

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 of IBM WebSphere Message Broker and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2000, 2007. All rights reserved.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this topic collection. v

Part 1. Security 1

Security. 3

Security overview 3
Planning for security when you install WebSphere
Message Broker 28
Setting up message flow security 28
Setting up broker domain security. 45
Setting up z/OS security 65
Publish/subscribe security 70
Securing the publish/subscribe domain 78

**Part 2. Configuring the broker
domain 87**

**Configuring WebSphere Message
Broker 89**

Planning a broker domain 89
Configuring databases. 99
Customizing the z/OS environment. 137
Configuring broker domain components 162
Configuring a broker domain in the workbench 222
Configuring a publish/subscribe topology. 235
Configuring global coordination of transactions
(two-phase commit) 253
Configuring the workbench 270
Changing locales 273

**Part 3. Administering the broker
domain 281**

Administering the broker domain. . . 283

Connecting to and disconnecting from the broker
domain 283
Connecting to and disconnecting from the broker
domain on z/OS 284
Starting and stopping message flows 286
Starting and stopping a broker 287
Starting and stopping a Configuration Manager 288
Starting and stopping the User Name Server 290

Starting a WebSphere MQ queue manager as a
Windows service 291
Stopping a WebSphere MQ queue manager when
you stop a component 292
Viewing broker domain log information 293
Refreshing broker domain log information. 293
Filtering broker domain log information 294
Saving broker domain log information 295
Clearing broker domain log information 296
Changing the location of the work path for
WebSphere Message Broker 296
Changing Event Log editor preferences. 298
Backing up resources. 298

Part 4. Reference 303

Databases. 305

odbc32.ini sample file 305
odbc64.ini sample file. 311

Operations 323

Broker properties 323
Commands 324
z/OS specific information 537

**Security requirements for
administrative tasks 591**

ACL permissions 591
Security requirements for Linux and UNIX
platforms. 592
Security requirements for Windows platforms 593
Security requirements for z/OS 596

Part 5. Appendixes 599

**Appendix. Notices for WebSphere
Message Broker 601**

Trademarks in the WebSphere Message Broker
information center. 603

Index 605

About this topic collection

This PDF has been created from the WebSphere Message Broker Version 6.1 (December 2007) 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 these attempt to link to a PDF that is called after the topic identifier (for example, ac12340_.pdf) and therefore fail. Use the online information to navigate freely between topics.

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

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

Part 1. Security

Security	3
Security overview	3
Message flow security	3
Authorization for configuration tasks	16
Security for development resources	16
Authorization to access runtime resources	16
Access Control Lists	18
Security exits	20
Public key cryptography	21
Digital certificates	22
Digital signatures	24
SSL authentication	25
Tunneling	26
Quality of protection	27
Domain awareness	27
Planning for security when you install WebSphere	
Message Broker	28
Setting up message flow security	28
Creating a security profile	29
Configuring identity extraction	34
Configuring authentication	36
Configuring identity mapping	38
Configuring authorization	39
Configuring for identity propagation	44
Setting up broker domain security	45
Creating user IDs	45
Considering security for the workbench	46
Considering security for a broker	48
Considering security for a Configuration Manager	50
Changing the security domain for the User Name Server	52
Implementing SSL authentication	52
Enabling topic-based security	61
Using security exits	63
Implementing HTTP tunneling	64
Implementing quality of protection	64
Database security	64
Setting up z/OS security	65
Setting up DB2 security on z/OS	67
Setting up WebSphere MQ	68
Setting up workbench access on z/OS	69
Creating Publish/Subscribe user IDs	69
Enabling the Configuration Manager on z/OS to obtain user ID information	69
Publish/subscribe security	70
Topic-based security	70
Authentication services	75
Message protection	77
Securing the publish/subscribe domain	78
Enabling topic-based security	78
Creating ACL entries	81
Enabling SSL for the Real-time nodes	81
Using message protection	84
Securing WebSphere MQ resources	84

Security

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 28
- “Setting up broker domain security” on page 45
- “Setting up z/OS security” on page 65
- “Publish/subscribe security” on page 70
- “Securing the publish/subscribe domain” on page 78

Security overview

An important aspect of securing an enterprise system is the ability to protect the information passed from third parties. This 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, there are properties that you need to set in the HTTP nodes. For information on this, see HTTPInput node, HTTPRequest node, and HTTPReply node.

There is some security configuration 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 is split into several areas:

- “Message flow security”
- WS-Security
- “Authorization for configuration tasks” on page 16
- “Security for development resources” on page 16
- “Authorization to access runtime resources” on page 16
- “Security exits” on page 20
- “SSL authentication” on page 25 (for the Real Time node only)
- “Tunneling” on page 26
- “Quality of protection” on page 27
- “Domain awareness” on page 27.

Message flow security

This section introduces the concepts related to message flow security. You can configure the broker to perform end-to-end processing of an identity carried in a message through a message flow. This enables you to configure security at flow level to control access based on the identity flowed in a message, and to provide a security mechanism that is both transport independent and message format independent.

If you do not enable message flow security, the default security facilities in WebSphere Message Broker are based on transport specific facilities. In this case,

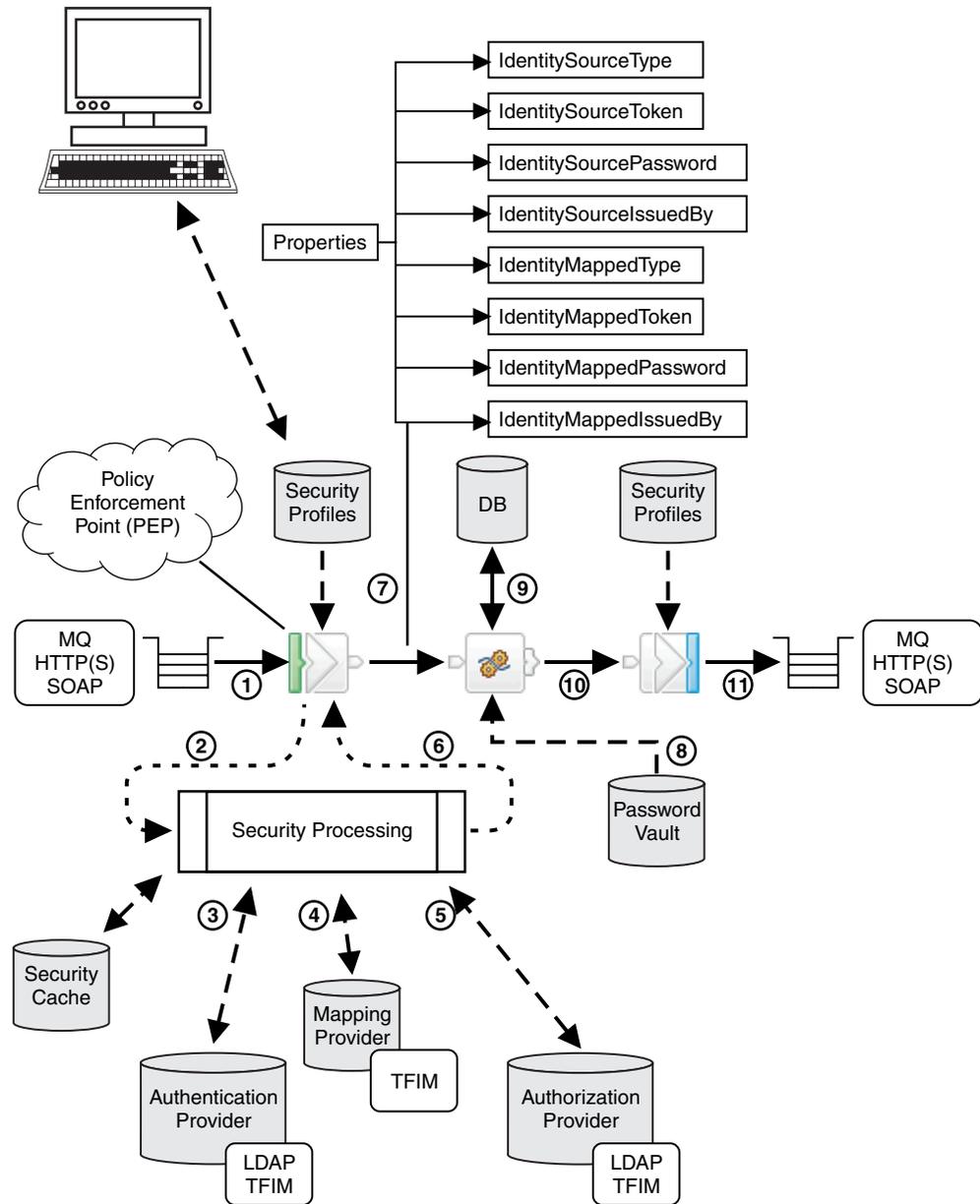
the broker processes all messages that can be physically delivered to it using the broker service identity as a proxy identity for all message instances. Any identity present in the incoming message is not used, preserved, or passed along.

WebSphere Message Broker Version 6.1 supports message flow security, and provides a security manager. The security manager enables access to message flows to be controlled on a per message basis using the identity of the message. Instead of delegating this authority to the transport, the broker can:

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

The actions taken for a given message flow are controlled using “Security profiles” on page 7, which are created by the broker administrator and are accessed by the security manager at runtime. In Version 6.1 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 in WebSphere Message Broker Version 6.1 are MQInput, HTTPInput, and SOAPInput. The output nodes that support identity propagation in Version 6.1 are MQOutput, HTTPRequest, SOAPRequest and SOAPAsyncRequest. If the message flow is a Web Service implemented using the Broker SOAP nodes, the Identity can be taken from the WS-Security header tokens 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, and correspond to the parts of the diagram above:

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 using either the `mqscreateconfigurable` service command or an editor in the Broker Administration perspective of the Message Broker Toolkit. You then use the Broker Archive editor to configure the security profile on the message flow.

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.

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 could be in a message header or in the message body itself, or a mixture of the two.
3. If authentication was specified in the security profile, the security manager calls the provider to authenticate the identity. A failure results in a SecurityException being thrown. Supported providers in Version 6.1 are LDAP and TFIM.
4. If identity mapping was 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 supported provider in Version 6.1 is TFIM.
5. If authorization was 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 thrown. Supported providers in Version 6.1 are LDAP and TFIM.
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 need to be used to access a resource such as a database. In WebSphere Message Broker Version 6.1, the identity used to access such a resource continues to be a proxy identity, either the broker's identity or an identity configured using the mqsisetdbparms command.
9. The resource is accessed using the appropriate proxy identity.
Flow developers can use the identity fields in the properties folder for their 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 is used to indicate 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, or if that is not set, the source identity is used. If no identity is set a SecurityException is thrown.
11. Any 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 mqsisreloadsecurity command to reload the cache and the mqsischangeproperties command to set the expiry and sweep intervals for the cache.

For a SOAPRequest or SOAPAsyncRequest node, 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:

- “Identity” on page 8
- “Authentication” on page 10
- “Identity mapping” on page 11
- “Authorization” on page 10
- “Identity propagation” on page 15
- “Security profiles”
- “Security exception processing” on page 15
- “Setting up message flow security” on page 28

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 prior to deploying a message flow, and are accessed by the security manager at runtime.

A security profile allows a broker administrator to specify whether or not identity propagation, authentication, authorization, and mapping are to be performed on the identity of messages in the message flow, and if so, which external security provider will be 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 and output/request nodes, and are configured by the administrator at deployment time in the Broker Archive editor. Input and output/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 or not runtime security is configured. If the property does not resolve to a real security profile, identity tokens will not be extracted to the identity fields in the properties folder.

The security profile also specifies whether or not propagation is required. A pre-configured profile specifying propagation is provided for use by output/request nodes. This 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 to input nodes. For more information, see “Authentication” on page 10.

authenticationConfig

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

mapping

Defines the type of mapping that is performed on the source identity. This property applies to input nodes. For more information, see “Identity mapping” on page 11.

mappingConfig

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

authorization

Defines the types of authorization checks that are performed on the mapped or source identity. This property applies to input nodes. For more information, see “Authorization” on page 10.

authorizationConfig

This is a provider-specific configuration string. It typically provides information on how the broker connects to the provider, and also some information that can be used to check access (for example, a group that can be checked for membership). This property applies to input nodes.

passwordValue

Defines how passwords are treated when they enter a 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 to input nodes.

Propagation

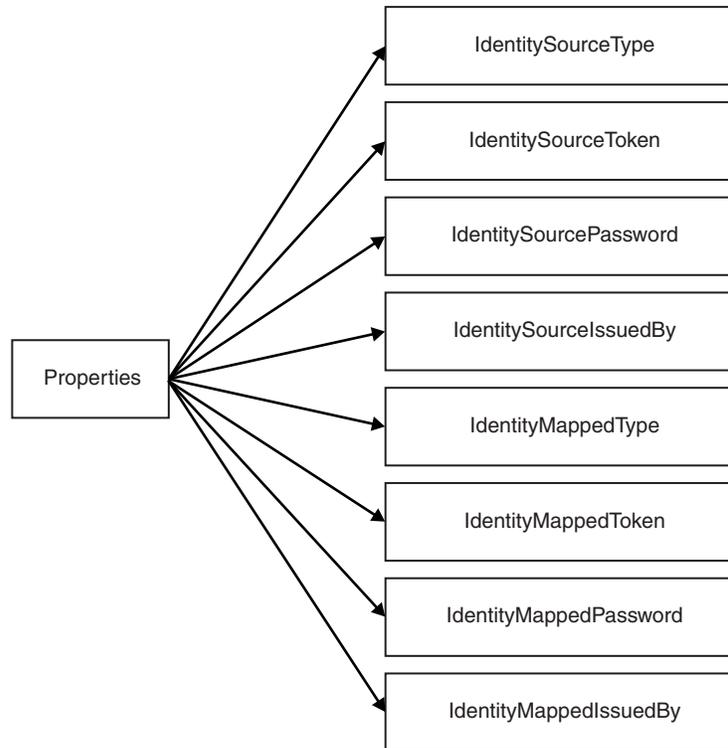
Enables or disables identity propagation on output and request nodes. For more information, see “Identity propagation” on page 15.

For information on configuring a security profile for LDAP or TFIM, see “Creating a security profile” on page 29.

Identity

This topic introduces the concept of identity in the message broker.

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 can be in the form of *Username*, *Username + Password* or *X.509 Certificate*. The Type property can also be *Transport Default*, which causes the transport's default to be used. For MQ the default is *Username* and for HTTP the default is *Username + Password*.

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

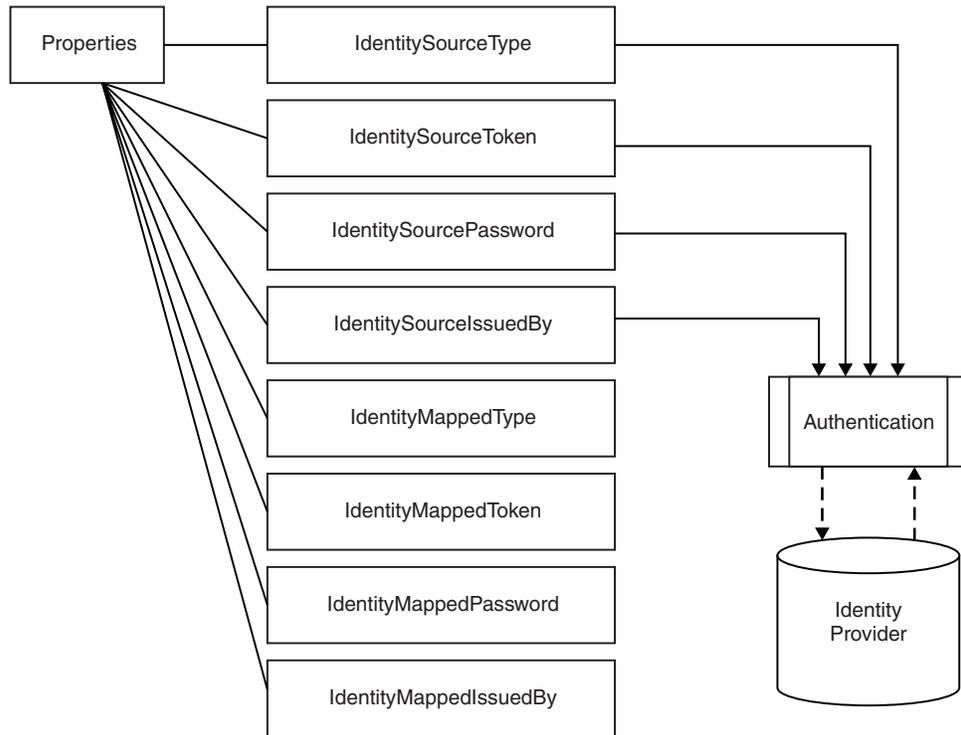
The source identity is only set by the input node if a security profile is associated with the node. The information to fill these fields is commonly found in the headers of a message but can also be located in the body, provided the node has been configured with an ESQL Path or XPath reference for the 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 are: Websphere MQ, HTTP(S), and SOAP. See MQInput node and HTTPInput node.

