitive; enclose the names in single quotation marks if they are in mixed case.
Do not include environment variables in this path; WebSphere Message Broker ignores them.

You must create your own directory for storing your `.lil` or `.jar` files. Do not save these files in the WebSphere Message Broker install directory.

-P *HTTPListenerPort*
(Optional) Enter the number of the port on which the Web services support is listening.

The broker starts this listener when a message flow that includes HTTP nodes or Web services support is started; the default is 7080.

Ensure that the port that you specify has not been specified for any other purpose.

-g *ConfigurationChangeTimeout*
(Optional) The maximum time (in seconds) that is allowed for a user configuration request to be processed. It defines the length of time taken within the broker to apply to an execution group a configuration change that you have initiated. For example, if you deploy a configuration from the workbench, the broker must respond to the Configuration Manager within this time.

A message flow cannot respond to a configuration change while it is processing an application message. An execution group that has been requested to change its configuration returns a negative response to the deployed configuration message if any one of its message flows does not finish processing an application message and apply the configuration change within this timeout.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-k *InternalConfigurationTimeout*
(Optional) The maximum time (in seconds) that is allowed for an internal configuration change to be processed. For example, it defines the length of time taken within the broker to start an execution group.

The response time of each execution group differs according to system load and the load of its own processes. The value must reflect the longest response time that any execution group takes to respond. If the value is too low, the broker returns a negative response, and might issue error messages to the local error log.
Specify the value in seconds, in the range 10 to 3600. The default is 60.
For information about how to set the value for this timeout, see "Setting configuration timeouts" on page 227.

-v *StatisticsMajorInterval*
(Optional) Specify the interval (in minutes) at which statistics and accounting archive records are to be written. The valid range is from 10 to 14400 minutes; the default value is 60.

An interval of zero minutes indicates that the operating system has an external method of notification (the ENF timer) and is not using an internal timer within WebSphere Message Broker.

-1 (Optional) The registry pass, which creates only the broker registry.

-2 (Optional) The WebSphere MQ pass, which creates only the broker WebSphere MQ queues.

-3 (Optional) The DB2 pass, which creates only the broker DB2 tables and indexes.

-y *LdapPrincipal*
(Optional, 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, but mandatory when *ldapPrincipal* is provided.) The user password for access to LDAP.

-c *ICUConverterPath*
(Optional) A delimited set of directories to search for additional code page converters.

The code page converters must be either of the form `icudt32_codepagename.cnv`, or in an ICU data package called `icudt32.dat`.

Do not use this parameter to set the converter path if you are using a converter that matches one of the built-in converters that are provided with Version 6.0, and that converter is the local code page for the broker. Use the **ICU_DATA** environment variable instead.

-x *UserExitPath*
(Optional) The path that contains the location of all user exits to be loaded for 32-bit execution groups in this broker. This path is added to the system library search path (`PATH,LIBPATH,LD_LIBRARY_PATH,SHLIBPATH`) for the execution group process only.

Examples:

To create a broker on z/OS by using a single command:

```
mqsicreatebroker CSQ1BRK -q CSQ1 -u BRKUSER -n DBA0
```

To create only the DB2 tables and indexes on z/OS:

```
mqsicreatebroker CSQ1BRK -q CSQ1 -u BRKUSER -n DBA0 -2
```

mqsicreateconfigmgr command

Use the mqsicreateconfigmgr command to create a Configuration Manager and its associated resources.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPCRCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

The mqsicreateconfigmgr command:

1. Checks that the specified WebSphere MQ queue manager already exists.
   If it does not exist:
   - If you run this command on z/OS, it reports an error and fails.
   - If you run this command on Linux, UNIX, or Windows, this command creates a queue manager.
     If the queue manager is created on Windows by this command, it is not started as a service. The queue manager stops if you log off. Therefore 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.

If you use WebSphere MQ clusters in your domain, define the queue manager before you run this command, and configure the queue manager in the cluster to benefit from reduced administration and increased availability.

2. Starts the WebSphere MQ queue manager if it is not already running, except on z/OS.
3. Connects to the associated queue manager.
4. Creates the WebSphere MQ queues and channel that are specific to the Configuration Manager, if they do not already exist.
5. 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.
6. If you run this command using the -n parameter, and then delete the Configuration Manager by using mqsideleteconfigmgr, without specifying the -n parameter on that command, the new database containing the configuration repository is not deleted.

   If you run the mqsicreateconfigmgr command again in this situation, and specify the -n parameter, the parameter is ignored because the new database still exists.
7. On Windows only, installs a service under which the Configuration Manager runs.
8. Creates a record for the component in the registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsicreateconfigmgr command - Windows" on page 527
- "mqsicreateconfigmgr command - Linux and UNIX systems" on page 529
- "mqsicreateconfigmgr command - z/OS" on page 530

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

On z/OS systems, the user ID used to run this command must be a member of a group that has READ and WRITE access to the component directory and access to WebSphere MQ.

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 Message Broker group mqbrkrs to all these queues. If the DLQ is enabled, it also has the same authority.

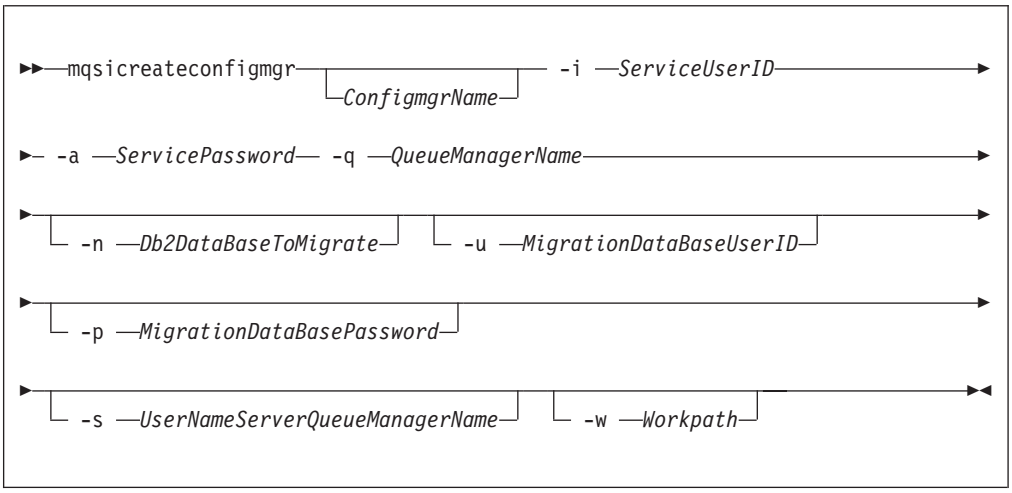
WebSphere MQ channels created:

- SYSTEM.BKR.CONFIG

Database tables created:
The database tables that this command creates are in a repository created by the Configuration Manager.

mqsicreateconfigmgr command - Windows:

Syntax:



Parameters:

ConfigmgrName
(Optional) The name of the Configuration Manager that you want to create.

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 user name 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 (either direct or indirect) of the local group **mqbrkrs**, and must be authorized to access the home directory (where WebSphere Message 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.

The security requirements for the ServiceUserID are detailed in "Security requirements for Windows platforms" on page 725.

-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 update 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 parameter 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 parameter 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*

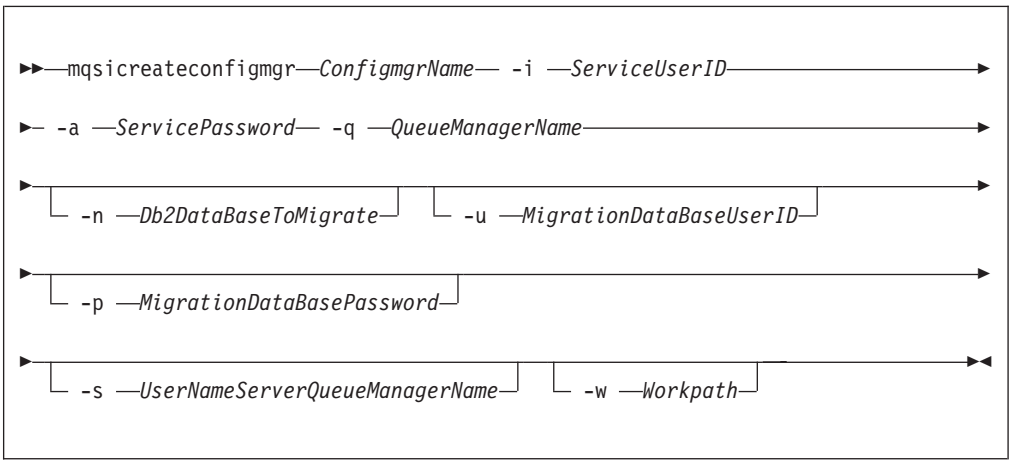
(Optional) The directory in which working files for the Configuration Manager are stored. If not specified, the default directory specified when the product was installed is used.

Examples:

```
mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_CONFIG_QM
```

mqsicreateconfigmgr command - Linux and UNIX systems:

Syntax:



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager that you want to create.

This must be the first parameter specified and the name is case-sensitive.

-i *ServiceUserID*
(Required) The user ID under which the service runs.

(Optional) This can be specified in any valid user name 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 Message 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**.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Linux and UNIX platforms" on page 724.

-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 update 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 parameter 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 parameter 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.

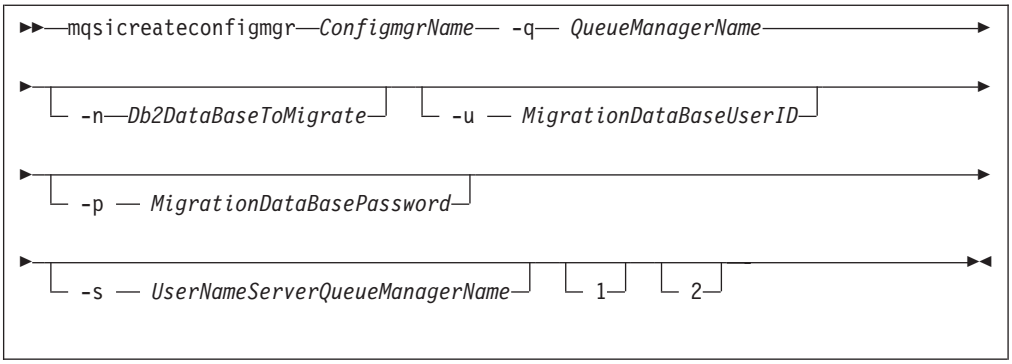
Examples:

```
mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_CONFIG_QM
```

mqsicreateconfigmgr command - z/OS:

Syntax:

z/OS command - BIPCRCM:



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager that you want to create.

This must be the first parameter specified and the name is case-sensitive.

-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 update 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.

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 parameter 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) The registry pass that creates only the Configuration Manager registry.

2 (Optional) The WebSphere MQ pass that creates only the Configuration Manager WebSphere MQ queues.

Note: This action can be performed only if the Configuration Manager registry exists.

Examples:

```
mqsicreateconfigmgr CMGR01   -q WBRK_CONFIG_QM -1
```

mqsicreateconfigurableservice command

Use the mqsicreateconfigurableservice command to create a new object name for a broker external resource, such as a JMS provider, JDBC provider, TCPIPClient, TCPIPServer, FTP server, monitoring profile, SAP, Siebel or Peoplesoft adapter, or IMS instance.

Supported platforms:

- Windows.
- Linux and UNIX systems.
- z/OS. Run this command by customizing and submitting BIPCRCS; see "Contents of the broker PDSE" on page 679.

Purpose:

For configurable services that you add by using the mqsicreateconfigurableservice command:

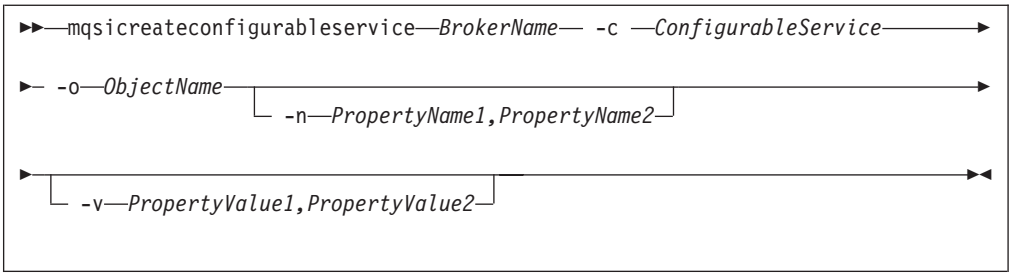
- Use the mqsireportproperties command to view the configurable services.
- Use the mqsichangeproperties command to modify the configurable services.
- Use the mqsideleteconfigurableservice command to delete configurable services.

You do not need to use the mqsicreateconfigurableservice command to create EIS providers, because they are already defined. Use the mqsichangeproperties command to modify EIS providers.

Usage notes:

- Before you run this command, ensure that the broker is running.
- Stop and restart the broker before you use any new broker resources and properties.

Syntax:



Parameters:

BrokerName
(Required) The name of the broker to modify. This parameter must be the first parameter.

-c *ConfigurableService*
(Required) The type of external resource (configurable service). Use the mqsireportproperties command to view the list of valid values.

The valid resource types are listed in "Configurable services properties" on page 444.

-o *ObjectName*
(Required) The name of the object whose properties you want to change.

The valid object names are listed in "Configurable services properties" on page 444.

For example, if the **-c** parameter is set to JDBCProviders, the expected object name is either an IBM defined JDBC provider name, or a user-defined JDBC provider name. Default services are provided for the supported databases to which you can connect over JDBC type 4 connections. Use the supplied services as a template when you create a new service using this command. Use the mqsireportproperties command to view the list of default provider names.

If the **-c** parameter is set to SecurityProfiles, the expected object name is the name of a new security profile for the broker.

If the **-c** parameter is set to FtpServer, the object name is the name of an FTP server specified as a property on the **FTP** tab of the FileInput or FileOutput node. It is a user-defined name that identifies the remote FTP server and the series of properties that are defined here.

If the **-c** parameter is set to IMSConnect, the expected object name is the name of an IMS Connect service.

If the **-c** parameter is set to MonitoringProfiles, the expected object name is the monitoring profile name.

If the **-c** parameter is set to PeopleSoftConnection, the expected object name is the name of a PeopleSoft service.

If the **-c** parameter is set to SAPConnection, the expected object name is the name of an SAP service.

If the **-c** parameter is set to SiebelConnection, the expected object name is the name of a Siebel service.

If the **-c** parameter is set to TCPIPClient, the expected object name is the name of a TCPIPClient configurable service.

If the **-c** parameter is set to TCPIPServer, the expected object name is the name of a TCPIPServer configurable service.

-n *PropertyName*
(Optional) The name of the property that is being changed.

The valid property names are listed in "Configurable services properties" on page 444.

-v *PropertyValue*
(Optional, but required if the **-n** parameter is specified) The value that is assigned to the property that is specified by the **-n** parameter. You can specify more than one property name and corresponding value using commas as separators; for example, `-n Name1,Name2 -v Value1,Value2`.

UNIX On UNIX, if the **-v** parameter contains a semi-colon (;), enclose the entire string in quotation marks, as shown in the following example:

```
mqsicreateconfigurableservice WBRK_BROKER -c JDBCProviders -o DB2EXTRA -n connectionUrlFormat
-v "jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];"
```

The property values are described in "Configurable services properties" on page 444.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all operating systems, the user ID must be a member of the mqbrkrs group.

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
- BIP8073 Invalid broker name
- BIP8983 Configurable service already exists
- BIP8984 Configurable service was not found

Examples:

Create an FtpServer configurable service:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c FtpServer -o Server01
-n serverName,scanDelay,transferMode,connectionType,securityIdentity
-v one.hursley.abc.com:123,20,ACTIVE,secId
```

Create an FtpServer configurable service to use SFTP without strict host key checking:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c FtpServer -o TEST1
-n protocol,serverName,scanDelay,remoteDirectory,securityIdentity,cipher,compression,
strictHostKeyChecking -v  SFTP,winlnx58,30,.,chbatey,blowfish-cbc,9,no
```

Create an IMSConnect configurable service for the IMS instance IMSA that is running on `test.ims.ibm.com` port 9999:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c IMSConnect -o myIMSConnectService
-n Hostname,PortNumber,DataStoreName -v test.ims.ibm.com,9999,IMSA
```

Add a JMS provider called MyProviderXYZ for broker WBRK61_DEFAULT_BROKER:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c JMSProviders -o JMS_MyProviderXYZ
```

Add a JMS provider called ProviderABC for broker WBRK61_DEFAULT_BROKER with default values for the resource properties:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_ProviderABC
```

Add a JMS provider called ProviderABC for broker WBRK61_DEFAULT_BROKER specifying a location for the provider's JAR files, and a library path for the local libraries that are associated with those JAR files:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_ProviderABC
-n jarsURL,nativeLibs -v file://D:\ProviderABC\java,D:\ProviderABC\libs
```

Add a JMS provider called BEAV91 for broker WBRK61_DEFAULT_BROKER, specifying the name of an IBM supplied Java class called `com.ibm.broker.apihandler.BEAWebLogicAPIHandler` to handle vendor specific API calls:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c JMSProviders -o BEAV91
-n proprietaryAPIHandler,proprietaryAPIAttr1,proprietaryAPIAttr2,proprietaryAPIAttr3
–v com.ibm.broker.apihandler.BEAWebLogicAPIHandler,weblogic.jndi.WLInitialContextFactory,
t3://19.21.194.126:7001,BEAServerName
```

Create a monitoring profile with the name 'mp1' to broker WBRK6_DEFAULT_BROKER:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c MonitoringProfiles -o mp1
```

Create a PeopleSoftConnection configurable service for the PeopleSoft instance that is running on *my.peoplesoft.qa.com*:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c PeopleSoftConnection
-o myPeopleSoftAdapter.outadapter -n hostName,port -v "my.peoplesoft.qa.com",9000
```

Create an SAPConnection configurable service for the SAP adapter *mySAPAdapter.outadapter* that connects to the SAP host `test.sap.ibm.com`, and uses client *001* for the connections into that server:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c SAPConnection -o mySAPAdapter.outadapter
-n applicationServerHost,client -v test.sap.ibm.com,001
```

Create a security profile for TFIM use:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c SecurityProfiles -o TFIM
-n authentication,mapping,authorization,propagation,mappingConfig
-v TFIM,TFIM,TFIM,TRUE,http://tfimhost1.ibm.com:9080
```

Create a SiebelConnection configurable service for the Siebel adapter *mySiebelAdapter.outadapter* that connects to the Siebel server *"siebel://my.siebel.qa.com/SBA_80/SSEObjMgr_enu"*.

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c SiebelConnection -o
mySiebelAdapter.outadapter -n connectString -v "siebel://my.siebel.qa.com/SBA_80/SSEObjMgr_enu"
```

Create a TCPIPServer configurable service:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c TCPIPServer -o ServerPort1452
-n Port,MaximumConnections,ExpireConnectionSec -v 1452,1000,15
```

Create a TCPIPClient configurable service:

```
mqsicreateconfigurableservice WBRK61_DEFAULT_BROKER -c TCPIPClient
-o ClientPort1452HostnameJsmith
-n Port,Hostname,MinimumConnections,MaximumConnections
-v 1452,jsmith.hursley.ibm.com,5,10
```

mqsicreatedb command

Use the mqsicreatedb command to create a database for a broker. On Windows XP and Windows Server 2003, you can create either a DB2 or a Derby database. On Windows Vista or Windows Server 2008, you can create only a DB2 database.

Supported platforms:

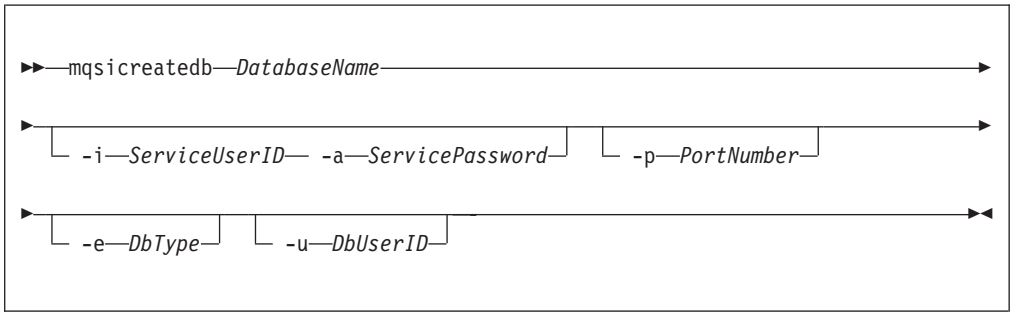
- Windows

Purpose:

The mqsicreatedb command creates a database and makes it accessible to the broker. The command creates the database, creates an ODBC data source name, and if necessary, depending on the database type, creates and starts a Windows service. The command creates at most a single instance of the Windows service for each installation of a major product version.

This command supports only the installed DB2 and Derby database engines.

Syntax:



Parameters:

DatabaseName
(Required) The name of the database you want to create, which is case sensitive. You must specify this parameter first. 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 379.

-i *ServiceUserID*
(Optional, Derby only) The user ID under which the DatabaseInstanceMgr service runs.

Specify this parameter in any valid user name 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 you have installed WebSphere Message Broker).

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

-a *ServicePassword*
(Optional, Derby only) The password for the ServiceUserID. Specify this parameter 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. 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.

On Windows Vista and Windows Server 2008, only DB2 is supported.

-u *DbUserID*
(Optional, DB2 only) An additional user name that requires access to the database that is created by this command.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID must be a member of the mqbrkrs group.

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

Use the mqsicreateexecutiongroup command to add a new execution group to a broker.

Supported platforms:

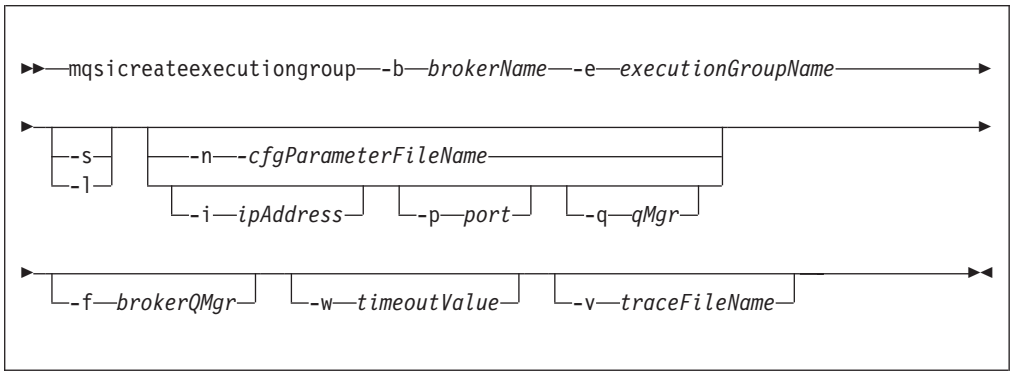
- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPCREG; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

You must start the Configuration Manager before you run the mqsicreateexecutiongroup command.

You can use the mqsicreateexecutiongroup command to add a new broker to the Configuration Manager.

Syntax:



Parameters:

-b *brokerName*
(Required) The name of the broker on which to create the execution group.

-e *executionGroupName*
(Required) The name of the new execution group.

-s (Optional) Create a 32 bit execution group.

-l (Optional) Create a 64 bit execution group.

-n *cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

If you are using this file on z/OS, you must remove the statement `encoding="UTF-8"` from the first line, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value 1414 is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used.

-f *brokerQMgr*
Forces a broker with the supplied queue manager name to be defined in the Configuration Manager, if it does not already exist. If the broker is already defined in the Configuration Manager, this flag is ignored.

-w *timeoutValue*
(Optional) This parameter is the time in seconds that the utility waits to ensure that the command completed; the default value is 60.

-v *traceFileName*
(Optional) This parameter sends internal debug trace information to the specified file.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the group mqm.

The command succeeds only if this user ID has the correct authority defined in the access control list for the Configuration Manager. To create an execution group, full control authority is required over the broker object; see "ACL permissions" on page 723 for a list of permissions that can be defined in the Configuration Manager.

Responses:

The mqsicreateexecutiongroup 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 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 the Configuration Manager with an associated queue manager called `QMGR`, which 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

Use the mqsicreateusernameserver command to create a User Name Server and its associated resources.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPCRUN; see "Contents of the User Name Server PDSE" on page 681

Purpose:

The mqsicreateusernameserver command:

1. Checks whether the specified WebSphere MQ queue manager already exists. If it does not exist:
   - If you run this command on z/OS, it reports an error and fails.
   - If you run this command on Linux, UNIX, or Windows, this command creates a queue manager.

     If the queue manager is created on Windows by this command, it is not started as a service. The queue manager stops if you log off. Therefore 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.

   If you use WebSphere MQ clusters in your domain, define the queue manager before you run this command, and configure the queue manager in the cluster to benefit from reduced administration and increased availability.
2. Starts the WebSphere MQ queue manager if it is not already running, except on z/OS.
3. Connects to the associated queue manager.
4. Creates the WebSphere MQ queues that are required by the User Name Server, if they do not already exist.
5. On Windows only, installs a service under which the User Name Server runs.
6. Creates a record for the component in the registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsicreateusernameserver command - Windows"
- "mqsicreateusernameserver command - Linux and UNIX systems" on page 543
- "mqsicreateusernameserver command - z/OS" on page 545

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

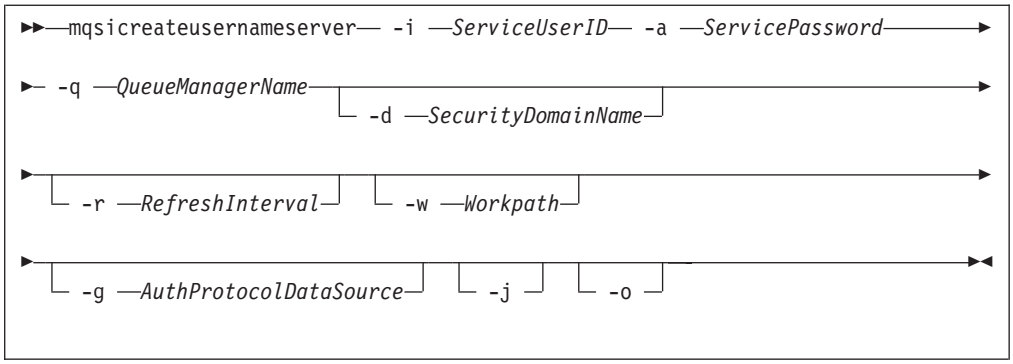
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsicreateusernameserver command - Windows:

Syntax:



Parameters:

-i *ServiceUserID*
(Required) The user ID under which the broker runs.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Windows platforms" on page 725.
`ServiceUserID` can be specified in any valid user name syntax. On Windows platforms, these are:

- `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 direct or indirect member of the local group **mqbrkrs**.
- A direct or indirect member of the local group **mqm**
- Authorized to access the home directory (where WebSphere Message Broker has been installed).

-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 User Name Server.

If the queue manager does not already exist, it is created by this command, however, it is not created as the default queue manager. If you want the queue manager to be the default queue manager on this system, you must create it 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) 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 Message Broker, see "Setting up broker domain security" on page 45.

-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) 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) 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.

(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.

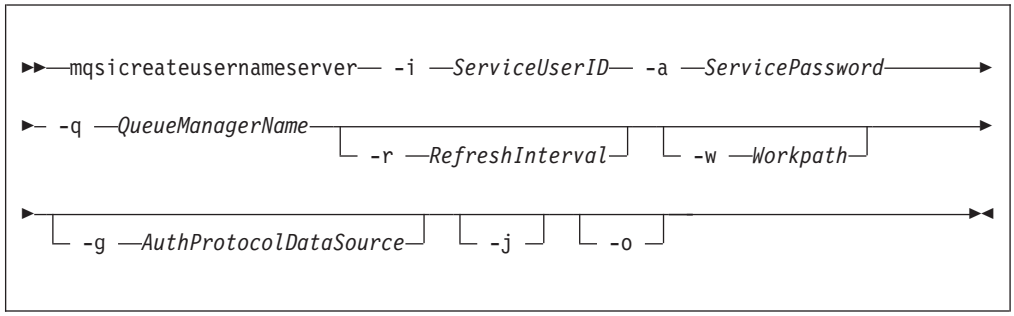
-o (Optional) Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.

Examples:

```
mqsicreateusernameserver -i wbrkuid -a wbrkpw
   -q WBRK_QM -r 1000
```

mqsicreateusernameserver command - Linux and UNIX systems:

Syntax:



Parameters:

-i *ServiceUserID*
(Required) The user ID under which the broker runs.

The security requirements for the `ServiceUserID` are detailed in "Security requirements for Linux and UNIX platforms" on page 724. The `ServiceUserID` specified must be:

- Specified in the form `username`.
- A direct or indirect member of the local group **mqbrkrs**.
- A direct or indirect member of the local group **mqm**
- Authorized to access the home directory (where WebSphere Message Broker has been installed).

-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 User Name Server.

If the queue manager does not already exist, it is created by this command, however, it is not created as the default queue manager. If you want the queue manager to be the default queue manager on this system, you must create it 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.

-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) 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) 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.

(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.

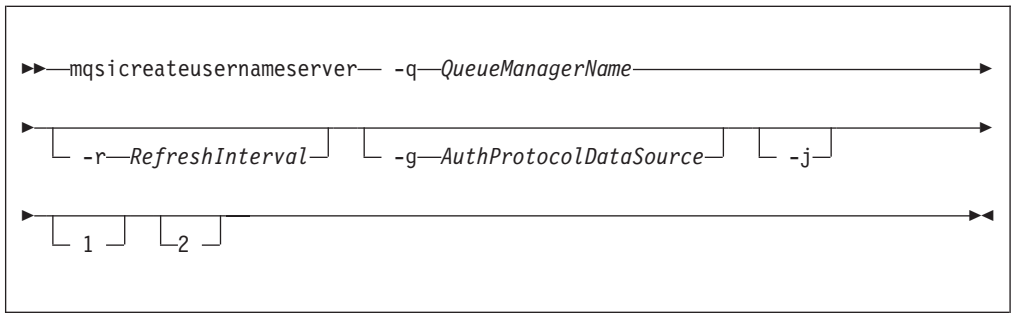
Examples:

```
mqsicreateusernameserver -i wbrkuid -a wbrkpw
   -q WBRK_QM -r 1000
```

mqsicreateusernameserver command - z/OS:

Syntax:

z/OS command - BIPCRUN:



Parameters:

-q *QueueManagerName*
(Required) The name of the queue manager associated with the User Name Server.

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.

-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.

-g *AuthProtocolDataSource*
(Optional) 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) Indicates that groups and group memberships are defined in the data source for the authentication protocol, rather than being drawn from the operating system.

1 (Optional) The registry pass, which creates only the User Name Server registry.

2 (Optional) The WebSphere MQ pass, which creates only the User Name Server WebSphere MQ queues.

Examples:

```
mqsicreateusernameserver -i wbrkuid -a wbrkpw
   -q WBRK_QM -r 1000
```

mqsicvp command

Use the mqsicvp command to perform verification tests on a broker or Configuration Manager.

Supported platforms:

- Windows
- Linux and UNIX systems

When you start a broker or Configuration Manager by using the mqsistart command, this command is run automatically to verify the component.

On z/OS, the same verification procedures are run automatically when you start a broker or a Configuration Manager.

You can run this command against a component that is running, or is not running. If the component is not running, the verification tests are performed, but the component is not started.

Purpose:

The mqsicvp command:

- Checks that the component environment is set up correctly; for example, that the installed level of Java is supported.
- Verifies that the WebSphere MQ queues are defined and accessible.
- If the component is a broker, checks that the expected database tables are present.

The mqsicvp command completes all these checks, and fails if one or more checks fail. It reports all errors in the system log, or to the command line, or both.

Syntax:

Parameters:

component
(Required) You must specify a broker name or a Configuration Manager name.

Linux UNIX All names are case sensitive on Linux and UNIX systems.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all Windows systems, you must run this command as the user ID that is defined as the service user ID for the component. You specified the service user ID on the **-i** parameter when you created the component by using the mqsicreatebroker or the mqsicreateconfigmgr command.

Therefore, on Windows Vista and Windows Server 2008, the service user ID must be a member of the group Administrators if you choose to run this command.

On Linux and UNIX systems, run this command as the user ID under which you started, or will start, the component by using the mqsistart command.

Responses:

- BIP8873I: Starting the component verification for component '*component_name*'.
- BIP8874I: The component verification for '*component_name*' has finished successfully.
- BIP8875W: The component verification for '*component_name*' has finished, but one or more checks failed.
- BIP8876I: Starting the environment verification for component '*component_name*'.
- BIP8877W: The environment verification for component '*component_name*' has finished, but one or more checks failed.
- BIP8878I: The environment verification for component '*component_name*' has finished successfully.
- BIP8879I: Starting the database verification for component '*component_name*'.
- BIP8880W: The database verification for component '*component_name*' has finished, but one or more checks failed.
- BIP8881I: The database verification for component '*component_name*' has finished successfully.
- BIP8882I: Starting the WebSphere MQ verification for component '*component_name*'.
- BIP8883W: The WebSphere MQ verification for component '*component_name*' has finished, but one or more checks failed.
- BIP8884I: The WebSphere MQ verification for component '*component_name*' has finished successfully.
- BIP8885E: Verification failed. Failed to connect to queue manager '*queue_manager_name*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8886I: Verification passed for queue '*queue_name*' on queue manager '*queue_manager_name*'.
- BIP8887E: Verification failed for queue '*queue_name*' on queue manager '*queue_manager_name*' while issuing '*operation*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8888E: Verification failed. Failed to disconnect from queue manager '*queue_manager_name*'. MQRC: *return_code* MQCC: *completion_code*
- BIP8889E: Verification failed for table '*table_name*' in broker database '*database_name*'. Subsequent messages contain further information.
- BIP8890I: Verification passed for table '*table_name*' in database '*database_name*'.

- BIP8891E: Verification failed. Failed to connect to database *'database_name'* with user ID *'user_ID'*. Subsequent messages contain further information.
- BIP8892E: Verification failed. The installed Java level *'level_installed'* does not meet the required Java level *'level_supported'*.
- BIP8893E: Verification failed for environment variable *'variable_name'*. Unable to access file *'file_name'* with user ID *'user_ID'*. Additional information for IBM support: *data1 data2*.
- BIP8894I: Verification passed for *'component_name'*.
- BIP8895E: Verification failed. Environment variable *'variable_name'* is incorrect or missing.
- BIP8896E: Verification failed. Unable to access the registry with user ID *'user_ID'*. Additional information for IBM support: *data1 data2*
- BIP8897E: Verification failed. Environment variable *'variable_name'* does not match the component name *'component_name'*.
- BIP8898E: Verification failed. Table *'table_name'* in database *'database_name'* contains no columns.
- BIP8899E: Verification failed. Table *'table_name'* in database *'database_name'* contains *'number_of_rows'* rows, but must contain exactly 1 row.
- BIP8900I: APF Authorization check successful for file *'file_name'*.
- BIP8903E: Verification failed. The APF Authorization check failed for file *'file_name'*.
- BIP8904E: Verification failed. Failed to stat file *'file_name*1' with return code *'return_code'* and errno *'error_number'*.

Examples:

Run verification checks on the broker named WBRK_BROKER:

```
mqsicvp WBRK_BROKER
```

Run verification checks on the Configuration Manager named `ConfigMgr`:

```
mqsicvp ConfigMgr
```

mqsideleteaclentry command

Use the mqsideleteaclentry command to delete a Configuration Manager access control list entry that you have defined.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPDLACL; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsideleteaclentry command - Windows" on page 549
- "mqsideleteaclentry command - Linux and UNIX systems" on page 551
- "mqsideleteaclentry command - z/OS" on page 553

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

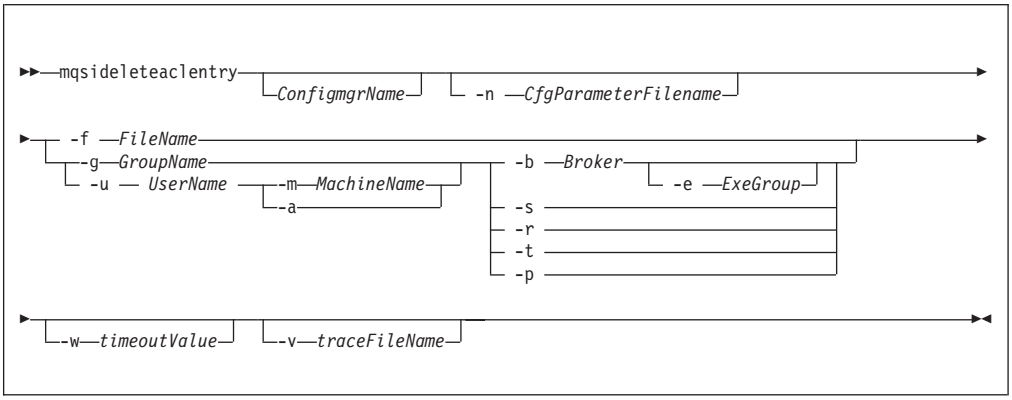
On all platforms, the user ID used to run this command must have full control permissions for the object being changed.

On Linux and UNIX, the user ID must be a member of mqbrkrs.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. Therefore the commands inherit a Full Control root ACL and you can carry out all operations. If you submit a console command to the Configuration Manager you can change all ACLs for that Configuration Manager.

mqsideleteaclentry command - Windows:

Syntax:



Parameters:

You must select:

- **-f**, or
  - **-u** and **-a**, or **-m**, or
  - **-g**, or
  - **-b**, or
  - **-s**, or
  - **-r**, or
  - **-t**, or
  - **-p**

ConfigmgrName
(Optional) The name of the Configuration Manager from which the access control lists are to be deleted.

The default name, if this parameter is not specified, is 'ConfigMgr'.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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) The specified user name can be on any machine. This option can not be used with **-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 *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.

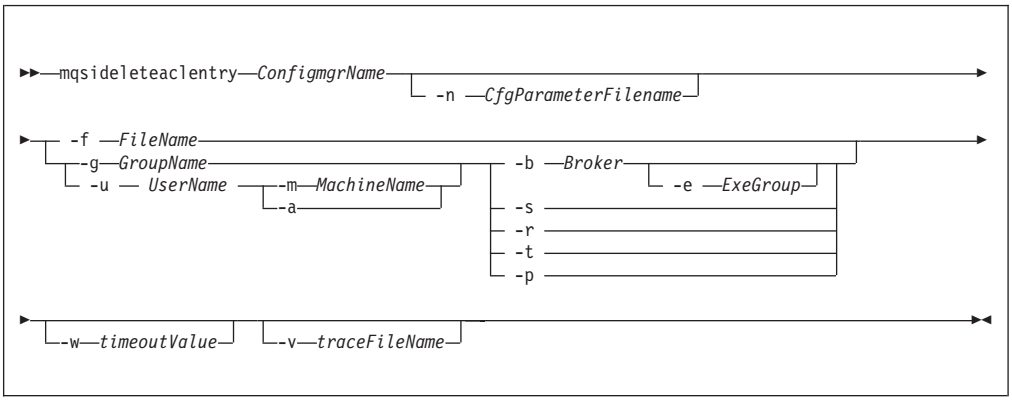
-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

```
mqsideleteaclentry CMGR01 -f c:\test\mylist
mqsideleteaclentry CMGR01 -g GROUPA -b MYBROKER
```

mqsideleteaclentry command - Linux and UNIX systems:

Syntax:



Parameters:

You must select:

- **-f**, or
  - **-u** and **-a**, or **-m**, or
  - **-g**, or
  - **-b**, or
  - **-s**, or
  - **-r**, or
  - **-t**, or
  - **-p**

ConfigmgrName
(Required) The name of the Configuration Manager from which the access control lists are to be deleted. This parameter must be the first parameter specified and its name is case-sensitive.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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) The specified user name can be on any machine. This option can not be used with **-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 *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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

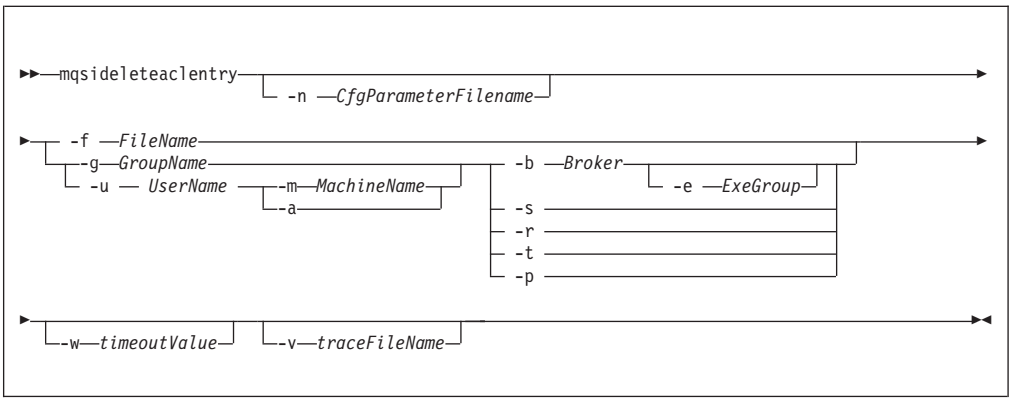
Examples:

```
mqsideleteaclentry CMGR01 -f c:\test\mylist
mqsideleteaclentry CMGR01 -g GROUPA -b MYBROKER
```

mqsideleteaclentry command - z/OS:

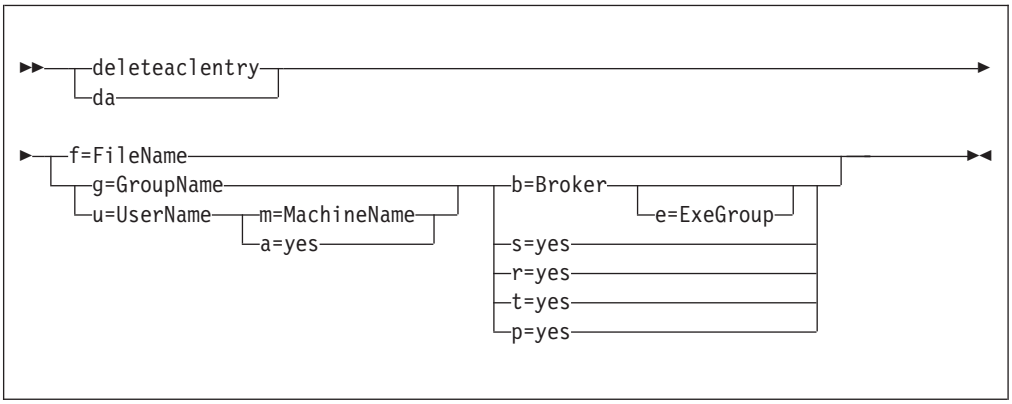
Syntax:

z/OS command - BIPDLACL:



z/OS console command:

Synonym: da



Parameters:

You must select:

- **-f**, or
  - **-u** and **-a**, or **-m**, or
  - **-g**, or
  - **-b**, or
  - **-s**, or

– **-r**, or

– **-t**, or

– **-p**

-n *CfgParameterFilename*

(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

Remove the statement `encoding="UTF-8"` from the first line of the `.configmgr` file, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*

(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*

(Optional) The 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) The user name to which this entry refers, for example, TEST\ANOTHER.

-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) The specified user name can be on any machine. This option can not be used with **-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 *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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

```
mqsideleteaclentry CMGR01 -g GROUPA -b MYBROKER
```

For the z/OS console command you must use a comma between each command option. The following example shows the z/OS console version of the preceding example:

```
/f CMGR01,da g='GROUPA' ,b='MYBROKER'
```

mqsideletebroker command

Use the mqsideletebroker command to delete a named broker. The command also deletes the queues on the associated 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.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPDLBK; see “Contents of the broker PDSE” on page 679

Purpose:

The mqsideletebroker 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.

  Although this command deletes all the data related to this broker from the broker database tables, it does not check if the tables are empty, and it does not delete the tables.
- Removes the record for the component in the registry.

If you delete a broker that has WebSphere MQ publish/subscribe broker neighbors, you must also run the clrmqbrk on each of these neighbors. You must identify the broker that you are deleting on this command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsideletebroker command - Windows, Linux, and UNIX systems"
- "mqsideletebroker command - z/OS" on page 557

Usage notes:

If you run the command against a component that does not exist (for example, the component has already been deleted, or you have mistyped the component name), the command returns with a successful completion message. The command does not inform you that the component does not exist.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

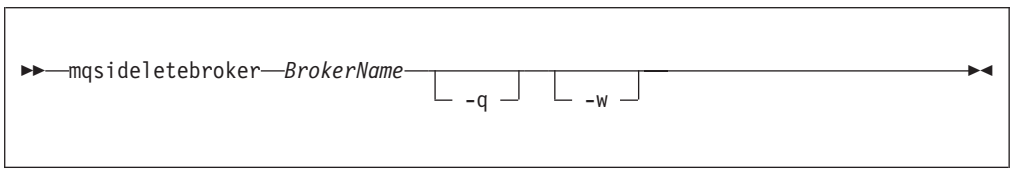
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsideletebroker command - Windows, Linux, and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required) The name of the broker that you want to delete. You must specify this parameter first.

-q (Optional) Specifies that the broker's queue manager is deleted. If you do not specify this parameter, only the WebSphere Message 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) Deletes all files related to this broker from the work path.

Examples:

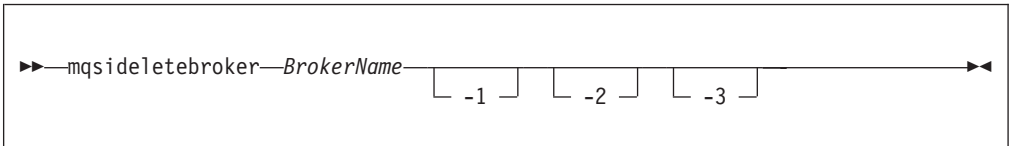
Delete the broker and its associated queue manager:

```
mqsideletebroker WBRK_BROKER -q
```

mqsideletebroker command - z/OS:

Syntax:

z/OS command - BIPDLBK:



Parameters:

BrokerName
(Required) The name of the broker that you want to delete. This must be the first parameter.

-1 (Optional) Deletes only the broker registry.

-2 (Optional) Deletes only the broker WebSphere MQ queues.

-3 (Optional) Deletes only the broker DB2 tables and indexes.

Examples:

Delete the broker registry for the broker CSQ1BRK on z/OS:

```
mqsideletebroker CSQ1BRK -1
```

Delete the broker CSQ2BRK and all its associated resources:

```
mqsideletebroker CSQ2BRK
```

mqsideleteconfigmgr command

Use the mqsideleteconfigmgr command to delete a named Configuration Manager. The command also deletes the queues on the associated queue manager (created when the Configuration Manager was created), and its internal repository. You can also specify that the queue manager is to be deleted.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPDLCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

The mqsideleteconfigmgr 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.

- Deletes the configuration repository, if requested.
- Removes the record for the component in the registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsideleteconfigmgr command - Windows"
- "mqsideleteconfigmgr command - Linux and UNIX systems" on page 559
- "mqsideleteconfigmgr command - z/OS" on page 560

Usage notes:

If you run the command against a component that does not exist (for example, the component has already been deleted, or you have mistyped the component name), the command returns with a successful completion message. The command does not inform you that the component does not exist.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

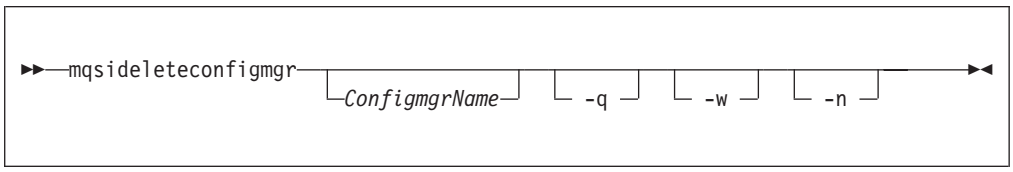
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsideleteconfigmgr command - Windows:

Syntax:



Parameters:

ConfigmgrName
(Optional) The name of the Configuration Manager that you want to delete.

The default name, if this parameter is not specified, is 'ConfigMgr'.

-q (Optional) Deletes the Configuration Manager's queue manager. (If this option is not specified, only the WebSphere Message 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) 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 must be done with extreme caution as all domain information for the Configuration Manager is deleted.

To delete the configuration repository manually:

1. Locate the working directory for the Configuration Manager. The default location is `%ALLUSERSPROFILE%\Application Data\IBM\MQSI\` where `%ALLUSERSPROFILE%` is the environment variable that defines the system working directory. The default directory depends on the operating system:
   - On Windows XP and Windows Server 2003: `C:\Documents and Settings\All Users\Application Data\IBM\MQSI\`
   - On Windows Vista and Windows Server 2008: `C:\ProgramData\IBM\MQSI\`

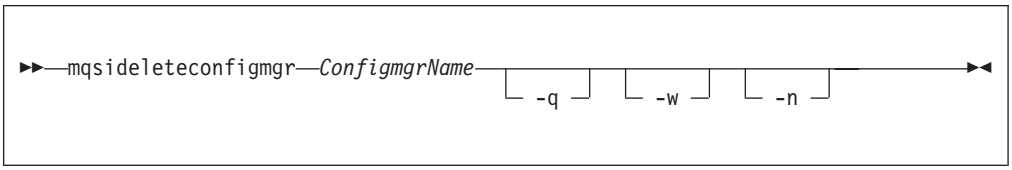
   The actual value might be different on your computer.
2. Locate the `components` directory. Delete the directory with the same name as the Configuration Manager.

Examples:

```
mqsideleteconfigmgr CMGR01 -q
```

mqsideleteconfigmgr command - Linux and UNIX systems:

Syntax:



Parameters:

ConfigmgrName

(Required) The name of the Configuration Manager that you want to delete.

This must be the first parameter specified and the name is case-sensitive.

-q (Optional) Deletes the Configuration Manager's queue manager. (If this option is not specified, only the WebSphere Message 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) 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 must 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`/var/mqsi`
2. Locate the `components` directory. Delete the directory with the same name as the Configuration Manager.

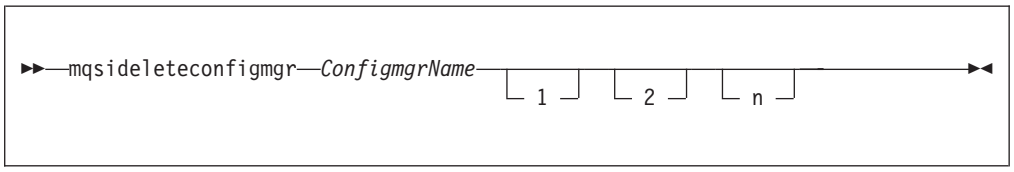
Examples:

```
mqsideleteconfigmgr CMGR01 -q
```

mqsideleteconfigmgr command - z/OS:

Syntax:

z/OS command - BIPDLCM:



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager that you want to delete.

This must be the first parameter specified and the name is case-sensitive.

-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 must 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 `++COMPONENTDIRECTORY++`
2. Locate the `components` directory. Delete the directory with the same name as the Configuration Manager.

1 (Optional) The registry pass that deletes only the Configuration Manager registry.

2 (Optional) 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.

Examples:

```
mqsideleteconfigmgr CMGR01 -q
```

mqsideleteconfigurableservice command

Use the mqsideleteconfigurableservice command to delete a configurable service, such as a JMS provider, JDBC provider, or FTP server, that you have created by using the mqsicreateconfigurableservice command.

Supported platforms:

- Windows systems
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPDLCS; see "Contents of the broker PDSE" on page 679

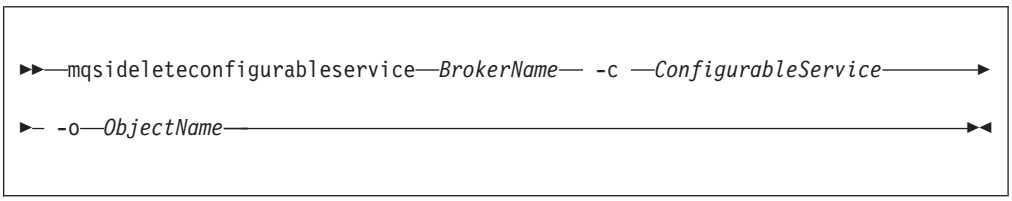
Purpose:

Use this command to delete a configurable service. Use the mqsireportproperties command to view the configurable services that are defined.

Usage notes:

- Before you run this command, ensure that the broker is running.
- After you have run this command, stop and restart the broker to ensure that deleted broker resources and properties are not being used.

Syntax:



Parameters:

BrokerName
(Required) The name of the broker to modify. This parameter must be the first parameter.

-c *ConfigurableService*
(Required) The type of configurable service. Use the mqsireportproperties command to view the list of all defined services.

For a list of supplied configurable services, and their properties and values, see "Configurable services properties" on page 444.

-o *ObjectName*
(Required) The name of the object for which you want to delete the properties. Use the mqsireportproperties command to view the list of properties that you can delete.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID must be a member of the mqbrkrs group.

Responses:

This command returns the following responses

- BIP8012 Unable to connect to system components
- BIP8014 Component cannot be created
- BIP8073 Invalid broker name
- BIP8984 Configurable service was not found

Examples:

Delete an FtpServer configurable service for broker WBRK6_DEFAULT_BROKER:

```
mqsideleteconfigurableservice WBRK6_DEFAULT_BROKER -c FtpServer -o Server01
```

Delete a JMS provider configurable service called *MyProviderXYZ*:

```
mqsideleteconfigurableservice WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_MyProviderXYZ
```

Delete a monitoring profile:

```
mqsideleteconfigurableservice myBroker -c MonitoringProfiles -o myMonitoringProfile
```

Delete the PeopleSoftConnection configurable service that is associated with *myPeopleSoftAdapter.outadapter*:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c PeopleSoftConnection
-o myPeopleSoftAdapter.outadapter
```

Delete a security profile for LDAP use:

```
mqsideleteconfigurableservice WBRK6_DEFAULT_BROKER -c SecurityProfiles -o MyLDAPProfile
```

Delete the SiebelConnection configurable service that is associated with *mySiebelAdapter.outadapter*:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c SiebelConnection
-o mySiebelAdapter.outadapter
```

Delete a TCPIPClient configurable service:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c TCPIPClient
-o ClientPort1452HostnameJsmith
```

Delete a TCPIPServer configurable service:

```
mqsideleteconfigurableservice WBRK61_DEFAULT_BROKER -c TCPIPServer -o ServerPort1452
```

mqsideletedb command

Use the mqsideletedb command to delete a database that you created by using the mqsicreatedb command.

Supported platforms:

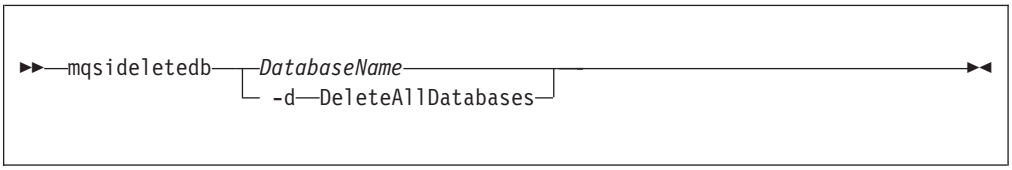
- Windows

Purpose:

The mqsideletedb command deletes the database and removes the ODBC data source name.

If this database is the last Derby database managed by DatabaseInstanceMgr, the Derby Network Server and DatabaseInstanceMgr service are also stopped and removed.

Syntax:



Parameters:

DatabaseName
This must be the only parameter. Specify the name of the database that you want to delete.

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.

All the data stored in **ALL** databases is permanently deleted.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

The user ID must also be a member of the mqbrkrs group.

Examples:

The following example deletes the database brokerdb:

```
mqsideletedb brokerdb
```

mqsideleteexecutiongroup command

Use the mqsideleteexecutiongroup command to remove an execution group from a broker.

Supported platforms:

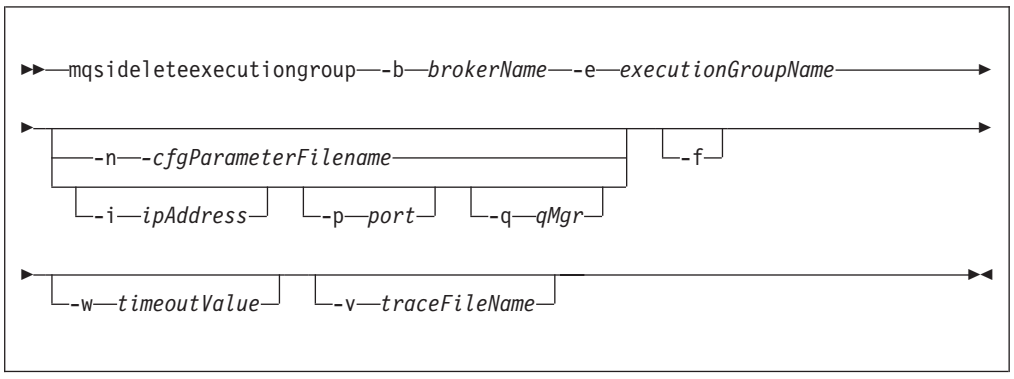
- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPDLEG; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

You must start the Configuration Manager 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:



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.

-n *cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

If you are using this file on z/OS, you must remove the statement `encoding="UTF-8"` from the first line, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value 1414 is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used.

-f Forces the broker to be removed from the Configuration Manager, if this is the last execution group. If this is not the last execution group in the broker, this flag is ignored.

-w *timeoutValue*
(Optional) This parameter is the time in seconds that the utility waits to ensure that the command completed; the default value is 60.

-v *traceFileName*
(Optional) This parameter sends internal debug trace information to the specified file.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the group mqm.

The command succeeds only if this user ID has the correct authority defined in the access control list for the Configuration Manager. To delete an execution group, full control authority is required over the broker object; see "ACL permissions" on page 723 for a list of permissions that can be defined in the Configuration Manager.

Responses:

The mqsideleteexecutiongroup 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 released 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 the Configuration 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 related resources, and send output to `trace.txt`.

```
mqsideleteexecutiongroup –n domain2.configmgr –b FRED –e EG3 –w 300 –v trace.txt
```

mqsideleteusernameserver command

Use the mqsideleteusernameserver command to delete the User Name Server. The command also deletes the queues on the associated queue manager (created when the User Name Server was created). You can also specify that the queue manager is to be deleted.

Supported platforms:

- Windows
- Linux and UNIX systems

- z/OS. Run this command by customizing and submitting BIPDLUN; see "Contents of the User Name Server PDSE" on page 681

Purpose:

The mqsideleteusernameserver command:

- On Windows platforms, 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 registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsideleteusernameserver command - Windows, Linux and UNIX systems"
- "mqsideleteusernameserver command - z/OS" on page 568

Usage notes:

If you run the command against a component that does not exist (for example, the component has already been deleted, or you have mistyped the component name), the command returns with a successful completion message. The command does not inform you that the component does not exist.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

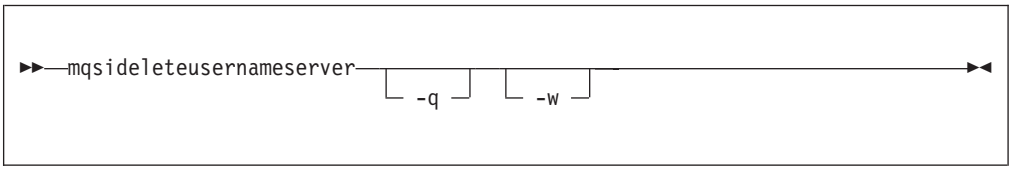
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

mqsideleteusernameserver command - Windows, Linux and UNIX systems:

Syntax:



Parameters:

-q (Optional) 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 Message 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) Deletes all files related to the User Name Server from the work path.

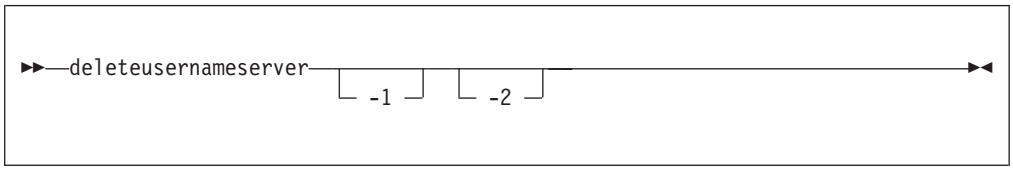
Examples:

```
mqsideleteusernameserver
```

mqsideleteusernameserver command - z/OS:

Syntax:

command - BIPDLUN:



Parameters:

-1 (Optional) Deletes only the User Name Server registry.

-2 (Optional) Deletes only the User Name Server WebSphere MQ queues.

Examples:

```
mqsideleteusernameserver
```

mqsideploy command

Use the mqsideploy command to make a deployment request to the Configuration Manager.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPDPLY; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Use the mqsideploy command to make deployment requests of all types from a batch command script, without the need for manual interaction.

The default operation is a delta or incremental deployment. Select **-m** to override the default operation and perform a complete deployment.

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.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsideploy command - Windows, Linux, and UNIX systems" on page 569

• “mqsideploy command - z/OS” on page 573

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID used to run this command must have sufficient authority defined in the Configuration Manager to successfully deploy. The permissions required are the same as the permission required to perform the equivalent task in the Message Broker Toolkit.

On Linux and UNIX systems, the user ID used to run this command must be a member of the group mqbrkrs. If you do not have mqbrkrs membership, the command fails with the following error message:

```
mqsideploy -i localhost -p 1414 -q qm2 -b brk2
BIP8081 An exception was caught while processing the command,
'Unable to format an ImbException message for output, ImbException message
number is BIP2164'.
```

Responses:

This command returns the following responses:

0 (Success) The Configuration Manager issued the deployment request and all of the relevant brokers responded successfully before the timeout period expired.

2 (Failure) The Configuration Manager issued the deployment request and at least one broker responded negatively. See the messages issued from the utility (or theConfiguration Manager event log) for more information.

3 (Initiated) The Configuration Manager has replied, stating that deployment has started, but that no broker responses were received before the timeout occurred.

5 (Submitted) The deployment message was sent to the Configuration Manager, but no response was received before the timeout occurred.

6 (SuccessSoFar) 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.

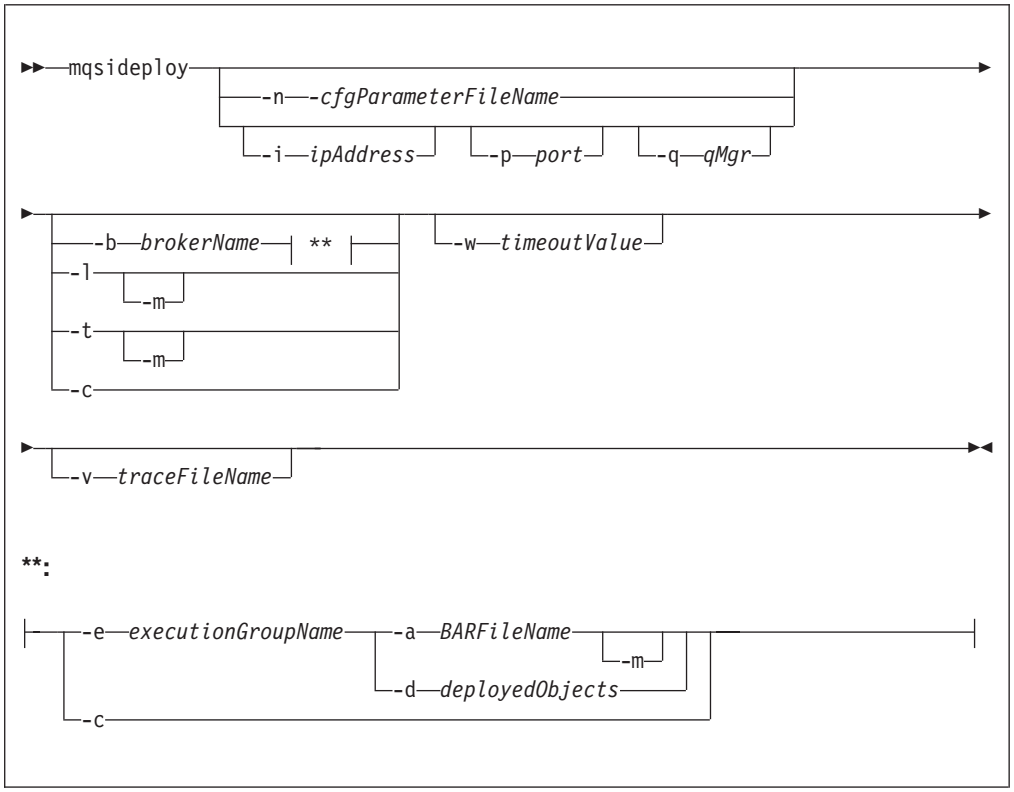
98 The Configuration Manager cannot be reached.

99 The arguments supplied to the utility are not valid.

mqsideploy command - Windows, Linux, and UNIX systems:

Use the mqsideploy command on Windows, Linux, and UNIX to make a deployment request to the Configuration Manager.

Syntax:



Parameters:

-n *-cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value `1414` is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used.

-b *brokerName*
(Optional) This parameter specifies the name of the broker to which to deploy. If you specify either of the **-t** or **-l** parameters, the **-b** parameter is ignored

because, when deploying topics or topology, all brokers in the domain are affected. Without the **-e** and **-a** parameters, a broker configuration deployment is initiated. If you do not specify the **-b** parameter, the command applies to all brokers in the domain.

-e *executionGroupName*
(Optional) This parameter specifies the name of the execution group to which to deploy. You must specify **-b** and **-a** with this parameter.

-a *BARFileName*
(Optional) This parameter specifies the name of the broker archive (BAR) file that is to be used for deployment of the message flow and other resources. You must also specify the **-b** and **-e** parameters with this option.

-t (Optional) This parameter specifies deployment of all the topics configuration information.

- When used with the **-m** parameter, the **-t** parameter causes the complete topics hierarchy to be deployed to all brokers in the domain.
- When used without the **-m** parameter, the **-t** parameter causes changes only to the topics hierarchy (since the last successful topics deployment) to be deployed to all brokers in the domain.

-l (Optional) This parameter specifies that the topology configuration should be deployed. Information is deployed to all brokers in the domain if you also set the **-m** parameter; otherwise, the information is deployed only to brokers with a changed topology configuration.

-c (Optional) This parameter instructs the Configuration Manager to stop waiting for responses to previously submitted deployment requests. If used with the **-b** parameter, the Configuration Manager stops waiting for outstanding deployment responses from the specified broker; without the **-b** parameter, the Configuration Manager stops waiting for responses to all outstanding deployment requests in the domain.

Specify the **-c** parameter with caution; it has the effect of canceling deployment requests. Use this option only if the affected brokers will respond to the outstanding requests. If a broker subsequently processes a deployment request that has been cancelled, the Configuration Manager ignores the response, and is therefore no longer synchronized with the broker.

-w *timeoutValue*
(Optional) This parameter specifies the time in seconds that the command waits for the broker to reply before returning control to the command line or batch file. The mqsideploy command polls the Configuration Manager log records, looking for the results of the deployment request that has just been sent. The relevant log records contain information indicating whether the deployment was successful. The *timeoutValue* is the number of seconds that the command waits before timing out.

You can set this parameter to a value in the range 1 - 2 145 336 164. If you do not provide a *timeoutValue* value, or you set a value less than 1 or greater than 2 145 336 164 is specified, an error is returned.

Set this parameter to a value greater than the sum of the configuration timeouts that you specified for the broker, the *ConfigurationChangeTimeout* and the *InternalConfigurationTimeout* parameters, if you want to ensure that a response is received within the *timeoutValue* period. If you set a smaller value, the response returned might indicate that the state of the deploy request is unknown.

-d *deployedObjects*
(Optional) This parameter describes the set of objects that you want to remove from the execution group. You can specify multiple files to deploy by separating the filenames with a colon (:).

You can specify objects of all types, but if you specify an ambiguous object name (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, you must specify the fully qualified name of the objects to remove; for example, ″top.dictionary:top.cmf″.