The values in the properties are can be modified (for example, from ESQL), although it is not recommended to write to the IdentitySource* values. This ability could be used to 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 the Broker, Authentication is the process of passing the Source Type and Token to an external authenticator. For more information about Source Type and Token, see “Identity” on page 8.



The external identity provider checks the identities and returns a value to report whether or not the identity is authentic. If the identity is not authentic a security exception is raised.

Some identity providers, including LDAP, support only a single type of authentication token. If a token of another type is passed in, an exception is raised. For example, LDAP supports only a Username + 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.

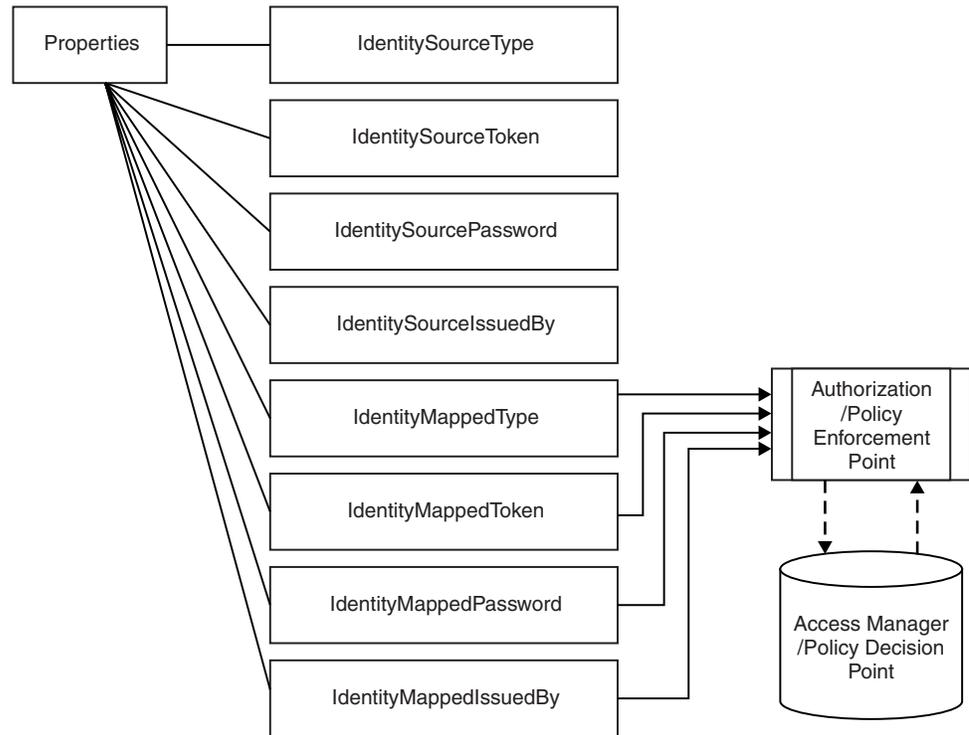
You can also select TFIM for authentication in the profile, but the TFIM module chain is invoked only once and the single module chain must be configured to perform all the required authentication, mapping, and authorization operations.

For more information about using TFIM for authentication, see “Authentication, Mapping, and Authorization with TFIM and TAM” on page 13.

Authorization

Authorization is the act of verifying that an identity has permission to execute a message flow.

If authentication and mapping are configured, they are applied to the identity to verify it before authorization of the identity is performed.



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

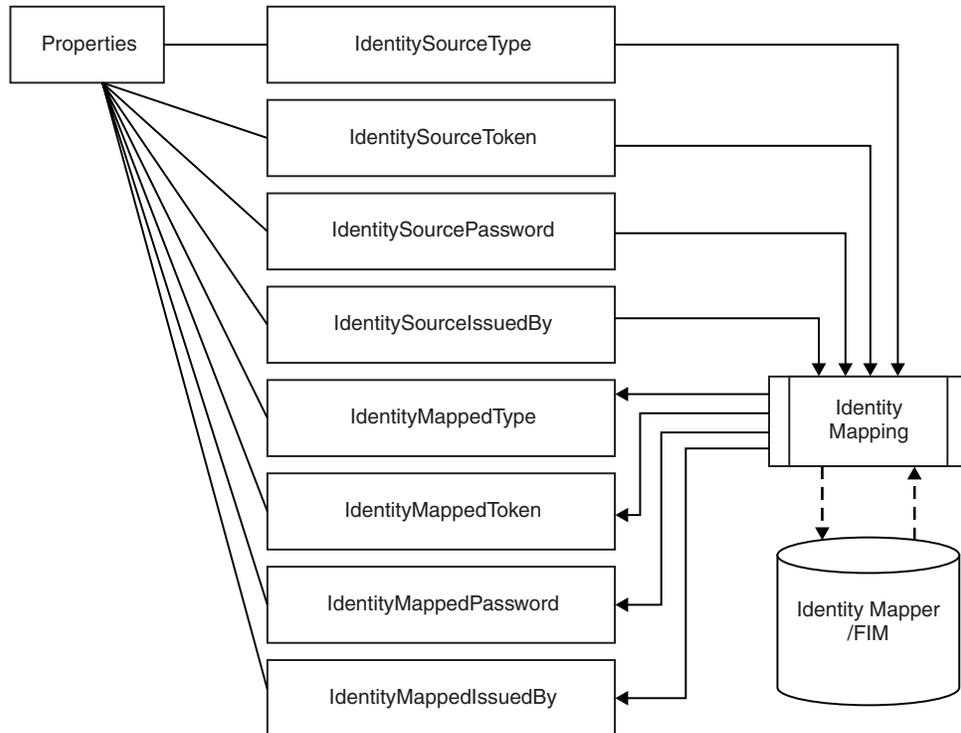
If you want to use LDAP for authorization, the identity must be a member of the relevant group configured in the security profile. The LDAP server must be LDAP Version 3 compliant.

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 13.

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

In WebSphere Message Broker Version 6.1 the supported mapper is IBM Tivoli Federated Identity Manager (TFIM). Mapping is performed only for input nodes with an associated security profile that selects TFIM 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 Version 6.1 supports the mapping from a username token to a Username token, and the mapping from an X.509 certificate to a Username token. There is no support for mapping to an X.509 token. When mapping from an X.509 certificate, TFIM can validate the certificate but cannot be used to 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 13.

User-defined mapping

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

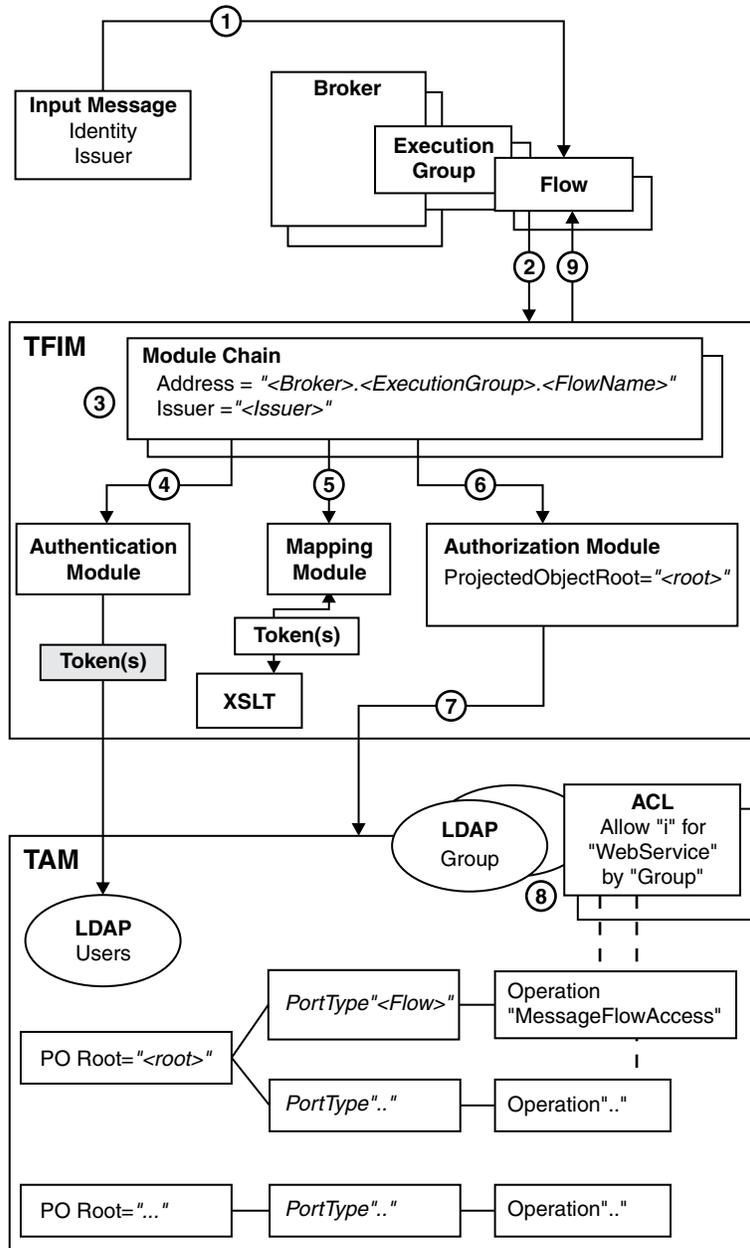
the Java Compute node you read the source identity values from the properties folder, process them, and then write the new identity values into the mapped identity fields. The mapped identity fields are then used in preference to the source identity fields by following nodes. However, in this case, any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

Authentication, Mapping, and Authorization with TFIM and TAM

This topic outlines the process of authentication, mapping, and authorization using Message Broker, TFIM, and TAM.

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

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



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

1. A message enters a 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 UsernameTokenSTSMModule or X509STSMModule) 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 AuthorizationSTSMModule in *other* mode. It must be configured with a *Protected Object Root* value.
7. The AuthorizationSTSMModule module 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 AuthorizationSTSMModule; and <FlowName> and "MessageFlowAccess" are the WS-Trust request PortType and Operation values.
8. TAM processes the authorization request by:
 - a. Finding the 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 is the result of a mapping request, it 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

This topic describes the propagation of an identity through the message flow.

In an enterprise system, different physical identities (such as usernames and certificates) can be used 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 username token might be required for back-end 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. The broker distinguishes between Session-based and Message-based security.

For more information see "Identity mapping" on page 11.

Security exception processing

This topic describes the processing of security exceptions in the broker. A security exception is raised when a runtime security failure occurs when carrying out security processing in an input node.

Security exceptions are processed in a different way to other errors on the input node. Whilst an error is usually caught on the input node and routed down the failure terminal for error processing, security exceptions are not processed in the same way. By default the broker does not allow security exceptions to be caught within the flow, and instead it backs the message out or returns an error (in the case of HTTP). 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 “Treat Security Exceptions as normal exceptions” on the input nodes. This causes security exceptions to be processed like any other exception in the message flow.

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 416.
- 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 61

Authorization for existing resources

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

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

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

Security for development resources

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

Authorization to access runtime resources

Runtime resources are WebSphere Message Broker objects that exist at run time in the broker domain. Each runtime object has an Access Control List (ACL) which determines which users and groups can access the object. The ACL entries for an

object can permit a user or group to view the object or view and modify the object from the workbench, the command line, or using the Configuration Manager Proxy (CMP).

ACLs allow or deny access for a user to an object but ACL entries do not secure the object; that is, the ACL entry cannot verify the user's identity.

Using ACL entries, you can control users' access to specific objects in the broker domain. For example, user JUNGLE\MPERRY might be given access to modify BROKERA, but have no access rights to BROKERB. In a further example the same user might have access to deploy to execution group EXEGRP1, but not to EXEGRP2, even though they are both members of BROKERA.

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. If your user ID is included in the ACL entry for the named object, you are authorized to perform the operation.

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

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.

Access Control Lists

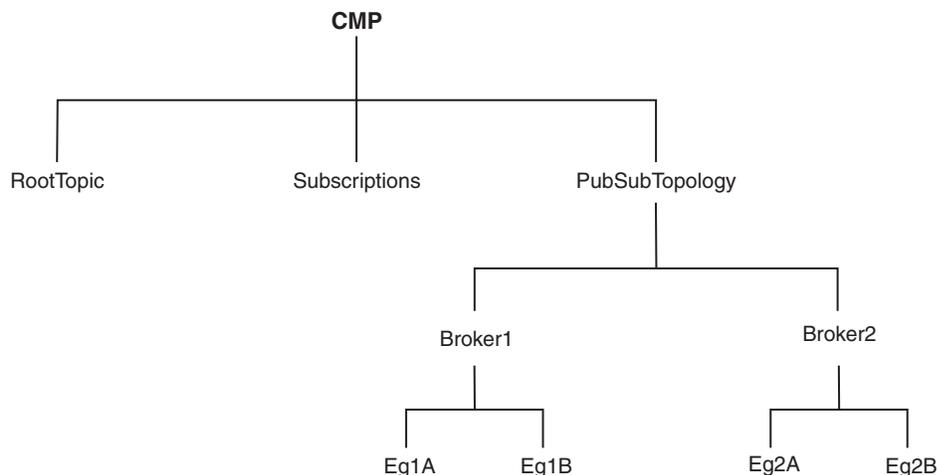
Access Control List (ACL) entries allow or deny a user to access an object. ACL entries do not, however, secure the object because the ACL entry cannot verify the user's identity. An ACL entry contains the user name and can specify a host name or domain name. It is possible, for example, for a user to get 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.

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; see "ACL permissions" on page 591 for a list of the permissions that can be applied to each object type and a summary of the actions that the user or group can perform.

To reduce the number of access control entries that a broker administrator must create, the ACL permissions behave in a hierarchical manner. The root of the tree is the ConfigManangerProxy object, which has three children: RootTopic, Subscriptions, and PubSubTopology. The PubSubTopology object has zero or more brokers as children, and each broker can have zero or more execution groups as children. When an ACL entry is added to a given object, permission is granted to that object and to all objects beneath it in the hierarchy, unless it is overridden by another ACL entry.

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.

The following diagram shows an example hierarchy:



The following examples show how this hierarchical behavior 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 the hierarchy.

Example 2

UserB has an ACL entry that gives Deploy authority to the execution group Eg1A. This 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 execution group's name). 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. This gives 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 authority for the CMP, and an ACL entry that gives View authority for Broker1. The View authority to access Broker1 means that UserD does not inherit Full 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 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:

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

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 above, with the exception that access to the ACLs cannot be removed by granting a lower permission further down the tree; this is necessary because otherwise a user would be able to give themselves a View entry and would not then be able to remove it.

ACL entries can be manipulated using the Java Configuration Manager Proxy API or using the `mqsicreateaclentry`, `mqsdeleteaclentry`, and `mqsilistaclentry` commands.

Security exits

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

When you start the Message Brokers 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 Brokers Toolkit. You can enable security exits on the connection from the Message Brokers 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 Brokers 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, refer to the topic "Channel security exit programs" in "Channel-exit programs" in *WebSphere MQ Intercommunication*. If you have v6 then locate "Channel-exit programs" within the Intercommunication section of the 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 what is called private key encryption. Public key encryption is the only challenge to private key encryption that has appeared within 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 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.
- The fact that keys must be shared between pairs of communicators means the keys must somehow be distributed to the participants. The need to transmit secret keys makes them vulnerable to theft.
- Participants can communicate only by prior arrangement. No way exists to 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, you have to be careful 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 that the owner of the public key is really the person you think it is. 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 they can be local entities, depending on the requirements of your application. Regardless, 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 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 upon 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 successfully authenticated, 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. Because public keys are distributed in certificates, 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 really 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 node, the HTTP listener, and the WebSphere MQ Java Client.

SSL authentication for the Real Time node

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

Asymmetric SSL

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

Symmetric SSL

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

In both instances, SSL authentication does not keep the SSL protocol up for the entire lifetime of a connection, because that would incur protection overheads on all messages. The SSL protocol remains in force long enough to accomplish mutual authentication and to establish a shared secret session key that can be used by message protection (see “Message protection” on page 77). Messages are then individually protected in accordance with the protection level specified for the given topic.

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

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

For more information on SSL authentication for the Real Time node, see “Enabling SSL for the Real-time nodes” on page 54.

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 57, and “Configuring an HTTPRequest node to use SSL (HTTPS)” on page 59.

SSL authentication for the MQ Java Client

For information on SSL authentication for the MQ Java Client, see “Enabling SSL on the WebSphere MQ Java Client” on page 52.