-v *traceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

-m
(Optional) This parameter specifies deployment of complete information:

- For a *BAR file deployment*, **-m** removes all currently-deployed message flows and message sets from the execution group as part of the deployment. If you do not set **-m**, the contents of the BAR file are deployed in addition to what is already deployed to the execution group. Any deployed objects with the same name as an item inside the BAR file are replaced by the version inside the BAR file.
- For a *topology configuration deployment*, **-m** deploys complete inter-broker configuration information to all brokers. If you do not set **-m**, only changed inter-broker configuration is deployed to brokers whose inter-broker configuration has changed.
- For a *broker configuration deployment* this parameter is not valid.
- For a *topic tree deployment*, **-m** deploys the entire topic tree to all brokers. If you do not set **-m**, only changes to the topic tree are deployed to all brokers.
- For a *remove message flow or message set operation*, the **-m** parameter is ignored.

Examples:

The following examples show the use both of the **i**, **p**, and **q** parameters, and the **-n** parameter, to define the connection parameters for the Configuration Manager; either form is valid.

Deploy a BAR file to the specified broker which is in the domain controlled by the Configuration Manager whose connection parameters are described in the file `cm1.configmgr`, and remove all currently-deployed message flows and message sets from the execution group as part of the deployment. Allow 10 minutes for the broker to reply.

```
mqsideploy -n cm1.configmgr -b broker1 -e default -a mybar.bar -m -w 600
```

Remove the message flow `top` and the dictionary `bar` from the execution group `default` on the broker `b1`, using the Configuration Manager whose connection parameters are described in the file `cm1.configmgr`.

(If there are no other objects called `top` and `bar` deployed to the execution group, you can shorten the value of the **-d** parameter to `top:bar`.)

```
mqsideploy –n cm1.configmgr –b B1 –e default –d top.cmf:bar.dictionary
```

Deploy publish/subscribe neighbors using a connection file whose parameters are described in the file `cm1.configmgr`, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -n cm1.configmgr -m -w 600
```

Deploy publish/subscribe neighbors using the **i**, **p**, and **q** parameters to connect to the Configuration Manager, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -i localhost -p 1414 -q QMNAME -m -w 600
```

Deploy a topics hierarchy using a connection file whose parameters are described in the file `cm1.configmgr`, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -n cm1.configmgr -t -m -w 600
```

Deploy a broker configuration using a connection file whose parameters are described in the file `cm1.configmgr`, to the specified broker, and allow 15 minutes for the broker to reply:

```
mqsideploy -n cm1.configmgr -b broker1 -w 900
```

Cancel a deployment using a connection file whose parameters are described in the file `cm1.configmgr`, and allow 15 minutes for the broker to reply. In this example the Configuration Manager stops waiting for all outstanding deployment requests in the domain.

```
mqsideploy -n cm1.configmgr -c -w 900
```

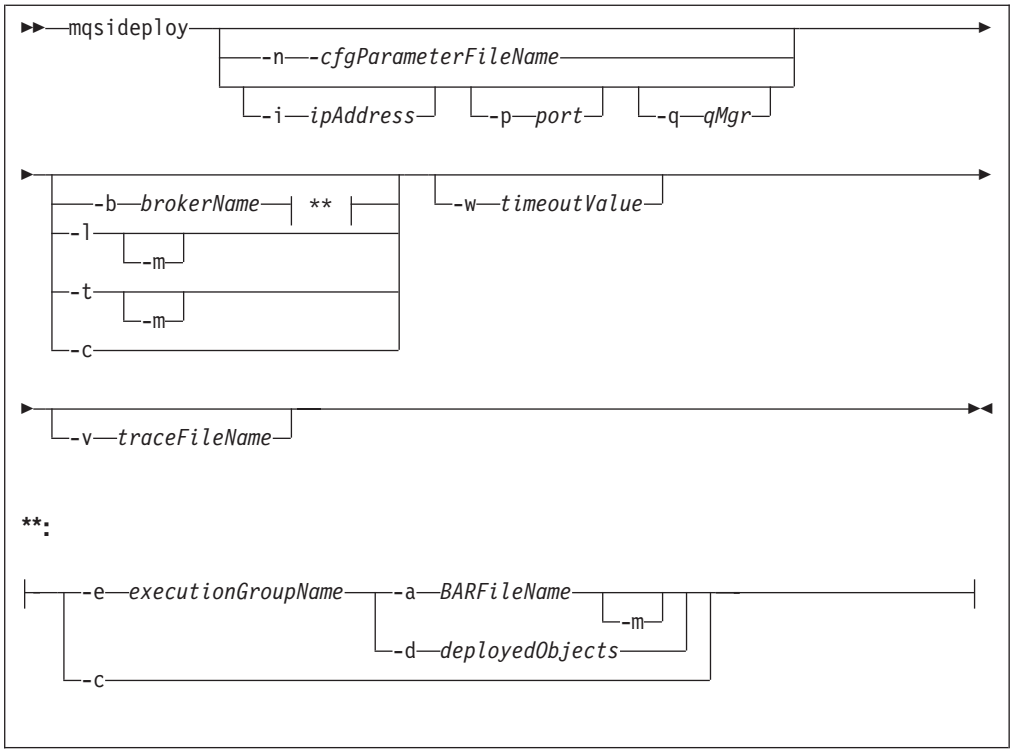
mqsideploy command - z/OS:

Use the mqsideploy command on z/OS to make a deployment request to the Configuration Manager.

BIPDPLY is used to run mqsideploy. On z/OS, the command uses Java bindings; see “Usage note” on page 577.

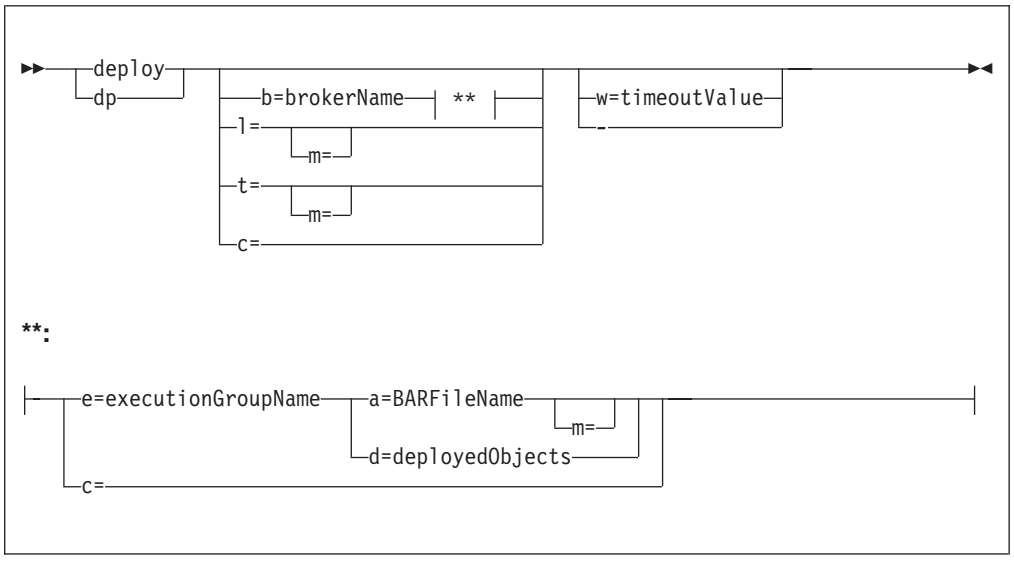
Syntax:

z/OS command - BIPDPLY:



z/OS console command:

Synonym: dp



Parameters:

-n *-cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

Remove the statement `encoding="UTF-8"` from the first line of the `.configmgr` file, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used, which results in a local binding connection.

To connect to the local host, set the value to a space (" ").

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value `1414` is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used.

-b *brokerName*
(Optional) This parameter specifies the name of the broker to which to deploy. If you specify either of the **-t** or **-l** parameters, the **-b** parameter is ignored because, when deploying topics or topology, all brokers in the domain are affected. Without the **-e** and **-a** parameters, a broker configuration deployment is initiated. If you do not specify the **-b** parameter, the command applies to all brokers in the domain.

Specify the **-c** parameter to cancel deployment to a specific broker.

-e *executionGroupName*
(Optional) This parameter specifies the name of the execution group to which to deploy. You must specify **-b** and **-a** with this parameter.

-a *BARFileName*
(Optional) This parameter specifies the name of the broker archive (BAR) file that is to be used for deployment of the message flow and other resources. You must also specify the **-b** and **-e** parameters with this option.

The BAR file can be in a local or remote file system, if the user ID or the Configuration Manager that is running the command can access the file and read it.

-t (Optional) This parameter specifies deployment of all the topics configuration information.

- When used with the **-m** parameter, the **-t** parameter causes the complete topics hierarchy to be deployed to all brokers in the domain.
- When used without the **-m** parameter, the **-t** parameter causes changes only to the topics hierarchy (since the last successful topics deployment) to be deployed to all brokers in the domain.

-l (Optional) This parameter specifies that the topology configuration should be deployed. Information is deployed to all brokers in the domain if you also set the **-m** parameter; otherwise, the information is deployed only to brokers with a changed topology configuration.

-c (Optional) This parameter instructs the Configuration Manager to stop waiting for responses to previously submitted deployment requests. If used with the **-b** parameter, the Configuration Manager stops waiting for outstanding deployment responses from the specified broker; without the **-b** parameter, the Configuration Manager stops waiting for responses to all outstanding deployment requests in the domain.

Specify the **-c** parameter with caution; it has the effect of canceling deployment requests. Use this option only if the affected brokers will respond to the outstanding requests. If a broker subsequently processes a deployment request that has been cancelled, the Configuration Manager ignores the response, and is therefore no longer synchronized with the broker.

-w *timeoutValue*
(Optional) This parameter specifies the time in seconds that the command waits for the broker to reply before returning control to the command line or batch file. The mqsideploy command polls the Configuration Manager log records, looking for the results of the deployment request that has just been sent. The relevant log records contain information indicating whether the deployment was successful. The *timeoutValue* is the number of seconds that the command waits before timing out.

You can set this parameter to a value in the range 1 - 2 145 336 164. If you do not provide a *timeoutValue* value, or you set a value less than 1 or greater than 2 145 336 164 is specified, an error is returned.

Set this parameter to a value greater than the sum of the configuration timeouts that you specified for the broker, the *ConfigurationChangeTimeout* and the *InternalConfigurationTimeout* parameters, if you want to ensure that a response is received within the *timeoutValue* period. If you set a smaller value, the response returned might indicate that the state of the deploy request is unknown.

-d *deployedObjects*
(Optional) This parameter describes the set of objects that you want to remove from the execution group. You can specify multiple files to deploy by separating the filenames with a colon (:).

You can specify objects of all types, but if you specify an ambiguous object name (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, you must specify the fully qualified name of the objects to remove; for example, ″top.dictionary:top.cmf″.

-v *traceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

-m
(Optional) This parameter specifies deployment of complete information:

- For a *BAR file deployment*, **-m** removes all currently-deployed message flows and message sets from the execution group as part of the deployment. If you do not set **-m**, the contents of the BAR file are deployed in addition to what is already deployed to the execution group. Any deployed objects with the same name as an item inside the BAR file are replaced by the version inside the BAR file.

- For a *topology configuration deployment*, **-m** deploys complete inter-broker configuration information to all brokers. If you do not set **-m**, only changed inter-broker configuration is deployed to brokers whose inter-broker configuration has changed.
- For a *broker configuration deployment* this parameter is not valid.
- For a *topic tree deployment*, **-m** deploys the entire topic tree to all brokers. If you do not set **-m**, only changes to the topic tree are deployed to all brokers.
- For a *remove message flow or message set operation*, the **-m** parameter is ignored.

Usage note:

If you enter the command and specify the **i**, **p**, and **q** parameters to identify the Configuration Manager to connect to, the command attempts to use WebSphere MQ Java client code, which is not supported on z/OS. The following error occurs:

```
BIP1046E: Unable to connect with the Configuration
Manager (name)
```

The reported reason code is `2298 0x000008fa MQRC_FUNCTION_NOT_SUPPORTED`.

Examples:

The following examples show the use both of the **i**, **p**, and **q** parameters, and the **-n** parameter, to define the connection parameters for the Configuration Manager; either form is valid.

Deploy a BAR file to the specified broker which is in the domain controlled by the Configuration Manager whose connection parameters are described in the file `cm1.configmgr`, and remove all currently-deployed message flows and message sets from the execution group as part of the deployment. Allow 10 minutes for the broker to reply.

```
mqsideploy -n cm1.configmgr -b broker1 -e default -a mybar.bar -m -w 600
```

Remove the message flow `top` and the dictionary `bar` from the execution group `default` on the broker `b1`, using the Configuration Manager whose connection parameters are described in the file `cm1.configmgr`.

(If there are no other objects called `top` and `bar` deployed to the execution group, you can shorten the value of the **-d** parameter to `top:bar`.)

```
mqsideploy –n cm1.configmgr –b B1 –e default –d top.cmf:bar.dictionary
```

Deploy publish/subscribe neighbors using a connection file whose parameters are described in the file `cm1.configmgr`, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -n cm1.configmgr -m -w 600
```

Deploy publish/subscribe neighbors using the **i**, **p**, and **q** parameters to connect to the Configuration Manager, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -i localhost -p 1414 -q QMNAME -m -w 600
```

You can use the **i**, **p**, and **q** parameters in the following examples instead of the **-n** parameter.

Deploy a topics hierarchy using a connection file whose parameters are described in the file `cm1.configmgr`, allow 10 minutes for the broker to reply, and deploy complete inter-broker configuration information:

```
mqsideploy -n cm1.configmgr -t -m -w 600
```

Deploy a broker configuration using a connection file whose parameters are described in the file `cm1.configmgr`, to the specified broker, and allow 15 minutes for the broker to reply:

```
mqsideploy -n cm1.configmgr -b broker1 -w 900
```

Cancel a deployment using a connection file whose parameters are described in the file `cm1.configmgr`, and allow 15 minutes for the broker to reply. In this example, the Configuration Manager stops waiting for all outstanding deployment requests in the domain.

```
mqsideploy -n cm1.configmgr -c -w 900
```

mqsiformatlog command

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 who runs the command.

Supported platforms:

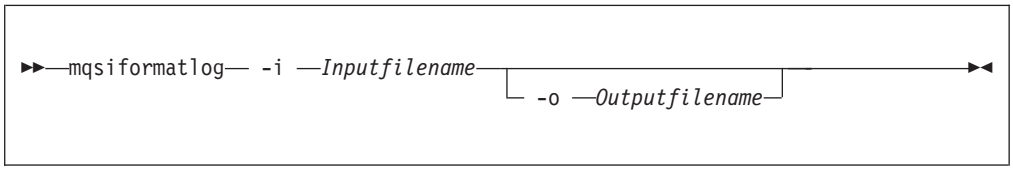
- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPFMLG; see "Contents of the broker PDSE" on page 679

Purpose:

The mqsiformatlog command interprets an input log file that has been created on any system in a platform-independent code page, `utf-8`. Use this command 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:



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:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID used to run 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.

Examples:

```
mqsiformatlog -i trace.xml -o formattrace.log
```

The following extract shows the output that is 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

Use the mqsijoinmqpubsub command to join this WebSphere Message 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 Message Broker broker.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPJNMP; see "Contents of the broker PDSE" on page 679

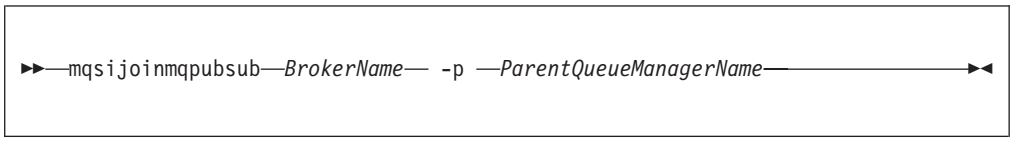
Purpose:

The mqsijoinmqpubsub command runs asynchronously. Successful completion of this command indicates that the WebSphere Message Broker broker has accepted the request, not that the required action has completed.

Use the mqsilist**mqpubsub** command to monitor the status of the asynchronous actions that result from this command.

Use this command only if you are integrating this WebSphere Message Broker broker with an WebSphere MQ Publish/Subscribe broker network. Before you issue this command, ensure that the WebSphere Message 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:



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 Message Broker broker is to be joined.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID used to run 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.

Examples:

```
mqsijoinmqpubsub WBRK_BROKER -p MQBroker1
```

mqsilist (list resources) command

Use the mqsilist command on Windows, Linux, and UNIX systems, or the list command from a z/OS console, 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.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPLIST; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

On Windows, Linux, and UNIX systems, the output is directed to STDOUT.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsilist (list resources) command - Windows, Linux, and UNIX systems"
- "mqsilist (list resources) command - z/OS" on page 582

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

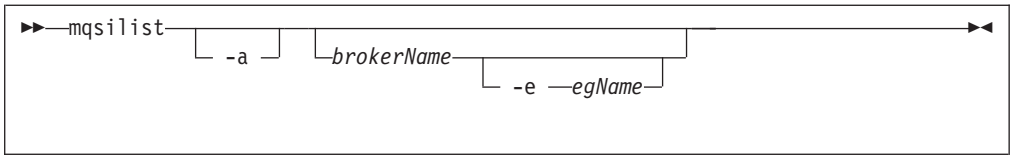
On Windows platforms, if you have specified a broker name and the `-e` flag, the user ID used to run this command must have mqbrkrs group membership.

On Linux and UNIX systems, the user ID must be a member of the mqbrkrs group.

mqsilist (list resources) command - Windows, Linux, and UNIX systems:

Use the mqsilist command to list all the components installed on the system.

Syntax:



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
```

Parameters:

-a (Optional) List all the components installed on the system.

If you use this option on brokers from a previous release of the product, you see the following message:

```
BIP8221I: <Component>: <ComponentName> (<Version>) - <Queue Manager>
```

brokerName
(Optional) The name of the broker for which you want to list resources. You must specify a deployed broker. A list of execution groups configured on this broker, and the process ID (pid) of each, is displayed.

-e *egName*
(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.

Examples:

List all the components defined on the local computer:

```
mqsilist -a
```

List the execution groups defined on a specific broker:

```
mqsilist WBRK_BROKER
```

List the message flows that are deployed to a specific execution group:

```
mqsilist WBRK_BROKER -e DefaultEG
```

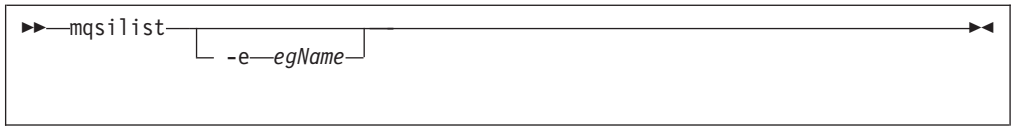
List information about the Database Instance Manager:

```
mqsilist DatabaseInstanceMgr
```

mqsilist (list resources) command - z/OS:

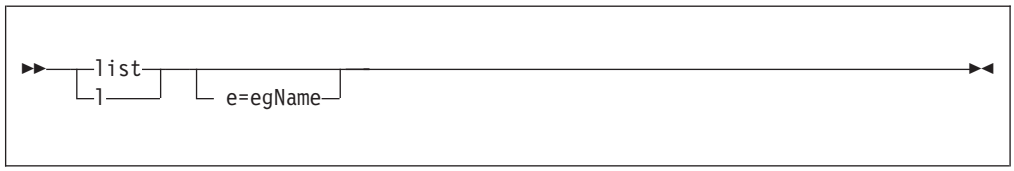
Syntax:

z/OS command - BIPLIST:



z/OS console command:

Synonym: l



If you do not specify any parameters when you issue this command, a list of the execution groups is displayed.

Parameters:

-e *egName*
(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.

Examples:

You can run 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
```

mqsilistaclentry command

Use the mqsilistaclentry command to view or list the user groups, users, objects, or access control lists that you have defined.

Supported platforms:

- Windows
- Linux and UNIX systems

- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPLIACL; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Use the mqsilistaclentry command to view or list the following currently defined resources:

- 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″.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsilistaclentry command - Windows”
- “mqsilistaclentry command - Linux and UNIX systems” on page 587
- “mqsilistaclentry command - z/OS” on page 589

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

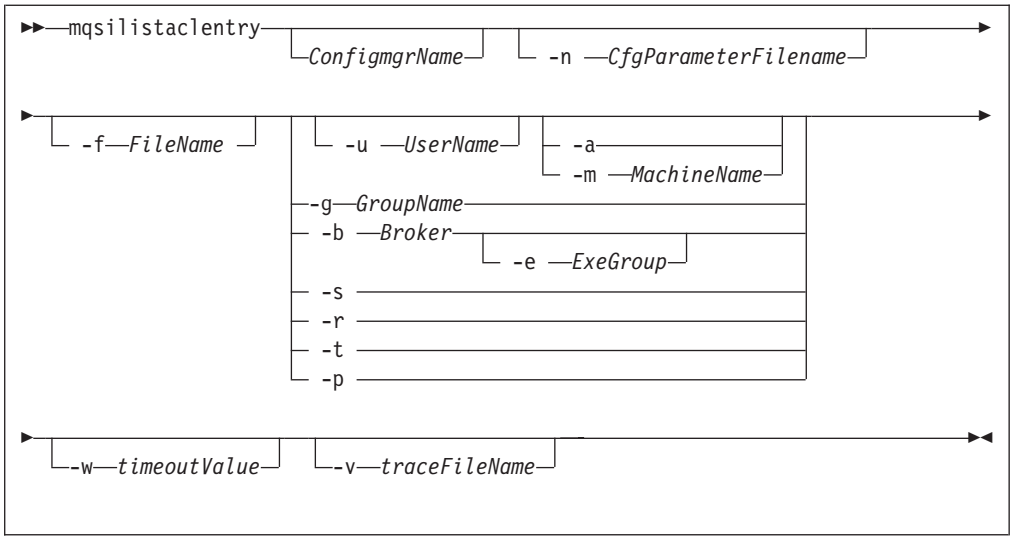
On all platforms, the user ID used to run this command must have full control permissions for the object being displayed.

On Linux and UNIX, the user ID must be a member of mqbrkrs.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. Therefore the commands inherit a Full Control root ACL and you can carry out all operations. If you submit a console command to the Configuration Manager you can change all ACLs for that Configuration Manager.

mqsilistaclentry command - Windows:

Syntax:



Parameters:

ConfigmgrName
(Optional) The name of the Configuration Manager for which the access control lists are to be displayed.

The default name, if this parameter is not specified, is 'ConfigMgr'.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The 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) The specified user may connect to all machines. This option can not be used with **-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 *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.

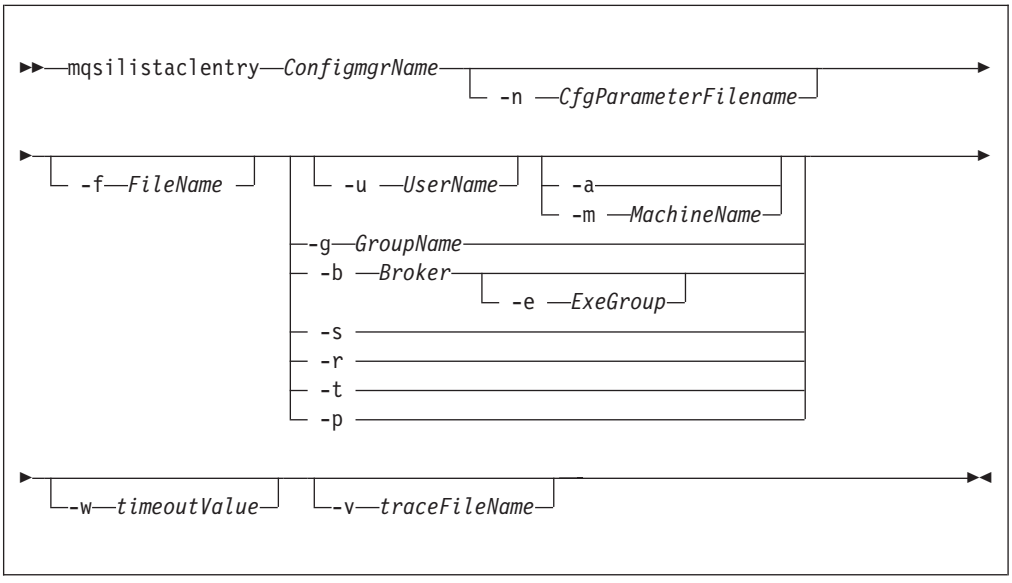
-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

```
mqsilistaclentry CMGR01 -b BROKER01
mqsilistaclentry CMGR01 -e BROKER01\ExeGrp01
mqsilistaclentry CMGR01 -g GROUPA
```

mqsilistaclentry command - Linux and UNIX systems:

Syntax:



Parameters:

ConfigmgrName
(Required) The name of the Configuration Manager for which the access control lists are to be displayed. This parameter must be the first parameter specified and its name is case-sensitive.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The 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) The specified user may connect to all machines. This option can not be used with **-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 *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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

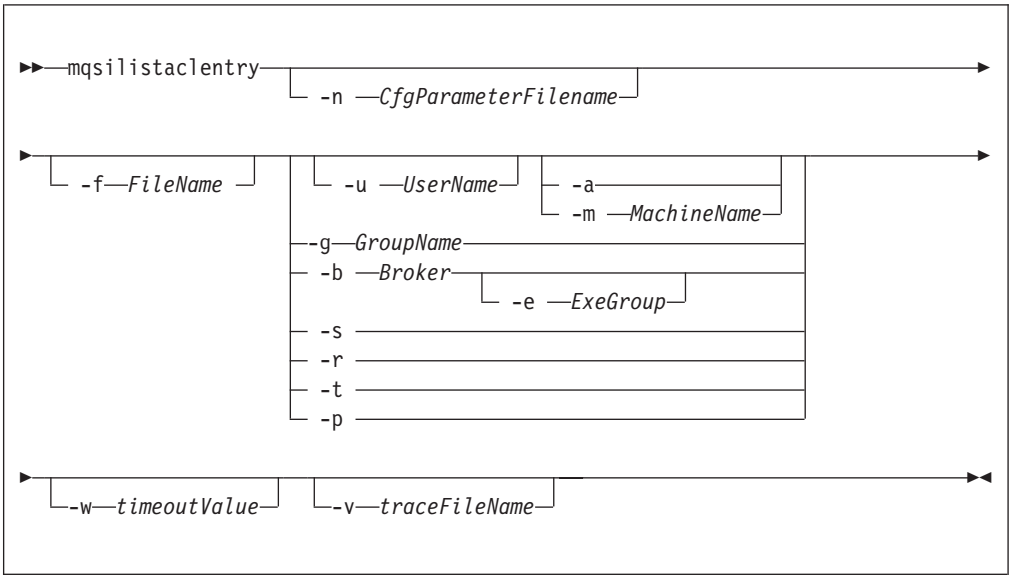
Examples:

```
mqsilistaclentry CMGR01 -b BROKER01
mqsilistaclentry CMGR01 -e BROKER01\ExeGrp01
mqsilistaclentry CMGR01 -g GROUPA
```

mqsilistaclentry command - z/OS:

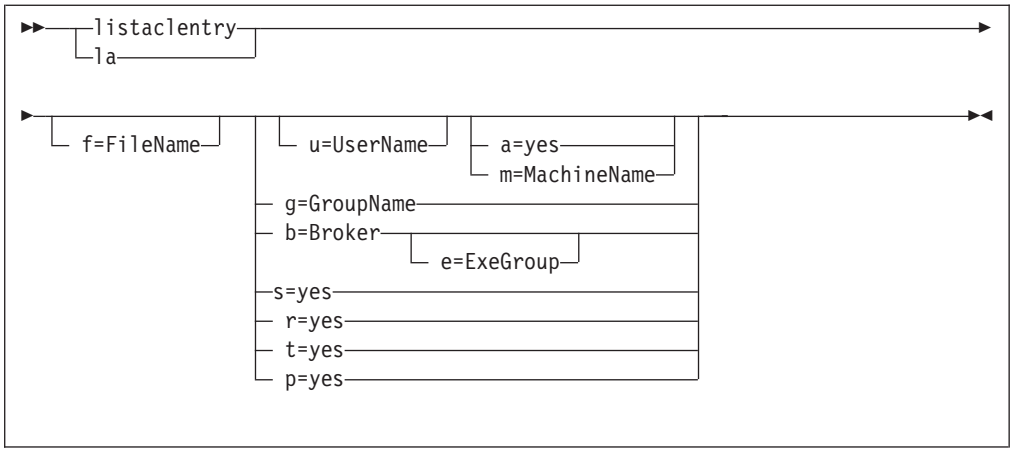
Syntax:

z/OS command - BIPLIACL:



z/OS console command:

Synonym: la



Parameters:

ConfigmgrName
This parameter is implicit because you specify the component that you want to MODIFY.

-n *CfgParameterFilename*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

Remove the statement `encoding="UTF-8"` from the first line of the `.configmgr` file, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-f *FileName*
(Optional) The file from which to import the configuration. The output of the **mqsilistaclentry** command, with output generated using the **-f** option, is the correct format.

-g *GroupName*
(Optional) The 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) The 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) The specified user may connect to all machines. This option can not be used with **-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 *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.

-v *TraceFileName*
(Optional) This parameter sends the internal Configuration Manager Proxy trace to the specified file.

Examples:

```
mqsilistaclentry CMGR01 -g GROUPA
```

For the console form of the command on z/OS you must use a comma between each command option. The following example shows the console version of the preceding example:

```
 /f CMGR01,la g='GROUPA'
```

mqsilistmqpubsub command

Use the mqsilistmqpubsub command to display the status of the WebSphere MQ Publish/Subscribe neighbor brokers to the specified WebSphere Message Broker broker.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPLSMP; see “Contents of the broker PDSE” on page 679

Purpose:

The mqsilistmqpubsub command returns the status of the activity started by a previous join request (see “mqsijoinmqpubsub command” on page 579). 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 Message 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:

Parameters:

BrokerName
(Required) The name of the broker for which you want a list of neighbors.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows XP and Windows Server 2003 systems, no authorization is required.

On Linux and UNIX systems, the user ID must be a member of the mqbrkrs group.

Examples:

If no WebSphere MQ Publish/Subscribe brokers exit, 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 Message Broker broker has three WebSphere MQ Publish/Subscribe neighbors. Two of these neighbors are active and have been successfully joined to the WebSphere Message Broker broker. The third is inactive and is in the process of being joined.

The list of streams that are common to the WebSphere Message 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 Message Broker and WebSphere MQ Publish/Subscribe brokers are both active.

mqsimanagexalinks command

Use the mqsimanagexalinks command to set up links for a supported database on Linux and UNIX systems for XA coordination under the control of WebSphere MQ.

Supported platforms:

- Linux and UNIX systems

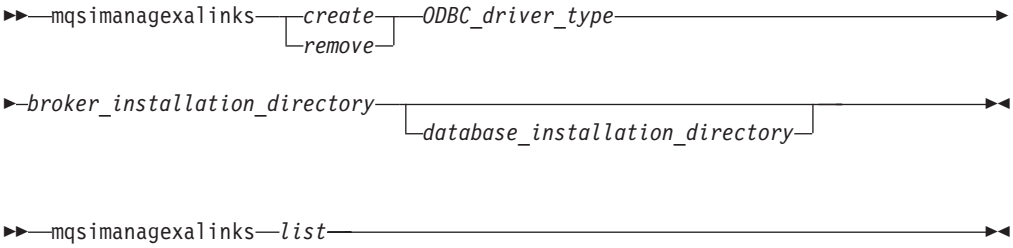
Purpose:

The command sets up or removes the links required by WebSphere MQ to include databases in XA transactions.

You can also use the mqsimanagexalinks command to return a list of supported ODBC drivers.

Run the mqsimanagexalinks command before you create the database that you want the broker to connect to. You can run the command before or after you install your database; specify either the intended or the actual database installation directory where indicated.

Syntax:



Parameters:

create
(Required) Specify that the required links are created.

remove
(Required) Specify that the defined links are removed.

ODBC_driver_type
(Required) The ODBC driver type. See the list command for the list of ODBC drivers that are supported.

This parameter is required if you specify **create**, and is optional if you specify **remove**. If you specify **remove** and omit this parameter, all links are removed.

broker_installation_directory
(Required) The full path to your broker installation.

This parameter is required only if you specify **create**.

database_installation_directory
(Optional). The full path to your database installation.

This parameter is required only if you have created a DB2 or Informix database. This parameter is required only if you specify **create**. If you are using an Informix database, set this parameter to the location of your Informix client installation.

list
(Required) Display the list of supported ODBC driver versions. Use the output to check what values you can specify for **ODBC_driver_type**.

For example, the following output is generated on AIX:

Supported information for the 'create' option

Supported version of the ODBC drivers	Version number for mqsimanagexalinks
DB2 Version 8	DB28

Supported information for the 'create' option

Supported version of the ODBC drivers	Version number for mqsimanagexalinks
DB2 Version 9	DB29
DataDirect Connect for ODBC V5.3	DD53
Informix 64-bit databases	INF64
Informix 32-bit databases	INF32

Supported information for the 'remove' option

Supported version of the ODBC drivers	Version number for mqsimanagexalinks
DB2 Version 8	DB28
DB2 Version 9	DB29
DataDirect Connect for ODBC V5.0	DD50 (on HP-UX on PA-RISC only)
DataDirect Connect for ODBC V5.2	DD52 (not on HP-UX on PA-RISC)
DataDirect Connect for ODBC V5.3	DD53
Informix 64-bit databases	INF64
Informix 32-bit databases	INF32

Links created in `/var/mqm/exits`:

The links that are created are shown later in this section for each supported database. Not all links are created on all platforms; the links created are determined by the database support on each platform. In the links shown, `x` represents the relevant library extension on your platform.

DB2 Version 8

database_install_dir/lib/libdb2.x

database_install_dir/lib/libdb2.x.1

broker_install_dir/sample/xatm/db2swit32

DB2 Version 9

database_install_dir/lib32/libdb2.x

database_install_dir/lib32/libdb2.x.1

broker_install_dir/sample/xatm/db2swit32

Informix

database_install_dir/lib

database_install_dir/lib/esql

For platforms that also support 64-bit mode: *broker_install_dir*/sample/xatm/infswit32

For platforms that do not support 64-bit mode: *broker_install_dir*/sample/xatm/infswit

Oracle

broker_install_dir/ODBC32/V5.3/lib/libUKicu23.*x*

broker_install_dir/ODBC32/V5.3/lib/UKor823.*x*

broker_install_dir/ODBC32/V5.3/lib/UKor8dtc23.*x*

broker_install_dir/ODBC32/V5.3/lib/libodbcinst.*x*

Where *.x* is the library extension on your platform: `.a`, `.so`, or `.sl`.

Sybase

broker_install_dir/ODBC32/V5.3/lib/libUKicu23.*x*
broker_install_dir/ODBC32/V5.3/lib/UKase23.*x*
broker_install_dir/ODBC32/V5.3/lib/UKasedtc23.*x*
broker_install_dir/ODBC32/V5.3/lib/libodbcinst.*x*

Where *.x* is the library extension on your platform: `.a`, `.so`, or `.sl`.

Links created in `/var/mqm/exits64`:

Links that are created for each supported database:

DB2 Version 8

database_install_dir/lib64/libdb2.x
database_install_dir/lib64/libdb2.x.1
broker_install_dir/sample/xatm/db2swit

DB2 Version 9

database_install_dir/lib64/libdb2.x
database_install_dir/lib64/libdb2.x.1
broker_install_dir/sample/xatm/db2swit

Informix

database_install_dir/lib
database_install_dir/lib/esql
broker_install_dir/sample/xatm/infswit

Oracle

broker_install_dir/ODBC64/V5.3/lib/libUKicu23.*x*
broker_install_dir/ODBC64/V5.3/lib/UKora23.*x*
broker_install_dir/ODBC64/V5.3/lib/UKor8dtc23.*x*
broker_install_dir/ODBC64/V5.3/lib/libodbcinst.*x*

Where *.x* is the library extension on your platform: `.a`, `.so`, or `.sl`.

Sybase

broker_install_dir/ODBC64/V5.3/lib/libUKicu23.*x*
broker_install_dir/ODBC64/V5.3/lib/UKase23.*x*
broker_install_dir/ODBC64/V5.3/lib/UKasedtc23.*x*
broker_install_dir/ODBC64/V5.3/lib/libodbcinst.*x*

Where *.x* is the library extension on your platform: `.a`, `.so`, or `.sl`.

Authorization:

The user ID with which you run this command must be root.

Examples:

To create the links required for DB2 Version 9:

```
mqsimanagexalinks create DB29 /opt/IBM/mqsi/V6.1 /opt/IBM/db2/V9.1
```

To create the links required for Informix 64-bit drivers:

```
mqsimanagexalinks create INF64 /opt/IBM/mqsi/V6.1 /informix/client
```

To create the links required for Oracle and Sybase databases (these databases use the DataDirect V5.3 drivers):

```
mqsimanagexalinks create DD53 /opt/IBM/mqsi/V6.1
```

To remove the links required for Oracle and Sybase databases (these databases previously used the DataDirect V5.2 drivers):

```
mqsimanagexalinks remove DD52
```

To list the supported database drivers:

```
mqsimanagexalinks list
```

mqsimigratecomponents command

Use the mqsimigratecomponents command to migrate a component from a previously installed version of the product to another version on the same computer to prepare it for participation in the broker domain of the target version.

Supported platforms:

- Windows.
- Linux and UNIX systems.
- z/OS. Run this command by customizing and submitting BIPMGCMP.

Purpose:

Migrate components to WebSphere Message Broker Version 6.1 from Version 6.0 or from Version 5.0. If you migrate from Version 5.0, you must have Version 5.0.0.4 (Fix Pack 4) or later; earlier releases of the product are not supported for migration.

You can also use this command to return a component from a later version to an earlier one to reverse the effects of forward migration, with one exception; you cannot return a Configuration Manager from Version 6.1 to Version 5.0.

You must run this command from whichever version of the installed product is the later, regardless of whether it is the source version or the target version.

You must have an installation of the product at both target and source versions, with the required component code installed, to issue this command successfully.

Before you start migration, stop any active debug sessions in the Message Broker Toolkit. You cannot migrate message flows that are being debugged.

Specify appropriate options on this command to perform one of the following actions:

- Check that the component is suitable for the required migration, without making any changes (**-c**).
- Move a component to a different version, in full or part (**-s** and **-t**).
- Undo a failed migration step (**-u**).
- Verify that a move has been successful (**-v**).

If you are using the mqsimigratecomponents command to migrate a broker that uses Sybase for its broker database, you must modify the database by performing the following actions:

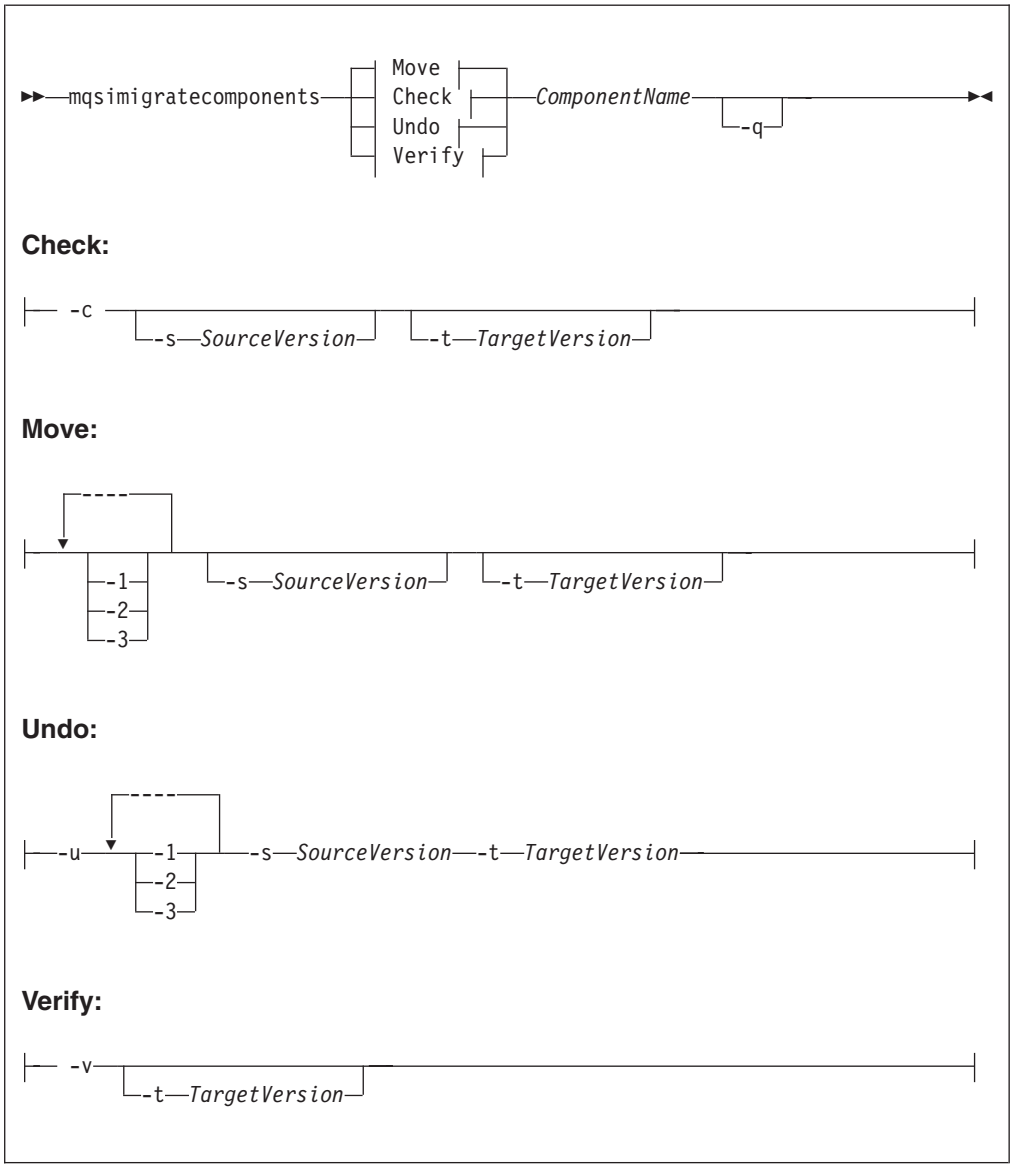
1. Log on to ISQL using a system administrator account.

2. Issue the following series of commands:

```
1> use master
2> go
1> sp_dboption "BROKER1","ddl in tran",TRUE
2> go
Database option 'ddl in tran' turned ON for database 'BROKER1'.
Run the CHECKPOINT command in the database that was changed.
(return status = 0)
1> use BROKER1
2> go
1> checkpoint
2> go
```

where *BROKER1* is the name of the Sybase broker database.

Syntax:



Parameters:

-c

(Optional) Check a specified component before migration, to ensure that:

- The auto-detected version of the broker matches any version specified on the command line.
- If you are migrating from Version 6.1 to Version 5.0, 64-bit execution groups are not supported.
- The database tables accessed in a previous release do not contain any rows that are incorrectly indexed.

You can check a running component. The check does not affect the component, apart from a slight impact on performance. On Linux and UNIX systems, you must migrate the ODBC configuration files (the files in which you have defined the data sources which are referenced by `ODBCINI` and `ODBCINI64`) before you run the check, because the checking command must be able to access the broker database.

The check command either succeeds or fails, and prints a message about whether the migration will succeed, but no modifications are made during the process.

-v

(Optional) Check a specified component after migration, to ensure that:

- The registry is in the correct format for the specified version.
- The correct queues exist for the specified version.
- The correct database tables exist for the specified version.

-q

(Optional) Print fewer status messages during the operation.

-1

(Optional) Do only registry and file system work.

- When you migrate to Version 6.1, use the **-1** parameter before the **-2** or **-3** parameters.
- When you migrate backwards from Version 6.1 to a previous version, use the **-2** or **-3** parameters before the **-1** parameter.

-2

(Optional) Do only WebSphere MQ work.

-3 (Optional) Do only database work.

If a broker that you are migrating shares a database schema with another broker, warning message BIP8678 is issued and the check fails. In this case, all the brokers that share a database schema must be migrated together.

1. Stop all the brokers that share the database schema.
2. Migrate the first broker. This action migrates the database tables for all brokers, as well as the file system and registry, and WebSphere MQ definitions for that broker only; for example:

   ```
   mqsimigratecomponents FIRSTBROKER -t 6.1.0.0
   ```

3. Migrate the file system and registry, and WebSphere MQ parts of each of the other brokers; the database part has already been migrated. Use the **-1** and **-2** parameters to do this, either in one step or two steps:
   - In one step:

     ```
     mqsimigratecomponents BROKERB -1 -2
     ```

   - In two steps:

```
mqsimigratecomponents BROKERB -1
mqsimigratecomponents BROKERB -2
```

-u (Optional) Undo a failed migration step; you must also specify at least one of **-1**, **-2**, or **-3**. Use this option only when migration has failed, and also failed to auto-recover (for example, if a failure occurs during split migration).

-s *SourceVersion*

(Optional) The previous version of the component.

- If not specified, this value is detected automatically.
- When you perform split migration to Version 6.1, the **-s** parameter is mandatory after you run the mqsimigratecomponents command with the **-1** parameter, as shown in the split migration example.
- See "Purpose" on page 597 for the restrictions to the version numbers of the product that are supported.

-t *TargetVersion*

(Optional) The destination version of the component.

- If not specified, this value is assumed to be the current version.
- When you perform split migration from Version 6.1 to a previous version, the **-t** parameter is mandatory. This requirement is shown in the split migration example.
- See "Purpose" on page 597 for the restrictions to the version numbers of the product that are supported.

ComponentName

(Required) The name of the component to migrate.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

The mqsimigratecomponents command updates your registry and file system, WebSphere MQ definitions, and database definitions.

If the user ID used to run this command does not have the authority to perform all of these steps, you can run the command one part at a time. Different users can run the part for which they are authorized in order to achieve the overall result. This approach is referred to as *split migration*, and is performed by using the **-1**, **-2**, and **-3** parameters.

If you run single-step migration, your user ID must have the ability to:

- Write to the registry and the file system for the product
- Modify queue definitions

- Modify databases associated with the component

If you run split migration, your user ID 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 write to the registry and the file system for the product
- **-2** requires the ability to modify queue definitions
- **-3** requires the ability to modify databases associated with the component

Responses:

This command can produce a large number of possible responses, depending on the results of the various operations. This command differs from other commands in the way it produces messages: they are displayed when they are generated, rather than being reported in a batch at the end of the program.

When you migrate database tables, z/OS produces more output than distributed systems. Use the **-q** parameter to reduce the number of messages displayed.

Examples:

The following example shows a split migration from Version 5.0 to Version 6.1:

```
mqsimigratecomponents BROKER -1
mqsimigratecomponents BROKER -s 5.0.0.5 -2
mqsimigratecomponents BROKER -s 5.0.0.5 -3
```

The following example shows a pre-migration check followed by a migration of BROKER1 from Version 6.0 to Version 6.1 on Windows XP:

```
mqsimigratecomponents -c BROKER1
BIP8849I: Broker 'BROKER1' (Version 6.0) with Queue Manager 'QM' and
Data Source 'BROKERDB' specified for migration.
BIP8680I: Pre-migration check succeeded.
BIP8071I: Successful command completion.

mqsimigratecomponents BROKER1
BIP8849I: Broker 'BROKER1' (Version 6.0) with Queue Manager 'QM' and
Data Source 'BROKERDB' specified for migration.
BIP8768I: Finished registry migration for component 'BROKER1'.
BIP8654I: Moving filesystem artefacts from
'C:\Documents and Settings\All Users\Application Data\IBM\MQSI' to
'C:\Documents and Settings\Allsers\Application Data\IBM\MQSI'
BIP8670I: Database migration started
BIP8663I: Creating temporary new tables
BIP8664I: Migrating from existing tables to temporary new tables
BIP8665I: Dropping existing tables
BIP8666I: Creating new tables
BIP8667I: Copying all rows from temporary new tables to new tables
BIP8668I: Dropping temporary new tables
BIP8669I: Database migration successful
WebSphere MQ queue manager running.
BIP8785I: Starting WebSphere MQ queue migration for component 'BROKER1'.
The setmqaut command completed successfully.
BIP8786I: Created WebSphere MQ queue 'SYSTEM.BROKER.EDA.COLLECTIONS'
The setmqaut command completed successfully.
BIP8786I: Created WebSphere MQ queue 'SYSTEM.BROKER.EDA.EVENTS'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.AGGR.REQUEST'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.AGGR.CONTROL'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.AGGR.REPLY'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.AGGR.TIMEOUT'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.AGGR.UNKNOWN'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.TIMEOUT.QUEUE'
```

```
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.ADMIN.QUEUE'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.EXECUTIONGROUP.QUEUE'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.WS.INPUT'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.WS.REPLY'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.WS.ACK'
BIP8787I: Cleared WebSphere MQ queue 'SYSTEM.BROKER.IPC.QUEUE'
BIP8789I: Finished WebSphere MQ queue migration for component 'BROKER1'.
BIP8071I: Successful command completion.
```

The following example shows a migration from Version 6.1 back to Version 6.0:

```
mqsimigratecomponents BROKER -t 6.0.0.3
```

mqsimode command

Use the mqsimode command to configure and retrieve operation mode information for one or more brokers in a domain.

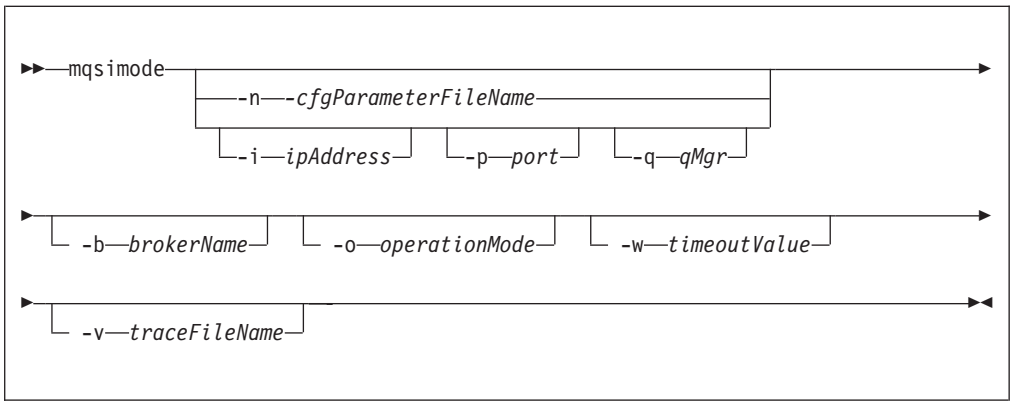
Supported platforms:

- Windows
- Linux and UNIX systems

Purpose:

Use the mqsimode command to change the operation mode for one or more brokers in a domain, or to retrieve information about the mode in which the broker is currently working.

Syntax:



Parameters:

-n *-cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the

Configuration Manager. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value 1414 is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used.

-b *brokerName*
(Optional) This parameter specifies the name of the broker for which you want to change the mode. If you do not specify the **-b** parameter, the command applies to all brokers in the domain. You cannot specify either multiple **-b** parameters in a single command, or insert the name of more than one broker into a single **-b** parameter.

-o *operationMode*
(Optional) This parameter sets the mode of the target broker or brokers. Valid values are enterprise (the full edition), starter (Starter Edition), and adapter (Remote Adapter Deployment mode). If you do not specify the **-o** parameter, the command displays the mode in which the broker is running.

-w *timeoutValue*
(Optional) This parameter is the time in seconds that the utility waits to ensure that the command completed; the default value is 60.

-v *traceFileName*
(Optional) This parameter sends internal debug trace information to the specified file.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all supported platforms, the user ID that is used to run this command to change the mode must have sufficient authority defined in the Configuration Manager. Therefore, the user ID must be a member of both the mqm and mqbrkrs groups, and must have full administration rights to the Configuration Manager.

Examples:

Running the mqsimode command specifying both the -o and -b parameters

When you run the mqsimode command using both the **-o** and **-b** parameters, it sets the mode for a given broker, and reports any errors. For example, if you run the following mqsimode command, and your broker is in the following state:

- *Broker_Name* is the name of your broker.
- Your broker contains too many execution groups for the Starter Edition mode.

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER –b Broker_Name
 –o starter
```

You receive the following messages:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1809: Deploying 'Broker_Mode' mode on broker 'Broker_Name'...
BIP1805: The mode for broker 'Broker_Name' has been changed to 'starter'.
BIP1821: WARNING: Broker 'Broker_Name' is in 'starter' mode but has '3' execution
 groups, which exceeds the allowed maximum for this mode.
BIP8229: The command completed with the following number of warnings: 1.
```

Running the mqsimode command specifying the -b parameter, but not the -o parameter

When you run the mqsimode command specifying the **-b** parameter, but not the **-o** parameter, you receive a report about the mode being used by your broker, and a report about all mode violations. For example, if you run the following mqsimode command, and your broker is in the following state:

- *Broker_Name* is the name of your broker.
- Your broker is in Starter Edition mode.
- Your broker has no violations.

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER –b Broker_Name
```

You receive the following messages:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1807: Discovering mode information from broker 'Broker_Name'...
BIP1802: Broker 'Broker_Name' is in 'starter' mode.
BIP8071: Successful command completion.
```

Running the mqsimode command specifying the -o parameter, but not the -b parameter

When you run the mqsimode command with the **-o** parameter but without the **-b** parameter, the mode is set for all of the brokers in the domain, and you receive a report about any errors. For example, if you run the following mqsimode command to change your brokers to the Remote Adapter Deployment mode, and your brokers are in the following state:

- *Broker_Name1* to *Broker_Name8* are the names of your brokers.
- *Message_Flow* is the name of your message flow.
- *Execution_Group* is the name of your execution group.
- The command changes brokers *Broker_Name1* to *Broker_Name8* to the Remote Adapter Deployment mode, but broker *Broker_Name6* contains a node *Node_Type* that is not valid in this mode.

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER –o adapter
```

You receive the following messages:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1805: The mode for broker 'Broker_Name1' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name2' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name3' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name4' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name5' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name6' has been changed to 'adapter'.
BIP1823: WARNING: Broker 'Broker_Name6' has a message flow called 'Message_Flow'
 in execution group 'Execution_Group', which contains one or more nodes that are not
 valid in this mode: Node_Type.
BIP1805: The mode for broker 'Broker_Name7' has been changed to 'adapter'.
BIP1805: The mode for broker 'Broker_Name8' has been changed to 'adapter'.
BIP8229: The command completed with the following number of warnings: 1.
```

Running the mqsimode command without the -o and -b parameters

When you run the mqsimode command without the **-o** and **-b** parameters, you receive reports about the modes in use in a given domain and all mode violations. For example, if you run the following mqsimode command, and your brokers are in the following state:

- *Broker_Name1* to *Broker_Name8* are the names of your brokers.
- *Message_Flow* is the name of your message flow.
- *Execution_Group* is the name of your execution group.
- Brokers *Broker_Name1* to *Broker_Name3* are in Starter Edition mode. Broker *Broker_Name2* contains too many execution groups for the Starter Edition mode. Broker *Broker_Name3* contains too many message flows for the Starter Edition mode.
- Brokers *Broker_Name4* and *Broker_Name5* are in Trial Edition mode. The trial period for broker *Broker_Name4* has expired. The trial period for broker *Broker_Name5* expires on 6th September 2009.
- Brokers *Broker_Name6* and *Broker_Name7* are in Remote Adapter Deployment mode. Broker *Broker_Name6* contains a node *Node_Type* that is not valid in the Remote Adapter Deployment mode.
- Broker *Broker_Name8* is in enterprise mode.

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER
```

You receive messages similar to these messages:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1807: Discovering mode information from broker 'Broker_Name1'...
BIP1802: Broker 'Broker_Name1' is in 'starter' mode.
BIP1807: Discovering mode information from broker 'Broker_Name2'...
BIP1802: Broker 'Broker_Name2' is in 'starter' mode.
BIP1821: WARNING: Broker 'Broker_Name2' is in 'starter' mode but has '3' execution
 groups, which exceeds the allowed maximum for this mode.
BIP1807: Discovering mode information from broker 'Broker_Name3'...
BIP1802: Broker 'Broker_Name3' is in 'starter' mode.
BIP1822: WARNING: Broker 'Broker_Name3' is in 'starter' mode but has '17' message
 flows deployed, which exceeds the allowed maximum for this mode.
BIP1807: Discovering mode information from broker 'Broker_Name4'...
BIP1824: WARNING: The trial period for broker 'Broker_Name4' expired on '06 September 2009'.
BIP1807: Discovering mode information from broker 'Broker_Name5'...
BIP1803: The trial period for broker 'Broker_Name5' expires on '06 Jun 2008'.
BIP1807: Discovering mode information from broker 'Broker_Name6'....
BIP1802: Broker 'Broker_Name6' is in 'adapter' mode.
BIP1823: WARNING: Broker 'Broker_Name6' has a message flow called 'Message_Flow'
in execution group 'Execution_Group', which contains one or more nodes that are not valid
 in this mode: Node_Type.
BIP1807: Discovering mode information from broker 'Broker_Name7'...
```

```
BIP1802: Broker 'Broker_Name7' is in 'adapter' mode.
BIP1807: Discovering mode information from broker 'Broker_Name8'...
BIP1802: Broker 'Broker_Name8' is in 'enterprise' mode.
BIP8229: The command completed with the following number of warnings: 4.
```

Running the mqsimode command specifying the -o parameter against a broker that is running a version before V6.1

When you run the **-o** parameter against a broker that is running a version before V6.1 you receive a report with an appropriate message. For example, where *Broker_Name* is the name of your broker:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1808: Broker 'Broker_Name' is not at the required software level to
 change the operation mode.
 BIP8229: The command completed with the following number of warnings: 1.
```

Example: Changing the Trial Edition to the full edition:

You want to convert all of your brokers from the Trial Edition mode to the full edition (enterprise mode).

Contact your IBM representative to upgrade your license, and change the brokers that you have created to conform to your new license.

Before, or after, the Trial Edition expires complete the following steps:

1. Open a command prompt.
   - Linux On Linux, run the mqsiprofile command to initialize the command environment.
   - Windows On Windows, click **Start** → **Programs** → **IBM WebSphere Message Brokers 6.1** → **Command Console** to open a command console.
2. Run the following mqsimode command:

   ```
   mqsimode –i localhost –p 2414 –q WBRK61_DEFAULT_CONFIGURATION_MANAGER –o enterprise
   ```

   When this command completes successfully, all brokers are made fully functional, but all new brokers are still created in trial mode by default.
3. To ensure that all new brokers start in enterprise mode, uninstall the trial package and reinstall using the full package; see “Moving from Trial Edition” on page 236.

Example: Changing the operation mode of your broker:

You want to convert from the Starter Edition (starter mode) to the full package (enterprise mode).

Contact your IBM representative to upgrade your license, and change the brokers that you have created to conform to your new license.

1. Open a command prompt.
   - Linux On Linux, run the mqsiprofile command to initialize the command environment.
   - Windows On Windows, click **Start** → **Programs** → **IBM WebSphere Message Brokers 6.1** → **Command Console** to open a command console.
2. Run the following mqsimode command to change the mode of your broker from starter to enterprise (where *Broker_Name* is the name of your broker):

```
mqsimode –i localhost –p 1414 –q WBRK61_DEFAULT_QUEUE_MANAGER –b Broker_Name
 –o enterprise
```

The following messages are displayed:

```
BIP1044: Connecting to the Configuration Manager's queue manager...
BIP1045: Connecting to the Configuration Manager...
BIP1809: Deploying 'enterprise' mode on broker 'Broker_Name'...
BIP1805: The mode for broker 'Broker_Name' has been changed to 'enterprise'.
```

mqsireadlog command

Use the mqsireadlog command to retrieve trace records for the specified component.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPRELG; see "Contents of the broker PDSE" on page 679

Purpose:

The mqsireadlog 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.

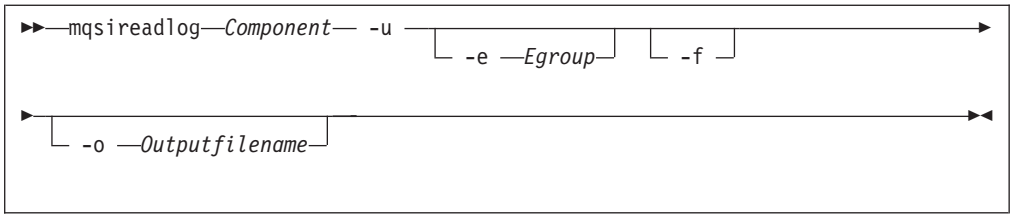
On HP-UX, set the `size` parameter of the mqsichangetrace command to be less than 500 MB because the size of the XML generated files is often half as much again as the original trace file, and setting the value of the `size` parameter to be greater than 500 MB can cause problems.

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).

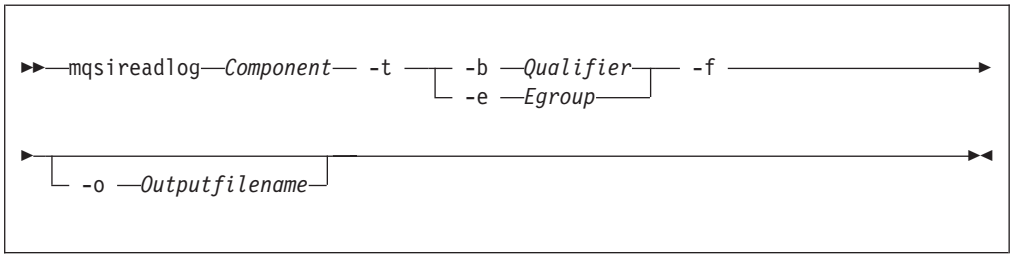
If you specify a broker, or any of its resources (execution group or message flow) you must have deployed them before you can start trace and read the log files.

Syntax:

User trace:



Service trace:



Parameters common to user trace and service trace:

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, and on z/OS).

-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 path name, 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 by using the mqsiformatlog command. If you do not specify a filename, the contents of the log are written to stdout. Use a file extension of `.xml`, which represents the format of the data.

-f (Optional for User trace; required for Service trace). 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.

If you specify this option, stop tracing (by using mqsichangetrace) before you run 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. Specify **-m safe** on the mqsichangetrace command to reduce the risk of partial records. 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 by using a text editor before running the mqsiformatlog command, as partial records cannot be read by the format command.

Additional parameter exclusive to user trace:

-u (Required) Read the log contents from the user trace log. This option is valid **only** if you select the `broker` component.

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 Message 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
```

Run this command only when directed to do so by a WebSphere Message Broker error message, or when instructed to do so by your IBM Support Center.

Qualifier	Component= *broker_name*	Component= *ConfigMgr_name*	Component= UserNameServer	Component= workbench	Component= utility	Component= Database instance manager(n)
mqsichangebroker	x					
mqsichangeconfigmgr		x				
mqsichangedbimgr					x	
mqsichangeflowmonitoring	x					
mqsichangeflowstats	x					
mqsichangeflowuserexits	x					
mqsichangeproperties	x					
mqsichangetrace	x	x	x			
mqsichangeusernameserver			x			
mqsiclearmqpubsub	x					
mqsicreateaclentry					x	

Qualifier	Component= *broker_name*	Component= *ConfigMgr_name*	Component= UserNameServer	Component= workbench	Component= utility	Component= Database instance manager(n)
mqsicreatebroker	x					
mqsicreateconfigmgr		x				
mqsicreateconfigurableservice	x					
mqsicreatedb						x
mqsicreateusernameserver			x			
mqsicvp	x	x				
mqsideleteaclentry					x	
mqsideletebroker	x					
mqsideleteconfigmgr		x				
mqsideleteconfigurableservice	x					
mqsideletedb						x
mqsideleteusernameserver			x			
mqsideploy		x				
mqsiformatlog[1]					x	
mqsijoinmqpubsub		x				
mqsilist[2]		x			x	
mqsilist	x					
mqsilistaclentry					x	
mqsilistmqpubsub		x				
mqsimigratecomponents	x	x	x			
mqsireadlog	x	x	x		x	
mqsireload	x					
mqsireloadsecurity	x					
mqsireportflowmonitoring	x					
mqsireportflowstats	x					
mqsireportflowuserexits	x					
mqsireportproperties	x					
mqsireporttrace		x		x		
mqsisetdbparms	x					
mqsistart	x	x	x			
mqsistop	x	x	x			
agent	x	x	x			
service	x	x	x			
workbench				x		
httplistener	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 the mqsichangetrace command.
2. If you run this command 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 the mqsichangetrace command.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all 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 run the command must be a member of the mqbrkrs group

On Linux and UNIX platforms, the user ID must be a member of the mqbrkrs group. If the `-f` flag is specified, the user ID used to run this command must also have access to the trace file.

Examples:

Retrieve the user trace for broker `WBRK_BROKER`:

```
mqsireadlog WBRK_BROKER -u -e default -o trace.xml
```

Retrieve the service trace for component `ConfigMgr`:

```
mqsireadlog ConfigMgr -t -b agent -f -o trace.xml
```

Retrieve service trace for utility mqsiformatlog:

```
mqsireadlog utility -t -b agent -f -o trace.xml
```

You can format the log file (`trace.xml` in the above examples) by using the command mqsiformatlog, or view it using an XML editor or viewer.

mqsireload command

Use the mqsireload command to request the broker to stop and restart execution groups.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS - as a console command

Purpose:

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 if you use the default form of this command to reload all execution groups you ensure state and data integrity is preserved.

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, therefore review any long-running message flows before running this command.

If you have included a user-defined 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 user-defined nodes and parsers are re-accessed and reacquired. However, it is better programming practice to ensure that user-defined nodes and parsers provide their own mechanism to reload persistent state and data dynamically, and do not rely on the use of this command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireload command - Windows, Linux, and UNIX systems"
- "mqsireload command - z/OS" on page 613

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

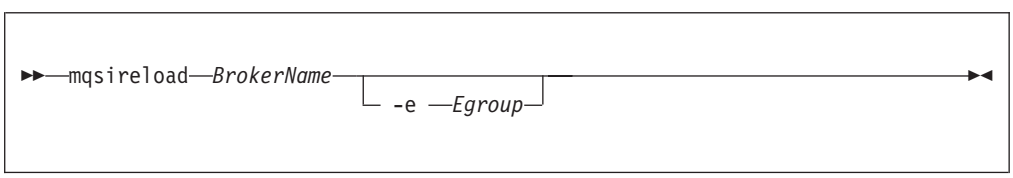
On Windows, Linux, and UNIX systems the user ID used to issue the command must be a member of the group mqbrkrs.

Responses:

No additional responses are returned.

mqsireload command - Windows, Linux, and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required) 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.

Examples:

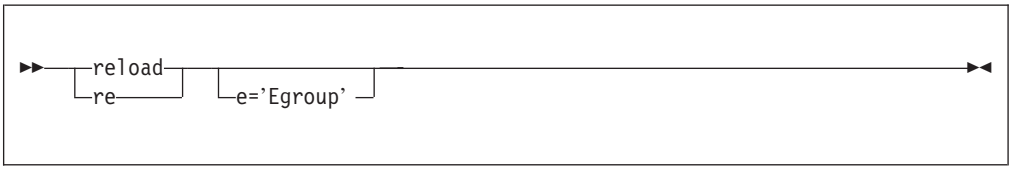
```
mqsireload broker1
```

mqsireload command - z/OS:

Syntax:

z/OS console command:

Synonym: re



Parameters:

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.

Examples:

```
F MQP1BRK,re e='EgName'
```

mqsireloadsecurity command

Use the mqsireloadsecurity command to force the immediate expiry of some or all of the entries in the security cache.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPRLSEC; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Entries in the security cache are valid for a specified length of time, after which the entries are marked as 'expired'. When an entry is marked as expired, it must be re-authenticated with the security provider before it can be reused, and its expiry

time must be reset. If re-authentification fails, the entry remains marked as expired. All entries in the security cache marked as expired are removed when the next sweep of the cache is performed.

Use the mqsichangeproperties command to set the time for which entries in the cache are valid, and also the value for the sweep of the security cache. When the entries in the security cache have expired, you must re-authenticate them.

Select the appropriate link for details of this command on the operating system that is used by your enterprise:

- "mqsireloadsecurity command - Windows, Linux, and UNIX systems"
- "mqsireloadsecurity command - z/OS" on page 616

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must be a member of the group mqbrkrs.

On z/OS, the user ID used to run this command must be a member of a group that has read and write access to the component directory.

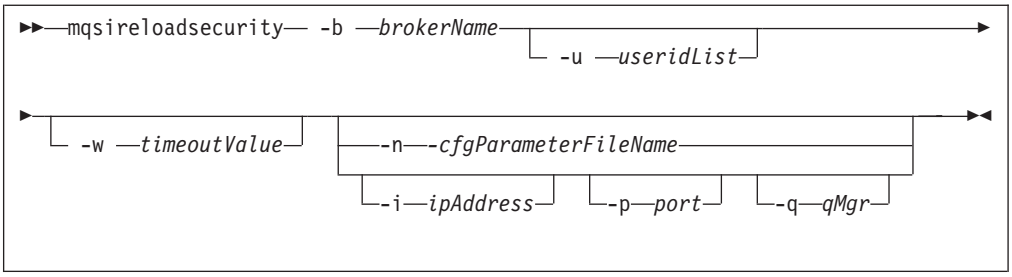
Responses:

No additional responses are returned.

mqsireloadsecurity command - Windows, Linux, and UNIX systems:

Use the mqsireloadsecurity command on Windows, Linux, and UNIX to force the immediate expiry of some or all of the entries in the security cache.

Syntax:



Parameters:

brokerName
(Required) The name of the broker to which the request is sent. If you specify only this option, all entries in the security cache are reloaded.

-u *useridList*
(Optional) This parameter reloads all entries in the security cache for the specified list of users (separated by colons). If you do not specify this parameter, all entries in the security cache are reloaded. For cached certificates, this value is the common name element value.

-w *timeoutValue*
(Optional) This parameter specifies the time in seconds that the utility waits for the broker to reply before returning control to the command line.

-n *-cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value `1414` is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value `""` (a pair of quotation marks) is used.

Examples:

Reload the cache for all users on the specified broker:

```
mqsireloadsecurity -b a_broker
```

Reload the cache for a single user on the specified broker:

```
mqsireloadsecurity -b a_broker -u user1
```

Reload the cache for a list of users on the specified broker, and wait for five seconds before returning:

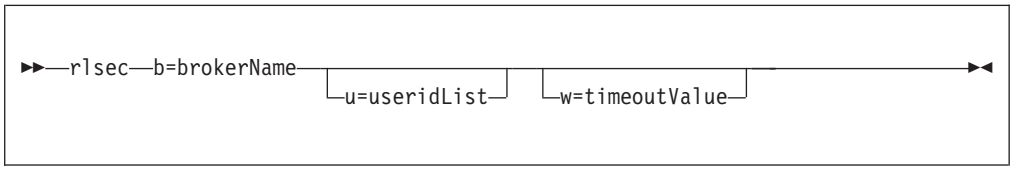
```
mqsireloadsecurity -b a_broker -u user1:user2:user3 -w 5
```

mqsireloadsecurity command - z/OS:

Use the mqsireloadsecurity command on z/OS to force the immediate expiry of some or all of the entries in the security cache.

Syntax:

z/OS console command:



Parameters:

b=*brokerName*
(Required) The name of the broker to which the request is sent. If you specify only this option, all entries in the security cache are reloaded.

u= *useridList*
(Optional) Reload all entries in the security cache for the specified list of users (separated by colons). If you do not specify this parameter, all entries in the security cache are reloaded. For cached certificates, this value is the common name element value.

w= *timeoutValue*
(Optional) This parameter specifies the time in seconds that the utility waits for the broker to reply before returning control to the command line.

Examples:

Reload the cache for all users on the specified broker:

```
F MQP1BRK, b=rlsec
```

Reload the cache for a single user on the specified broker:

```
F MQP1BRK, b=rlsec u=user1
```

Reload the cache for a list of users on the specified broker, and wait for five seconds before returning:

```
F MQP1BRK, b=rlsec u=user1:user2:user3 w=5
```

mqsireportbroker command

Use the mqsireportbroker command to display broker registry entries.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command either as a console command, or by customizing and submitting BIPRPBK; see “Contents of the broker PDSE” on page 679

Purpose:

You can use the mqsireportbroker command to view the property values set when you create or change the broker.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireportbroker command - Windows, Linux, and UNIX systems" on page 618
- "mqsireportbroker command - z/OS" on page 620

Usage notes:

The following rules apply on the report:

- You can run this command if the broker is stopped or running.
- If values for a password are set, eight asterisks are displayed.
- Units are shown in the output for all numeric values.

Responses:

The following table shows the output returned when you run the mqsireportbroker command.

Property (Units)	Notes
brokerName	1
Service userId	2, 3, 4
Service password	2, 3, 4
Queue manager	1
Broker database name or (z/OS only) DB2Location	1
Broker database userId or (z/OS only) DB2TableOwner	1
Broker database password	2, 3
User Name Server queue manager	2
Pubsub access control - true	2
Pubsub access control - false	5
Work path	1
Trusted (fathpath) queue manager application - true	2, 3
Trusted (fathpath) queue manager application - false	3, 5
Pubsub migration - true	1, 3
User lil path	2
Configuration change timeout (seconds)	2
Internal configuration timeout (seconds)	2
HTTP listener port	2
Statistics major interval (minutes)	2
LDAP principal	2
LDAP credentials	2
ICU converter path	2
User exit path	2

Property (Units)	Notes
Operation mode	2, 3
User lil path64	2, 3, 6
User exit path64	2, 3, 6
Fixpack capability level	5
Active user exits	5
Broker UUID	7
Install path	8
Process id	7

Notes:

1. This value is set by the mqsicreatebroker command.
2. This value is set by the mqsicreatebroker command or later modified by the mqsichangebroker command.
3. This option is applicable only on distributed systems.
4. This option is not relevant on Windows, because this value is stored in the service.
5. This option can be set only by the mqsichangebroker command.
6. This option is not applicable on 32-bit only platforms.
7. This value is generated automatically when the broker is started.
8. You cannot set this value by using a command.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

On z/OS systems, the user ID used to run this command must be a member of a group that has READ access to the component directory.

mqsireportbroker command - Windows, Linux, and UNIX systems:

Use the mqsireportbroker command to display broker registry entries.

Syntax:

Parameters:

brokerName
(Required) The label of the broker for which registry entries are to be reported.

Examples: Windows

Display registry entries of the specified broker on Windows:

```
mqsireportbroker SOAPBR
```

The following output is returned:

```
BIP8927I: Broker Name 'SOAPBR'
Install path = 'C:\Program Files\IBM\MQSI\6.1.0.3'
Work path = 'C:\Documents and Settings\All Users\Application Data\IBM\MQSI'
Broker UUID = '91c31c2a-1a01-0000-0080-d9cfce7fc2e8'
Process id = '6928'
Queue manager = 'SOAPQM'
User Name Server queue manager = 'UNSQM'
Broker database name = 'DATAFLOW'
Broker database userId = 'db2admin'
Broker database password = '********'
User lil path = 'C:\Documents and Settings\All Users\Application Data\lilPath'
User exit path = 'C:\Documents and Settings\All Users\Application Data\userExits'
Active user exits = 'activeUE'
LDAP principal = 'ldapuser'
LDAP credentials = '********'
ICU converter path = 'C:\Documents and Settings\All Users\Application Data\converters'
HTTP listener port = '8081'
Pubsub migration = 'false'
Pubsub access control = 'true'
Trusted (fathpath) queue manager application = 'true'
Configuration change timeout = '320' seconds
Internal configuration timeout = '70' seconds
Statistics major interval = '61' minutes
Operation mode = 'enterprise'
Fixpack capability level = 'all' (effective level '6.1.0.3')
Broker registry format = 'v6.1'
```

Linux UNIX

Display registry entries of the specified broker on Linux:

```
mqsireportbroker SOAPBR
```

The following output is returned:

```
BIP8926I: Broker Name 'SOAPBR'
Install path = 'usr/lpp/ibm/mqsi/6.1.0.3/'
Work path = '/var/mqsi'
Broker UUID = '91c31c2a-1a01-0000-0080-d9cfce7fc2e8'
Process id = '0'
Service userId = 'db2admin'
Service password = '********'
Queue manager = 'SOAPQM'
User Name Server queue manager = ''
Broker database name = 'DATAFLOW'
```

```
Broker database userId = 'db2admin'
Broker database password = '********
User lil path = ''
User lil path64 = ''
User exit path = ''
User exit path64 = ''
Active user exits = ''
LDAP principal = ''
LDAP credentials = ''
ICU converter path = ''
HTTP listener port = '8080'
Pubsub migration = 'false'
Pubsub access control = 'false'
Trusted (fathpath) queue manager application = 'false'
Configuration change timeout = '300' seconds
Internal configuration timeout = '60' seconds
Statistics major interval = '60' minutes
Operation mode = ''
Fixpack capability level = '' (effective level '6.1.0.2')
Broker registry format = 'v6.1'
```

mqsireportbroker command - z/OS:

Use the mqsireportbroker command to display broker registry entries.

Syntax:

Parameters:

brokerName
(Required) The label of the broker for which registry entries are to be reported.

Examples: z/OS

Display registry entries of the specified broker on z/OS:

```
mqsireportbroker MQ81BRK
```

The following output is returned:

```
BIP8928I: Broker Name 'MQ81BRK'
Install path = '/usr/lpp/ibm/mqsi/6.1.0.3/'
Work path = '/u/wmqi81/broker'
Broker UUID = '91c31c2a-1a01-0000-0080-d9cfce7fc2e8'
Process id = '6928'
Queue manager = 'MQ81'
User Name Server queue manager = 'MQ81'
DB2 location name = 'DHS1'
DB2 table owner = 'wmqi81'
Broker database password = '********'
User lil path = '/u/wmqi81/broker/userlilpath'
User exit path = '/u/wmqi81/broker/userexitpath'
Active user exits = 'activeUE'
LDAP principal = 'wmqi81'
LDAP credentials = '********'
ICU converter path = '/u/wmqi81/broker/converterPath'
HTTP listener port = '8080'
Pubsub access control = 'true'
Configuration change timeout = '320' seconds
```

```
Internal configuration timeout = '70' seconds
Statistics major interval = '61' minutes
Fixpack capability level = 'all' (effective level '6.1.0.3')
Broker registry format = 'v6.1'
```

mqsireportconfigmgr command

Use the mqsireportconfigmgr command to display Configuration Manager registry entries.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways; as a console command, or by customizing and submitting BIPRPCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

You can use the mqsireportconfigmgr command to view the property values set when you create or change the Configuration Manager.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireportconfigmgr command - Windows" on page 622
- "mqsireportconfigmgr command - Linux, and UNIX systems" on page 623
- "mqsireportconfigmgr command - z/OS" on page 624

Usage notes:

The following rules apply on the report:

- You can run this command if the Configuration Manager is stopped or running.
- If values for a password are set, eight asterisks are displayed.
- Units are shown in the output for all numeric values.

Responses:

The following table shows the output returned when you run the mqsireportconfigmgr command.

Property (Units)	Notes
ConfigmgrName	1
Service userId	2, 3, 4
Service password	2, 3, 4
Queue manager	1
Migration database name	1, 3
Migration database userId	1, 3
Migration database password	2, 3
User Name Server queue manager	2
Work path	1
JVM heap size (MB)	2
Install path	5

Property (Units)	Notes
Process id	6

Notes:

1. This value is set by the mqsicreateconfigmgr command.
2. This value is set by the mqsicreateconfigmgr command or later modified by the mqsichangeconfigmgr command.
3. This option is applicable only on distributed systems.
4. This option is not relevant on Windows because this value is stored in the service.
5. You cannot set this value by using a command.
6. This value is generated automatically when the Configuration Manager is started.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

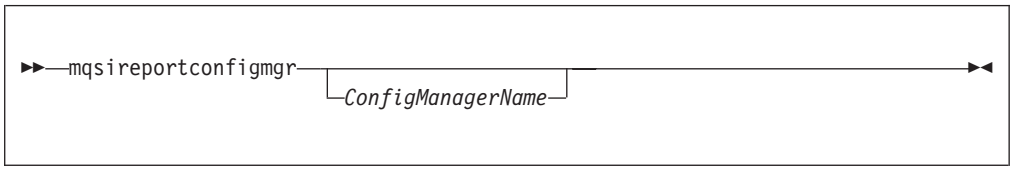
On Linux and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

On z/OS systems, the user ID used to run this command must be a member of a group that has READ access to the component directory.

mqsireportconfigmgr command - Windows:

Use the mqsireportconfigmgr command to display Configuration Manager registry entries.

Syntax:



Parameters:

ConfigManagerName
(Optional) The label of the Configuration Manager for which registry entries are to be reported. If you do not enter a name, the default name ″ConfigMgr″ is used.

Examples: Windows

Display registry entries of the specified Configuration Manager:

```
mqsireportconfigmgr SOAPCM
```

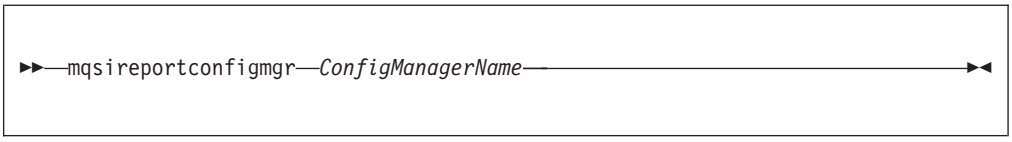
The following output is returned:

```
BIP8931I: Config Mgr Name 'SOAPCM'
Install path = 'C:\Program Files\IBM\MQSI\6.1.0.3'
Work path = 'C:\Documents and Settings\All Users\Application Data\IBM\MQSI'
Process id = '1862'
Queue manager = 'SOAPQM'
User Name Server queue manager = 'SOAPQM'
JVM heap size = '256' MB
Migration database name = 'DATACONF'
Migration database userId = 'db2admin'
Migration database password = '********'
Configuration Manager registry format = 'v6.1'
```

mqsireportconfigmgr command - Linux, and UNIX systems:

Use the mqsireportconfigmgr command on to display Configuration Manager registry entries.

Syntax:



Parameters:

ConfigManagerName
(Required) The label of the Configuration Manager for which registry entries are to be reported.

Examples: Linux UNIX

Display registry entries of the specified Configuration Manager:

```
mqsireportconfigmgr SOAPCM
```

The following output is returned:

```
BIP8930I: Config Mgr Name 'SOAPCM'
Install path = '/usr/lpp/ibm/mqsi/6.1.0.3/'
Work path = '/var/mqsi'
Process id = '1862'
Queue manager = 'SOAPQM'
Service userId = 'db2admin'
Service password = '********'
User Name Server queue manager = 'SOAPQM'
JVM heap size = '128' MB
Migration database name = 'DATACONF'
```

```
Migration database userId = 'db2admin'
Migration database password = '********'
Configuration Manager registry format = 'v6.1'
```

mqsireportconfigmgr command - z/OS:

Use the mqsireportconfigmgr command to display Configuration Manager registry entries.

Syntax:

Parameters:

ConfigManagerName
(Required) The label of the Configuration Manager for which registry entries are to be reported.

Examples: z/OS

Display registry entries of the specified Configuration Manager:

```
mqsireportconfigmgr MQ81CMGR
```

The following output is returned:

```
BIP8932I: Config Mgr Name 'MQ81CMGR'
Install path = 'usr/lpp/ibm/mqsi/6.1.0.3/'
Work path = '/u/wmqi81/config'
Process id = '1862'
Queue manager = 'MQ81'
User Name Server queue manager = 'MQ81'
JVM heap size = '128' MB
Configuration Manager registry format = 'v6.1'
```

mqsireportflowmonitoring command

Use the mqsireportflowmonitoring command to display the current options for monitoring that have been set using the mqsichangeflowmonitoring command.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS: Run this command either as a console command, or by customizing and submitting BIPRPME; see "Contents of the broker PDSE" on page 679.

Purpose:

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireportflowmonitoring command - Windows, Linux and UNIX systems" on page 625
- "mqsireportflowmonitoring command - z/OS" on page 627

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

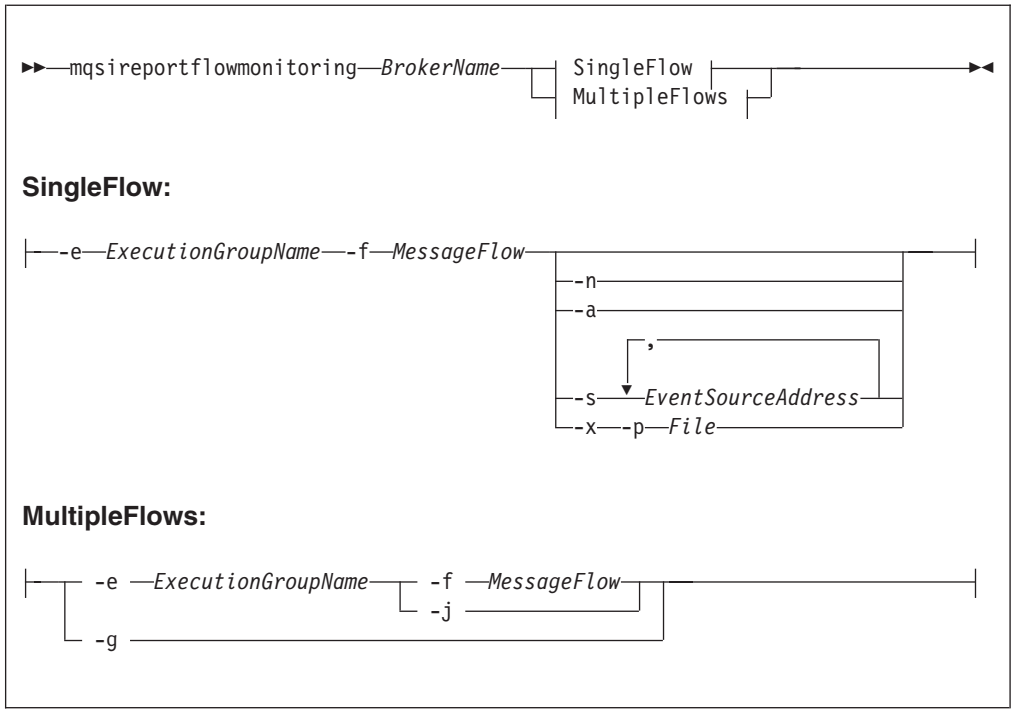
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

mqsireportflowmonitoring command - Windows, Linux and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required) Specify the label of the broker, for which monitoring options are to be reported.

-n
(Optional) Report the flow-level monitoring properties and the properties of any configured event sources within the flow.

-a (Optional) Report the flow-level monitoring properties and the properties of all event sources within the flow, whether configured or not.

-s *EventSourceAddress*
(Optional) Report the flow-level monitoring properties and the properties of the specified event sources within the flow, whether configured or not.

You must provide a comma-separated list of the event source addresses for which monitoring options are to be reported. An event source address takes the form <*node name*>.<*event source*>, where <*event source*> is one of the following values:

'terminal.<*terminal name*>'

'transaction.Start'

'transaction.End'

'transaction.Rollback'

If a message flow contains two or more nodes with identical names, the event sources on those nodes cannot be accurately addressed. If this is attempted, behavior is undefined.

Note: <*node name*> is the label of the node as known by the broker runtime components. If the node is in a subflow the label reflects this. For example, flow A contains an instance of flow B as a subflow labeled 'myB'; flow B contains an instance of a Compute node labeled 'myCompute'. The <*node name*> for the Compute node is 'myB.myCompute'.

-x (Optional; if you specify -x you must also specify the -p parameter.) Outputs the current monitoring properties for the specified message flow as a monitoring profile XML file.

-p (Optional unless -x is specified.) The file name to which the monitoring profile will be written, in XML format.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which monitoring options are to be reported.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

-f *MessageFlow*
(Required) Specify the label for the message flow, for which monitoring options are to be reported.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

-g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

-j (Required) Specifies that the command applies to **all** message flows that belong to the execution group.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

Note:

- If all flags are omitted, only the flow-level properties are reported.
- Flags -n, -a, -s and -x can be used only when -f is present.
- Flags -n, -a, -s and -x are alternatives and cannot be used together.

Examples:

Request a report of the monitoring options for message flow ″MyFlow1″ in the execution group ″default″ for broker ″BrokerA″:

```
mqsireportflowmonitoring BrokerA -e default -f MyFlow1
```

Request a report of the current monitoring options for all message flows in all execution groups for broker ″BrokerA″ :

```
mqsireportflowmonitoring BrokerA -g -j
```

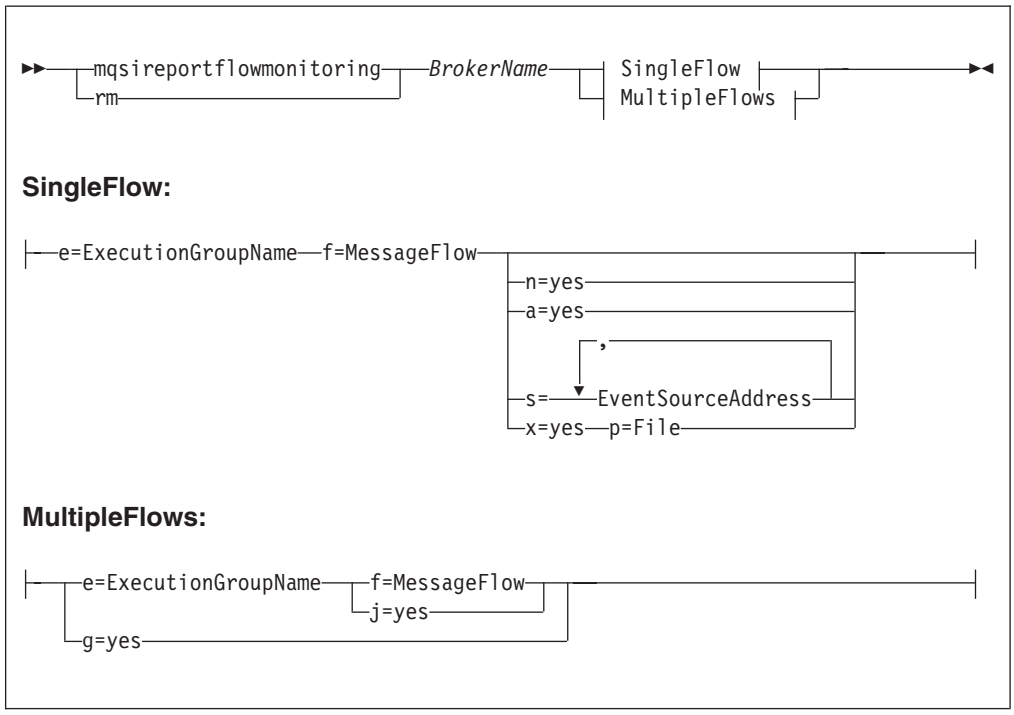
For more examples, see Reporting monitoring settings.

mqsireportflowmonitoring command - z/OS:

Syntax:

z/OS console command:

Synonym: rm



Parameters:

BrokerName
(Required) Specify the label of the broker, for which monitoring options are to be reported.

n (Optional) Report the flow-level monitoring properties and the properties of any configured event sources within the flow.

a (Optional) Report the flow-level monitoring properties and the properties of all event sources within the flow, whether configured or not.

s *EventSourceAddress*
(Optional) Report the flow-level monitoring properties and the properties of the specified event sources within the flow, whether configured or not.

You must provide a comma-separated list of the event source addresses for which monitoring options are to be reported. An event source address takes the form <*node name*>.<*event source*>, where <*event source*> is one of the following values:

'terminal.<*terminal name*>'

'transaction.Start'

'transaction.End'

'transaction.Rollback'

If a message flow contains two or more nodes with identical names, the event sources on those nodes cannot be accurately addressed. If this is attempted, behavior is undefined.

Note: <*node name*> is the label of the node as known by the broker runtime components. If the node is in a subflow the label reflects this. For example, flow A contains an instance of flow B as a subflow labeled 'myB'; flow B contains an instance of a Compute node labeled 'myCompute'. The <*node name*> for the Compute node is 'myB.myCompute'.

x (Optional; if you specify -x you must also specify the -p parameter.) Outputs the current monitoring properties for the specified message flow as a monitoring profile XML file.

p (Optional unless -x is specified.) The file name to which the monitoring profile will be written, in XML format.

e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which monitoring options are to be reported.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

f *MessageFlow*
(Required) Specify the label for the message flow, for which monitoring options are to be reported.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

g (Required) Specifies that the command applies to **all** execution groups that belong to the broker.

You must specify either **-e** or **-g**. If you do not specify one of these arguments you receive an error message.

j (Required) Specifies that the command applies to **all** message flows that belong to the execution group.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive an error message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

s (Optional) Report the flow-level monitoring properties and the properties of the specified event sources within the flow, whether configured or not.

You must provide a comma-separated list of the event source addresses for which monitoring options are to be reported. An event source address takes the form <*node name*>.<*event source*>, where <*event source*> is one of the following values:

'terminal.<*terminal name*>'

'transaction.Start'

'transaction.End'

'transaction.Rollback'

If a message flow contains two or more nodes with identical names, the event sources on those nodes cannot be accurately addressed. If this is attempted, behavior is undefined.

Note: <*node name*> is the label of the node as known by the broker runtime components. If the node is in a subflow the label reflects this. For example, flow A contains an instance of flow B as a subflow labeled 'myB'; flow B contains an instance of a Compute node labeled 'myCompute'. The <*node name*> for the Compute node is 'myB.myCompute'.

Note:

- If all flags are omitted, only the flow-level properties are reported.
- Flags -n, -a, -s and -x can be used only when -f is present.
- Flags -n, -a, -s and -x are alternatives and cannot be used together.

Examples:

Request a report of monitoring options for message flow "MFlow1" in the execution group "default":

```
F MI10BRK,rm e='default',f='MFlow1'
```

Request a report of the current monitoring options for all message flows in all execution groups:

```
F MI10BRK,rm g=yes,j=yes
```

For more examples, see Reporting monitoring settings.

mqsireportflowstats command

Use the mqsireportflowstats command to display the current options for accounting and statistics that have been set using the mqsichangeflowstats command.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPRPMS; see "Contents of the broker PDSE" on page 679

Purpose:

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireportflowstats command - Windows, Linux and UNIX systems"
- "mqsireportflowstats command - z/OS" on page 631

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

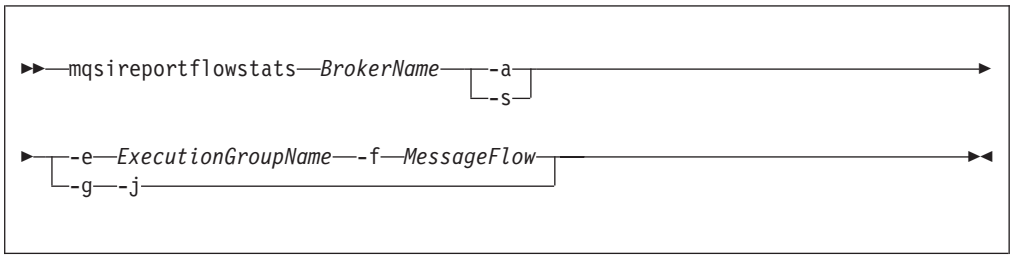
- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

mqsireportflowstats command - Windows, Linux and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required) 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.

You must specify **-a** or **-s**, or both arguments. 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.

You must specify **-a** or **-s**, or both arguments. If you do not specify at least one of these arguments you receive a warning message.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which accounting and statistics options are to be reported.

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 reported.

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.

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.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

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
```

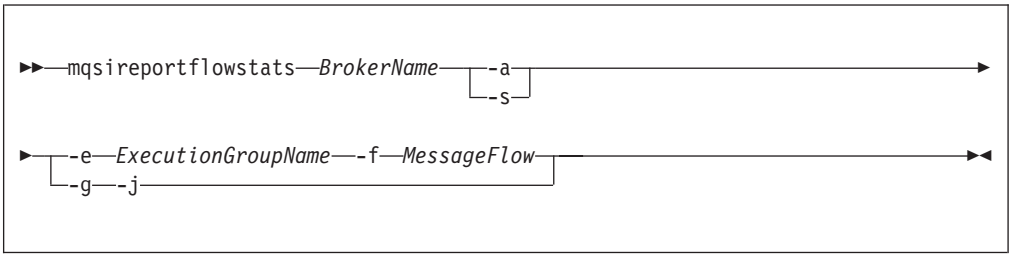
Request a report of the snapshot options that are currently stored for all message flows in all execution groups for broker ″BrokerA″ :

```
mqsireportflowstats BrokerA -s -g -j
```

mqsireportflowstats command - z/OS:

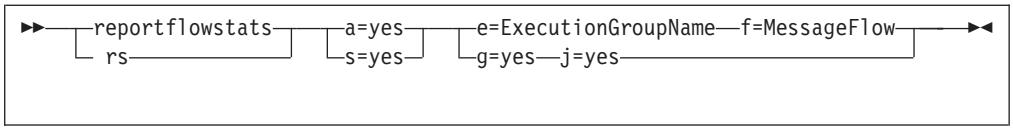
Syntax:

z/OS command - BIPRPMS:



z/OS console command:

Synonym: rs



Parameters:

BrokerName
(Required) 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.

You must specify **-a** or **-s**, or both arguments. 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.

You must specify **-a** or **-s**, or both arguments. If you do not specify at least one of these arguments you receive a warning message.

-e *ExecutionGroupName*
(Required) Specify the name for the execution group, for which accounting and statistics options are to be reported.

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 reported.

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.

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.

You must specify either **-f** or **-j**. If you do not specify one of these arguments you receive a warning message.

Note: If you set the **-g** option for all execution groups, you must use **-j** instead of **-f**.

Examples:

Using the command BIPRPMS:

- Request a report for message flow "MyFlow1" in the execution group "default" 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:

```
mqsireportflowstats BrokerA -s -g -j
```

Using the console form of the command, request a report for message flow ″MyFlow1″ in the execution group ″default″ for both archive and snapshot statistics collection:

```
mqsireportflowstats  s= a= e=default f=MyFlow1
```

mqsireportflowuserexits command

Use the mqsireportflowuserexits command to report the list of active and inactive user exits for the specified broker, execution group, or message flow.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPRPUE; see “Contents of the broker PDSE” on page 679

Purpose:

Use the mqsireportflowuserexits command to generate a report about the status of user exits:

- If you specify only the broker name, the command returns a list of all user exits that you have set to be active at the broker level by using the mqsichangebroker command.
- If you specify the broker and execution group names, the command returns the list returned for the broker name only, plus:
  - A list of all user exits that you have set to be active at the execution group level by using the mqsichangeflowuserexits command.
  - A list of all user exits that you have set to be inactive at the execution group level by using the mqsichangeflowuserexits command.
- If you specify the broker, execution group, and message flow names, the command returned the list returned for the broker and execution group names, plus:
  - A list of all user exits that you have set to be active at the message flow level by using the mqsichangeflowuserexits command.
  - A list of all user exits that you have set to be inactive at the message flow level by using the mqsichangeflowuserexits command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsireportflowuserexits command - Windows, Linux, and UNIX systems” on page 634
- “mqsireportflowuserexits command - z/OS” on page 634

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

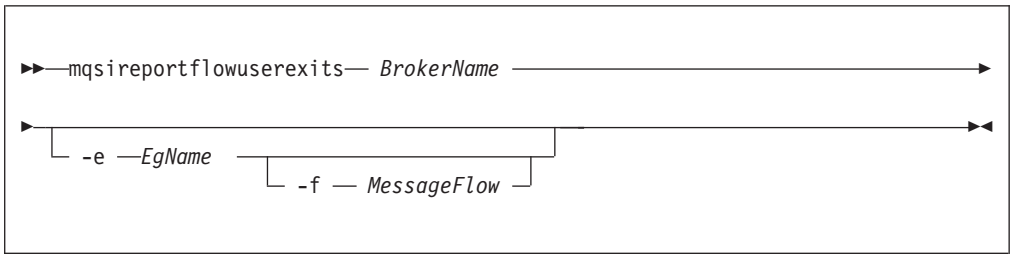
If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows, Linux, and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs group.

mqsireportflowuserexits command - Windows, Linux, and UNIX systems:

Syntax:



Parameters:

BrokerName
(Required). The name of the broker.

-e *EgName*
(Optional). The name of the execution group.

-f *MessageFlow*
(Optional). The name of the message flow.

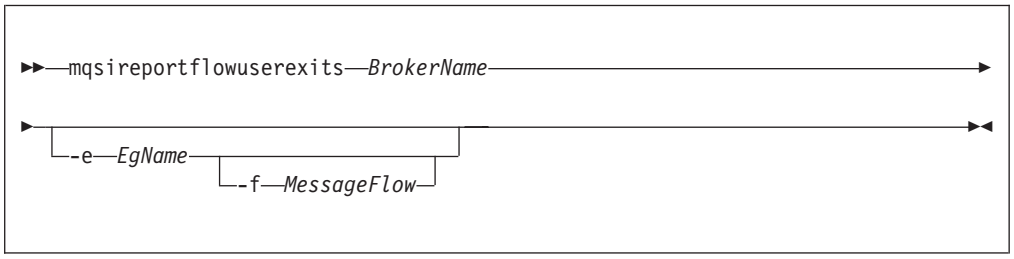
Examples:

```
mqsireportflowuserexits WBRK_BROKER -e default -f MYFLOW
BIP8859 User Exits active for broker MYBROKER: exit1, exit2
BIP8854 User Exits active for Execution Group default: exit1,exit3
BIP8855 User Exits inactive for Execution Group default: exit2
BIP8856 User Exits active for Message Flow MYFLOW: exit2
BIP8857 User Exits inactive for Message Flow MYFLOW: exit1
```

mqsireportflowuserexits command - z/OS:

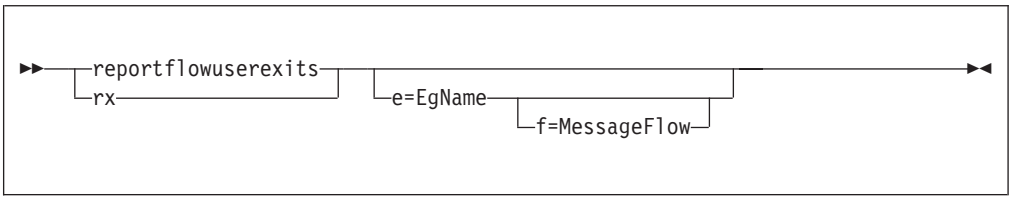
Syntax:

z/OS command - BIPRPUE:



z/OS console command:

Synonym: rx



Parameters:

BrokerName
(Required). The name of the broker.