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

Enabling domain awareness means that access to the Message Brokers Toolkit is not restricted to users from one Windows domain.

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

You can, however, configure whether the Message Brokers Toolkit sends domain aware information about the user to the Configuration Manager. If the user is logged on to a Windows domain and the Message Brokers Toolkit is configured to send domain aware information, the Message Brokers Toolkit sends the user's domain membership information as well as the user's ID. If the user is not logged on to a Windows domain, the Message Brokers Toolkit sends only the user's ID and their computer name.

With domain awareness enabled, users from different Windows domains who are either trusted by the primary domain, or in a Windows 2000 transitive trust relationship, can perform Message Brokers Toolkit tasks, provided that they are in the correct user role definition groups or object level security groups.

When using domain awareness:

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

Planning for security when you install WebSphere Message Broker

On Linux and UNIX systems, you must complete security tasks before you install WebSphere Message Broker; these tasks are described in 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 45 for further security considerations.

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

Setting up message flow security

This topic introduces the tasks involved in setting up security on a message flow.

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

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

1. "Creating a security profile" on page 29
2. "Configuring identity extraction" on page 34
3. "Configuring authentication" on page 36
4. "Configuring identity mapping" on page 38

5. “Configuring authorization” on page 39
6. “Configuring for identity propagation” on page 44

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 configure them on the relevant SOAP Nodes (in addition to the security profile). See Associating policy sets and bindings with message flows and nodes. For working with a Username + Password identity you need to configure for Username token capabilities. For working with an X.509 Certificate identity, configure 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*. It is set to *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 bindings: Authentication and Protection Tokens.

Creating a security profile

This topic describes how to create a security profile for use with LDAP or TFIM.

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 by using either the `mqsicreateconfigurableservice` command or an editor in the Broker Administration perspective of the Message Broker Toolkit.

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. For information on creating security profiles, see:

- “Creating a security profile for LDAP”
- “Creating a security profile for TFIM” on page 33.

Creating a security profile for LDAP

This topic describes how to create a security profile for use with the Lightweight Directory Access Protocol (LDAP).

Before you start:

Ensure that you have one of the following LDAP servers:

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

You can create a security profile for LDAP using either the `mqsicreateconfigurableservice` command or an editor in the Broker Administration perspective of the Message Broker Toolkit (see “Creating a security profile using the Message Broker Toolkit” on page 31).

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.

1. To create a security profile that ensures that each message has an authenticated ID and is authorized for the flow, enter the `mqsicreateconfigurable-service` command on the command line. For example:

```
mqsicreateconfigurable-service 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\""
```

As shown in the example above, you must enclose the LDAP URL (which contains commas) with escaped double quotes `\"` and `\"` so that the URL commas are not confused with the comma separator of the value parameter of `mqsicreateconfigurable-service`.

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 surrounded in double quotes, (`"`)

For more information about the structure of the command, refer to the “`mqsicreateconfigurable-service` command” on page 435.

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 authorization to LDAP. This ensures that the incoming identity is checked for group membership in LDAP.
- c. 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 or not 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 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 should map, typically `uid`, `CN`, or email 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 `uid_attr + username + baseDN`. If *sub* is selected, a search must be performed before the DN can be resolved. Default is *sub*.

- 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 or not 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.

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 the ones 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_attr + 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), these 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.

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

Creating a security profile using the Message Broker Toolkit:

This section describes how to use the Broker Administration perspective in the Message Broker Toolkit to create a security profile for LDAP.

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** required. This can be LDAP, TFIM, or NONE. If you select LDAP for authentication (or for authorization), the **LDAP Parameters** fields at the bottom of the pane are enabled.
 - 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. If you select TFIM for mapping (or for authentication or authorization) the **TFIM Configuration** field of the **TFIM Parameters** section is enabled.
 - 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. If you select LDAP for authorization (or for authentication), the **LDAP Parameters** fields at the bottom of the pane are enabled.
 - 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 (****).

5. 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

This topic describes how to create a security profile for Tivoli Federated Identity Manager (TFIM).

You can create a security profile for TFIM using either the `mqsicreateconfigurableservice` command or an editor in the Broker Administration perspective of the Message Broker Toolkit (see “Creating a profile for TFIM using the Message Broker Toolkit”).

Creating a profile using `mqsicreateconfigurableservice`:

You can use the `mqsicreateconfigurableservice` command to create a new security profile that uses Tivoli Federated Identity Manager (TFIM) for any combination of the following security functions: authentication, authorization, and mapping.

To create a security profile that uses TFIM, use the `mqsicreateconfigurableservice` command.

The configuration parameter must be set 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:

This section describes how to 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 mqsicreateconfigurable-service” on page 33.
 - c. In the **Propagation** field, specify whether or not 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 appears in the Properties folder as plain text.
 - OBFUSSATE*
The password appears in the Properties folder as base64 encoding.
 - MASK* The password appears in the Properties folder as four asterisks (****).
5. Click **Finish** to deploy the security profile to the broker.

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

Configuring identity extraction

This topic explains how to configure the extraction of an identity in the message flow.

Before you start:

Before you can configure a message flow to perform identity extraction, you need to check that an appropriate security profile exists, or create a new security profile. See “Creating a security profile” on page 29.

Input nodes can retrieve identity from the bitstream.

- An MQInput node retrieves the UserIdentifier from the message descriptor (MQMD) and puts it in 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 *usernameAndPassword* and the Identity Source Issued By element to the HTTP header *UserAgent* property.
- A SOAPInput node either retrieves 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, but with WebSphere MQ only a username is available so 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 this, the flow developer must include a path into the relevant locations in the message to locate the security information. You specify this information 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 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 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 is in the form of an ESQL Path, XPath expression, or literal 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: Promote the properties to the message flow. This ensures that all input nodes share the same information.

To enable the extraction of an identity in the message flow, you also need to 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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
5. In the **Security Profile Name** field, select a security profile.
6. Save the bar file.

Configuring authentication

This topic describes how to configure authentication of an identity in the message flow.

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” on page 29.

You can use either LDAP or TFIM to perform authentication of an identity. For information about configuring authentication with LDAP or TFIM, see:

- “Configuring authentication with LDAP”
- “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 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 now 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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
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” on page 34.

Configuring authentication with TFIM

This topic describes how to configure a message flow to perform identity authentication using 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 33.

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 13.

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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
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” 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 29.

Configure 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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
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

This topic explains how to configure a message flow to perform authorization on an identity.

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” on page 29.

You can use either LDAP or TFIM to perform authorization of an identity. For information about configuring authorization for LDAP or TFIM, see:

- “Configuring authorization with LDAP”
- “Configuring authorization with TFIM” on page 41

Configuring authorization with LDAP

This topic describes the steps involved in configuring a message flow to perform authorization on an identity using 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 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.
- 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 now 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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.

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 33.
- 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 13.

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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
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 TFIM.

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

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 TAM configuration (see “Configuring TAM”).

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. Check 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 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 <AcIName>
```

```
acl modify <AcIName> set Group iv-admin TcmdbsvaBRx1[WebService]i
```

```
acl modify <AcIName> set User sec_master TcmdbsvaBRx1[WebService]i
```

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

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

To add a specific group, use the form:

```
acl modify <AcIName> 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 protect 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
```

ispolicyattachable 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 in general the broker does not know at the input node which operation a flow is going to perform (because it might not be implementing a Web service). 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> <Ac1Name>
```

To attach an ACL to the leaf node use:

```
acl attach /<ContainerObjectName>/<FlowName>/MessageFlowAccess <Ac1Name>
```

For further information about how to configure 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 29.

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 **Configure** tab.
4. Click the flow or node on which you want to set the security profile.
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” 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.

Setting up broker domain security

This section introduces the things to consider when you are setting up security for a broker configuration 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 65 and “Summary of required access (z/OS)” on page 541 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 28, which contains links to security information that you need before, during, and after installation of WebSphere Message Broker.

You can 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 configuration. The answers to the questions provide the security information that you need to configure your WebSphere Message Broker components and also give you information about other security controls that you might want to deploy.

- “Creating user IDs”
- “Considering security for a broker” on page 48
- “Considering security for a Configuration Manager” on page 50
- “Considering security for the workbench” on page 46
- “Enabling topic-based security” on page 61
- “Using security exits” on page 63
- “Database security” on page 64

Creating user IDs

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

- Administrator user IDs that can issue mqsi* commands. Refer to:
 - “Deciding which user accounts can execute broker commands” on page 48
 - “Deciding which user accounts can execute User Name Server commands” on page 62
 - “Deciding which user accounts can execute Configuration Manager commands” on page 50
- Service user IDs under which components run. Refer to:
 - “Deciding which user account to use for the broker service ID” on page 49
 - “Deciding which user account to use for the User Name Server service ID” on page 63
 - “Deciding which user account to use for the Configuration Manager service ID” on page 51

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 the databases” on page 111 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 61.

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

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

Consider the following:

1. “Are you running with domain awareness enabled?”
2. “Are you running with domain awareness disabled?” on page 47
3. “Securing the channel between the workbench and the Configuration Manager” on page 48

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

Are you running with domain awareness enabled?

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

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

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

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

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

Go to "Securing the channel between the workbench and the Configuration Manager" on page 48.

Are you running with domain awareness disabled?

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

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

To set domain awareness to disabled, answer the following questions:

1. Are your workbench users drawn from a local domain?
 - a. No: Go to the next question.
 - b. Yes: Add any users to the local groups that you will be using in your ACLs. Go to "Securing the channel between the workbench and the Configuration Manager" on page 48.
2. Are your workbench users drawn from another domain?
 - a. Yes: Make the other domain a trusted domain of the Configuration Manager's computer then add the groups and users from the trusted domain to the local groups of the Configuration Manager.

For additional security, run with both domain awareness and security exits enabled. For more information about security exits, refer to "Security exits" on page 20.

Turning off the toolkit domain awareness

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

To turn off the domain awareness on the toolkit, run the toolkit in the following way:

1. Change to the *install_dir*\eclipse directory.
2. Run the toolkit using the command `mqsisstudio -vmargs -DDomainAware=0`.

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

Go to “Securing the channel between the workbench and the Configuration Manager”

Securing the channel between the workbench and the Configuration Manager

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

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

Considering security for a broker

Consider the factors for deciding which users can:

- Execute broker commands
- Control security for other broker resources.

Consider the following steps:

1. “Deciding which user accounts can execute broker commands”
2. “Deciding which user account to use for the broker service ID” on page 49
3. “Setting security on the broker’s queues” on page 49
4. “Enabling topic-based security in the broker” on page 50
5. “Securing the broker registry” on page 50

Deciding which user accounts can execute broker commands

Decide what permissions are required for the user IDs that:

- Create, change, list, delete, start, and stop brokers
- Display, retrieve, and change trace information.

Answer the following questions:

1. Is your broker installed on a Linux or UNIX operating system?
 - a. No: Go to the next question.
 - b. Yes: Go to “Deciding which user account to use for the broker service ID” on page 49
2. Are you executing broker commands under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a broker, ensure that your user ID is defined in your local domain. When you create or start a broker, ensure that your user ID is a member of WKSTN1\Administrators.
Go to “Deciding which user account to use for the broker service ID” on page 49.
3. Are you executing broker commands under a Windows domain account?

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

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

Deciding which user account to use for the broker service ID

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

Answer the following questions:

1. Is your broker installed on a Linux or UNIX operating system?
 - a. No: Go to the next question.
 - b. Yes: Go to “Setting security on the broker’s queues”
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 is defined in your local domain and is a member of **mqbrkrs**.
Go to “Setting security on the broker’s queues”
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: DOMAIN1\user1 is a member of DOMAIN1\Domain **mqbrkrs** and DOMAIN1\Domain **mqbrkrs** is a member of WKSTN1\mqbrkrs.
Go to “Setting security on the broker’s queues”

Setting security on the broker’s queues

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

```
SYSTEM.BROKER.ADMIN.QUEUE
SYSTEM.BROKER.CONTROL.QUEUE
SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
SYSTEM.BROKER.EXECUTIONGROUP.REPLY
SYSTEM.BROKER.INTERBROKER.QUEUE
SYSTEM.BROKER.MODEL.QUEUE
```

- If you have created WebSphere Message Broker components to run on different queue managers, set the permissions for the transmission queues that you define to handle the message traffic between the queue managers. Grant put and setall authority to the local **mqbrkrs** group, or to the service user ID of the component supported by the queue manager on which you defined the transmission queue.
- Ensure that the broker’s service user ID has authority to:

1. Get messages from each input queue included in a message flow
 2. Put messages to any output, reply, and failure queues included in a message flow.
- Ensure that the user IDs under which applications that interact with the broker run have authority to:
 1. Put messages to each input queue included in a message flow
 2. Get messages from any output, reply, and failure queue included in a message flow.
 - Go to “Enabling topic-based security in the broker”

Enabling topic-based security in the broker

Perform this task by responding to the following question:

Do you want to enable topic-based security in the broker?

1. Yes: Go to “Enabling topic-based security” on page 61
2. No: Go to “Considering security for a Configuration Manager”

Securing the broker registry

The broker registry (in the Windows registry or UNIX[®] filesystem) must be further secured. Set the following keys so that only the service user ID can read or write it:

- `<brokername>/CurrentVersion/DSN` (including all sub-keys)
- `<brokername>/CurrentVersion/ServicePassword` (including all sub-keys)
- `<brokername>/CurrentVersion/DataSourcePassword` (including all sub-keys)

Considering security for a Configuration Manager

During this task you consider the group membership that is required for:

- Users that execute Configuration Manager commands
- Service IDs
- Access to the Configuration Manager’s queues.

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

Read the appropriate sections in this list:

1. “Deciding which user accounts can execute Configuration Manager commands”
2. “Deciding which user account to use for the Configuration Manager service ID” on page 51
3. “Setting security on the Configuration Manager’s queues” on page 52
4. “Running the Configuration Manager in a domain environment” on page 52

Deciding which user accounts can execute Configuration Manager commands

During this task you decide what permissions are required for the user IDs that:

- Create, change, list, delete, start, and stop a Configuration Manager
- Display, retrieve, and change trace information.

Answer the following questions:

1. Are you executing Configuration Manager commands under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Assume that your local account is on a computer named, for example, WKSTN1. When you create a Configuration Manager, ensure that your user ID is defined in your local domain. When you create or start a Configuration Manager, ensure that your user ID is a member of WKSTN1\Administrators.
Go to "Deciding which user account to use for the Configuration Manager service ID"
2. Are you executing Configuration Manager commands under a Windows domain account?
 - a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you create a Configuration Manager using, for example, DOMAIN1\user1, ensure that DOMAIN1\user1 is a member of WKSTN1\Administrators.
Go to "Deciding which user account to use for the Configuration Manager service ID"

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

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

Answer the following questions:

1. Do you want your Configuration Manager to run under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Ensure that your user ID has the following characteristics:
 - It is defined in your local domain.
 - It is a member of **mqbrkrs**.
 - It is a member of **mqm**.
 - It is a member of Administrators.Go to "Setting security on the Configuration Manager's queues" on page 52.
2. Do you want your Configuration Manager to run under a Windows domain account?
 - a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a Configuration Manager using, for example, DOMAIN1\user1, ensure that:
 - 1) DOMAIN1\user1 is a member of DOMAIN1\Domain **mqbrkrs**.
 - 2) DOMAIN1\user1 is a member of WKSTN1\mqm.
 - 3) DOMAIN1\Domain **mqbrkrs** is a member of WKSTN1\mqbrkrs.
 - 4) DOMAIN1\user1 is a member of WKSTN1\Administrators.Alternatively, complete the following steps:
 - 1) Define user1 in DOMAIN1.
 - 2) Add user1 to the DOMAIN1\Domain **mqm** group.
 - 3) Add user1 to the DOMAIN1\Domain **mqbrkrs** group.

- 4) Add the DOMAIN1\Domain **mqm** group to the WKSTN1**mqm** group.
- 5) Add the DOMAIN1\Domain **mqbrkrs** group to the WKSTN1**mqbrkrs** group.
- 6) Add user1 to the WKSTN1**Administrators** group.

Go to “Setting security on the Configuration Manager’s queues.”

Setting security on the Configuration Manager’s queues

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

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

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

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

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

Go to “Running the Configuration Manager in a domain environment.”

Running the Configuration Manager in a domain environment

- If you want to *enable* domain awareness, go to “Are you running with domain awareness disabled?” on page 47.
- If you want to *disable* domain awareness, go to “Are you running with domain awareness enabled?” on page 46.

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 404.

Implementing SSL authentication

The following topics contain instructions for implementing SSL authentication:

- “Enabling SSL on the WebSphere MQ Java Client”
- “Enabling SSL for the Real-time nodes” on page 54
- “Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 57
- “Configuring an HTTPRequest node to use SSL (HTTPS)” on page 59

Enabling SSL on the WebSphere MQ Java Client

The WebSphere MQ Java Client supports SSL-encrypted connections over the server-connection (SVRCONN) channel between the application and the queue

manager. This topic tells you how to make use of this SSL support when communicating between the Configuration Manager Proxy (CMP) and the Configuration Manager.

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

1. Generate or obtain all the appropriate keys and certificates. This includes a signed pkcs12 certificate for the server and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the pkcs12 certificate to the queue manager certificate store and assign it to the queue manager. Use the standard WebSphere MQ facilities, for example, WebSphere MQ Explorer (for WebSphere MQ Version 6) or WebSphere MQ Services (for WebSphere MQ Version 5).
3. Add the certificate of the certificate authority to the JSEE truststore of the Java Virtual Machine (JVM) at the Configuration Manager Proxy end using a tool such as Keytool.
4. Decide which cipher suite to use.
5. Change the properties on the server-connection channel using WebSphere MQ Explorer, to specify the cipher suite to be used. This channel has a default name of SYSTEM.BKR.CONFIG and 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 223 and “Modifying domain connection properties” on page 225.
6. Add the required parameters (cipher suite, for example) to the Configuration Manager Proxy. If a truststore other than the default is used, its full path must be passed in via the truststore parameter.

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

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

1. Generate or obtain all the appropriate keys and certificates. This includes a signed pkcs12 certificate for the client and the appropriate public key for the certificate authority that signed the pkcs12 certificate.
2. Add the certificate of the certificate authority to the queue manager certificate store; use the standard WebSphere MQ facilities (for example, WebSphere MQ Explorer for WebSphere MQ Version 6).
3. Set the server-connection channel to always authenticate. You can use SSLCAUTH(REQUIRED) in runmqsc, or WebSphere MQ Explorer.
4. Add the pkcs12 certificate to the JSEE keystore of the JVM at the Configuration Manager Proxy end using a tool such as Keytool.
5. If not using the default keystore, its full path must be passed into the Configuration Manager Proxy via the keystore parameter

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

Further restrictions can be made using the sslPeerName field; for example, you can allow connections only from certificate holders with a specific company or

department name in their certificates. In addition, you can invoke a security exit for communications between the Configuration Manager Proxy and the Configuration Manager; see “Using security exits” on page 63.

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.

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 `mqscreateusernameserver` or the `mqschangeusernameserver` 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 `mqschangeusernameserver` 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 mqsicchangebroker 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)

This topic explains how to configure the HTTPInput and HTTPReply nodes to communicate with other applications using HTTP over SSL.

1. "Create a key store file to store the broker's certificates"
2. "Configuring the broker to use SSL on a particular port"
3. "Creating a message flow to process HTTPS requests" on page 58
4. "Testing your example" on page 58

Create a key store 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 invoke 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 key store: In the command console, type the following command:

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

password

the password used for the keystore

keystore file

the fully qualified name of the keystore file. This file is typically called `.keystore` and is located in the WebSphere Message Broker home directory.

The command then prompts you for some personal details that are used to generate the certificates. Your personal details are then added to a key store, if it already exists, or a key store is generated.

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

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

Configuring the broker to use SSL on a particular port:

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

- Turn on SSL support in message broker, by setting a value for **enableSSLConnector**
`mqsichangeproperties broker name -b httplistener -o HTTPListener
-n enableSSLConnector -v true`
- Choose the keystore file to be used, by setting a value for **keystoreFile**
`mqsichangeproperties broker name -b httplistener -o HTTPSConnector
-n keystoreFile -v fully qualified file path to keystore file`
- Specify the password for the keystore file, by setting a value for **keystorePass**
`mqsichangeproperties broker name -b httplistener -o HTTPSConnector
-n keystorePass -v password for keystore`
- Specify the **port** on which WebSphere Message Broker should listen for HTTPS requests
`mqsichangeproperties broker name -b httplistener -o HTTPSConnector
-n port -v Port to listen on for https`

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

Creating a message flow to process HTTPS requests:

The most simple message flow that shows the HTTPS functionality working is one that contains an HTTPInput node connected directly to an HTTPReply node: The two important properties to set on the HTTPInput node are:

- **Path suffix for URL.** For example `/*` or `/testHTTPS`
- **Use HTTPS.** Tick the box if you want to use HTTPS

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

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

You can now test the system.

Testing your example:

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

Start a web browser and type the following URL:

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

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

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

and in Internet Explorer the following information should be displayed:
XML document must have a top level element. Error processing resource
'https://localhost:7083/testHTTPS'

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

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

Another HTTPS client, such as a Java or .net client, could be used instead of the web browser. Depending on the type of client, the certificate that was created with keytool might have to be exported from the http listener's keystore file and then imported into that client's own keystore. Consult the client documentation to find out how you should configure the client to make client connections over SSL.

Configuring an HTTPRequest node to use SSL (HTTPS)

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

The steps required are as follows:

1. "Adding certificates to the cacerts file"
2. "Creating a message flow to make HTTPS requests" on page 60
3. "Testing your example" on page 61

Adding certificates to the cacerts file:

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

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

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

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

Importing a certificate into the cacerts file

1. Select **Start** → **IBM WebSphere Message 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  
-keypass changeit
```

name of certificate file

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

changeit

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

Extracting a certificate from another keystore

1. Select **Start** → **IBM WebSphere Message 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 tomcat  
-file name of certificate file -keystore keystore file  
-keypass changeit
```

name of certificate file

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

keystore file

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

changeit

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

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

Creating a message flow to make HTTPS requests:

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

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

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

```

CREATE PROCEDURE CopyMessageHeaders() BEGIN
    DECLARE I INTEGER;
    DECLARE J INTEGER;
    SET I = 1;
    SET J = CARDINALITY(InputRoot.*[]);
    WHILE I < J DO
        SET OutputRoot.*[I] = InputRoot.*[I];
        SET I = I + 1;
    END WHILE;
END;

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

```

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

Testing your example:

To test that the example works, complete the following steps

1. Follow all of the instructions given in “Configuring HTTPInput and HTTPReply nodes to use SSL (HTTPS)” on page 57, including testing the example.
2. Deploy the HTTPRequest message flow.
3. Put a message to the MQSeries queue HTTPS.IN1. A message should appear on the output queue. If it fails, an error appears in the local error log (event log on windows).

Enabling topic-based security

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

Complete the following steps:

1. Before you create a User Name Server, refer to “Considering security for a User Name Server” on page 62.
2. Create a User Name Server. Refer to “Creating a User Name Server” on page 186.
3. Select the -j flag and set the -s parameter to the name of the queue manager for the User Name Server on the mqsicreatebroker or mqsichangebroker command.
4. Set the -s parameter on the mqsicreateconfigmgr or mqsichangeconfigmgr command to the name of the queue manager for the User Name Server.
5. Create ACLs for the topics that require additional security. For more information, see “Creating ACL entries” on page 81.
6. Ensure that the broker’s service user ID has authority to:
 - a. Get messages from each input queue included in a message flow
 - b. Put messages to any output, reply, and failure queues included in a message flow.
7. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put to and get from message flow queues:
 - a. Authorize publish applications to put messages to the input queue of the message flow.

- b. Authorize applications that register subscriptions to put to the SYSTEM.BROKER.CONTROL.QUEUE queue.
- c. Authorize subscribe applications to get from the queue to which messages are published.
- d. Authorize publish and subscribe applications to get from the reply queue.

If you are issuing publish/subscribe requests from a JMS client, additional security options are available. Refer to “SSL authentication” on page 25, “Quality of protection” on page 27, and “Authentication services” on page 75.

Go to “Considering security for a Configuration Manager” on page 50.

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 50.
2. Yes: You need a User Name Server. Go to “Deciding which user accounts can execute User Name Server commands.”

Deciding which user accounts can execute User Name Server commands

During this task you decide what permissions are required for the user IDs that:

- Create, change, list, delete, start, and stop a User Name Server
- Display, retrieve, and change trace information.

Answer the following questions:

1. Is your User Name Server installed on a Linux or UNIX operating system?
 - a. No: Go to the next question.
 - b. Yes: The information about executing User Name Server commands on Linux and UNIX is not yet available.
Go to “Deciding which user account to use for the User Name Server service ID” on page 63.
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 63.
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 63.

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 `mqsicreateusernameeserver` or `mqsichangeusernameeserver` command, you determine the user ID under which the User Name Server component process runs.

Answer the following questions:

1. Is your User Name Server installed on a Linux or UNIX operating system?
 - a. No: Go to the next question.
 - b. Yes: The information about running your User Name Server on Linux and UNIX is not yet available.
Go to "Setting security on the User Name Server's queues"
2. Do you want your User Name Server to run under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Ensure that your user ID is defined in your local domain and is a member of **mqbrkrs**.
Go to "Setting security on the User Name Server's queues"
3. Do you want your User Name Server to run under a Windows domain account?
 - a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a User Name Server using, for example, DOMAIN1\user1, ensure that:DOMAIN1\user1 is a member of DOMAIN1\Domain **mqbrkrs** and DOMAIN1\Domain **mqbrkrs** is a member of WKSTN1\mqbrkrs.
Go to "Setting security on the User Name Server's queues."

Setting security on the User Name Server's queues

When you run the `mqsicreateusernameeserver` 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 `mqsicreateusernameeserver` 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

This topic describes how to define a security exit when you create a domain connection.

To create a security exit for communications between the Message Brokers Toolkit and the Configuration Manager, you must define a security exit when you create a domain connection.

1. Click **File** → **New** → **Domain Connection**. The Create a Domain Connection wizard opens.
2. Enter the values for **Queue Manager Name**, **Host**, and **Port** that you want to use.
3. Enter the Security Exit **Class** name. The name must be a valid Java class name.
4. 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 52.

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 381.

Once set, HTTP tunneling is administered using the “mqsichangeproperties command” on page 381 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 381.

From the broker properties window, you can set the values for temporary topics using:

- Temporary Topic Quality of Protection
- Sys Topic Quality of Protection
- ISys Topic Quality of Protection .

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

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

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

Database security

The service user ID for the brokers must be authorized to access databases:

- Each broker service user ID must be authorized for create and update tasks on the database that contains the broker internal tables.

In addition to the broker service user ID, this also applies to any data source user ID if defined (using “mqsicreatebroker command” on page 416), or to any user ID specified for a data source by use of the “mqsisetdbparms command” on page 520.

Derby databases, created either by the Default Configuration wizard or by the “mqsicreatedb command” on page 438 on Windows, do not require security authorization. Refer to “Using Derby databases on Windows” on page 108 for more information.

Database security is defined and modified using various commands dependent upon the database that is being used. See the following:

- “mqsicreatebroker command” on page 416
- “mqsicchangebroker command” on page 356

For information relating to specific databases see:

- “Oracle: granting and revoking user access to databases”
- “SQL Server: granting and revoking user access to databases”
- “Sybase: granting and revoking user access to databases”

Oracle: granting and revoking user access to databases

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

SQL Server: granting and revoking user access to databases

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

You must enable SQL Server Authentication mode as well as the default Windows Authentication mode.

Sybase: granting and revoking user access to databases

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

Setting up z/OS security

You need to complete some security configuration tasks before WebSphere 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 67
- “Setting up WebSphere MQ” on page 68
- “Setting up workbench access on z/OS” on page 69
- “Creating Publish/Subscribe user IDs” on page 69
- “Enabling the Configuration Manager on z/OS to obtain user ID information” on page 69

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 need to 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 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 need a minimum of 400 000 blocks free; this is needed if a dump is taken.

Associate the started task procedure with the user ID to be used. For example, you can use the STARTED class in RACF®. The WebSphere 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 described above.

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 DSNTPE2 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.RRSAP profile in the DSNR class, if present. In this case, subsystem is the DB2 subsystem name. For example, the following RACF command lists all the resources in DSNR class:

```
RLIST DSNR *
```

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

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

Setting up WebSphere MQ

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

The user ID of the person running the create component (BIPCBRK, BIPCRCM, and BIPCRUN) jobs needs *UPDATE* access to the component PDSE, *READ/EXECUTE* access to the installation directory, and *READ/WRITE/EXECUTE* access to the component directory. If you do not use queue manager security, you do not need to read the rest of this topic. Topic “Creating the broker component” on page 174 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 (x86 platform). The workbench must run on Windows or Linux (x86 platform). See “Considering security for a Configuration Manager” on page 50 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 `mqsicreateusername` 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.

mqrbrks authorizations

WebSphere Message Broker grants special publish/subscribe access control privileges to members of the `mqrbrks` group, and to the corresponding Domain `mqrbrks` 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 `mqrbrks` as the service user ID for the broker. The `mqrbrks` 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 `mqrbrks` 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 be published from, and subscribed to, by members of the `mqrbrks` 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 "\$SYS". 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.
QoS Level	Specifies the level of message protection that is enforced. One of the following four values can be chosen: <ul style="list-style-type: none"> • None • Channel Integrity • Message Integrity • Encrypted The default value is 'None'.

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

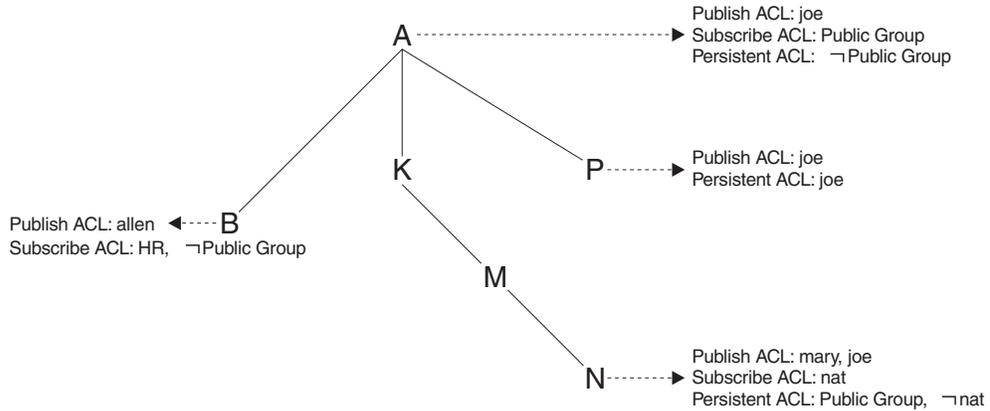
- Clients that are denied Publish access have their publication messages refused.
- Clients that are denied Subscribe access do not receive the publication.
- The persistent access control does not deny the message to subscribers, but denies them persistence, so denied subscribers always receive messages, 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 76 and “Mutual challenge-response password authentication” on page 77 respectively. Asymmetric and symmetric SSL protocols are described in “SSL authentication” on page 25.

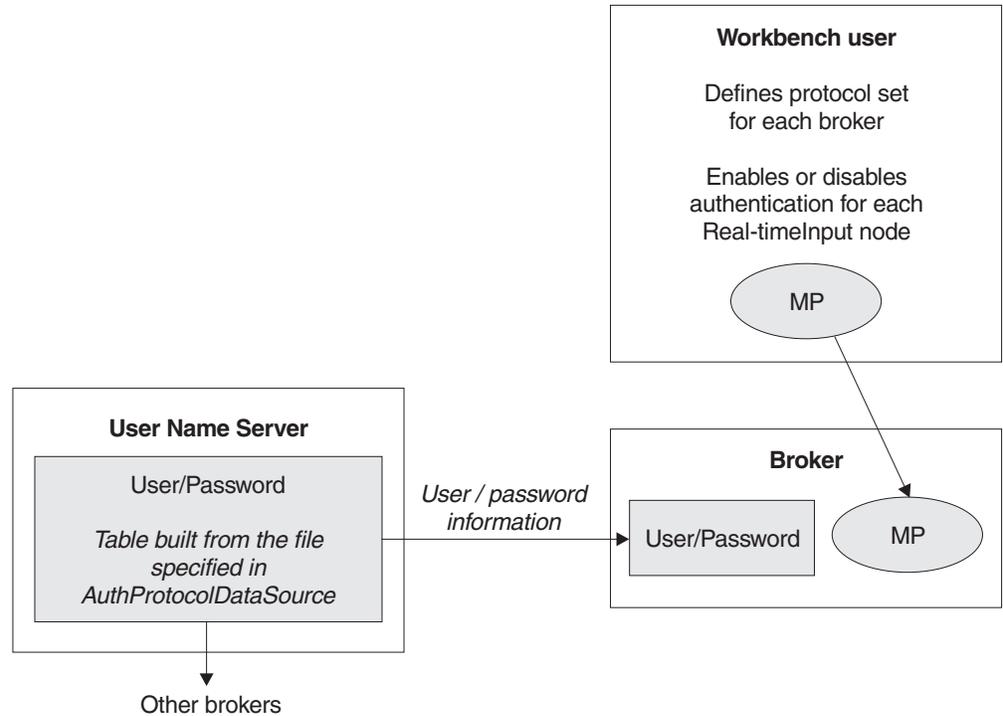
The protocols vary in strength, in terms of providing protection against participants that are not valid participants in the session; P is the weakest and R is the strongest.