-e *EgName*
(Optional). The name of the execution group.

-f *MessageFlow*
(Optional). The name of the message flow.

Examples:

```
mqsireportflowuserexits WBRK_BROKER -e default -f MYFLOW
BIP8859 User Exits active for broker MYBROKER: exit1, exit2
BIP8854 User Exits active for Execution Group default: exit1,exit3
BIP8855 User Exits inactive for Execution Group default: exit2
BIP8856 User Exits active for Message Flow MYFLOW: exit2
BIP8857 User Exits inactive for Message Flow MYFLOW: exit1
```

mqsireportproperties command

Use the mqsireportproperties command to display properties that relate to a broker or its associated configurable services.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPRPPR; see "Contents of the broker PDSE" on page 679.

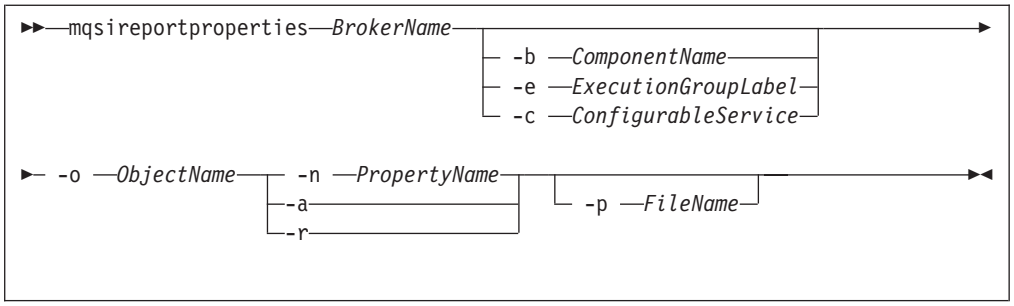
Purpose:

Use the mqsireportproperties command to examine the values of properties or broker resources that are set by using the mqsichangeproperties command, or created by using the mqsicreateconfigurableservice command.

Usage notes:

- Before you run the mqsireportproperties command, ensure that the broker is running.
- If you use the mqsichangeproperties command to change any value, stop and restart the broker for the change to take effect.

Syntax:



Parameters:

BrokerName
(Required) The name of the broker.

-b *ComponentName*
(Optional) The name of the component selected. Valid values are *httplistener* and *securitycache*.

-c *ConfigurableService*
(Optional) The type of the configurable service. Specify a value of `AllTypes` to report on all configurable service types.

For a list of supplied configurable services, and their properties and values, see "Configurable services properties" on page 444.

-e *ExecutionGroupLabel*
(Optional) The label of the execution group for which a report is required.

You can specify the special name `AllExecutionGroups` if you also specify either HTTPConnector or HTTPSConnector. This option returns broker-wide settings that affect SOAP nodes deployed to all execution groups for these objects.

-o *ObjectName*
(Required) The name of the object whose properties you want to read.

You must also specify **-b**, **-e**, or **-c** after **-o**, except if you specify the object name BrokerRegistry.

Set *ObjectName* to match other parameters:

- Specify BrokerRegistry for broker registry parameters.
- Specify the name of a configurable service, predefined or user-defined, of a type that you have specified with **-c**.

  For example:

  – **-c EISProviders** with SAP, Siebel, or PeopleSoft for a predefined WebSphere Adapters configurable service.
  – **-c JMSProviders** with the name of a predefined or user-defined service, for example, WebSphere_MQ.
  – **-c SecurityProfiles** with the name of a predefined or user-defined service, for example, Default Propagation.
- Specify a communications object for the execution group that you have specified with **-e**; one of HTTPListener, HTTPConnector, or HTTPSConnector. Values are defined for all SOAP nodes that you have deployed to the specified execution group.

- Specify a communications object for the httplistener component that you have specified with **-b**; one of HTTPListener, HTTPConnector, or HTTPSConnector. Values are defined for all HTTP nodes that you have deployed to the broker.
- Specify DynamicSubscriptionEngine for inter-broker communications properties for the execution group that you have specified with **-e**. These properties apply to brokers that you have configured in collectives, or cloned.
- Specify DynamicSubscriptionEngine for all Real-time nodes that you have deployed to the execution group that you have specified with **-e**.
- Specify SecurityCache for properties for the securitycache component that you have specified with **-b**.

Specify a value of `AllReportableEntityNames` to return a list of all valid object names. If you run the mqsireportproperties command on the command line without any properties, the `AllReportableEntityNames` is used.

-n *PropertyName*
(Optional) Display only the named property.

You must select only one option from **-n**, **-a**, and **-r**.

-a (Optional) Indicates that all property values of the object are displayed, and does not act in a recursive manner on 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.

-p *FileName*
(Optional) The location and name of the file to which the command writes all selected properties. If you do not specify **-n**, the property values, but not the property names, are written.

For more information about objects, properties, and values, and valid combinations of these parameters, see "mqsichangeproperties command" on page 437.

For the *httplistener* component, the mqsireportproperties command does not report those properties that have not been explicitly set with the mqsichangeproperties command, 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 you have changed it from this default with mqsichangeproperties. To see the default values for all properties that mqsireportproperties can report, see the mqsichangeproperties command description.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On other platforms, no specific authority is required to run this command.

Responses:

Responses are of the form:

- `HTTPConnector`
  - PortNumber = 7800
- `HTTPSConnector`
  - PortNumber = 7843

Examples:

Always enter the command on a single line; in some examples, a line break has been added to enhance readability.

Displaying properties associated with broker components

The following examples include the **-b** parameter to specify the component to view.

- Display all the current HTTPListener settings associated with HTTP nodes (defined in the httplistener component):

  ```
  mqsireportproperties TEST -b httplistener -o HTTPListener -a
  ```

- Display the HTTPSConnector port setting for the HTTP nodes (defined in the httplistener component):

  ```
  mqsireportproperties TEST -b httplistener -o HTTPSConnector -n port
  ```

Displaying properties associated with execution groups

The following example includes the **-e** parameter to specify the execution group to view.

- Display recursively all the current settings for the interbroker communication:

  ```
  mqsireportproperties TEST -e default -o DynamicSubscriptionEngine -r
  ```

Displaying properties associated with configurable services

The following examples include the **-c** parameter to specify the configurable service to view.

- Display all properties of the FtpServer configurable service called TEST1:

  ```
  mqsireportproperties BROKER2 -c FtpServer -o TEST1 -r
  ```

- Display the **protocol** property setting of the FtpServer configurable service called TEST1:

  ```
  mqsireportproperties BROKER2 -c FtpServer -o TEST1 -n protocol
  ```

- Display all IMSConnect configurable services:

  ```
  mqsireportproperties WBRK61_DEFAULT_BROKER -c IMSConnect -o AllReportableEntityNames -r
  ```

- Report the properties of the Oracle JDBCProvider configurable service:

```
mqsireportproperties WBRK_BROKER -c JDBCProviders -o Oracle -r
```

- Display the properties for all the broker's JMS provider resources (default JMS provider resources and those configurable services that you have defined by using the mqsicreateconfigurableservice command):

```
mqsireportproperties WBRK6_DEFAULT_BROKER -c JMSProviders
     -o AllReportableEntityNames -r
```

- Display the properties for all the JMS provider resources of WebSphere MQ.

```
mqsireportproperties WBRK6_DEFAULT_BROKER -c JMSProviders
     -o WebSphere_MQ -r
```

- Display the properties for all the broker's JMS provider resources (default JMS provider resources and those configurable services that you have defined with the mqsicreateconfigurableservice command):

```
mqsireportproperties WBRK6_DEFAULT_BROKER -c JMSProviders
     -o BEA_WebLogic –r
```

The result of this command has the following format:

```
ReportableEntityName=''
JMSProviders
  BEA_Weblogic=''
    jarsURL='default_Path'
    nativeLibs='default_Path'
    proprietaryAPIAttr1='weblogic.jndi.WLInitialContextFactory'
    proprietaryAPIAttr2='t3://9.20.94.16:7001'
    proprietaryAPIAttr3='BEAServerName'
    proprietaryAPIAttr4='default_none'
    proprietaryAPIAttr5='default_none'proprietaryAPIHandler='BEAWebLogicAPIHandler.jar'
```

- Display all PeopleSoftConnection configurable services:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c PeopleSoftConnection
-o AllReportableEntityNames -r
```

- Display all policy sets for broker WBRK_BROKER:

```
mqsireportproperties WBRK_BROKER -c PolicySets
     -o AllReportableEntityNames -a
```

- Display all policy set bindings for broker WBRK_BROKER:

```
mqsireportproperties WBRK_BROKER -c PolicySetBindings
     -o AllReportableEntityNames -a
```

- Export policy set Policy_2 in broker WBRK_BROKER to file `policyset.xml`:

```
mqsireportproperties WBRK_BROKER -c PolicySets
     -o Policy_2 -n ws-security -p policyset.xml
```

You can use the output file to move policy sets between brokers, and for backup.

- Export a policy set bindings from a broker to a file:

```
mqsireportproperties WBRK_BROKER -c PolicySetBindings
     -o Bindings_2 -n ws-security -p bindings.xml
```

This command writes the Policy Set Bindings file Bindings_2 in broker WBRK_BROKER to file `bindings.xml`. You can use this file to move policy set bindings between brokers, and for backup.

- Display the current status of the WebSphere Message Broker publish/subscribe engine:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c PublishSubscribe
      -o AllReportableEntityNames -r
```

If the psmode returned has the value enabled, the broker's publish/subscribe engine is enabled and the broker handles all publish/subscribe application messages and operations. If the value returned is disabled, the queue manager's publish/subscribe engine is handling all application publications and

subscriptions. You can disable the broker's publish/subscribe engine only if you have installed WebSphere MQ Version 7.0 on this computer.

- Report the dependencies for the WebSphere Adapter for SAP:

```
mqsireportproperties WBRK_BROKER -c EISProviders -o SAP -r
```

- Display all SAPConnection configurable services:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c SAPConnection -o AllReportableEntityNames -r
```

- Display the properties for all the broker's security profiles (default security profiles and any that you have defined by using the mqsicreateconfigurableservice command):

```
mqsireportproperties WBRK6_DEFAULT_BROKER -c SecurityProfiles
     -o AllReportableEntityNames -r
```

The result of this command has the following format:

```
ReportableEntityName=''
SecurityProfiles

 Default_Propagation=''
  Authentication = 'NONE'
  AuthenticationConfig = ''
  Mapping = 'NONE'
  MappingConfig = ''
  Authorization = 'NONE'
  AuthorizationConfig = ''
  Propagation = 'TRUE'
  passwordValue = 'PLAIN'
```

- Display the properties for the security profile called MyFirstSecurityProfile:

```
mqsireportproperties WBRK_6_DEFAULT_BROKER -c SecurityProfiles
     -o MyFirstSecurityProfile -r
```

The result of this command has the following format:

```
ReportableEntityName=''
SecurityProfiles
 MyFirstSecurityProfile=''
  Authentication = 'LDAP'
  AuthenticationConfig = 'ldap://localhost:389/ou=users,o=ibm'
  Mapping = 'TFIM'
  MappingConfig = 'http://tfimhost1:80'
  Authorization = 'NONE'
  AuthorizationConfig = ''
  Propagation = 'TRUE'
  passwordValue = 'PLAIN'
```

- Display all SiebelConnection configurable services:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c SiebelConnection -o AllReportableEntityNames -r
```

- Display all TCPIPclient configurable services:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c TCPIPClient
      -o AllReportableEntityNames -r
```

- Display all TCPIPServer configurable services:

```
mqsireportproperties WBRK61_DEFAULT_BROKER -c TCPIPServer
      -o AllReportableEntityNames -r
```

mqsireporttrace command

Use the mqsireporttrace command to display the trace options currently in effect.

Supported platforms:

- Windows
- Linux and UNIX systems

- z/OS as a console command

Purpose:

The mqsireporttrace command is valid for the following options:

- User trace. Specify the **-u** option.

  On z/OS, user trace is run against a specific execution group for a broker. You can specify an extra flag on the command 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). You can specify an extra flag on the command to trace a specific message flow.

  Service trace can also be run only against a broker, or any of its resources.
- Trace nodes. Specify the **-n** option.

If you specify a broker, or any of its resources (execution group or message flow), you must have deployed those resources, and the broker must be running, before you can query trace settings.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsireporttrace command - Windows, Linux, and UNIX systems"
- "mqsireporttrace command - z/OS" on page 643

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

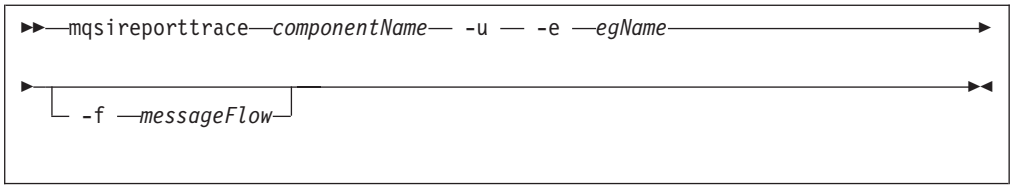
On Windows, Linux, and UNIX systems, the user ID that is used to run this command must be a member of the mqbrkrs group.

mqsireporttrace command - Windows, Linux, and UNIX systems:

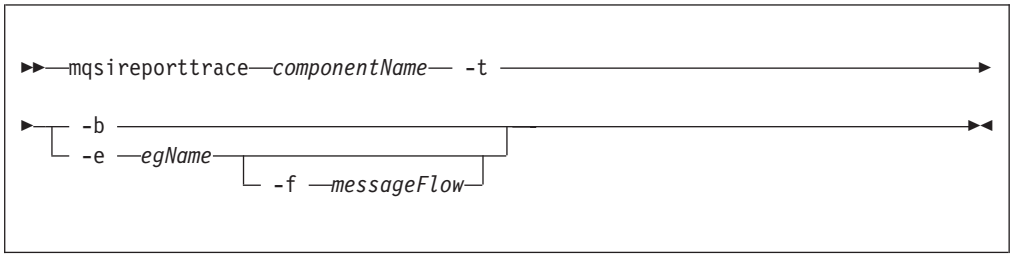
Use the mqsireporttrace command to display the trace options currently in effect.

Syntax:

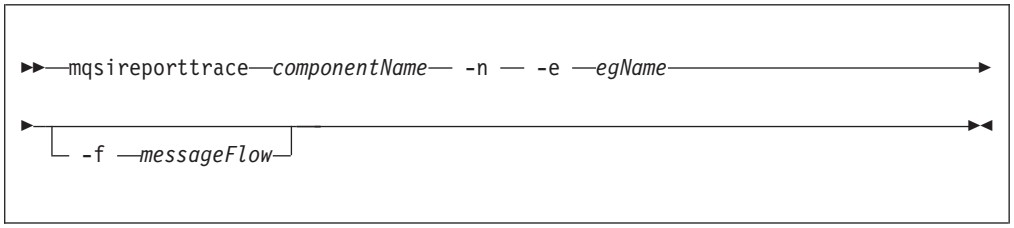
User trace:



Service trace:



Trace nodes:



Parameters:

componentName
(Required) The name of a broker, a Configuration Manager, or the fixed name `User Name Server`; this name is case sensitive on Linux and UNIX systems.

-u (Required for user trace) Derive report information from the user trace.

-e *egName*
(Required for user trace, otherwise optional) The label of the execution group for which a report is required. This option is valid only if you have specified a broker as the component.

-f *messageFlow*
(Optional) The label of the message flow for which a report is required. This option is valid only if you have specified a broker as the component, and an execution group.

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.

Additional parameters exclusive to Trace nodes:

-n Report the setting of the Trace node switch. One BIP message is reported for each message flow.

Examples:

To derive report information from service trace for execution group exgrp1 in broker WBRK_BROKER, enter the command:

```
mqsireporttrace WBRK_BROKER -t -e "exgrp1"
```

To report the setting of the Trace node switch for execution group exgrp1 in broker WBRK_BROKER, enter the command:

```
mqsireporttrace WBRK_BROKER -n -e "exgrp1"
```

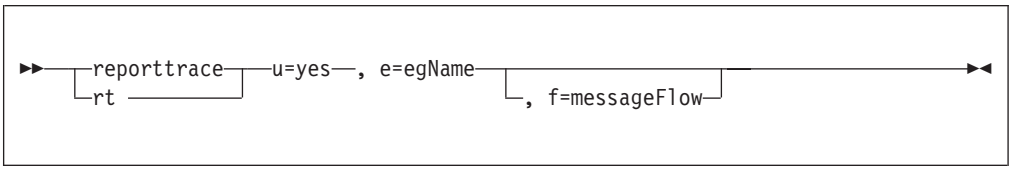
mqsireporttrace command - z/OS:

Use the mqsireporttrace command to display the trace options currently in effect.

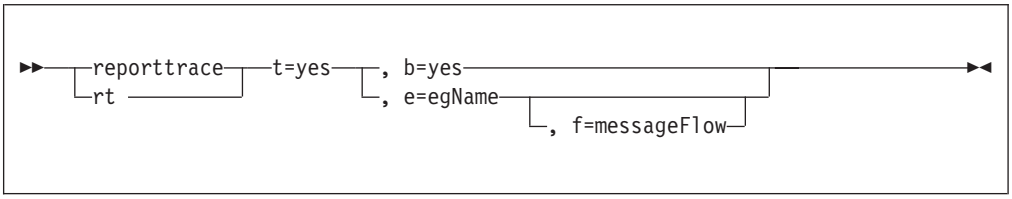
Syntax:

z/OS console command:

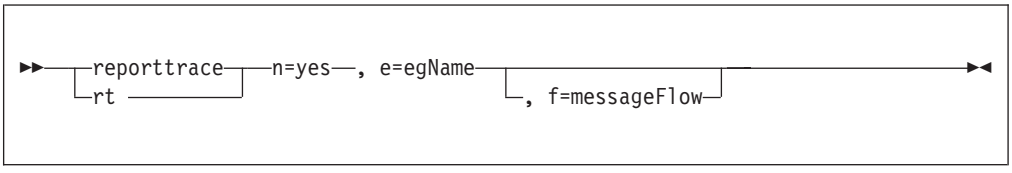
User trace:



Service trace:



Trace nodes:



Parameters:

-u (Required for user trace) Derive report information from the user trace.

-e *egName*
(Required for user trace, otherwise optional) The label of the execution group for which a report is required. This option is valid only if you have specified a broker as the component.

This name is case sensitive; include the name in single quotes if it contains mixed-case characters.

-f *messageFlow*
(Optional) The label of the message flow for which a report is required. This option is valid only if you have specified a broker as the component, and an execution group.

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.

Additional parameters exclusive to Trace nodes:

n=yes
Report the setting of the Trace node switch. One BIP message is reported for each message flow.

Examples:

To report information from service trace for execution group exgrp1, enter the command:

```
F MQP1BRK,rt t=yes, e='exgrp1'
```

To report the setting of the Trace node switch for execution group exgrp1, enter the command:

```
F MQP1BRK,rt n=yes, e='exgrp1'
```

mqsirestoreconfigmgr command

Use the mqsirestoreconfigmgr command to restore Configuration Manager archive files that you created by earlier use of the mqsibackupconfigmgr command.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPRSCM; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

You can use the mqsirestoreconfigmgr command to recreate a Configuration Manager from backed up files on the original system on which it existed, or on another system.

Usage notes:

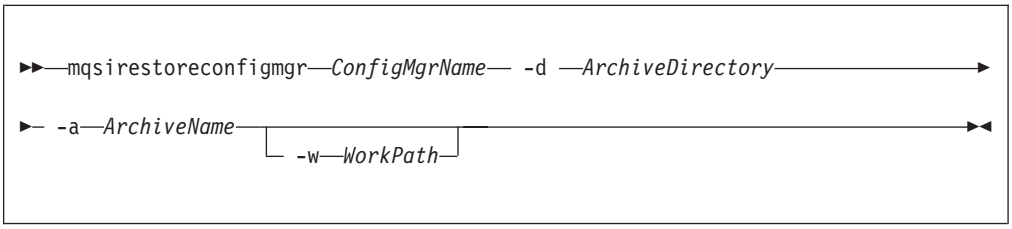
1. Before you run the command, stop the Configuration Manager.
2. Run this command on the computer on which the Configuration Manager has been created.

3. If you are recovering the Configuration Manager because the configuration repository is damaged, restore the repository from a previously successful backup version.
4. If you plan to restore the Configuration Manager repository on z/OS and the platform from which it was originally backed up was not z/OS, or the other way round, you must copy the saved `service.properties` file to:

```
<Configuration
Manager directory>/components/<component name>/<directory name>/service.properties
```

after running the mqsirestoreconfigmgr command.

Syntax:



Parameters:

ConfigMgrName
(Required) The name of the Configuration Manager.

-d *ArchiveDirectory*
(Required) The directory where the archive is placed.

-a *ArchiveName*
(Required) The backup archive name.

-w *WorkPath*
(Optional) The path for the Configuration Manager repository.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID must be a member of he group mqbrkrs.

mqsisetdbparms command

Use the mqsisetdbparms command to associate a specific user ID and password (or SSH identity file) with one or more resources that are accessed by the broker.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPSDBP.

Purpose:

Use the mqsisetdbparms command to set security credentials for a specific user ID and password (or SSH identity file) for the following resources:

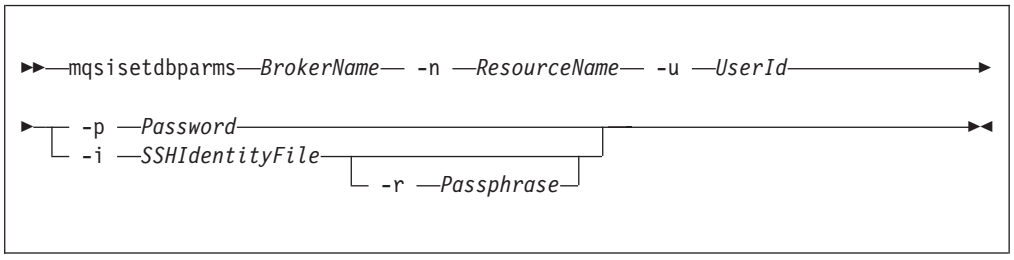
- A data source name (DSN) that is accessed from a message flow.
- A JDBCProvider configurable service.
- A JMS or JNDI resources, for example a JMSProvider configurable service.
- Lightweight Directory Access Protocol (LDAP) bind credentials for the broker security manager.
- An account name, with a user name and password, for the adapter nodes.
- An IMSConnect configurable service.
- An SMTP configurable service.
- An FtpServer configurable service.
- The broker keystore password.
- A WebSphere Service Registry and Repository (WSRR) configurable service.

The user ID and password pair is created in the DSN folder under the broker's registry folder.

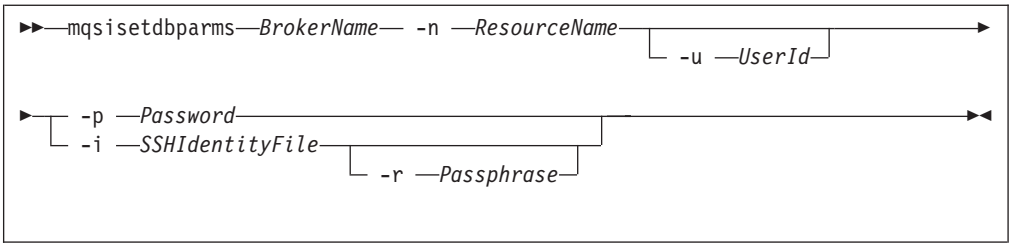
This command does not run if the broker is running. You must stop the broker before you run this command.

Syntax:

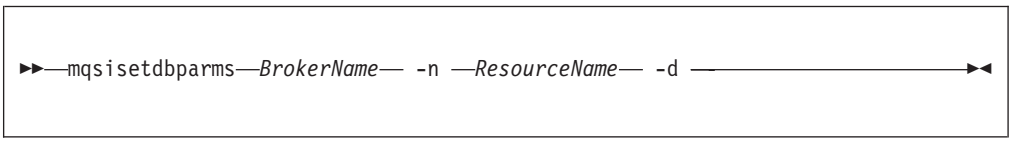
Create:



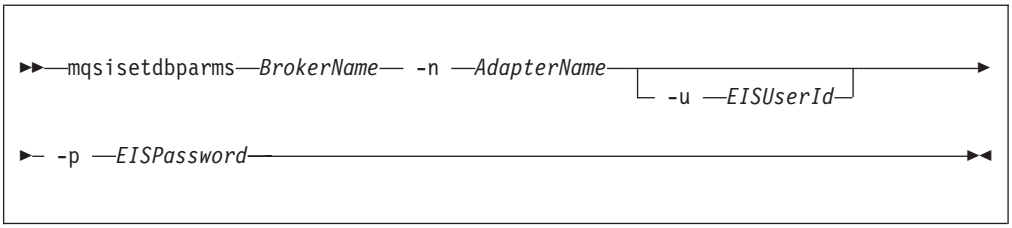
Alter:



Delete:



Adapter connection:



Parameters:

BrokerName
(Required) The name of the broker for which settings are to be created, altered, or deleted.

-n *ResourceName* **or** *AdapterName*
(Required) This parameter identifies one of the following resources:

- The data source for which the user ID and password pair are to be created or modified.

  Data source names are used by the following nodes:

  – Compute
  – Database
  – DatabaseRetrieve
  – DatabaseRoute
  – DataDelete
  – DataInsert
  – DataUpdate
  – Filter
  – Mapping
  – Warehouse

  If you use the same DSN to refer to the same database instance from multiple nodes, the same user ID and password combination is used.

  The mqsisetdbparms command does not apply to the broker database. Although you can issue this command successfully to specify the data source name that is used by the broker to access its database, it has no effect

on the user ID that is used to access any database, including all user databases that are referenced by that data source name.

- The name of the security identity that is used to authenticate a JDBC type 4 connection. The *ResourceName* takes the form `jdbc::secId`, where `secId` is specified as the value of the **-n securityIdentity** property of the associated JDBCProvider configurable service on the mqsicreateconfigurableservice or mqsichangeproperties command.
- The name of the security identity that is used to authenticate a connection to a JMS or JNDI resource. The *ResourceName* takes the form `jms::secId` or `jndi::secId`, where `secId` is specified as the value.
- The name of the security identity that is used to authenticate an LDAP directory.

  Specify `ldap::<servername>` to define credentials for an individual server. If you want the broker to bind anonymously to this server, specify `anonymous` as the user ID.

  Specify `ldap::LDAP` to define a default setting. The broker uses the specified user ID and password values for all servers that do not have an explicit `ldap::<servername>` entry. Therefore, all servers that were previously using anonymous bind by default start using the details defined in an `ldap::LDAP` entry.
- The name of the Adapter connection to the external EIS.
- The name of the IMS connection. The *ResourceName* takes the form `ims::secId`, where `secId` is the same as the value of the Security identity property on the IMSRequest node or in the **-n securityIdentity** property of the associated IMSConnect configurable service.
- The name of the security identity that is used to authenticate an SMTP server.
- The name of the security identity that is used to authenticate a connection to an FTP server. The *ResourceName* takes the form `ftp::secId`, where `secId` is specified as the value of the Security identity property of the FileInput or FileOutput node, or in the **-n securityIdentity** property of the associated FtpServer configurable service on the mqsicreateconfigurableservice or mqsichangeproperties command.
- The name of the security identity that is used to authenticate a connection to an SFTP server. The security identity is used to locate the user name and password or the Secure Shell (SSH) identity file. The *ResourceName* takes the form `sftp::secId`, where `secId` is specified as the value of the Security identity property of the FileInput or FileOutput node, or in the **-n securityIdentity** property of the associated FtpServer configurable service on the mqsicreateconfigurableservice or mqsichangeproperties command.
- The name of the security identity that is used to authenticate a broker keystore.
- The name of the security identity that is used to authenticate a WSRR configurable service.

-u *UserId* **or** *EISUserId*
(Required for Create and Adapter connection; Optional for Alter) The user ID to be associated with this resource or EIS.

-p *Password*
(Required for Create, Alter, and Adapter connection) The password to be associated with this resource or EIS.

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.

This parameter is required with the `ftp::` resource type, but is optional with the `sftp::` resource type. However, if you do not specify a password with an `sftp::` resource, you must specify the *SSHIdentityFile* parameter.

-i *SSHIdentityFile*
(Optional) The name of an identity file, in the OpenSSH format, to be used for authentication with SFTP, in place of a password. You must specify either a password or an identity file, but not both. If you specify an identity file you can also specify a pass phrase with the *Passphrase* parameter.

On z/OS systems, known hosts files and SSH identity files are stored in EBCDIC format, and on other operating systems they are stored in ASCII format.

-r *Passphrase*
(Optional) The pass phrase used for authentication with SFTP. This parameter is valid only when the *SSHIdentityFile* parameter is also specified. The pass phrase is used during decryption of the identity file.

-d (Required for Delete) This parameter deletes the user ID and password pair for this resource from the registry.

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the group Administrators on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID that is used to run this command must be a member of the mqbrkrs group.

On z/OS, the user ID that is used to run this command must be a member of a group that has *READ* and *WRITE* access to the component directory.

Ensure that the registry is appropriately secured to prevent unauthorized access. mqsisetdbparms is not required for correct operation of the broker. However, if the broker administrator does not assign specific user IDs and passwords to specific resources, the broker user ID (and password on Windows and UNIX systems) is used. The password is not stored in clear text in the file system.

Examples:

Data source names

The following example show use of the command for a specific data source name (no Universal Record Identifier (URI) prefix is required for this purpose):

```
mqsisetdbparms WBRK_BROKER -n Database1 -u MQUserId -p password
```

The following example deletes the values that were defined for a specific data source name:

```
mqsisetdbparms WBRK_BROKER -n ClientDB -d
```

JDBC type 4 connections

Use the mqsisetdbparms command to associate a user ID and password pair with a JDBC type 4 connection. The value that you specify for the **-n ResourceName** must have a prefix of `jdbc::`, followed by the value that matches the **-n securityIdentity** property of the associated JDBCProvider configurable service.