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

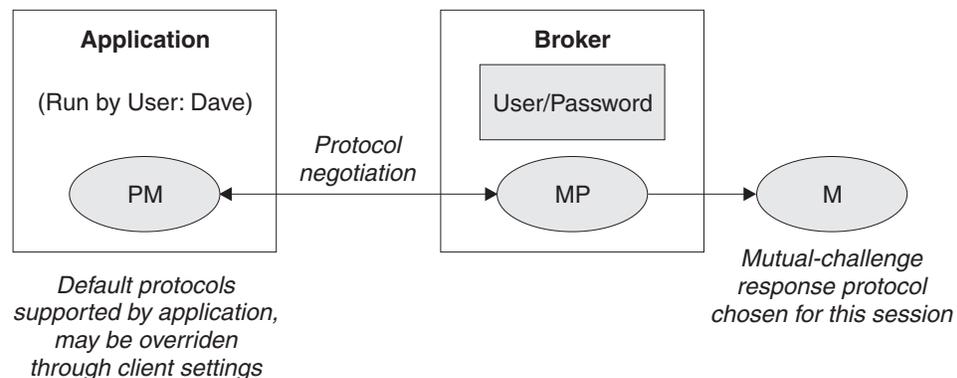
authentication is enabled at a Real-timeInput node, that node supports the full set of protocols specified for its corresponding broker. The configuration options are illustrated in the following diagrams:

Overview of authentication configuration

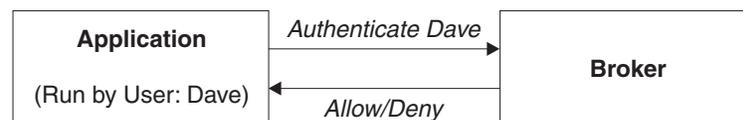


Two-stage runtime authentication process

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 the User 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 are 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 61
- "Creating ACL entries" on page 81
- "Enabling SSL for the Real-time nodes" on page 54
- "Using message protection" on page 84
- "Securing WebSphere MQ resources" on page 84

For more information, see "Publish/subscribe security" on page 70 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.

Complete the following steps:

1. Before you create a User Name Server, refer to “Considering security for a User Name Server” on page 62.
2. Create a User Name Server. Refer to “Creating a User Name Server” on page 186.
3. Select the `-j` flag and set the `-s` parameter to the name of the queue manager for the User Name Server on the `mqsicreatebroker` or `mqsichangebroker` command.
4. Set the `-s` parameter on the `mqsicreateconfigmgr` or `mqsichangeconfigmgr` command to the name of the queue manager for the User Name Server.
5. Create ACLs for the topics that require additional security. For more information, see “Creating ACL entries” on page 81.
6. Ensure that the broker’s service user ID has authority to:
 - a. Get messages from each input queue included in a message flow
 - b. Put messages to any output, reply, and failure queues included in a message flow.
7. Ensure that the user IDs under which publish and subscribe applications run have sufficient authority to put to and get from message flow queues:
 - a. Authorize publish applications to put messages to the input queue of the message flow.
 - b. Authorize applications that register subscriptions to put to the `SYSTEM.BROKER.CONTROL.QUEUE` queue.
 - c. Authorize subscribe applications to get from the queue to which messages are published.
 - d. Authorize publish and subscribe applications to get from the reply queue.

If you are issuing publish/subscribe requests from a JMS client, additional security options are available. Refer to “SSL authentication” on page 25, “Quality of protection” on page 27, and “Authentication services” on page 75.

Go to “Considering security for a Configuration Manager” on page 50.

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 50.
2. Yes: You need a User Name Server. Go to “Deciding which user accounts can execute User Name Server commands” on page 62.

Deciding which user accounts can execute User Name Server commands

During this task you decide what permissions are required for the user IDs that:

- Create, change, list, delete, start, and stop a User Name Server
- Display, retrieve, and change trace information.

Answer the following questions:

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

Go to “Deciding which user account to use for the User Name Server service ID” on page 63.

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 63.
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 63.

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 `mqsicreateusername` or `mqsichangeusername` command, you determine the user ID under which the User Name Server component process runs.

Answer the following questions:

1. Is your User Name Server installed on a Linux or UNIX operating system?
 - a. No: Go to the next question.
 - b. Yes: The information about running your User Name Server on Linux and UNIX is not yet available.
Go to “Setting security on the User Name Server’s queues” on page 63
2. Do you want your User Name Server to run under a Windows local account?
 - a. No: Go to the next question.
 - b. Yes: Ensure that your user ID is defined in your local domain and is a member of **mqbrkrs**.
Go to “Setting security on the User Name Server’s queues” on page 63
3. Do you want your User Name Server to run under a Windows domain account?
 - a. Yes: Assume that your computer named, for example, WKSTN1, is a member of a domain named DOMAIN1. When you run a User Name Server using, for example, DOMAIN1\user1, ensure that:DOMAIN1\user1 is a member of DOMAIN1\Domain **mqbrkrs** and DOMAIN1\Domain **mqbrkrs** is a member of WKSTN1\mqbrkrs.
Go to “Setting security on the User Name Server’s queues” on page 63.

Setting security on the User Name Server’s queues

When you run the `mqsicreateusername` 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 63.

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 `mqscreateusernameserver` 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 252 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.

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 `mqscreateusernameserver` or the `mqschangeusernameserver` command:

- *AuthProtocolDataSource* describes the location of a 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 `mqschangeusernameserver` 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 `mqscreatebroker` or `mqschangebroker` 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 configuration requires.

This section does not apply to z/OS.

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

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

- When you issue the command `mqsicreatebroker`, it grants put and get authority on your behalf to the group `mqbrkrs` for the following queues:

- SYSTEM.BROKER.ADMIN.QUEUE
- SYSTEM.BROKER.CONTROL.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- SYSTEM.BROKER.INTERBROKER.QUEUE
- SYSTEM.BROKER.MODEL.QUEUE
- When you issue the command **mqsicreateconfigmgr** 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 **mqsicreateusername**, it grants put and get authority on your behalf to the group **mqbrkrs** for the following queues:
 - SYSTEM.BROKER.SECURITY.QUEUE
 - SYSTEM.BROKER.MODEL.QUEUE
- When you issue the command **mqsicreateaclentry**, it grants put and get authority on your behalf to the resource or user that you have specified for the command parameters **-p** or **-u** for the following queues:
 - SYSTEM.BROKER.CONFIG.QUEUE
 - SYSTEM.BROKER.CONFIG.REPLY
- If you have created WebSphere Message Broker components to run on different queue managers, the transmission queues that you define to handle the message traffic between the queue managers must have put and setall authority granted to the local **mqbrkrs** group, or to the service user ID of the component supported by the queue manager on which the transmission queue is defined.
- When you start the workbench, it connects to the Configuration Manager using a WebSphere MQ client/server connection. For details of WebSphere MQ channel security refer to "Setting up WebSphere MQ client security" in the *WebSphere MQ Clients* book.
- When you create and deploy a message flow, grant:
 1. get and inq authority to each input queue identified in an MQInput node, for the broker's ServiceUserID.
 2. put and inq authority to each output queue identified in an MQOutput node, or by an MQReply node, for the broker's ServiceUserID.
 3. get authority to each output queue identified in an MQOutput node or an MQReply node to the user ID under which a receiving or subscribing client application runs.
 4. put authority to each input queue identified in an MQInput node to the user ID under which a sending or publishing client application runs.

Part 2. Configuring the broker domain

Configuring WebSphere Message Broker	89
Planning a broker domain	89
Considering resource naming conventions	90
Designing the WebSphere MQ infrastructure	92
Considering performance in the domain.	95
Configuring databases.	99
Databases overview	100
Creating the databases	105
Authorizing access to the databases	111
Enabling ODBC connections to the databases	113
Enabling JDBC connections to the databases	128
Using retained publications with a Sybase database	134
Configuring databases for global coordination of transactions	135
Customizing the z/OS environment.	137
z/OS customization overview	138
Customizing UNIX System Services on z/OS	148
DB2 planning on z/OS	150
WebSphere MQ planning for z/OS	153
Resource Recovery Service planning on z/OS	154
Defining the started tasks to z/OS Workload Manager (WLM)	154
Automatic Restart Manager planning	155
Mounting file systems	155
Checking the permission of the installation directory	156
Customizing the level of Java on z/OS.	157
Checking APF attributes of bipmain on z/OS	157
Collecting broker statistics on z/OS	157
Configuring an execution group address space as non-swappable on z/OS.	158
Creating WebSphere Message Broker components on z/OS.	158
WebSphere Message Broker and WebSphere MQ setup verification	161
Configuring broker domain components	162
Creating a broker	163
Adding an execution group to a broker using the command line	175
Adding an execution group to a broker on z/OS	175
Creating a Configuration Manager	176
Enabling a User Name Server	186
Creating a User Name Server	186
Using the Default Configuration wizard	200
Using the Command Assistant wizard	202
Verifying components	204
Connecting components	204
Preparing the environment for WebSphere Adapters nodes	206
Modifying a broker	207
Modifying a Configuration Manager	210
Modifying a User Name Server	213
Modifying access to the broker database	215
Moving from WebSphere Message Broker on a distributed platform to z/OS	216
Deleting an execution group from a broker using the command line.	216
Deleting a broker	217
Deleting a Configuration Manager	219
Disabling a User Name Server.	220
Deleting a User Name Server	221
Configuring a broker domain in the workbench	222
Creating a domain connection	223
Modifying domain connection properties	225
Deleting a domain connection	226
Adding a broker to a broker domain	227
Copying a broker	228
Modifying broker properties	229
Renaming a broker	230
Removing a broker from a broker domain.	230
Removing deployed children from a broker	231
Adding an execution group to a broker in the workbench	232
Copying an execution group	233
Modifying execution group properties	233
Renaming an execution group.	234
Deleting an execution group	234
Removing deployed children from an execution group	235
Configuring a publish/subscribe topology.	235
Setting up the broker domain for publish/subscribe	235
Operating a publish/subscribe domain.	251
Configuring global coordination of transactions (two-phase commit)	253
The Transactional model.	254
Configuring global coordination with DB2 using a 32-bit queue manager	257
Configuring global coordination with DB2 using a 64-bit queue manager	259
Configuring global coordination with Oracle using a 32-bit queue manager	263
Configuring global coordination with Oracle using a 64-bit queue manager	265
Configuring global coordination with Sybase	268
Configuring the workbench	270
Changing workbench preferences.	270
Changing Broker Administration preferences	271
Configuring CVS to run with the Message Broker Toolkit	271
Displaying selected projects in working sets	272
Changing locales	273
Changing your locale on UNIX and Linux systems	273
Changing your locale on Windows	275
Changing your locale on z/OS	275
Code page converters	276

Configuring WebSphere Message Broker

Configuring WebSphere Message Broker involves creating and setting up all the databases, components, and connections that are required for a broker domain to which message flow applications can be deployed.

If you want to create a simple configuration on Windows or Linux (x86 platform) 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 200.

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 16 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 the broker domain will be used for publish/subscribe messaging: Configure the domain for publish/subscribe messaging.

Planning a broker domain

When you plan a new WebSphere Message Broker domain, you must first have considered your resource naming conventions, the design of the WebSphere MQ infrastructure, and any performance issues. The following topics describe these considerations:

- “Considering resource naming conventions” on page 90
- “Designing the WebSphere MQ infrastructure” on page 92
- “Considering performance in the domain” on page 95

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

1. **Brokers:** The number of brokers that you need in your domain depends on the following factors:
 - a. **Performance:** What is the required message throughput? (Refer to “Optimizing message flow throughput” on page 96.) 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 the broker domain’s performance. Refer to “Considering performance in the domain” on page 95.
 - b. Do you need to isolate applications from each other? You might want to separate applications that serve different functions, for example personnel and finance.

- c. Do the brokers need to handle publish/subscribe? Refer to Developing publish/subscribe applications.
2. **User Name Server:** Consider the following if you have a User Name Server in your broker domain:
 - a. Performance: If you have a large number of brokers in your broker domain, the requests that they send to the User Name Server can be handled more quickly if there is more than one User Name Server. More than one User Name Server might also be beneficial (in terms of network traffic) if your broker domain is complex.
 - b. Resilience: Although no standby mechanism is provided by WebSphere 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.
3. **Configuration Manager:** This acts as an interface between the configuration repository and the set of brokers in the domain and the workbench. It uses WebSphere MQ messages to communicate with the brokers, and thus a large number of brokers in a broker domain (if poorly designed) can cause congestion at the Configuration Manager. To solve this, consider dividing the brokers into more than one domain where related brokers are kept together. You can then establish connections with each domain (refer to “Creating a domain connection” on page 223).

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

Product component naming conventions

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

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

Configuration Managers

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

Brokers

When you create a broker, give it a name that is unique within your broker domain. You must use the same name for that broker when you create it on the system in which it is installed (using the command **mqscreatebroker**) and when you create a reference to that broker in the broker domain topology in the workbench. The latter is a representation of the physical broker (created by **mqscreatebroker**) 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.

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

Message sets and messages

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

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

The User Name Server is not allocated 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 539.

Database naming conventions

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

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

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

Designing the WebSphere MQ infrastructure

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

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

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

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

For further information see:

- “WebSphere MQ resources for the broker” on page 93
- “WebSphere MQ resources for the Configuration Manager” on page 93
- “WebSphere MQ resources for the User Name Server” on page 94

WebSphere MQ resources for the broker

Each broker depends on a number of WebSphere MQ resources: some must be available, others depend on the broker domain setup. Some of these resources are created for you: you must define others yourself.

WebSphere MQ resources created for you

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

- The broker's queue manager. Each broker must be associated with a queue manager. Specify a queue manager name when you create the broker. If this queue manager does not exist, it is created for you. (For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your broker. See "Creating a broker on z/OS" on page 167 for more details.) This can be shared with the Configuration Manager and User Name Server
- Fixed-name queues on the broker's queue manager. These allow information to be exchanged with other components in the 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 can create WebSphere MQ resources using commands and utilities such as `runmqsc` and the PCF interface, or you can use WebSphere MQ Services (with WebSphere MQ Version 5) or WebSphere MQ Explorer (with WebSphere MQ Version 6).

- For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your broker. See "Creating a broker on z/OS" on page 167 for more details.
- If the broker and Configuration Manager do not share a queue manager, you must define the channels and transmission queues to support communication between the broker's queue manager and the Configuration Manager's queue manager.
- If the broker and User Name Server do not share a queue manager, you must define the channels and transmission queues to support communication between the broker's queue manager and the Configuration Manager's queue manager.
- You must define listener connections on the broker's queue manager. You must define one listener connection for every protocol used.
- For connection between brokers, you must define the channels and transmission queues to permit two-way communication between them.

For more information about creating WebSphere MQ resources, see *WebSphere MQ Intercommunication*, available on the WebSphere MQ library Web page.

WebSphere MQ resources for the Configuration Manager

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

- The Configuration Manager's queue 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. (For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your Configuration Manager. See "Creating a Configuration Manager on z/OS" on page 181 for more details.)
- Fixed-name queues on the Configuration Manager's queue manager. These allow information to be exchanged with other components in the broker domain.
- A server connection for use by the Workbench.

WebSphere MQ resources that you must create yourself

Depending on the setup of your Configuration Manager, you might need to create some WebSphere MQ resources yourself. You can create WebSphere MQ resources using commands and utilities such as `runmqsc` and the PCF interface, or you can use WebSphere MQ Services (with WebSphere MQ Version 5) or WebSphere MQ Explorer (with WebSphere MQ Version 6).

- For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your Configuration Manager. See "Creating a Configuration Manager on z/OS" on page 181 for more details.
- You must define the transmission queues and channels between the Configuration Manager and every broker in the domain (apart from the broker that shares a queue manager with the Configuration Manager).
- If the Configuration Manager and User Name Server do not share a queue manager, you must define the channels and transmission queues to support communication between the broker's queue manager and the Configuration Manager's queue manager.
- You must define listener connections on the Configuration Manager's queue manager for the Workbench and other components and clients that do not share the Configuration Manager's queue manager. You must define one listener connection for every protocol used.

For more information about creating WebSphere MQ resources, see *WebSphere MQ Intercommunication*, available on the WebSphere MQ library Web page.

WebSphere MQ resources for the User Name Server

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

- The User Name Server's queue manager. 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. (For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your User Name Server. See "Creating a User Name Server on z/OS" on page 190 for more details.)
- Fixed-name queues on the User Name Server's queue manager. These allow information to be exchanged with other components in the broker domain.

WebSphere MQ resources that you must create yourself

Depending on the setup of your User Name Server, you might need to create some WebSphere MQ resources yourself. You can create WebSphere MQ resources using commands and utilities such as `runmqsc` and the PCF interface, or you can use WebSphere MQ Services (with WebSphere MQ Version 5) or WebSphere MQ Explorer (with WebSphere MQ Version 6).

- For WebSphere Message Broker on z/OS, you must create a queue manager in WebSphere MQ for your User Name Server. See “Creating a User Name Server on z/OS” on page 190 for more details.
- If the Configuration Manager and User Name Server do not share a queue manager, you must define the channels and transmission queues to support communication between them.
- You must define the transmission queues and channels between the User Name Server and every broker in the 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.

For more information about creating WebSphere MQ resources, see *WebSphere MQ Intercommunication*, available on the WebSphere MQ library Web page.

Considering performance in the domain

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

Message flows

A message flow includes an input node that receives a message from an application over a particular protocol (for example WebSphere MQ). The message must be parsed, although some parsers support partial parsing which might reduce processing. 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 need to consider how you split your business logic; how much work should the application do, and how much should the message flow do? Every interaction between an application and a message flow involves I/O and message parsing and therefore adds to processing time. Design your message flows, and design or restructure your 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, these have to be stored for safekeeping.

You might need to process message with a well defined structure; if so you can create MRM models for your messages. If you have no need to interrogate the structure, you can work with BLOB messages. 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. These can play a part in how message flows perform, and how efficiently messages can be processed.

For more information about these factors, see *“Optimizing message flow throughput” 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 has a search facility; enter “performance” and review the links that are returned.
- The developerWorks article on message flow performance.
- The developerWorks article on monitoring resource use.
- A developerWorks article that describes other available resources including best practices, tools, and performance data.

Optimizing message flow throughput

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

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

Consider the following options for optimizing message flow throughput:

Multiple threads processing messages in a single message flow

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

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

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

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

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

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

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

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

WebSphere MQ Everyplace and SCADA applications use a different method of message ordering as described in WebSphere MQ Mobile Transport and WebSphere MQ Telemetry Transport respectively. The broker does not provide message ordering for messages received across WebSphere MQ 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 has similar effects to increasing the number of processing threads in a single message flow, although usually provides less noticeable gains.

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

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

Copies of the message flow in multiple brokers

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

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 this, be aware that it is not possible to execute the separate message flows in the same unit of work, and if there are transactional aspects to your message flow (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 RouteToLabel, the input queue has become a bottleneck. You could use another copy of the message flow in a second execution group, but this is not appropriate if you want all messages to be handled in the order in which they appear 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 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 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 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 could 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 represents the number of messages processed from the input queue before an MQCMIT is issued.
- *Commit Interval*. This represents the time interval that elapses before an MQCMIT is invoked.

Configuring databases

Create and configure the database that is required by each broker in your domain.

A broker stores internal performance and operational data in a database that you must create and configure before you create the broker. Multiple brokers can store their tables in a single database.

You might also choose to access databases to hold application or business data. These databases, known as user databases, 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, the data that you hold in user databases might be significant. You might need to coordinate table updates, or 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 100.

z/OS For additional information about setting up databases on z/OS, see “DB2 planning on z/OS” on page 150 and “Customizing the z/OS environment” on page 137.

Configuration of databases has three phases:

1. Required: Create and configure a broker database. 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 three phases:

1. 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.

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. Enable an ODBC connection or Enable a JDBC connection.
 3. Optional: If you want your databases to participate in globally coordinated transactions, configure the environment for global coordination.

Databases overview

WebSphere Message Broker components 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 also configure them 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 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 in the specified broker database. You can create a database for each broker, or you can use one database for multiple brokers if the platforms are compatible.

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. 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 database tables required by that component 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*.

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 (x86 platform)), 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 102

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 are also supported; for example, Informix® can be used for user databases even though it is not supported for broker databases, while SQL Server running on Windows can be accessed from some Linux and UNIX systems. 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 103.

Databases created by the Default Configuration wizard

On Windows or Linux (x86 platform), 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

Broker databases contain internal operational data that is written and accessed by broker components and execution groups. You must create connections from each broker to its database using ODBC.

The number of connections it requires depends on the actions of the message flows that it processes. The following connections are required for each broker:

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

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

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

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 becomes idle for one minute

The broker is stopped

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 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. If you believe the connections that the broker might require exceeds the value for *maxappls*, increase this parameter and the associated parameter *maxagents* to new values based on your calculations.

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 access 32-bit databases” on page 127).

The execution group 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 (Integrity platform), Linux (POWER platform), Linux (zSeries platform), and Solaris (x86-64 platform), the 64-bit broker supports only 64-bit execution groups, therefore the 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 113.

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 to the data source name (DSN) for each 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 becomes idle
- 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.

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 Additionally on /OS, database connections for globally-coordinated message flows are 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 102.

If you stop the broker, it releases all current database connections.

If you are using DB2 for your database, DB2’s 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 to database quiesce, where 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, you must 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 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 102).

If a message flow that accesses the user database is deployed to a 64-bit execution group, you must 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 so 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 102.

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 113.

Creating the databases

Before you can create a broker, you must create a database for the broker to use to store its internal data; multiple brokers can store their tables in a single database.

If you need to create user databases to store your application and business data so that message flows can store, update, or retrieve the data, create the databases using these same instructions.

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 150 and “Customizing the z/OS environment” on page 137.

To create the databases on distributed systems:

1. If you are creating Oracle databases for 32-bit brokers on UNIX systems, run the `mqssetupdatabase` command before you create any database. For details, see “`mqssetupdatabase` command” on page 525. Do not complete this task 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 `WBRKKBKDB`, and keep a note of it for when you create the broker.
 - For DB2, follow the instructions in “Creating a DB2 database on Windows” on page 106 or “Creating a DB2 database on Linux and UNIX systems” on page 107.
 - For Derby, see the “`mqscreatedb` command” on page 438. Derby database support is described in “Using Derby databases on Windows” on page 108.
 - 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 `mqsprofile`. For more information about command environments on Linux and UNIX platforms, see Command environment: Linux and UNIX systems

You have now created a database for your broker and, if required, databases for your business data.

Next: return to the instructions in “Configuring databases” on page 99.

Creating a DB2 database on Windows

You can use the **mqsicreatedb** command or the DB2 Control Center to create a DB2 database. Alternatively, if you prefer, you can use any other method supported by DB2 (including command line or batch files); refer to the DB2 documentation for details of how to do this.

When you create a broker, you specify the user 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 using operating system user management; you do not have to define the user name to DB2 itself.

1. If you want to create a database with the **mqsicreatedb** command, enter the command at the command line, specifying the appropriate parameters. You must provide a user name and password that is known to DB2. Indicate that you want to create a DB2 database (not a Derby database). For more information about these and other parameters, refer to the **mqsicreatedb** command.

If you want to use the Control Center instead of the **mqsicreatedb** command, complete step 2.

2. Start the DB2 Control Center. For each database you want to create:
 - a. Expand **All Systems** in the object tree in the DB2 Control Center until you find **Databases**. Right-click **Databases** and click **Create Database** → **Standard**.
 - b. Enter a name and alias for your database. If you have a naming convention for databases, choose a compatible name. The alias name can be the same as the database name. Database names are limited to eight characters. For example, enter WBRKKBDB.
 - c. Click **Done**.
3. You must increase the database heap size to ensure it is sufficient for the broker. This task is described in “Changing the Database Heap Size on DB2 broker databases” on page 108.
4. When you have completed these steps for every database you have created, click **OK**.

If you use the DB2 command line to create the databases, 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.

1. Open a DB2 Command Line Processor window.
2. Connect to the broker database using the following command:

```
db2cmd db2 CONNECT to <YourBrokerDatabaseName>
```
3. Enter the following commands, where c:\ is the drive on which you installed DB2. You must enter your full DB2 installation path; do not use spaces or quotes.

```
db2 bind C:\SQLLIB\BND\@db2ubind.lst GRANT PUBLIC
db2 bind C:\SQLLIB\BND\@db2cli.lst GRANT PUBLIC
```
4. Repeat the previous two steps for every broker database.

Creating a DB2 database on Linux and UNIX systems

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 using operating system user management; you do not have to define the user name to DB2 itself.

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.

a. On AIX:

```
/usr/lpp/db2_08_01/instance/db2icrt -u fence_userID username
```

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

```
/opt/IBM/db2/V8.1/instance/db2icrt -u fence_userID username
```

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

3. Log on as *username*
4. Create a database (in this example called WBRKBKDB) 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 it is sufficient for the broker. This task is described in "Changing the Database Heap Size on DB2 broker databases" on page 108.
6. If you are using 32-bit execution groups, set the environment variable MQSI_LIBPATH32 to include the 32-bit database libraries.

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

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, expand the database name, then click **Configure Parameters**. The Database Configuration dialog opens.
 - b. In the Database Configuration window, click **Performance** → **DBHEAP**.
 - c. Click the cell in the **Value** column, then click the button labelled ... The Change Database Configuration Parameters dialog opens.
 - d. Set the value of the DBHEAP parameter to at least 900 (the maximum value that you can enter is 60000), then click **OK**. The Change Database Configuration Parameters dialog closes and the new value is shown in the cell in the **Pending Value** column.
 - e. Click **OK**. The new value is applied and a message is displayed to show that it 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 command line:

1. Open a DB2 command window.
2. Enter the following command, where *WBRKBD* is the name of the broker database and *900* is the number of 4 KB pages:

```
db2 update database configuration for WBRKBD using dbheap 900
```

The heap size of the broker database is changed.

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 Derby databases. Derby is a native Java database engine without ODBC support. The DB2 Run-Time Client provides the drivers that allow ODBC to access Derby databases. The DB2 Run-Time Client is used only for providing and managing the ODBC connection between 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 `mqscreatedb` command, a Windows service is also created. The service is called IBM MQSeries® Broker DatabaseInstanceMgr6, and starts automatically when Windows starts, under the user name that you supplied with the `mqscreatedb` command. The service is referred to as DatabaseInstanceMgr component, and starts the network server. DbInstMgr is the internal component name of the Database Instance Manager.

All Derby databases that you create using the `mqscreatedb` command are served by one instance of the DatabaseInstanceMgr and network server. Before the network server can function, it requires a TCP/IP port number. The default port number for Derby is 1527 (use this port when you create a Derby database). You can specify a different port number when you issue the `mqscreatedb` 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 `mqsdeletedb` to remove all Derby databases.

Run the command `mqsilist DatabaseInstanceMgr` to produce a list all of the databases that have been created by the `mqscreatedb` command. You can remove the DatabaseInstanceMgr and the network server after the last Derby database has been deleted, using the `mqsdeletedb` command.

If the password of the user name under which the Windows service runs is changed, use the `mqschangedbimgr` command to update the service with the new password. You can also use the `mqschangedbimgr` command to change the user name of the service. Use the `mqsistart` and `mqsistop` commands to start and stop the DatabaseInstanceMgr component.

Multiplicity (brokers, dbiMgrs, installations, databases)

The number of databases that you can create with the `mqscreatedb` command is limited only by availability of system resources. A maximum of one DatabaseInstanceMgr is created irrespective of how many databases have been created. If you have installed multiple instances of WebSphere Message Broker, all instances use a single instance of the DatabaseInstanceMgr component.

Removing databases and the DatabaseInstanceMgr component

Use the `mqsidedeletedb` command to clear all resources created by the `mqsicreatedb` command. When the last Derby database is deleted, the `DatabaseInstanceMgr` and network server are also stopped and removed. If the database files cannot be deleted using the `mqsidedeletedb` 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 438
- “`mqsidedeletedb` command” on page 461
- “`mqsichangedbimgr` command” on page 371

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

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

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 page applies only to the `mqsicreatedb`, `mqsidedeletedb`, and `mqsichangedbimgr` commands.

Supported database engines

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

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

Database Instance manager

The databases that are created by `mqsicreatedb` are managed by a component called the Database Instance manager. This component exists only on Windows. The component stores a list of all the databases created and which database engine is used for each database. 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 `mqsistat DatabaseInstanceMgr6` command lists all the databases that have been created using the `mqsicreatedb` command on this Windows computer. Use the `mqsichangedbimgr` command to change the user name and password under which the Database Instance manager Windows service is run. 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 108.

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. Not only is the database itself created (in either DB2 or Derby), but the ODBC data source name (DSN) is also created (with the same name).

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

Authorizing access to the 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 specify a second user ID specifically for the broker to use 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.

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.

You specify the service user ID and its password with the `-i` and `-a` flags, and the optional database connection user ID and password with the `-u` and `-p` flags.

The way that you authorize access depends on the database manager you are using, and the platform on which you have created it. This topic provides instructions for the DB2 and Oracle database managers:

- “DB2 authorization”
- “Oracle authorization” on page 113

DB2 authorization

To authorize access to a DB2 database using the DB2 Control Center:

1. Start the DB2 Control Center.
2. Expand the object tree until you find the database that you created for the broker.
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, `mqsuid`). The user ID that you select must be the user ID that you specify to be used for database access when you create the broker. 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 that is labelled **Choose the appropriate authorities to grant to the selected user** for the database. The available options are:
 - Connect database
 - Create tables
 - Create packages
 - Register functions to execute in database manager’s process
7. Click **OK**. The authorities are granted. The dialog closes.
8. Close the DB2 Control Center.

If you prefer, you can use the following commands instead of using the DB2 Control Center.

To authorize access to a DB2 database:

1. Connect to the database with a user ID that has DB2 system administration (SYSADM or DBADM) authority (substitute the correct database and ID in this command):

```
db2 connect to broker_db user SysAd_id
```

2. Issue the following command to grant the required privileges to the user ID that the broker will use to connect to the database (substitute the correct ID for your broker in this command if you are not using the sample `mqsuid`):

```
db2 grant connect, createtab, bindadd, create_external_routine on database to user mqsuid
```

For more information, see the documentation that is supplied with the DB2 Control Center. Your database administrator might also be able to offer advice and assistance.

Next: return to the instructions in “Configuring databases” on page 99.

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 DBA (database administrator) 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. The user ID needs quota in its tablespaces and sufficient privilege to allow the creation of, and updates to, the broker tables:

```
GRANT CREATE SESSION TO dbid;  
GRANT CREATE TABLE TO dbid;
```

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 135.

For further information, refer to the Oracle documentation.

Next: return to the instructions in “Configuring databases” on page 99.

Enabling ODBC connections to the databases

Set up the environment that is required for the broker to connect to broker and user databases with ODBC.

The broker uses ODBC to connect to the broker database. You can configure both ODBC and JDBC connections for access to user databases. To set up ODBC connections to the broker database, and for ODBC connections to user databases, follow the instructions in this section. For JDBC connections to user databases, see “Enabling JDBC connections to the databases” on page 128.

z/OS On z/OS systems, see Data sources on z/OS for information about enabling connections to databases.

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:

- If the broker core components operate in 32-bit mode, define a 32-bit ODBC DSN for the broker to connect to the broker database.
- If the broker core components operate in 64-bit mode, define a 64-bit ODBC DSN 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, 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 database instances are required for 64-bit connections, 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 64-bit platforms.

On distributed systems, when you define the DSNs, consider the following two factors that determine if you must define a 32-bit DSN for the database, a 64-bit DSN, or both:

- Whether the execution group and the database instance are 32-bit or 64-bit
- Whether the message flow transactions will be globally coordinated

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 referenced in your ODBCINI32 environment variable.
- Define 64-bit DSNs in the file referenced in 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; the file should be a copy of `odbc64.ini` on all platforms except Linux (x86 platform) where it should 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`. You can delete the ODBCINI64 variable; it is no longer required.

For more information about the 32-bit and 64-bit considerations, see “Broker database connections” on page 102 and “User database connections” on page 103.

When you have defined the appropriate DSNs, you must also configure the environment so that the broker can access the correct database libraries (see “Setting your environment to access 32 bit databases” on page 127).

To enable connections on distributed systems:

1. Define the ODBC DSNs according to your platform:

On Windows:

Windows provides only 32-bit support. Follow the instructions in “Connecting to a database from Windows” on page 115.

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.

Refer to the following tables to check which DSNs you must define, and follow the links for the appropriate instructions. For more information about 32-bit and 64-bit considerations, see “Broker database connections” on page 102 and “User database connections” on page 103.

	32-bit execution group	64-bit execution group
32-bit brokerLinux (x86 platform)	Broker database: 32-bit User database: 32-bit	Not possible

	32-bit execution group	64-bit execution group
64-bit broker AIX, Solaris (SPARC platform), HP-UX (PA-RISC platform), Linux (x86-64 platform)	Broker database: 32-bit and 64-bit User database: 64-bit	Broker database: 64-bit User database: 64-bit
64-bit broker HP-UX (Integrity platform), Linux (POWER platform), Linux (zSeries platform), Solaris (x86-64 platform)	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 queue managers on 64-bit platforms run in 64-bit mode.