```
mqsisetdbparms broker name -n resource_name -u userID -p password
```

For example:

```
mqsisetdbparms BROKER1 -n jdbc::mySecurityIdentity -u myuserid -p secretpw
```

JMS and JNDI resource names

The following examples show the use of the command when the URI for a JMS or JNDI resource name is substituted for the **-n ResourceName** parameter.

For a JMS resource, the URL prefix is `"jms::"`; for JNDI, the prefix is `"jndi::"`.

On Linux and UNIX systems, if the parameter string includes a back slash (\) character, you must escape from this character by using a second back slash character (\\) when you enter the mqsisetdbparms command.

For example, to specify a user ID of `myuserid` and password `secret` for JMS topic connection factory `tcf1` in broker `MyBroker1`, use the following syntax:

```
mqsisetdbparms MyBroker1 -n jms::tcf1 -u myuserid -p secret
```

Similarly, to specify the same security for a JNDI initial context `com.sun.jndi.fscontext.RefFSContextFactory`, enter the following command:

```
mqsisetdbparms MyBroker1 -n jndi::com.sun.jndi.fscontext.RefFSContextFactory
     -u myuserid -p secret
```

JMS node account names

The preceding examples describe how to configure security for JMS and JNDI resources for *all* JMS nodes that use those resources in a broker.

To increase the degree of control that you have in the security of JMS nodes, you can associate a resource with an *account name*. The account name comprises the message flow name concatenated with the node label by using the underscore character (_):

```
Message Flow Name_Node label
```

For example, where the message flow name is `MyJMSFlow1`, and you require a specific user ID and password for JMSInput node `MyJMSInput1`, the resulting account name is:

MyJMSFlow1_MyJMSInput1

You can then specify the account name string in the **-n ResourceName** parameter on the mqsisetdbparms command by prefixing the account name with the *resource type*, and concatenating the account name with an at sign (@) character followed by the resource name :

resource typeaccount name@resource name

Therefore, assuming a JMS resource name of tcf1, used by JMSInput node MyJMSInput1 in message flow MyJMSFlow1, the following resource name is used:

jms::MyJMSFlow1_MyJMSInput1@tcf1

To specify a user ID of myuserid, a password of secret, a broker name of MyBroker1, and the resource name created from the account name, as described previously, use the following syntax:

```
mqsisetdbparms MyBroker1 -n jms::MyJMSFlow1_MyJMSInput1@tcf1
                -u myuserid -p secret
```

LDAP servers

Use the mqsisetdbparms command to set up authorization credentials for the LDAP server ldap.mydomain.com:

```
mqsisetdbparms BRK1 -n ldap::ldap.mydomain.com -u ldapuid -p ********
```

To set up authorization for other servers, use the command to set up default credentials:

```
mqsisetdbparms BRK1 -n ldap::LDAP -u ldapother -p ********
```

If you want the broker to bind anonymously to an LDAP server, specify the server name and the user ID anonymous:

```
mqsisetdbparms BRK1 -n ldap::ldap.mydomain2.com -u anonymous -p ********
```

For the user ID anonymous, the password is always ignored.

WebSphere Adapters account names

Use the mqsisetdbparms command in the following format to configure an account name with a user name and password for the WebSphere Adapters.

mqsisetdbparms *broker name* -n *adapter name* -u *user name* -p *password*

For example:

```
mqsisetdbparms BRK1  -n SAPCustomerInbound.inadapter -u sapuid -p ********
mqsisetdbparms BRK1  -n TwineballInbound.inadapter -u mqbroker -p ********
```

IMS connections

Use the mqsisetdbparms command in the following format to associate a user ID and password pair with an IMS Connect connection.

mqsisetdbparms *broker name* -n *resource_name* -u *userID* -p *password*

For example:

```
mqsisetdbparms MyBroker1 -n ims::mySecurityIdentity -u myuserid -p mypassword
```

FTP and SFTP server connections

Use the mqsisetdbparms command to associate a user ID and password with an FTP server connection:

```
mqsisetdbparms Broker1 -n ftp::identityA -u user1 -p MyPassword
```

Use the mqsisetdbparms command to associate a user ID and password with an SFTP server connection:

```
mqsisetdbparms Broker2 -n sftp::identityB -u user2 -p MyPassword
```

Use the mqsisetdbparms command to associate a user ID and SSH identity file with an SFTP server connection:

```
mqsisetdbparms Broker3 -n sftp::identityC -u user3 -i C:\key_rsa_no_pp
```

Use the mqsisetdbparms command to associate a user ID, SSH identity file, and pass phrase with an SFTP server connection:

```
mqsisetdbparms Broker4 -n sftp::identityD -u user4 -i C:\key_rsa_pp -r MyPassPhrase
```

mqsisetsecurity command

Use the mqsisetsecurity command to create the Windows groups that WebSphere Message Broker requires for secure access to its runtime libraries and data.

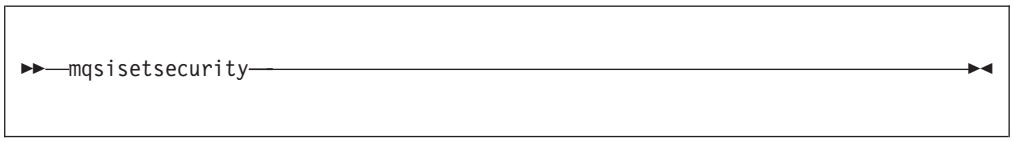
Supported platforms:

- Windows

Purpose:

The mqsisetsecurity command runs automatically as part of the installation process of WebSphere Message Broker. If WebSphere MQ is installed after WebSphere Message Broker, you can issue this command to add your account to the group mqm.

Syntax:



Parameters:

None

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged

command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Windows XP and Windows Server 2003, the user ID used to run this command must belong to the Administrators group on the local system.

mqsi_setupdatabase command

Use the mqsi_setupdatabase command to set up database links for an Oracle database from 32-bit execution groups defined to brokers running on the listed Linux and UNIX systems.

Supported platforms:

- AIX
- HP-UX on PA-RISC
- Linux on x86
- Linux on x86-64
- Solaris on SPARC

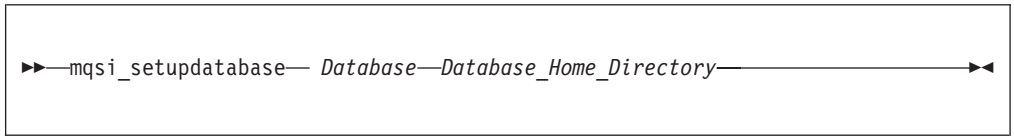
Purpose:

Do not run this command for 64-bit brokers or for a database from any other supported vendor.

Run the command for both broker and user databases. The command works for a remote database in the same way that it does for a local database.

Run this command after you have installed the Oracle database manager but before you create the database that you want the broker to connect to. If necessary, though, you can run the command before you install the Oracle database manager if you specify the intended database installation directory correctly.

Syntax:



Parameters:

Database
(Required) The database version that you have installed. The following versions are supported:

- oracle9
- oracle10
- oracle11

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 run this command must be a member of the mqbrkrs group.

Examples:

Set up the required links to an Oracle 9 database:

```
mqsi_setupdatabase oracle9 /oracle/product/9i/Db_1
```

Set up the required links to an Oracle 11 database:

```
mqsi_setupdatabase oracle11 /oracle/product/11g/Db_1
```

mqsistart command

Use the mqsistart command to start the specified component if all initial verification tests complete successfully.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS as a console command.

Purpose:

The mqsistart command:

- On distributed systems, starts the associated queue manager if it is not already running.
- If the component is a broker or a Configuration Manager:
  - Checks that the component environment is set up correctly; for example, that the installed level of Java is supported.
  - Verifies that the WebSphere MQ queues are defined and accessible.
  - If the component is a broker, checks that the expected database tables are present.

  The command completes all these checks, and reports all errors in the system log, or to the command line, or both. If one or more checks fail, the command does not start the component.
- Starts the component.
- If the component is a broker, and WebSphere MQ Version 7.0 is installed, checks the status of the broker's publish/subscribe engine, and that the setting of the queue manager's publish/subscribe mode (attribute PSMODE) is compatible. The default scenario is that the broker's publish/subscribe engine is enabled, and the queue manager's engine must be disabled. If you have disabled the broker's publish/subscribe engine, the command checks that the queue manager's publish/subscribe engine is enabled. It reports the status of the broker's publish/subscribe engine in the system log.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsistart command - Windows, Linux, and UNIX platforms" on page 655
- "mqsistart command - z/OS" on page 656

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the Administrators group and the mqbrkrs group on the local system. The queue manager used by the component must already be running before this command is executed, unless the user ID is also a member of the mqm group.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the Administrators group and the mqbrkrs group on the local system.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes. The queue manager used by the component must already be running before you run this command, unless the user ID is also a member of the mqm group.

On Linux and UNIX systems, the user ID used to run this command must be a member of the mqbrkrs. The queue manager used by the component must already be running before you run this command, unless the user ID is also a member of the mqm group.

When the Windows service starts, it runs under the user ID specified by the **-i** flag on the appropriate mqsicreatexxxx command.

On Windows, Linux, and UNIX systems, the component starts only if the *ServiceUserID* specified is authorized to access both the following locations:

- Home directory, where WebSphere Message Broker has been installed.
- Working directory, if specified by the **-w** flag on the mqsicreatexxxx command.

On z/OS, the user ID used to run this command must have **UPDATE** authority in class **OPERCMDS** to the `MVS.START.STC.message_broker_component_started_task` resource.

mqsistart command - Windows, Linux, and UNIX platforms:

Use the mqsistart command to start a component.

Purpose:

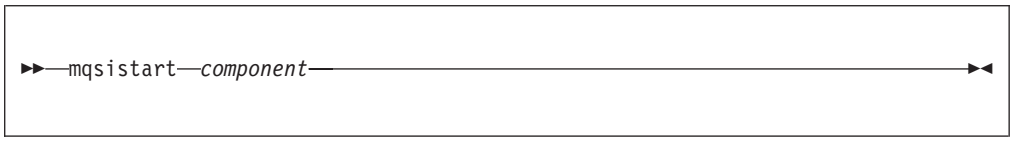
If the queue manager associated with this component (defined by the corresponding create command) is not already running, it is also started by this command. However, no listeners, channels, or channel initiators associated with the started component are started. Use WebSphere MQ Explorer to start any required listeners, channels, or channel initiators.

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 or UNIX syslog to determine if the component and all related software have started successfully, are initially active, and remain in an active state.

All errors that have prevented successful startup, that are detected by the component, are recorded in the log. Continue to monitor the Windows system event log or Linux or UNIX syslog.

On Windows platforms, the queue manager is not started as a service, and stops if you log off. To avoid this happening, either remain logged on, or change the startup status of the queue manager service as described in "Starting a WebSphere MQ queue manager as a Windows service" on page 329. (If you lock your workstation, the queue manager does not stop).

Syntax:



Parameters:

component
(Required) The component must have a broker name, a Configuration Manager name, or one of the following fixed values:

- Windows `ConfigMgr`, `UserNameServer`, or `DatabaseInstanceMgr` on Windows platforms.
- Linux UNIX `UserNameServer` on Linux and UNIX systems.

On Linux and UNIX systems, all names are case sensitive.

Do not use the mqsistop command on a `DatabaseInstanceMgr` unless you are using the Derby database.

Responses:

- 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:

To start the broker WBRK_BROKER:

```
mqsistart WBRK_BROKER
```

To start the Database Instance manager on Windows:

```
mqsistart DatabaseInstanceMgr
```

mqsistart command - z/OS:

Use the startcomponent command to start a component when its controller (control process) is already running, or the start (/S) command to bring a component into a state in which you can run the appropriate change command.

Purpose:

The following table distinguishes between the start (/S) and startcomponent commands, and lists the available options.

Component	Command	Description
Broker	/S *Broker started task name* /F *Broker started task name*,SC	Starts the broker. Starts the broker from a 'stop component' state.
Configuration Manager	/S *Configuration Manager started task name* /F *Configuration Manager started task name*,SC	Starts the Configuration Manager. Starts the Configuration Manager from a 'stop component' state.
User Name Server	/S *User Name Server started task name* /F *User Name Server started task name*,SC	Starts the User Name Server. Starts the User Name Server from a 'stop component' state.

When the controller address space is started, the component starts automatically. You can change this behavior by using an optional start parameter in the started task. If you set the parameter to MAN, the component does not start automatically; the default is AUTO.

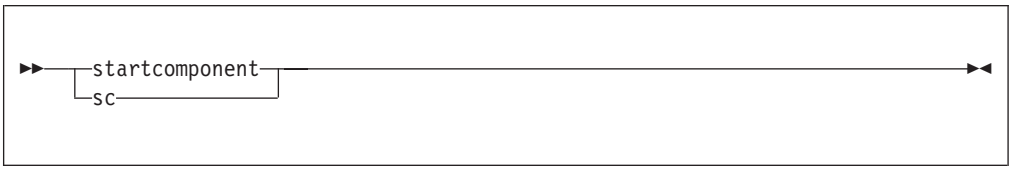
Issuing commands against the controller means issuing start, stop, or modify commands from the console to the controller address space. Two scenarios apply when you use this command:

1. The controller is started with the parameter MAN instead of AUTO.
2. After a stopcomponent command, you must restart the component.

Syntax:

z/OS console command - startcomponent:

Synonym: sc



Examples:

```
F MQ00BRK,sc
```

mqsistartmsgflowcommand

Use the mqsistartmsgflow command to start message flows.

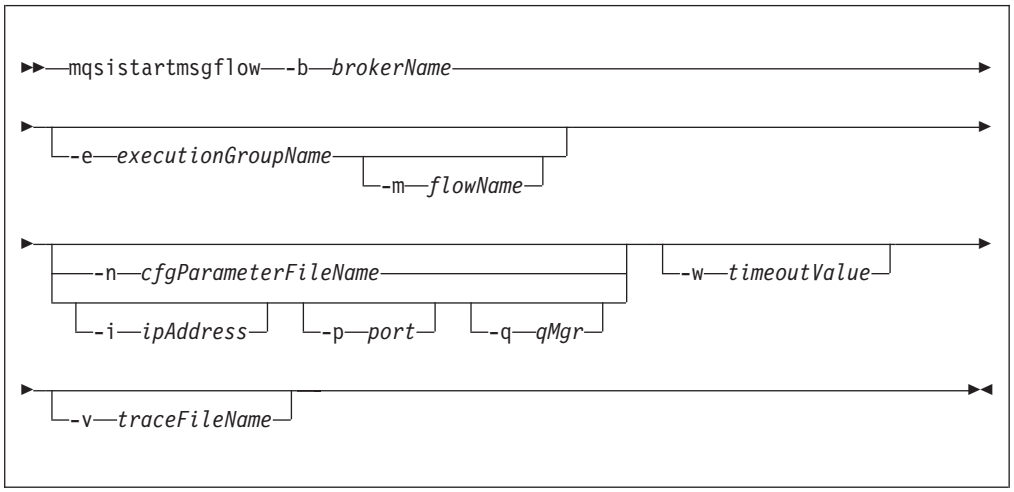
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPSTMF; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Start a single message flow, or all message flows in an execution group, on a specified broker. You must have previously deployed the message flows to the broker in a BAR file.

Syntax:



Parameters:

-b *brokerName*
(Required) The name of the broker on which to start message flows.

If you do not specify **-e** and **-m**, all message flows on the broker are started.

-e *executionGroupName*
(Optional) The name of the execution group on which message flows are started.

-m *flowName*
(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 **-m**, you must also specify **-e**.

-n *cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

If you are using this file on z/OS, you must remove the statement `encoding="UTF-8"` from the first line, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value 1414 is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used.

-w *timeoutValue*
(Optional) This parameter is the time in seconds that the utility waits to ensure that the command completed; the default value is 60.

-v *traceFileName*
(Optional) This parameter sends internal debug trace information to the specified file.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID used to run this command must have sufficient authority defined in the access control list defined to the Configuration Manager. The permissions required are the same as the permission required to do the equivalent function in the Message Broker Toolkit

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 command are not valid.

Examples:

Start 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
```

Start 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 –n cm1.configmgr –b B1 –w 120
```

Start all message flows on broker B1, which is controlled by the Configuration Manager. The Configuration Manager is hosted by the queue manager QM1 on the local computer, listening on port 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
'-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

Use the mqsistop command to stop the specified component.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS as a console command

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- "mqsistop command - Windows, Linux, and UNIX systems"
- "mqsistop command - z/OS" on page 663

Authorization:

On Windows XP and Windows Server 2003 systems, the user ID used to run this command must be a member of the Administrators group and the mqbrkrs group on the local system.

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the Administrators group and the mqbrkrs group on the local system.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On Linux and UNIX systems, the user ID used to run this command must conform to the following requirements:

- The user ID must be a member of the mqbrkrs group.
- The user ID must be the user ID that started the component by using the mqsistart command.
- If you specify the `-q` parameter, the user ID must be a member of the mqm group.

The security requirements for using the mqsistop command are summarized in the following topics:

- "Security requirements for Windows platforms" on page 725
- "Security requirements for Linux and UNIX platforms" on page 724

On z/OS, the user ID used to run this command must have **UPDATE** authority in class **OPERCMDS** to the `MVS.STOP.STC.message_broker_component_started_task` resource.

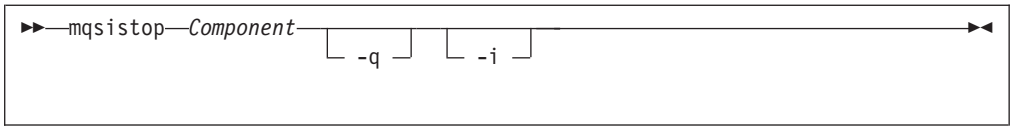
mqsistop command - Windows, Linux, and UNIX systems:

How to use the mqsistop command on Windows, Linux, and UNIX systems.

Purpose:

Use the mqsistop command to stop a WebSphere Message Broker component.

Syntax:



Parameters:

Component
(Required) The component must have a broker name, a Configuration Manager name, or one of the following fixed values:

- Windows `ConfigMgr`, `UserNameServer`, or `DatabaseInstanceMgr` on Windows platforms.
- Linux UNIX `UserNameServer` on Linux and UNIX systems.

On Linux and UNIX systems, all names are case sensitive.

Do not use the mqsistop command on a `DatabaseInstanceMgr` unless you are using the Derby database.

-q (Optional) Stops the WebSphere MQ queue manager associated with this WebSphere Message Broker component.

Specify this flag only if the WebSphere Message Broker component is the last (or only) WebSphere Message Broker component active on this queue manager. The mqsistop command initiates a controlled shutdown of the queue manager, and informs other users of the queue manager that it is closing.

Stop other WebSphere Message Broker components that use this queue manager before you issue the mqsistop command with this option; alternatively 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 the mqsistop command.

-i (Optional) Immediately stops the broker.

Specify this flag only if you have already tried, and failed, to stop the broker in a controlled fashion by using the mqsistop command without the **-i** flag.

Responses:

- 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:

To stop the broker, *mybroker*, and the WebSphere MQ queue manager associated with it:

```
mqsistop mybroker -q
```

To stop the Database Instance manager on Windows:

```
mqsistop DatabaseInstanceMgr
```

mqsistop command - z/OS:

Use the mqsistop command to stop a WebSphere Message Broker component; the controller must be running.

Purpose:

The following table distinguishes between the **stop (/P)** and **stopcomponent** commands, and lists the available options:

- Use the **stop (/P)** command to bring a component into a state in which you can run the appropriate **change** command.
- Use the **stopcomponent** command to stop a broker, Configuration Manager, or User Name Server when its controller (control process) is already running.

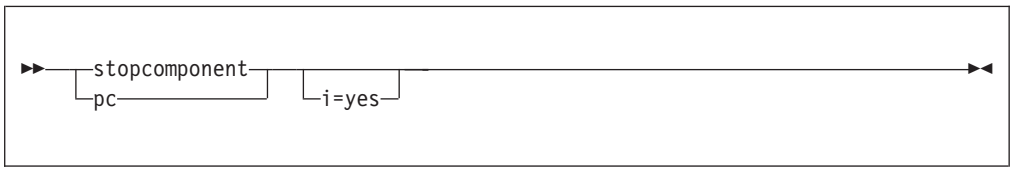
Component	Command	Description
Broker	/P <Broker started task name>	Stop broker.
	/F <Broker started task name>,P	Stop broker; this is the same as /P. You can also use /F <broker started task name>,STOP
	/F <Broker started task name>,PC	Stop broker component. This stops the broker process (including any execution group address spaces), but leaves the message broker console command server running inside the controller address space. This allows you to run the "mqsichangebroker command" on page 404 console command. Restart the broker afterwards by running the "mqsistart command" on page 654 (SC) console command.
Configuration Manager	/P <Configuration Manager started task name>	Stop Configuration Manager.
	/F <Configuration Manager started task name>,P	Stop Configuration Manager; this is the same as /P. You can also use /F <configMgr started task name>,STOP
	/F <Configuration Manager started task name>,PC	Stop Configuration Manager component. This stops the Configuration Manager process, but leaves the message broker console command server running inside the controller address space. This allows you to run the "mqsichangeconfigmgr command" on page 415 console command. Restart the broker afterwards by running the "mqsistart command" on page 654 (SC) console command.

User Name Server	/P <User Name Server started task name> /F <User Name Server started task name>,P /F <User Name Server started task name>,PC	Stop User Name Server. Stop User Name Server; this is the same as /P. You can also use /F <User Name Server started task name>,STOP Stop User Name Server started task name component. This stops the User Name Server process, but leaves the message broker console command server running inside the controller address space. This allows you to run the "mqsichangeusernameserver command" on page 499 console command. Restart the broker afterwards by running the "mqsistart command" on page 654 (SC) console command.

Syntax:

z/OS console command - stopcomponent:

Synonym: pc



Parameters:

-i (Optional) Immediately stop the broker.

Specify this flag only if you have already tried, and failed, to stop the broker in a controlled fashion by using the mqsistop command without the **-i** flag.

This command is rejected by the WebSphere Message Broker console command server if a previous stop command has failed to complete. This can occur, for example, if one or more execution group address spaces cannot shutdown.

Examples:

```
F MQ00BRK,pc
```

mqsistopmsgflow command

Use the mqsistopmsgflow command to stop message flows.

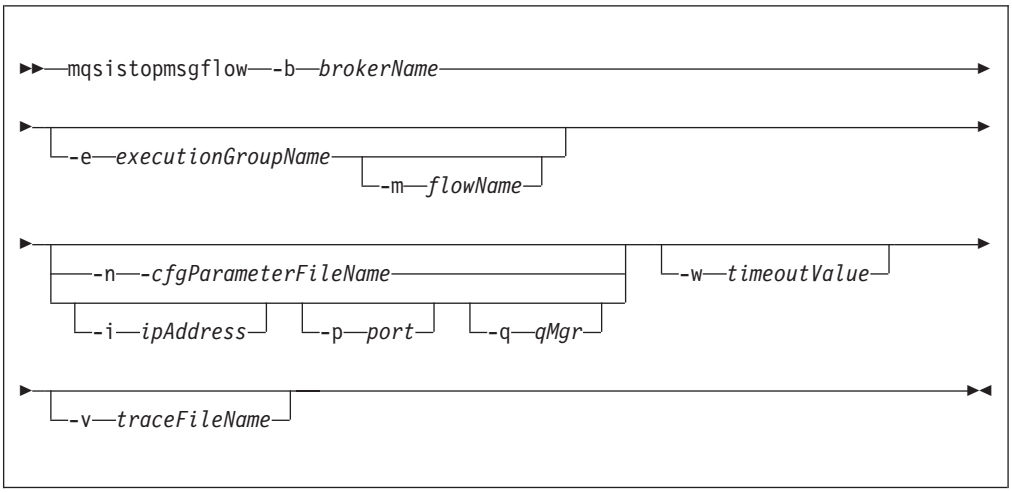
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command in one of two ways - as a console command, or by customizing and submitting BIPSPMF; see "Contents of the Configuration Manager PDSE" on page 682

Purpose:

Stop a single message flow, or all message flows in an execution group, on a specified broker. You must have previously deployed the message flows to the broker in a BAR file.

Syntax:



Parameters:

-b *brokerName*
(Required) The name of the broker on which to stop message flows.

If you do not specify **-e** and **-m**, all message flows on the broker are stopped.

-e *executionGroupName*
(Optional) The name of the execution group on which message flows are stopped.

-m *flowName*
(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 **-m**, you must also specify **-e**.

-n *-cfgParameterFileName*
(Optional) This parameter specifies the name of a `.configmgr` file that describes the connection parameters to the Configuration Manager.

The file is in XML, using the `.configmgr` format that is saved by the workbench; for example:

```
<?xml version="1.0" encoding="UTF-8"?>
<configmgr crlNameList="" domainName="" host="winmvsd0" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

If you are using this file on z/OS, you must remove the statement `encoding="UTF-8"` from the first line, and remove the value for the host attribute, to leave the statement with the following format:

```
<?xml version="1.0"?>
<configmgr crlNameList="" domainName="" host="" listenerPort="2206"
queueManager="mq06" securityExit="" securityExitJar="" sslCipherSuite="NONE"
sslDistinguishedNames="" sslKeyStore="" sslTrustStore="" svrconn="SYSTEM.BKR.CONFIG"/>
```

-i *ipAddress*
(Optional) This parameter specifies the host name or IP address of the Configuration Manager. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used, which results in a local binding connection.

-p *port*
(Optional) This parameter is the port number of the Configuration Manager. If you do not specify this parameter, the default value 1414 is used.

-q *qMgr*
(Optional) This parameter specifies the name of the WebSphere MQ queue manager that the Configuration Manager is using. If you do not specify this parameter, the default value "" (a pair of quotation marks) is used.

-w *timeoutValue*
(Optional) This parameter is the time in seconds that the utility waits to ensure that the command completed; the default value is 60.

-v *traceFileName*
(Optional) This parameter sends internal debug trace information to the specified file.

Authorization:

On Windows Vista and Windows Server 2008 systems, the user ID used to run this command must be running with elevated privileges on the local system:

- The user ID must be a member of the group Administrators.
- The command must be started from an environment that has **Run as Administrator** authority.

If you do not run the command from a privileged environment, you are asked to confirm that you want to continue. When you click **OK**, a new privileged command console is created and the command completed, but all responses are written to the privileged environment and are lost when that console closes when the command completes.

On all platforms, the user ID used to run this command must have sufficient authority defined in the access control list defined to the Configuration Manager. The permissions required are the same as the permission required to do the equivalent function in the Message Broker Toolkit

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:

Stop 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 –n 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
'-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 671
- "z/OS JCL variables" on page 683

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.

Stepname

The processes BIPSERVICE and BIPBROKER are in the same address space (control address space). After an Execution Group address space is started on z/OS, the stepname of the address space has the following value:

- The last eight characters are taken from the Execution Group label.
- Any lowercase characters are folded to upper case.
- Any non alphanumeric characters are changed to the character @.

- If the first character is not an alpha character, it is changed to A.

The stepname is not guaranteed to be a unique value. However, you are strongly recommended to ensure that the last eight characters of any Execution Group labels that are deployed to a z/OS broker are unique, and contain only alphanumeric characters; note that an Execution Group label has a maximum size of 3000 bytes.

See the following topics for more information:

- "Issuing commands to the z/OS console"
- "Guidance for issuing console commands in z/OS"
- "START and STOP commands on z/OS" on page 669

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 S/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'
```

START and STOP commands on z/OS

These examples use a broker called `MQP1BRK`. Start and stop the broker, Configuration Manager, 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 MQP1BRK,rt
```

The MVS command `START (S)` starts a broker, Configuration Manager, or User Name Server (server component) on MVS. This initiates the control address space, plus other address spaces as needed, to run that component.

If you have problems starting the broker, use the command:

```
S <broker name>,STRTP=MAN
```

This command starts the administration task but not the execution groups.

The MVS command `STOP (P)` stops a server component completely, including its control address space.

For administration purposes you can bring the server component into another state, where the control address space is still running, but all other components are stopped.

For example, this is needed in order to change broker startup parameters, see "mqsichangebroker command" on page 404. You can do this by using the `MODIFY (/F)` command on the started component, and using the `startcomponent (SC)` or

`stopcomponent (PC)` options. See “mqsistart command” on page 654 and “mqsistop command” on page 660 for more information.

Usage data on z/OS

This topic introduces SMF 89 subtype 1 records as a method of recording accumulated usage data on z/OS.

Whenever an execution group (DataFlowEngine) address space starts on z/OS, the system registers the address space for usage data collection; the system writes this usage data information to SMF 89 subtype 1 records.

When the execution group address space stops, the system deregisters the address space. The data that is collected in the SMF 89 subtype 1 records does not correspond to the data that is collected by the Accounting and Statistics SMF 117 records.

The system writes the accumulated usage data at a scheduled interval, the maximum period of which is one hour. The timing of the interval is set by the INTERVAL value that is specified for the SMF address space.

The system writes any usage data that is produced by other products to the same SMF 89 subtype 1 record.

On registering for usage data collection, the system writes the following message to the system log:

```
BIP9272I MQ05BRK default 0 THE DATAFLOWENGINE PROCESS HAS REGISTERED SMF 89
         SUBTYPE 1 RECORD COLLECTION. RETURN CODE '0', : ImbMain(316)
```

On deregistering from usage data collection, the system writes the following message to the system log:

```
BIP9273I MQ05BRK default 0 THE DATAFLOWENGINE PROCESS HAS DEREGISTERED SMF 89
         SUBTYPE 1 RECORD COLLECTION. RETURN CODE '0', : ImbMain(1079)
```

The values set in the SMF 89 subtype 1 records are defined in the following table..

Name	Description	WebSphere Message Broker value
SMF89UPO	Product owner	″IBM CORP″
SMF89UPN	Product name	″WMB″ WebSphere Message Broker ″WRF″ WebSphere Message Broker Rules and Formatter
SMF89UPV	Product version	″NOTUSAGE″
SMF89UPQ	Product qualifier	″″
SMF89UPI	Product ID	″5655-74″ for WMB ″5697-J09″ for WRF

In all cases, the values are left-aligned and padded with blanks.

All execution groups that are started and stopped on a single system provide the same values when registering and deregistering. Therefore, the usage data that is written at the end of an interval in the SMF 89 subtype 1 record is an accumulation of all execution group address spaces.

For further information about SMF 89 subtype 1 records, see the *MVS System Management Facilities (SMF)* manual.

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 Message Broker for z/OS

Decide upon a naming convention for your WebSphere Message Broker for z/OS 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 or `UNS` 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 on z/OS

Systems programmers do most of the customization of WebSphere Message Broker.

Tasks that need to be completed by other people in your organization are identified in the following table.

Role	Task
z/OS systems programmer	"Customizing the z/OS environment" on page 152
	"Setting up z/OS security" on page 73
	"Summary of required access (z/OS)" on page 673
	"Customizing UNIX System Services on z/OS" on page 163
	"Creating the broker component" on page 190
	"Starting and stopping a broker on z/OS" on page 324
	"Checking APF attributes of `bipimain` on z/OS" on page 172
DB2 administrator	"Setting up DB2 security on z/OS" on page 75
	"Summary of required access (z/OS)" on page 673
	"DB2 planning on z/OS" on page 165
	"Priming DB2" on page 189
WebSphere MQ administrator	"Setting up WebSphere MQ" on page 76
	"Summary of required access (z/OS)" on page 673
	"WebSphere MQ planning for z/OS" on page 168
	"Creating the broker component" on page 190
	"Defining the started tasks to z/OS Workload Manager (WLM)" on page 169
WebSphere Message Broker administrator	"Setting up workbench access on z/OS" on page 77
	"Summary of required access (z/OS)" on page 673
	"Creating a broker on z/OS" on page 183
	"Creating the broker component" on page 190
	"Checking APF attributes of `bipimain` on z/OS" on page 172
Performance specialist	"Defining the started tasks to z/OS Workload Manager (WLM)" on page 169
Security administrator	"Setting up z/OS security" on page 73
	"Setting up DB2 security on z/OS" on page 75
	"Summary of required access (z/OS)" on page 673
	"Creating Publish/Subscribe user IDs" on page 77
Data administrator	"Setting up z/OS security" on page 73
	"Summary of required access (z/OS)" on page 673
	"Disk space requirements on z/OS" on page 677
	"Starting and stopping a broker on z/OS" on page 324

	"Using the file system on z/OS" on page 157
	"Binding a DB2 plan to use data-sharing groups on z/OS" on page 678

Some tasks, for example defining queue security, overlap two different roles.

Summary of required access (z/OS):

The professionals in your organization require access to components and resources on z/OS.

Authorizations required for the WebSphere Message Broker started-task user ID:

The following directory authorizations are required for all WebSphere Message Broker components:

- *READ/EXECUTE* access to `<INSTPATH>`, where `<INSTPATH>` is the directory where WebSphere Message Broker for z/OS is installed by SMP/E.
- *READ/WRITE/EXECUTE* access to the component directory `++COMPONENTDIRECTORY++`.
- *READ/WRITE* access to the home directory.
- *READ/WRITE* access to the directory identified by `++HOME++`.
- In UNIX System Services, the started task user ID and the WebSphere Message 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.
- *READ/WRITE* access to RACF class `BPX.SMF`, when you need to create SMF 117 records for accounting and statistics.

The following PDSE and DB2 authorizations are required only for a broker component, that is, not a Configuration Manager or User Name Server.

READ access to the component PDSE is required.

DB2 authorizations for the started task user ID and the table owner ID are required:

- If a profile for db2subsystem.RRSAF exists 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 Message Broker component is connected to, and TASKID represents the WebSphere Message 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.
  - Model 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 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 Message Broker default configuration.

For users connecting remotely from either the Message Broker Toolkit or from a CMP API application to the Configuration Manager on z/OS, the following authorizations are required :

- Connection security: *READ* access to profile <MQ_QMNAME>.CHIN of class MQCONN. For example, for queue manager MQP1 and started task ID TASKID, use the following RACF commands:

  ```
  RDEFINE MQCONN MQP1.CHIN UACC(NONE)
  PERMIT MQP1.CHIN CLASS(MQCONN) ID(TASKID) ACCESS(READ)
  ```

- 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 user ID of the Message Broker Toolkit or CMP API application. For example, for queue manager MQP1, started task ID TASKID, and user ID USERID, use the following RACF commands:

```
RDEFINE MQADMIN MQP1.ALTERNATE.USER.USERID UACC(NONE)
PERMIT MQP1.ALTERNATE.USER.USERID  CLASS(MQADMIN) ID(TASKID) ACCESS(UPDATE)
```

Authorizations required for the WebSphere Message Broker administrator:

The broker administrator requires the following authorizations:

- *ALTER* access to the component PDSE.
- *READ*, *WRITE*, and *EXECUTE* access to the component directory ++COMPONENTDIRECTORY++.
- *READ/EXECUTE* access to <INSTPATH>, where <INSTPATH> is the directory where WebSphere Message Broker for z/OS is installed by SMP/E.
- *READ/WRITE* access to the directory identified by ++HOME++.
- In UNIX System Services, the started task user ID and the WebSphere Message 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.
- To run the DB2 pass when creating and deleting components DBADM authority for the broker database is required.

Authorizations required for the DB2 administrator:

The DB2 administrator needs to have the following authorizations to run the DB2 configuration jobs BIPCRDB and BIPDLDB:

- *ALTER* access to the component PDSE.
- DB2 authorizations: SYSCTRL or SYSADM authority.
- CREATE STOGROUP, CREATE DATABASE, and CREATE TABLESPACEs.
- DROP DATABASE and DROP STOGROUP.

If the DB2 administrator runs the DB2 pass when creating and deleting a component, the administrator user ID also needs the following authorizations. Alternatively, you can grant authorization to the WebSphere Message Broker administrator to run the DB2 pass.

- *READ*, *WRITE*, and *EXECUTE* access to the component directory ++COMPONENTDIRECTORY++.
- *READ/EXECUTE* access to <INSTPATH>, where <INSTPATH> is the directory where WebSphere Message Broker for z/OS is installed by SMP/E.
- *READ/WRITE* access to the directory identified by ++HOME++.
- In UNIX System Services, the started task user ID and the WebSphere Message 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 must give the appropriate permissions to this group.

Authorizations required for the WebSphere MQ administrator:

If the WebSphere MQ administrator runs the WebSphere MQ pass when creating a component, the administrator user ID requires the following authorizations.

Alternatively, you can grant authorization to the WebSphere Message Broker administrator to run the WebSphere MQ pass.

- *ALTER* access to the component PDSE.
- Directory authorizations:
  - *READ/EXECUTE* access to <INSTPATH>, where <INSTPATH> is the directory where WebSphere Message Broker for z/OS is installed by SMP/E.
  - *READ*, *WRITE*, and *EXECUTE* access to the component directory ++COMPONENTDIRECTORY++.
  - *READ/WRITE* access to the directory identified by ++HOME++.

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 Message 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)
  ```

- System command server: *UPDATE* access to profile <MQ_QMNAME>.queue of class MQQUEUE for SYSTEM.COMMAND.**. For example, for queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the following RACF commands to restrict access to the system command server:

  ```
  RDEFINE MQQUEUE MQP1.SYSTEM.COMMAND.** UACC(NONE)
  PERMIT MQP1.SYSTEM.COMMAND.** 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 WebSphere MQ pass when creating a component you need:
    - *ALTER* access to <MQ_QMNAME>.DEFINE.QLOCAL of class MQCMDS.
    - *ALTER* access to <MQ_QMNAME>.DEFINE.QMODEL of class MQCMDS.
    - *ALTER* access to <MQ_QMNAME>.DEFINE.CHANNEL of class MQCMDS.
  - To run the WebSphere MQ pass when deleting a component you need:
    - *ALTER* access to <MQ_QMNAME>.DELETE.QLOCAL of class MQCMDS.
    - *ALTER* access to <MQ_QMNAME>.DELETE.QMODEL of class MQCMDS.
    - *ALTER* access to <MQ_QMNAME>.DELETE.CHANNEL of class MQCMDS.