	32-bit execution group	64-bit execution group
64-bit broker AIX, Solaris (SPARC platform), HP-UX (PA-RISC platform), Linux (x86-64 platform)	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 (Integrity platform), Linux (POWER platform), Linux (zSeries platform), Solaris (x86-64 platform)	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	Broker database: 32-bit User database: 32-bit	Not possible
64-bit broker	Not possible	Not possible

You have now configured the ODBC DSN for your broker database and the ODBC DSNs for any 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 access 32 bit databases” on page 127.

You have now enabled the broker to make connections to the broker database and to any user databases.

Next, return to the instructions in “Configuring databases” on page 99.

Connecting to a database from Windows

This topic describes how to connect to supported databases from Windows.

Configure an ODBC data source using the ODBC Data Source Administrator:

1. Click **Start** → **Settings** → **Control Panel** → **Administrative Tools** → **Data Sources (ODBC)**.

2. Click the **System DSN** tab.

When you define a new data source, select the appropriate driver for your database and complete the dialog that is displayed. Refer to your relevant database product documentation for more information.

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

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

When you define a data source for DB2 UDB you must choose the driver:

- IBM DB2 ODBC DRIVER

1. Enter the data source name (DSN) and description.
2. Select the correct database alias from the list.

You must register the data source as a system data source.

You might find it easier to use the Configuration Assistant:

1. Open the DB2 Configuration Assistant.
2. Right-click the database. 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

When you define a data source for Informix Dynamic Server, choose the driver:

- IBM INFORMIX ODBC DRIVER

Complete these steps to configure the driver:

1. 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, olsotcp).
 - The Informix data source name.
 - The user identifier to access the data source within.
 - The password for that user identifier.
2. Click **Apply**.
3. Click **Test Connection** to check your supplied values.
4. Click **OK** to finish.

Microsoft® SQL Server

When you define a data source for Microsoft SQL Server you must choose the driver:

- SQL Server

The driver level must be Version 3.60, or later.

1. Specify a name and description.
2. Select the correct server from the list.

Oracle

When you define a data source for Oracle, choose the driver:

- MQSeries DataDirect Technologies 5.20 32-BIT Oracle

The ODBC Oracle Driver Setup dialog opens.

Complete these steps to configure the driver:

1. 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.
2. 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.
3. Click **Start** → **Run**.
4. Type REGEDIT in the **Open** field.
5. Click **OK**.
6. Use REGEDIT to navigate to the correct location. Create a new registry subkey for each of your DSNs that reference an Oracle database.


```
HKEY_LOCAL_MACHINE
  SOFTWARE
    ODBC
      ODBC32.INI
```
7. Click **OK** to close ODBC Data Source Administrator.
8. Right-click **DSN**. Select **New** → **String Value**.
9. Specify **WorkArrounds** for the string.
 - a. Right-click **WorkArrounds**.
 - b. Select **Modify**.
 - c. Type the data value 536870912.

Sybase Adaptive Server Enterprise

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

- MQSeries DataDirect Technologies 5.20 32-BIT Sybase Wire Protocol

Complete these steps to configure the driver:

1. Enter the DSN name, description, and network address of the server.
2. Select **Enable Describe Parameter**. This parameter is on the **Advanced** tab.
3. Ensure the **Prepare Method** setting is 1 - Partial. This parameter is on the **Performance** tab.
4. Use REGEDIT to navigate to the correct location. Create a new registry subkey for each of your DSNs that reference an Sybase database.

```
HKEY_LOCAL_MACHINE
  SOFTWARE
    ODBC
      ODBC32.INI
```

5. Right-click the DSN, and select **New** → **String Value**. Specify **SelectUserName** for the string, and set the value to 1.

You have now configured your ODBC data source names on Windows.

Next:

You must 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 access 32 bit databases” on page 127.

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
- Ensure that the broker is authorized to access the database

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:

- On a Linux or UNIX system, if a message flow is deployed to a 32-bit execution group, define a 32-bit DSN for the broker and any user databases.
- You are using Linux (x86 platform).

To configure a 32-bit DSN for a database:

1. Copy the `odbc32.ini` sample file that is supplied either in the `ODBC32/V5.0` directory (for the HP-UX (PA-RISC platform) or in the `ODBC32/V5.2` directory (for all other platforms)) of your WebSphere Message Broker installation to a location of your choice; for example, copy the file to your user ID’s home directory. Copying this file enables each broker service user ID to use its own DSN definitions.

See the sample file contents in “`odbc32.ini` sample file” on page 305.

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 `yourodbc32.ini` file, specifying a full path and file name.

For Linux (x86 platform) only, use the `ODBCINI` environment variable.

4. On AIX, HP-UX, Solaris (SPARC platform) and Linux (x86-64 platform) 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/sql1lib/lib32
```

On Linux (x86 platform) 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. Use TCP/IP mode to connect to the database instance. For detailed instructions on how to do this, see DB2 error message SQL1224N is issued when you connect to DB2. 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.
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 using the following command:

```
vi $ODBCINI32
```

or vi \$ODBCINI on Linux (x86 platform)

- 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 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]  
Trace=0 # To turn on ODBC trace set = 1  
TraceFile=<A Directory with plenty of free space to hold trace output>/odbc32.out  
TraceDll=<Your Broker install directory>/ODBC32/V5.2/lib/odbc32.dll  
InstallDir=<Your Broker install directory>/ODBC32/V5.2  
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:

```
#####  
##### List of data sources stanza #####  
#####  
[ODBC Data Sources]  
DB2V8DB=IBM DB2 Version 8 ODBC Driver  
DB2V9DB=IBM DB2 Version 9 ODBC Driver  
ORACLEDB=DataDirect 5.2 Oracle Wire Protocol  
SYBASEDB=DataDirect 5.2 Sybase Wire Protocol  
SYBASEUTF8=DataDirect 5.2 Sybase UTF8 Wire Protocol  
SQLSERVERDB=DataDirect 5.2 SQL Server Wire Protocol  
IFORMIXDB=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 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 text 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 text 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 text 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/V5.2/lib/UKor822.so
Description=DataDirect 5.2 Oracle Wire Protocol
```

```
ApplicationsUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle server name>
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 text 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/V5.2/lib/UKase22.so
Description=DataDirect 5.2 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
ApplicationsUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
```

If you wish 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 text 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/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 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 text 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 will not work, and the broker will not be able to connect to the database.

You have now configured 32-bit ODBC database connections.

Now 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 access 32 bit databases” on page 127.

Connecting to a database from Linux and UNIX systems: 64-bit considerations

To enable the 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
- Ensure that the broker is authorized to access the database

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. This is 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 118
- If the message flows that are deployed to the broker access one or more user databases, you must also define a **64-bit** DSN for each user database too. On Linux and UNIX systems, 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 113.

On Linux and UNIX systems, 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 broker 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 113.

To configure a 64-bit DSN for a database:

1. Copy the `odbc64.ini` sample file that is supplied either in the `ODBC64/V5.0` directory (for the HP-UX (PA-RISC platform)) or in the `ODBC64/V5.2` directory (for all other platforms) of your WebSphere Message Broker installation to a location of your choice; for example, to your user ID’s home directory. This enables each broker service user ID on the system to use its own DSN definitions.

See the sample file contents in “`odbc64.ini` sample file” on page 311.

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:

- On Linux and Solaris: `LD_LIBRARY_PATH`
 - On HP-UX: `SHLIB_PATH`
 - 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. For detailed instructions on how to do this, 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]
Trace=0                # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbc64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbc64.dll
InstallDir=<Your Broker install directory>/ODBC64/V5.2
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 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 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 text 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 text 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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 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 `odbc.ini` file.
- b. In **Description**, type a meaningful description of the database. This text 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 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 which 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/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
```

If you wish to use a UNICODE UTF8 Sybase data source, add the following line to the end of your Sybase stanza:

```
Charset=UTF8
```

For an **SQLServer** 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 text field is for information only and does not affect the connection.
- c. In **Address**, type the network address of your database server (this 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/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 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 text 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 will not work and the broker will not be able to connect to the database.

10. If you are running DB2 Version 9.1 on HP-UX (PA-RISC platform), 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.

Now, you must 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 access 32 bit databases.”

Setting your environment to access 32 bit databases

When you have configured your ODBC data source names, you must also configure the environment for issuing console commands, and for running the broker, so that it can access the required database libraries. For example, if you have a DB2 broker database, you must add the DB2 client libraries to your library search path.

On Windows platforms, this is likely to have been done for you when you installed the database product.

On UNIX systems, and Linux, you need to run a profile for each database you need to access. For example, on DB2 you must run `db2profile`; other database vendors have similar profiles.

If you intend to use 32-bit execution groups on AIX, HP-UX, Solaris (SPARC platform) or Linux (x86-64 platform) 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, ensure that you add `<DB2 instance directory>/sql/lib/lib` for DB2 version 8 or `<DB2 instance directory>/sql/lib/lib32` for DB2 version 9 to the end of the `MQSI_LIBPATH32` library search path environment variable after running `db2profile`.
- If the broker needs to access an Informix user database, add `$INFORMIX/lib:$INFORMIX/lib/esql:$INFORMIX/lib/cli` to the `MQSI_LIBPATH32` library search path environment variable (see the Informix product

documentation for more information). You must ensure that any WebSphere Message Broker libraries appear *before* any Informix libraries in the library search path.

- If you are using a 64-bit Oracle instance, ensure that you add `$ORACLE_HOME/lib32` to the end of the `MQSI_LIBPATH32` library search path environment variable.

When you have set your environment to access databases, return to the instructions in “Configuring databases” on page 99.

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: Authorize access to JDBCProvider resources

Next: return to the instructions in “Configuring databases” on page 99.

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 connections 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 is connected to by your Java applications.

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 `mqsiereportproperties` command to view the list of available JDBC providers. Substitute the name of your broker in place of `broker_name`.

```
mqsiereportproperties 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:

```
mqsiereportproperties broker_name -c JDBCProviders -o Oracle -a
```

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 this to identify the specific database that you want to connect to.

A JDBCProvider has the following properties:

- `databaseType`. The database type; for example, DB2.
- `databaseVersion`. The database version; for example, 8.1.

- `type4DriverClassName`; The name of the JDBC driver class name that is used to establish a connection; for example, `com.ibm.db2.jcc.DB2Driver`.
- `connectionUrlFormat`. A pattern that represents a connection URL definition; for example, `jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password]`;
- `connectionUrlFormat Attr1-5`. If the specified URL format contains non-standard JDBC data source properties, such as a server identifier, specify these properties in addition to the standard attributes using one of five general purpose connection URL attributes.

For example, if `connectionUrlFormat = jdbc:oracle:thin:[user]/[password]@[serverName]:[portNumber]:[connectionUrlFormatAttr1]`, and `connectionUrlFormatAttr1 = Oracle SID`, the server identifier property is not included as part of the standard set of JDBC data source properties within the registry, therefore for the Oracle-specific connection URL pattern, the general purpose connection URL attribute `connectionUrlFormatAttr1` is used to represent this non-standard property in the URL definition. This attribute is substituted with its assigned value, *Oracle SID*, which is a user-defined SID.

- `databaseName`. The name of the database to which the data source entry enables connections; for example, `employees`.
- `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`.
- `serverName`. The name of the server; for example, `host1`.
- `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`. Create a new DSN entry using the `mqssetdbparms` command, as described in "Securing a JDBC type 4 connection" on page 131.

The DSN entry provides a user ID and password value pair, which are used to access the specified data source 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. If the `securityIdentity` is blank or left as the default (`default_User@default_Server`), but it is required for the connection URL pattern, or if a valid unique security identity key is entered and it cannot be found under the DSN key, the `DataSourceUserId` and `DataSourcePassword` properties under which the broker was created are used.

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. (You should assign a password to a started task user ID only after considering all of the potential security ramifications.)

4. If you want to use the provided definition, run the `mqschangeproperties` command to replace default values with those specific to your database and environment. If you are in any doubt about the required values, talk to 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. 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. For example, DB2 requires a data source login name and password on all connections.

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 database.

```
mqsisetdbparms broker_name -n datasource_name -u userID -p password
```

For example, if you want user ID `myuserid` with a password of `secretpw` to access database `OracleXADataSource1` on broker `BROKER1`, run the following command:

```
mqsisetdbparms BROKER1 -n OracleXADataSource1 -u myuserid -p secretpw
```

3. Update the **securityIdentity** property for the JDBCProvider to associate the connection with the user ID. Enter the following command:

```
mqsichangeproperties broker_name -c JDBCProviders -o service_name -n securityIdentity -v datasource
```

For example, if you are using the supplied JDBCProvider definition for Oracle:

```
mqsichangeproperties BROKER1 -c JDBCProviders -o Oracle -n securityIdentity -v OracleXADataSource1
```

You can use the same user ID and password definition for more than one JDBCProvider if appropriate; specify the same data source name 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 required, set up the environment to include the JDBCProvider service in 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 in the required way.

Complete the following steps:

1. Check that the definition of your JDBCProvider service is appropriate for coordinated transactions. 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 to the broker's queue manager.

Linux **UNIX** 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).

JDBC_DataSource is the identifier of the JDBCProvider configurable service (the value that you specified for the `-o` parameter on the `mqsichangeproperties` command).

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.

JDBCSwitch is the name of the switch file that is required to perform XA coordination. The file names for each platform are shown in the table.

Windows 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.

Platform	32-bit file	64-bit file
Linux (x86 platform)	libJDBCSwitch.so	
Windows	JDBCSwitch.dll	
HP-UX (PA-RISC platform)	libJDBCSwitch.sl	libJDBCSwitch64.sl

Platform	32-bit file	64-bit file
AIX Linux (x86-64 platform) Solaris (SPARC platform)	libJDBCSwitch.so	libJDBCSwitch64.so
HP-UX (Integrity platform) Linux (POWER™ platform) Linux (zSeries® platform) Solaris (x86-64 platform)		libJDBCSwitch.so

3. On Windows, JDBCSwitch.dll is installed in the *install_dir*\bin directory. Copy this file to the \exits subdirectory in the WebSphere MQ installation directory.
4. On Linux and UNIX platforms, switch files are installed in the *install_dir* /lib directory.

To set up access by WebSphere MQ to these switch load files, create a symbolic link to the files that are supplied in your *install_dir*/lib directory. Links are needed in the /var/mqm/exits directory, or the /var/mqm/exits64 directory, or both. The preceding table shows which platforms need links in /var/mqm/exits, /var/mqm/exits64, or both. You must specify the same name of the switch file 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
```

5. You must define a stanza for each database (DSN) that you connect to from this broker. Separate definitions are required 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.
6. 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. Open the message flow that includes the node or nodes that connect to the database that is to participate in a globally-coordinated transaction. Set the message flow property Coordinated to Yes.

Next: authorize the broker and its queue manager to access resources associated with the JDBCProvider configurable service.

Authorizing access to JDBC type 4 JDBCProvider resources

Authorize the broker and queue manager to access shared resources that are associated with the JDBCProvider. This step is required only if you want the database updates to be included in globally-coordinated transactions.

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.

1. 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.
2. 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:
 - **Windows** Complete the following steps:
 - a. Click **Start** → **Run** and enter `dcomcnfg`. The Component Services window opens.
 - b. In the left pane, expand **Component Services** → **Computers** → **My Computer** and click **DCOM Config**.
 - c. In the right pane, right-click the WebSphere MQ service labelled **IBM MQSeries Services**, and click **Properties**.
 - d. Click the **Identity** tab.
 - e. Select **This user** and enter the user ID and password for the broker service ID to associate that ID with the queue manager.
 - f. Click **OK** to confirm the change.

Next: return to the instructions in “Configuring databases” on page 99.

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 `mqsuid` and `mqsipw` in that command.

2. Connect to the broker database with this command:

```
use WBRKBKDB
```

If you have created your broker database with a different name, substitute your broker database name for `WBRKBKDB` in that command.

3. Update the retained publications table to use row-level locking with this command:

```
alter table mqsiuid.BRETAINEDPUBS lock datarows
```

If the owner of this database instance is not `mqsuid`, substitute the correct schema name in that command.

4. Apply the change with this command:

```
go
```

You can check that the change has been successfully applied by entering the commands:

```
sp_help BRETAINEDPUBS
go
```

The locking scheme is displayed: lock scheme datarows.

If the change has **not** completed, it is displayed as: lock scheme allpages.

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, set up the required ODBC or JDBC connections to the databases; see “Enabling ODBC connections to the databases” on page 113 or “Enabling JDBC connections to the databases” on page 128.