  For queue manager MQP1 and WebSphere MQ administrator ID MQADMIN, use the following RACF commands:

```
RDEFINE MQCMDS MQP1.DELETE.QLOCAL UACC(NONE)
PERMIT MQP1.DELETE.QLOCAL CLASS(MQCMDS) 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 Message Broker default configuration.

For a description of how to implement WebSphere MQ security using RACF, see "Setting up WebSphere MQ" on page 76.

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 on z/OS

The installation of WebSphere Message Broker for z/OS uses approximately 400 MB of disk space; plan on using 500 MB to allow for the component directories, and for new service fixes to be applied.

When you apply service, if you do not replace your existing installation (for example, you apply the new fix pack level alongside your existing installation), you must plan the same amount of disk space for the higher service level libraries.

If you are transferring the files by 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 a file system by 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 Message Broker for each component implemented on z/OS.

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 information includes trace files and other user problem determination data, which might 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	Collects diagnostic materials: for example, dumps. Dumps are usually more than 500 MB in size. 8 GB of space must be available in the file system, but many user IDs can have their home directory in the file system.

The Component directory and the Started task user ID home directory must be separate 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 has to write to the Component directory.

Binding a DB2 plan to use data-sharing groups on z/OS

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 on z/OS.

For further information, see:

- "Installation information - broker and User Name Server" on page 184
- "Installation information - Configuration Manager" on page 198
- "DB2 information" on page 184
- "Component information - broker" on page 185
- "Component information - Configuration Manager" on page 198
- "Component information - User Name Server" on page 207

Contents of the broker PDSE

After you have successfully customized the broker, the broker PDSE members have been set up.

Broker PDSE members originating in <hlq>.SBIPSAMP

The PDSE members are described in the following table.

Description	Name
Broker profile	BIPBPROF
Job to run the broker DB2 database backup	BIPBUDB
Broker DB2 dsnaoini file "Sample BIPDSNAO file" on page 712 lists the shipped BIPDSNAO file.	BIPDSNAO
Job to run the broker DB2 database QUIESCE	BIPQSDB
Job to run the broker DB2 database report	BIPRPDB
Job to run the broker DB2 database restore	BIPRSDB
Job to run the broker DB2 run statistics example	BIPRUNST

Broker PDSE members originating in <hlq>.SBIPPROC

The PDSE members are described in the following table.

Description	Name
Commented sample job to identify database resources, including tables and table spaces for performance tuning purposes	BIPALDB
Commented sample job to identify WebSphere MQ resources, including queues for performance tuning purposes	BIPALMQ

Job to run the mqsiapplybaroverride command	BIPOBAR
Job to run mqsibrowse command. Use this command only at the request of IBM service.	BIPBRWS
Job to run the mqsichangebroker command	BIPCHBK
Job to run the mqsichangeflowmonitoring command	BIPCHME
Job to run the mqsichangeflowstats command	BIPCHMS
Job to run the mqsichangeflowuserexits command	BIPCHUE
Job to run the mqsichangeproperties command	BIPCHPR
Job to run the mqsiclearmqpubsub command to remove an MQSeries Publish/Subscribe broker as a neighbor	BIPCLMP
Job to run the mqsicreatebroker command to: • Create the DB2 tables • Create the WebSphere MQ resources • Create the broker registry "Sample BIPCBRK file" on page 693 lists the shipped BIPCRBK file.	BIPCRBK
Job to run the mqsicreateconfigurableservice command.	BIPCRCS
Job to create the DB2 storage group, database and table spaces. "Sample BIPCRDB file" on page 702 lists the shipped BIPCRDB file.	BIPCRDB
Job to run the mqsideletebroker command	BIPDLBK
Job to drop the DB2 database and storage groups	BIPDLDB
Edit macro for customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDBK "Sample BIPEDIT file" on page 713 lists the shipped BIPEDIT file.	BIPEDIT (MQP1BRK)
Job to run the mqsiformatlog command.	BIPFMLG
Job to convert BIPBPROF profile to a valid ENVFILE. "Sample BIPGEN file" on page 714 lists the shipped BIPGEN file.	BIPGEN
Job to run the mqsijoinmqpubsub command	BIPJNMP
Job to run the mqsilist command	BIPLIST
Job to run the mqsilistmqpubsub command	BIPLSMP
Job to run the mqsimigratecomponents command	BIPMGCMP
Job to run the mqsireadbar command	BIPRBAR
Job to run the mqsireadlog command.	BIPRELG
Job to run the mqsireportbroker command	BIPRPBK
Job to run the mqsireportflowmonitoring command	BIPRPME
Job to run the mqsireportflowstats command	BIPRPMS
Job to run the mqsireportproperties command	BIPRPPR
Job to run the mqsireportflowuserexits command	BIPRPUE
Job to run the mqsisetdbparms command to define a data source, user ID, and password for user data sources.	BIPSDBP
(started task). Rename BIPBRKP to **the same as the ++STARTEDTASKNAME++**. "Sample BIPBRKP file" on page 690 lists the shipped BIPBRKP file.	BIPBRKP

Use the standard z/OS IPCS facilities to take a dump. You require access to the following resources:

- 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:

User Name Server PDSE members originating in <hlq>.SBIPSAMP

Description	Name
User Name Server profile "Sample BIPUPROF file" on page 718 lists the shipped BIPUPROF file.	BIPUROF

User Name Server PDSE members originating in <hlq>.SBIPPROC

Description	Name
Commented sample job to identify WebSphere MQ resources, including queues for performance tuning purposes	BIPALMQ
Job to run the mqsichangeusernameserver command.	BIPCHUN
Job to run the mqsicreateusernameserver command to: • Create the WebSphere MQ resources • Create the User Name Server registry "Sample BIPCRUN file" on page 710 lists the shipped BIPCRUN file.	BIPCRUN
Job to run the mqsideleteusernameserver command.	BIPDLUN
Edit macro for customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDUN. "Sample BIPEDIT file" on page 713 lists the shipped BIPEDIT file.	BIPEDIT
Job to run the mqsiformatlog command.	BIPFMLG
Generate ENVFILE. "Sample BIPGEN file" on page 714 lists the shipped BIPGEN file.	BIPGEN
Job to run the mqsilist command.	BIPLIST
Job to run the mqsireadlog command.	BIPRELG
Started task. Rename BIPUNSP to **the same as the ++STARTEDTASKNAME++**. "Sample BIPUNSP file" on page 716 lists the shipped BIPUNSP file.	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

Descriptions of the Configuration Manager PDSE members originating in <hlq>.SBIPSAMP and <hlq>.SBIPPROC are detailed in this topic.

After you have customized the Configuration Manager, the members of your PDSE are as described in the following tables:

Configuration Manager PDSE members originating in <hlq>.SBIPSAMP

Description	Name
Configuration Manager profile "Sample BIPCPROF file" on page 697 lists the shipped BIPCPROF file.	BIPCPROF

Configuration Manager PDSE members originating in <hlq>.SBIPPROC

Description	Name
Commented sample job to identify WebSphere MQ resources, including queues for performance tuning purposes	BIPALMQ
Job to run the "mqsibackupconfigmgr command" on page 403	BIPBUCM
Job to run the "mqsichangeconfigmgr command" on page 415	BIPCHCM
Job to run the "mqsicreateaclentry command" on page 504	BIPCRACL
Job to run the "mqsicreateconfigmgr command" on page 525 to: • Create the WebSphere MQ resources • Create the Configuration Manager registry "Sample BIPCRCM file" on page 700 lists the shipped BIPCRCM file.	BIPCRCM
Job to run the "mqsicreateexecutiongroup command" on page 538	BIPCREG
Job to run the "mqsideleteaclentry command" on page 548	BIPDLACL
Job to run the "mqsideleteconfigmgr command" on page 557	BIPDLCM
Job to run the "mqsideleteexecutiongroup command" on page 564	BIPDLEG
Job to run the "mqsideploy command" on page 568	BIPDPLY
Job to run the "mqsideploy command" on page 568	BIPOBAR
Job to run the "mqsideploy command" on page 568	BIPRBAR
Edit macro for customization. Rename BIPEDIT to a unique name that identifies it to the current component; for example, MQ01EDCM. "Sample BIPEDIT file" on page 713 lists the shipped BIPEDIT file.	BIPEDIT
Job to run the "mqsiformatlog command" on page 578	BIPFMLG
Job to convert BIPCPROF profile to a valid ENVFILE. "Sample BIPGEN file" on page 714 lists the shipped BIPGEN file.	BIPGEN

Description	Name
Job to run the “mqsilistaclentry command” on page 583	BIPLIACL
Job to run the “mqsilist (list resources) command” on page 581	BIPLIST
Job to run the “mqsireadlog command” on page 607	BIPRELG
Job to run the “mqsireloadsecurity command” on page 613	BIPRLSEC
Job to run the “mqsireportconfigmgr command” on page 621	BIPRPCM
Job to run the “mqsirestoreconfigmgr command” on page 644	BIPRSCM
Job to run the “mqsistopmsgflow command” on page 664	BIPSPMF
Job to run the “mqsistartmsgflowcommand” on page 657	BIPSTMF
Started task. Rename BIPCMGRP to the same as the **++STARTEDTASKNAME++**. “Sample BIPCMGRP file” on page 695 lists the shipped BIPCMGRP file.	BIPCMGRP

Use the standard z/OS IPCS facilities to create a memory dump. You require access to these resources:

- COMPONENT_PDSE
- ComponentDirectory
- SYS1.MIGLIB
- SYS1.SBLSCLI0
- SYS1.PARMLIB
- IPCS

z/OS JCL variables

The following table lists the JCL variables that you can customize in BIPEDIT, in alphabetical order, together with a description and an example value. For JCL variables that apply to specific commands, see the information that relates to that command.

JCL variable	Description	Example value	Corresponding value in BIPEDIT
++COMPONENTDATASET++	The dataset where all JCL relevant to a particular component is saved.	TESTDEV.MQP1BRK.BROKER	componentdataset_value
++COMPONENTDIRECTORY++	The file system directory where the component exists. This directory includes subdirectories, for example, /log and /registry	mqsi/brokers/MQP1BRK	compdir_value
++COMPONENTNAME++	The name you give the component when you create it.	MQP1BRK	MQ01BRK
++COMPONENTPROFILE++	Profile name.	BIPBPROF, BIPCPROF, or BIPUPROF	componentprofile_value
++DB2BUFFERPOOL++	The name of the DB2 buffer pool associated with the component for which the JCL is submitted.	BP0	BP0
++DB2CONVERSION++	Specifies the DB2 Converter.	SINGLE	SINGLE

JCL variable	Description	Example value	Corresponding value in BIPEDIT
++DB2CURRENTSQLID++	The DB2 user ID for the component and commands.	MQP1BRK	MQ01GRP
++DB2DATABASE++	The name of the DB2 database associated with the component for which the JCL is submitted.	DMQP1BRK	DMQ01BRK
++DB2HLQ++	DB2 high-level-qualifier	SYS2.DB2.V810	SYS2.DB2.V810
++DB2INDEXBP++	DB2 index buffer pool	BP0	BP0
++DB2LOBBP++	DB2 LOB table buffer pool	BP0	BP0
++DB2LOCATION++	The DB2 location value of the DB2 subsystem to which the component connects.	DSN810PK	db2location_value
++DB2DSNACLIPLAN++	The DB2 plan name.	DSNACLI	dsnacli_plan
++DB2RUNLIB++	The DB2 run library value.	DSN810PK.RUNLIB.LOAD	runlib_value
++DB2SAMPLEPROGRAM++	Sample program to run SQL statements dynamically.	DSNTEP2	sampleprogram_value
++DB2SAMPLEPROGRAMPLAN++	Plan required for sample program.	DSNTEP81	sampleprogramplan_value
++DB2STORAGEGROUP++	The name of the DB2 storage group associated with the component for which the JCL is submitted.	MQP1STOR	MQ01STOR
++DB2SUBSYSTEM++	The DB2 subsystem ID which the component connects.	DFK4	dbdsubsystem_value
++DB2TABLEOWNER++	Broker tables schema name.	MQP1BRK	MQ01GRP
++HOME++	The file system home directory for the component's user ID. This is required to dynamically generate the `ENVFILE`. You must have the appropriate external security manager, for example RACF, authorities to write to this file system directory when submitting JCL to run a command.	/u/mqp1brk	/u/mq01brk
++INSTALL++	The directory where you install the product.	/usr/lpp/mqsi	install_value
++JAVA++	Location of Java installation.	/usr/lpp/java/IBM/J1.5	/usr/lpp/java/IBM/J1.5
++LANGLETTER++	The letter for the language in which you want messages shown.	E (English)	E
++LOCALE++	Locale of environment where commands are run by submitting JCL.	C	C
++MQPATH++	WebSphere MQ location.	/usr/lpp/mqm	/usr/lpp/mqm
++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	options_value

JCL variable	Description	Example value	Corresponding value in BIPEDIT
++QUEUEMANAGER++	The name of the QueueManager associated with the component for which you submit the JCL.	MQP1	MQ01
++STARTEDTASKNAME++	Name of the Started Task JCL. This can be a maximum of 8 characters.	MQP1BRK	MQ01BRK
++TIMEZONE++	Time zome of environment where commands are run by submitting JCL.	GMT0BST	GMT0BST
++WMQHLQ++	WebSphere MQ high-level-qualifier	MQM.V600	MQM.V600
++XMLTOOLKIT++	IBM XML Toolkit location	/usr/lpp/ixm/IBM/xml4c-5_6	/usr/lpp/ixm/IBM/xml4c-5_6

z/OS sample files supplied

Several samples files are provided for you to customize on z/OS.

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.                                   *
#*                                                                 *
#*******************************************************************
#* 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++
#*                      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/J1.5'
#*
#*    Replace   ++XMLTOOLKIT++
#*                      IBM XML Toolkit location.
#*                      e.g. '/usr/lpp/ixm/IBM/xml4c-5_6'
#*
#*    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_LILPATH32 needed by components to find LILs
#     e.g. MQSI_LILPATH32=usr/lpp/mqsi/V6R0M0/lil
#
# 1.5 MQSI_SECURITY_PROVIDER_PATH32 needed by components to find LSLs
#     e.g. MQSI_SECURITY_PROVIDER_PATH32=usr/lpp/mqsi/V6R0M0/SecurityProviders
#
export MQSI_REGISTRY=++COMPONENTDIRECTORY++
export MQSI_COMPONENT_NAME=++COMPONENTNAME++
export MQSI_FILEPATH=++INSTALL++
export MQSI_LILPATH32=$MQSI_FILEPATH/lil
export MQSI_SECURITY_PROVIDER_PATH32=$MQSI_FILEPATH/SecurityProviders


#*****************************************************************
# 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/J1.5
#     Note that the Java version must be at least 1.5.0
#
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. Message Broker with Rules and Formatter Extensions 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
#     $MQSI_FILEPATH/bin directory be removed, and placed in
#     environment specific locations, to preserve multi-user
#     capability.
#     e.g. NN_CONFIG_FILE_PATH=/usr/lpp/NNSY/bin
#
# See documentation supplied with Rules and Formatter
# Extensions for further information.
#
export NNSY_ROOT=$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:$JAVAHOME/lib
CP=$CP:$MQSI_FILEPATH/messages
export CLASSPATH=$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
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
```

Note that the NNSY values apply only if you are using the Rules and Formatter Extension. You should read the documentation supplied with this feature for further information.

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.V600'
//*
//*    Replace   ++WMBHLQ++
//*                      WebSphere Message Broker
//*                      high-level-qualifier.
//*
//*                      ONLY NEEDED IF USING EVENT NOTIFICATION
//*
//*                      e.g. 'MB.V6R0M0'
//*
//*******************************************************************
//*
//* Following variables are changed when starting a DataFlowEngine:
//*     Semaphore ID
//*       SE=&lt;Semaphore ID&gt; (default is '')
//*     Shared Memory Segment ID
//*       SH=&lt;Shared Memory Segment ID&gt; (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++',
//         SE='',
//         SH='',
//         HOME='++HOME++',
//         DB2HLQ='++DB2HLQ++',
//         WMQHLQ='++WMQHLQ++'
//*
//*********************************************************************
//* 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=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/mqsicvp 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=BPXBATA8,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
//*        APF Authorized Library of Message Broker
//*        Required if using Event Notification
//*        (All librarys in concatenation
//*         need to be APF authorized)
//*        DD DISP=SHR,DSN=++WMBHLQ++.SBIPAUTH
//STDENV   DD PATH='&HOME./ENVFILE'
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//*
//         ENDIF
//*
//*-----------------------------------------------------------------
//         PEND
//*-----------------------------------------------------------------
//*
//
```

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.V600'
//*
//*********************************************************************
//*
```

```
//*****************************************************************
//* 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 *
BPXBATSL PGM -
  ++INSTALL++/bin/-
mqsicreatebroker -
  ++COMPONENTNAME++ -
  -q ++QUEUEMANAGER++ -
  -u ++DB2TABLEOWNER++ -
  -n ++DB2LOCATION++ -
  ++OPTIONS++
/*
//
```

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.V600'
//*
//*****************************************************************
//*
//++STARTEDTASKNAME++ PROC INSTP='++INSTALL++',
//         MAINP='bipimain',
//         SRVMP='bipservice',
//         COMPK='++STARTEDTASKNAME++',
//         STRTP='AUTO',
//*        COMPDIR='++COMPONENTDIRECTORY++',
//         HOME='++HOME++',
//         WMQHLQ='++WMQHLQ++'
//*
//*****************************************************************
//* 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=BPXBATSL,REGION=0M,TIME=NOLIMIT,
//         PARM='PGM &INSTP./bin/mqsicvp 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=BPXBATA8,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 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/J1.5'
#*
#*    Replace   ++XMLTOOLKIT++
#*                      IBM XML Toolkit location.
#*                      e.g. '/usr/lpp/ixm/IBM/xml4c-5_6'
#*
#*    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/J1.5
#     Note that the Java version must be at least 1.5.0
#
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:$MQSI_FILEPATH/classes/configutil.jar
CP=$CP:$MQSI_FILEPATH/messages
export CLASSPATH=$CP
XMLTOOLKIT=++XMLTOOLKIT++
LP=++MQPATH++/java/lib
LP=$LP:$MQSI_FILEPATH/lib/wbirf
LP=$LP:$MQSI_FILEPATH/lib/wbimb
LP=$LP:$MQSI_FILEPATH/lib
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   ++WMQHLQ++
//*                      WebSphere MQ high-level-qualifier.
//*                      e.g. 'MQM.V600'
//*
//*******************************************************************
//*
//*******************************************************************
//* 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
//*        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 *
BPXBATSL PGM -
  ++INSTALL++/bin/-
mqsicreateconfigmgr -
  ++COMPONENTNAME++ -
  -q ++QUEUEMANAGER++ -
  ++OPTIONS++
/*
//
```

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 buffer pool.
//*                     e.g. 'BP0'
//*
//*    Replace   ++DB2INDEXBP++
//*                     DB2 index buffer pool.
//*                     e.g. 'BP0'
//*
//*    Replace   ++DB2LOBBP++
//*                     DB2 LOB table buffer pool.
//*                     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.V600'
//*
//*********************************************************************
//*
//*********************************************************************
//* 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 *
BPXBATSL 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 BIPEDIT file

```
//* REXX  */
/*********************************************************************/
/*                                                                   */
/*  @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                  */
/*                                                                   */
/* REXX utility to customize JCL.                                    */
/*                                                                   */
/*********************************************************************/
/* MORE INFORMATION - See:                                           */
/*                                                                   */
/*    WebSphere Event/Message Brokers Information Centre.            */
/*                                                                   */
/*********************************************************************/
/*                                                                   */
/*  This edit macro can be used to assist you in configuring your    */
/*  Broker configuration files.                                      */
/*  1. Edit this file to alter the change commands so the second     */
/*     parameter is the value for your installaton                   */
/*  2. Save the file                                                 */
/*  3. Rename it to a broker related file, for example VCP1EDIT for  */
/*     Broker called VCP1BRK                                         */
/*  4. In TSO or TSO command shell ( often ISPF option 6) execute    */
/*     ALTLIB ACTIVATE APPLICATION(EXEC) DA('WBI.V5.SBIPPROC')       */
/*     where WBI.V5.SBIPPROC is the name of the PDS contaning this   */
/*     macro                                                         */
/*  5. Using the same ISPF session, edit a configuration file        */
/*  6. Type the macro name at the commands line and press enter      */
/*     This will execute the macro.                                  */
/*     If you get Command not found, then check you issued the command */
/*     in 4. in the same ISPF session                                */
/*  7. If the macro has any errors, cancel from the file being edited */
/*     resolve the errors in the macro and retry                     */
/*                                                                   */
/*********************************************************************/
/*   See the product docmentation on the meaning of the fields below=*/
/*********************************************************************/
ISREDIT MACRO NOPROCESS
ADDRESS ISREDIT
/* All components (Broker , Configuration Manager and User Name Server) */
"change ++INSTALL++ install_value                    all"
"change ++COMPONENTDIRECTORY++ compdir_value         all"
"change ++COMPONENTNAME++ MQ01BRK                    all"
"change ++HOME++ /u/mq01brk                          all"
"change ++OPTIONS++ options_value                    all"
"change ++LOCALE++ C                                 all"
"change ++TIMEZONE++ GMT0BST                         all"
```

```
"change ++JAVA++ /usr/lpp/java/IBM/J1.5                  all"
"change ++WMQHLQ++ MQM.V600                              all"
"change ++QUEUEMANAGER++ MQ01                            all"
"change ++COMPONENTDATASET++ componentdataset_value      all"
"change ++STARTEDTASKNAME++ MQ01BRK                      all"
"change ++COMPONENTPROFILE++ componentprofile_value      all"
"change ++XMLTOOLKIT++ /usr/lpp/ixm/IBM/xml4c-5_6        all"
/* Broker only                                                          */
/* (Delete if Configuration Manager or User Name Server)                */
"change ++DB2CONVERSION++ SINGLE                         all"
"change ++DB2SUBSYSTEM++ dbdsubsystem_value              all"
"change ++DB2LOCATION++ db2location_value                all"
"change ++DB2TABLEOWNER++ MQ01GRP                        all"
"change ++DB2CURRENTSQLID++ MQ01GRP                      all"
"change ++DB2DSNACLIPLAN++ dsnacli_value                 all"
"change ++DB2HLQ++ SYS2.DB2.V810                         all"
"change ++DB2RUNLIB++ runlib_value                       all"
"change ++DB2SAMPLEPROGRAM++ sampleprogram_value         all"
"change ++DB2SAMPLEPROGRAMPLAN++ sampleprogramplan_value all"
"change ++DB2DATABASE++ DMQ01BRK                         all"
"change ++DB2STORAGEGROUP++ MQ01STOR                     all"
"change ++DB2BUFFERPOOL++ BP0                            all"
"change ++DB2INDEXBP++ BP0                               all"
"change ++DB2LOBBP++ BP0                                 all"
/* Configuration Manager only                                           */
/* (Delete if Broker or User Name Server)                               */
"change ++MQPATH++ /usr/lpp/mqm                          all"
```

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 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.V600'
//*
//*************************************************************
//*
//++STARTEDTASKNAME++ PROC INSTP='++INSTALL++',
//         MAINP='bipimain',
//         SRVMP='bipservice',
//         COMPK='++STARTEDTASKNAME++',
//         STRTP='AUTO',
//         COMPDIR='++COMPONENTDIRECTORY++',
//         STDD='OTRUNC',
//         HOME='++HOME++',
//         WMQHLQ='++WMQHLQ++'
//*
//*************************************************************
```

```
//* 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=BPXBATSL,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 PATHOPTS=(OWRONLY,OCREAT,&STDD),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='&COMPDIR./output/stdout'
//STDERR   DD PATHOPTS=(OWRONLY,OCREAT,&STDD),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='&COMPDIR./output/stderr'
//SYSMDUMP DD SYSOUT=*
//*
//*********************************************************************
//* Copy stdout to SYSOUT
//*********************************************************************
//*
//COPYOUT  EXEC PGM=IKJEFT01,COND=EVEN,
//         PARM='OCOPY INDD(BIPFROM) OUTDD(STDOUT)'
//SYSTSPRT DD DUMMY
```

```
//BIPFROM  DD PATHOPTS=(ORDONLY),
//            PATH='&COMPDIR./output/stdout'
//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='&COMPDIR./output/stderr'
//STDERR   DD SYSOUT=*,DCB=(RECFM=V,LRECL=256)
//SYSTSIN  DD DUMMY
//*
//         ENDIF
//*
//*--------------------------------------------------------------
//         PEND
//*--------------------------------------------------------------
//*
//
```

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/J1.5'
#*
#*    Replace   ++XMLTOOLKIT++
#*                      IBM XML Toolkit location.
#*                      e.g. '/usr/lpp/ixm/IBM/xml4c-5_6'
#*
#*********************************************************************
# 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/J1.5
#     Note that the Java version must be at least 1.5.0
#
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:$MQSI_FILEPATH/messages
CP=$CP:$JAVAHOME/lib
export CLASSPATH=$CP
XMLTOOLKIT=++XMLTOOLKIT++
LP=$MQSI_FILEPATH/lib/wbirf
LP=$LP:$MQSI_FILEPATH/lib/wbimb
LP=$LP:$MQSI_FILEPATH/lib
LP=$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
```

Security requirements for administrative tasks

You can configure Access Control List (ACL) entries to govern which users and groups can manipulate objects in the broker domain. Security requirements for administrative tasks depend on the platform that you use.

This section contains the following topics:

ACL permissions

WebSphere Message Broker uses Access Control List (ACL) entries to govern which users and groups can manipulate objects in the broker domain. 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 following table lists 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 • Full control permission for all brokers.
	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.

Object	Permission	Rights
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.
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.
Configuration Manager Proxy	Full control	• Full control permissions for Topology. • Full control permissions for Root topic. • Full control permissions for Subscriptions.
	View	• View topology configuration and any managed subcomponents. • View all topics, including child topics, and any managed subcomponents. • View or query all subscriptions and any managed subcomponents.

Security requirements for Linux and UNIX platforms

View a summary of the authorizations in a Linux or UNIX environment.

Make sure that you add the required user IDs to the appropriate group to enable them to complete the relevant tasks.

User is...	Linux or UNIX domain
Creating a component	• Member of mqbrkrs and mqm. • When root is used to run the create command, it can nominate any user to run the component.
Installing	• Superuser.
Uninstalling	• Superuser.
Changing a component	• Member of mqbrkrs.

User is...	Linux or UNIX domain
Viewing the properties of a component	• Member of mqbrkrs and mqm.
Deleting a component	• Member of mqbrkrs and mqm.
Starting a component, or verifying a broker or Configuration Manager	• Member of mqbrkrs.
Stopping a component	• Member of mqbrkrs. • The user ID must be the same as the user ID that started the component. • Member of mqm if -q is specified.
Listing a component	• Member of mqbrkrs.
Changing, displaying, retrieving trace information.	• Member of mqbrkrs.
Running a User Name Server (login ID).	• Member of mqbrkrs. The User Name Server runs under the login ID specified in the create command.
Running a broker (WebSphere MQ non-trusted application) (login ID).	• Member of mqbrkrs. • The broker runs under the login ID that started it. • If root started the component, the broker runs under the login ID specified as the service ID in the create command.
Running a 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.
Running publish/subscribe applications.	• Any user, subject to topic and WebSphere MQ queue access control.

Ensure that mqbrkrs has access to all user-defined queues that are referenced in your message flows. You can use the setmqaut command to set permissions.

Set the following permissions on all input queues:

```
setmqaut -m WBRK_BROKER -n TEST_INPUT -t queue -g mqbrkrs +get +inq
```

Set the following permissions on all output queues:

```
setmqaut -m WBRK61_DEFAULT_QUEUE_MANAGER -n TEST_OUTPUT -t queue -g mqbrkrs  +put +inq +setall
```

You might also need to add **+passid +passall +setid +setall**, depending on your requirements.

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

Security requirements for Windows platforms

Security requirements depend on the administrative task that you want to perform.

The following table summarizes the requirements for administrative tasks. It shows 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 a broker, Configuration Manager, User Name Server, or database (with mqsicreatedb)	• 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 a 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
Viewing the properties of a broker, Configuration Manager, User Name Server, or database (with mqsideletedb)	• Member of Administrators	• Member of SALONE\Administrators	• Member of SALONE\Administrators
Deleting a broker, Configuration Manager, User Name Server, or database (with mqsideletedb)	• Member of Administrators	• Member of SALONE\Administrators	• Member of SALONE\Administrators
Starting a broker, Configuration Manager, User Name Server, or DatabaseInstanceMgr, or running the verification command mqsicvp	• Member of Administrators and mqbrkrs	• Member of SALONE\Administrators and member of PRIMARY\Domain mqbrkrs	• Member of SALONE\Administrators and member of TRUSTED\Domain mqbrkrs
Listing a 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
Running a User Name Server (service user ID)	• Must be a user ID defined in SALONE • Member of mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy

User is...	Local domain (SALONE)	Primary domain (PRIMARY) / Windows Single domain (PRIMARY)	Trusted domain (TRUSTED) / Windows Parent/Child domain in domain tree (TRUSTED)
Running a 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 a Configuration Manager (service user ID)	• Must be a user ID defined in SALONE • Member of mqbrkrs • Member of Adminstrators • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Member of SALONE/Adminstrators • Must have the *Logon as a service* privilege in the Windows Local Security Policy
Running a broker (WebSphere MQ fastpath off) (service user ID)	• Must be a user ID defined in SALONE • Member of mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Must have the *Logon as a service* privilege in the Windows Local Security Policy
Running a broker (WebSphere MQ fastpath on) (service user ID)	• Must be a user ID defined in SALONE • Member of mqbrkrs • Member of mqm • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Member of SALONE\mqm • Must have the *Logon as a service* privilege in the Windows Local Security Policy	• Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Member of SALONE\mqm • Must have the *Logon as a service* privilege in the Windows Local Security Policy
Clearing, joining, or 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 a Message Broker Toolkit[1]	• Must be a user ID defined in SALONE[2]. For example, SALONE\User1 is valid, PRIMARY\User2 and TRUSTED\User3 are not.	• Regardless of whether domain awareness is enabled, when using Message Broker Toolkit ACLs, user IDs must be members of any local ACL groups created on SALONE.	• Regardless of whether domain awareness is enabled, when using Message Broker 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.

Ensure that mqbrkrs has access to any user-defined queues defined in your message flows. You can use the setmqaut command to set permissions.

Set the following permissions on any input queues:

```
setmqaut -m WBRK61_DEFAULT_QUEUE_MANAGER -n TEST_INPUT -t queue -g mqbrkrs  +get +inq
```

Set the following permissions on any output queues:

```
setmqaut -m WBRK61_DEFAULT_QUEUE_MANAGER -n TEST_OUTPUT -t queue -g mqbrkrs +put +inq +setall
```

You might also need to add **+passid +passall +setid +setall**, depending on your requirements.

Notes:

1. All Message Broker Toolkit users need read access to the WebSphere MQ java \lib subdirectory of the WebSphere MQ home directory (the default location 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 Message Broker installation overrides this restriction and gives read access for this subdirectory to all users.
2. 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 Broker Toolkit with the authorities of PRIMARY\User4.

The following general notes also apply:

1. 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 you have set a workpath for any component to a non-default value, ensure that the services user ID has appropriate access to this location.
2. 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 and message sets to the broker. To avoid this error:
   - Define the user ID for the broker on the computer where the Configuration Manager is running.
   - Define the user ID for the Configuration Manager on the computer where the broker is running.
   - Ensure that all user IDs are in lowercase so that they are compatible between computers.

Broker security requirements on Windows Vista and Windows Server 2008

On Windows Vista and Windows Server 2008, the service user ID must be a member of the Administrators group.

Broker security requirements on Windows XP and Windows Server 2003

On Windows XP and Windows Server 2003, the service user ID must be a member of the mqbrkrs group and optionally a member of the Administrators group. As a member of the Administrators group, the service user ID has permission to access the registry keys of the broker so that it can access broker information. If the service user ID does not belong to the Administrators group, you can edit the Windows registry so that the service user ID can access the registry keys without having Administrators permissions.

The instructions for both operating systems are identical except where stated.

1. Click **Start** → **Run**, enter `regedit`, and click **OK**. The Registry Editor opens.
2. In the left pane, navigate to `HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebSphereMQIntegrator` (on 32–bit operating system editions) or `HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\IBM\WebSphereMQIntegrator` (for 64–bit operating system editions).
3. Right-click **WebSphereMQIntegrator** and select **Permissions**. The Permissions for WebSphereMQIntegrator window opens.
4. Click **Add** below the list of Group or user names. The Select Users or Groups window opens.
5. Click **Advanced** and then **Find Now** to list the current users and groups. From the list, select the mqbrkrs group to highlight it, and click **OK** twice.
6. Click **Advanced** on the Permissions for WebSphereMQIntegrator window to set special permissions. The Advanced Security Settings for WebSphereMQIntegrator window opens.
7. Highlight mqbrkrs and click **Edit**. The Permission Entry for WebSphereMQIntegrator window opens.
8. Select **Set Value**, **Create Subkey**, and **Delete** and click **OK**.
9. Ensure on Windows Server 2003 that **Allow inheritable permissions** is selected, or on Windows XP that **Inherit from parent** is selected, and click **OK**.
10. Click **OK** to close the remaining windows, and then close the Registry Editor.

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
Viewing properties of a 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
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

User is...	File access
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.

Part 5. Appendixes

Appendix. Notices for WebSphere Message Broker

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 in the WebSphere Message Broker information center

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at `www.ibm.com/legal/copytrade.shtml`.

Adobe is a registered trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Intel, Itanium, and Pentium are trademarks of Intel Corporation in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

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

Index

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

W

X

Z

Printed in USA