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, therefore WebSphere MQ retains its connections to the database. When the database starts again, the broker tries and fails to use these old connections.

To configure databases for coordinated message flows, follow the instructions relevant to your database manager:

- “Configuring DB2 for global coordination of transactions”
- “Configuring Oracle for global coordination of transactions” on page 136
- “Configuring Sybase for global coordination of transactions” on page 137

Configuring DB2 for global coordination of transactions

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 (x86 platform) 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 (x86 platform)), 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 shared memory:

- a. On the DB2 server, run the following commands to turn on extended shared memory:

```
export EXTSHM=ON
db2set DB2ENVLIST=EXTSHM
db2stop
db2start
```

- b. Ensure that shared memory support is turned on in the broker environment. For more information, see “Configuring global coordination with DB2 using a 32-bit queue manager” on page 257 or “Configuring global coordination with DB2 using a 64-bit queue manager” on page 259.
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 253.

Configuring Oracle for global coordination of transactions

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 `initjvm.sql` and `initxa.sql` scripts which are supplied with the Oracle installation; your database administrator (DBA) 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 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 253.

Configuring Sybase for global coordination of transactions

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 253

Customizing the z/OS 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 159, 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 160.

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 416.

When planning to work in a z/OS environment, you must complete the following tasks:

- Create started task procedures for the broker, User Name Server, and Configuration Managers 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:

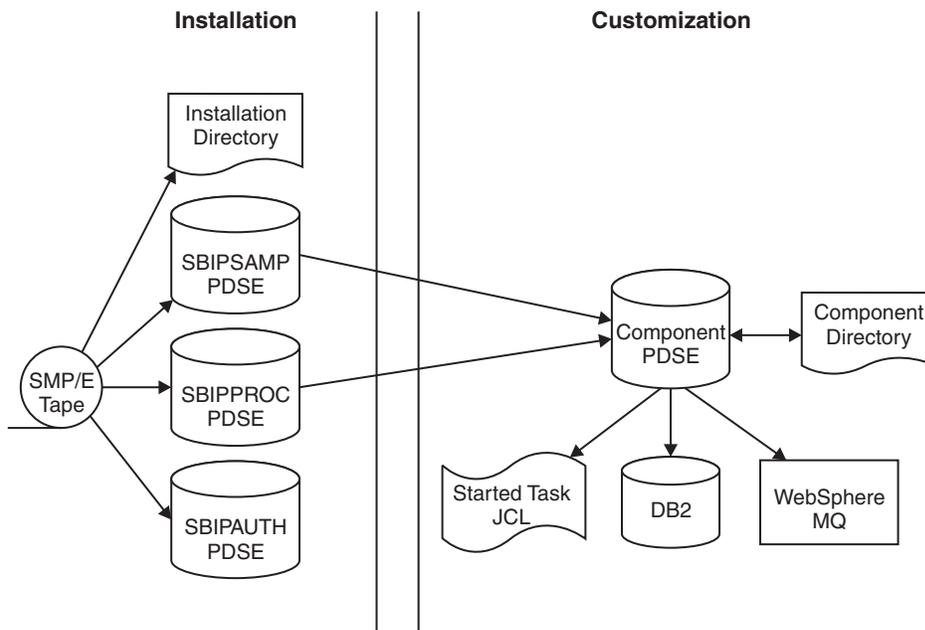
- “z/OS customization overview”
- “Customizing UNIX System Services on z/OS” on page 148
- “DB2 planning on z/OS” on page 150
- “WebSphere MQ planning for z/OS” on page 153
- “Resource Recovery Service planning on z/OS” on page 154
- “Defining the started tasks to z/OS Workload Manager (WLM)” on page 154
- “Automatic Restart Manager planning” on page 155
- “Mounting file systems” on page 155
- “Checking the permission of the installation directory” on page 156
- “Customizing the level of Java on z/OS” on page 157
- “Checking APF attributes of bipimain on z/OS” on page 157
- “Collecting broker statistics on z/OS” on page 157
- “Configuring an execution group address space as non-swappable on z/OS” on page 158

For an overview of how to create WebSphere Message Broker components, see “Creating WebSphere Message Broker components on z/OS” on page 158.

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 <h1q>.SBIPSAMP, the JCL procedures are located in the PDS <h1q>.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 140
- “Component PDSE on z/OS” on page 140
- “XPLink on z/OS” on page 141
- “Binding a DB2 plan to use data-sharing groups on z/OS” on page 141
- “Using the file system on z/OS” on page 142
- “Event log messages on z/OS” on page 142
- “Security considerations on z/OS” on page 142
- “Overview of message serialization on z/OS” on page 143

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 155

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, although a broker, Configuration Manager, and User Name Server can share a single queue manager. A broker component also needs access to a database.

Component directory 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 155

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 so that it looks like this:

```

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

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

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

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

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

For further information on mounting file systems and allocating space see “Mounting file systems” on page 155

Space requirements:

For details of the disk space required, see “Disk space requirements on z/OS” on page 545.

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 65.

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 145
3. "Serialization of input within an execution group on z/OS" on page 146

Note the following:

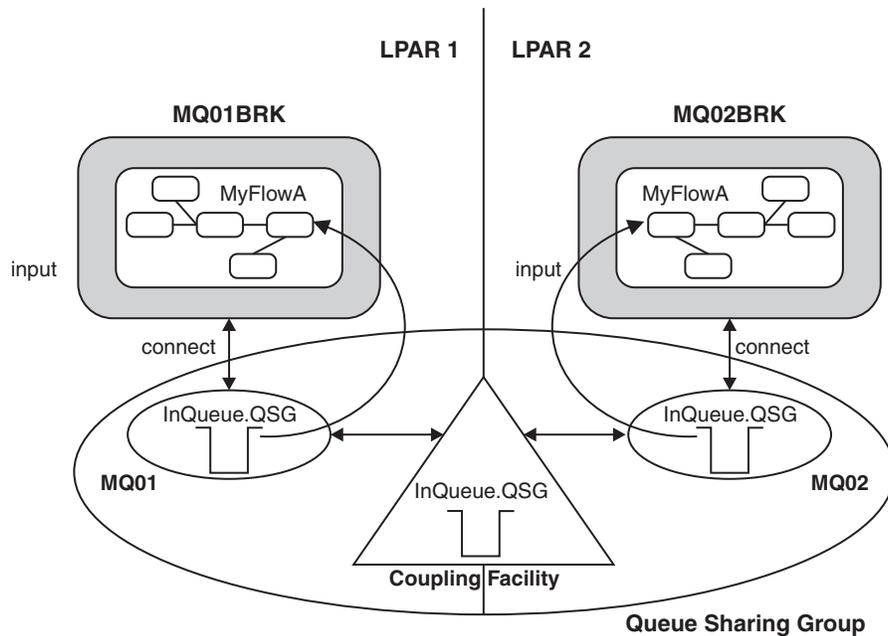
- In each of the first two examples there are two brokers configured, referred to as MQ01BRK and MQ02BRK; the broker's respective queue managers are called MQ01 and MQ02.
- The queue managers participate in the same queue sharing group. Each queue manager has a shared queue INQueue.QSG that has been 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 WEBSHERE 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 WEBSHERE 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.

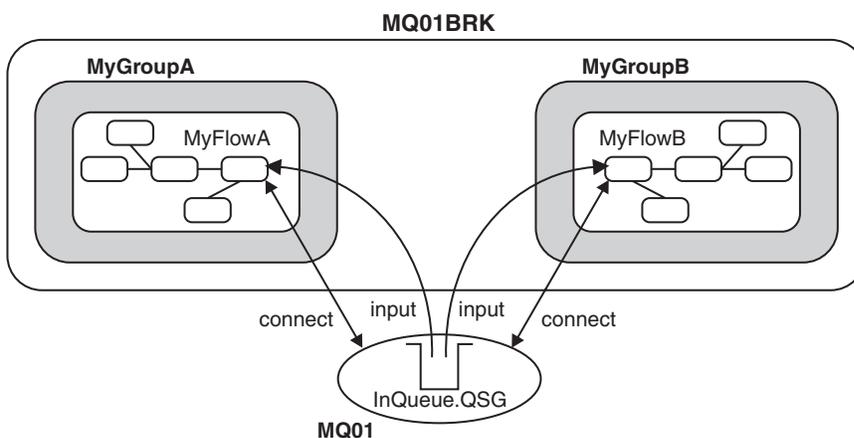
Serialization of input between separate execution groups running on the same broker on z/OS: This example demonstrates that only one MQInput node at a time is allowed to take messages from a shared queue when the same serialization token is used by message flows running in separate execution groups on 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 143:

- 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 WEBSPPHERE 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.

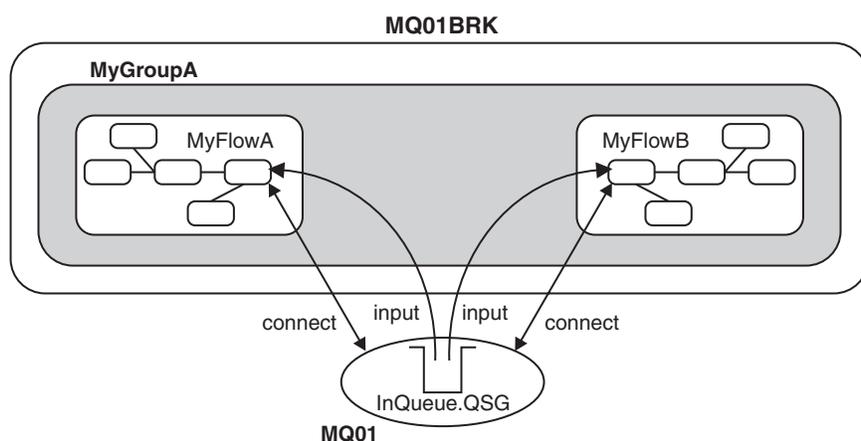
If the first execution group is cancelled by the operator, abends, or is deleted during a redeployment of the broker configuration, then the input node in the second execution group is now able to get input messages from queue INQueue. A sequence of SDSF console messages is logged, of which the following one is relevant:

```
BIP2091I MQ01BRK MyGroupB 11 THE BROKER HAS  
RECONNECTED TO WEBSPPHERE 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:

This topic gives an overview of the steps needed to:

- Configure a shared input queue for message flows
- Define a serialization token

Configure a shared input queue for message flows

The WebSphere Message Broker 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 *WebSphere MQ for z/OS Concepts and Planning Guide* 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 2048 (MQRC_PERSISTENT_NOT_ALLOWED)
- Set the Backout Threshold for the shared queue to at least 2.
This is to avoid 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.

You can use the D OMVS,O command to display your current OMVS options (view the syntax for this command).

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 as follows:

<i>Description</i>	<i>Parameter</i>	<i>Value</i>
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.	MAXCPU TIME	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 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: <ol style="list-style-type: none"> 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. In particular, 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 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 will abend 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/lib/*
/usr/lpp/mqsi/lib/wbirf/*
/usr/lpp/mqsi/lib/wbimb/*
/usr/lpp/mqsi/lib/wbieb/*
```

To set the shared-library attribute, use the **extattr** command with the +l 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 **ls -E bipimain** command:

```
-rwxr-x--- a-s- 1 USER GROUP 139264 Mar 15 10:05 bipimain
```

where **s** 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 (view the syntax for this command):

```
/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 component. The Configuration Manager and User Name Server do not require access to DB2.

WebSphere Message Broker for z/OS accesses DB2 tables using ODBC. To connect to DB2 using ODBC, the location name of the DB2 subsystem is used.

See the *DB2 for z/OS V8 Data Sharing: Planning and Administration* manual for more details.

You need to give certain user IDs access to DB2 resources and these are summarized below:

- 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 there should be a message **DSNL004I DDF START COMPLETE**. The location name is displayed just after this message. When you

customize a broker component on z/OS you create a `dsnaoini` file called `BIPDSNA0` in the broker PDSE. It contains necessary information to establish the ODBC connection.

See the *DB2 for z/OS V8 ODBC Guide and Reference* manual for more details.

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

If you choose to use the same value for the data source name and subsystem ID, you must edit `BIPDSNA0` in the broker PDSE so that the `Datasource` and `Subsystem` keywords are in one section.

See the *DB2 for z/OS V8 ODBC Guide and Reference* manual for more information on customizing this file.

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

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

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

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

You will get significant performance benefits from using the `CACHE DYNAMIC SQL` facility of DB2, because this eliminates the need to reprocess DB2 statements.

See `CACHEDYN=YES` in the DB2 information center (z/OS).

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 on z/OS 1.5 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.BROKERA
    ,DB2_TABLE_OWNER.BROKERAEG
    ,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.

Note the following:

- When accessing DB2 resources, you specify a CURRENT SQLID as a DB2 command, or through the BIPDSNA0 member.
If this Id is a group, then 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, then 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, then, if the granting user Id is removed from DB2 any permissions that it gave are also removed. This 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, then 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, then 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, then, if the granting user Id is removed from DB2 any permissions that it gave are also removed. This can prevent other users working.

See “z/OS JCL variables” on page 551 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 223 for details.

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

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

Resource Recovery Service planning 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*

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 545

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 TS0USER 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 TS0USER 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 have to 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 level of Java on z/OS

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.)

1. Enter the following command from an OMVS window to check the current version of Java:

```
java -version
```

2. Enter the following command from an OMVS window to check the current version of Java:

```
java -version
```

To check for the version of java for the current broker you are running, run this command in the java home directory that is referenced within your mqsi profile. 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 required level for this platform.

3. If you want to change the current Java version, edit 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 158.

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
```

Note that you need 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 “mqsichange flowstats command” on page 372 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 172 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

This is 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`.

Note: 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 137 section, and follow any recommendations 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 168
 - “DB2 information” on page 168
 - “Component information - broker” on page 169
 - Configuration Manager:
 - “Installation information - Configuration Manager” on page 181
 - “Component information - Configuration Manager” on page 182
 - User Name Server:
 - “Installation information - broker and User Name Server” on page 168
 - “Component information - User Name Server” on page 191
2. Set up security for the started task user IDs. Start with “Setting up z/OS security” on page 65.
 3. Plan your DB2 requirements; start with “DB2 planning on z/OS” on page 150
 4. Create the broker by carrying out the tasks listed in “Creating a broker on z/OS” on page 167. Start at “Creating the broker PDSE” on page 170.
 5. Create the Configuration Manager by carrying out the tasks listed in “Creating a Configuration Manager on z/OS” on page 181. Start at “Creating the Configuration Manager PDSE” on page 182
 - 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 160.
 6. Optionally, create the User Name Server by carrying out the tasks listed in “Creating a User Name Server on z/OS” on page 190. Start at “Creating the User Name Server PDSE” on page 191.

Connecting directly to a Configuration Manager on z/OS

Connection to the z/OS Configuration Manager directly, can be 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 connect through 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.

An alternative is to connect through an intermediate queue manager; see “Connecting to a z/OS Configuration Manager through an intermediate queue manager.”

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 158.

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 289.
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 223.

Connecting to a z/OS Configuration Manager through an intermediate queue manager

This procedure enables you to connect a Message Broker Toolkit to a z/OS Configuration Manager without the need for the WebSphere MQ Client Attach feature.

You do this by using an intermediate queue manager on another platform, for example, Windows.

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 158

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 Windows 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 289.
6. Start the Message Broker Toolkit on Windows.
7. Create a new domain connection; see “Creating a domain connection” on page 223.
8. Enter the following connection parameters for the domain connection:
hostname = (windows machine name, for example, localhost)
port = (listener port for Windows queue manager, for example, 1414)
queue manager = (z/OS queue manager, for example, MQ05)
9. Right click on **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 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

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 591.
- On Windows platforms, you have created a new user ID, the service user ID. This is specified during component creation and is used to run the component (the Configuration Manager, broker, and User Name Server).
- Refer to “Planning for security when you install WebSphere Message Broker” on page 28 for more information on user ID authorization and creation.
- 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:

- Select the task for the component and action that you require from the following list.
- Select the platform that you require from within the task.

or

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

On Windows, you can create, modify, and delete physical components “Using the Command Assistant wizard” on page 202.

If you require a default broker domain configuration on Linux or Windows, you can use the Default Configuration wizard. The Default Configuration wizard creates all the components you need to start exploring WebSphere Message Broker and run the supplied samples. See “Using the Default Configuration wizard” on page 200.

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

The following set of tasks describes creating and configuring component databases, and creating, modifying, and deleting the physical broker domain components and associated resources using the command line. For information on the using the Configuration Manager Proxy Java API, see *Developing applications using the CMP*.

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

- “Configuring databases” on page 99
- “Creating a broker”
- “Adding an execution group to a broker using the command line” on page 175
- “Adding an execution group to a broker on z/OS” on page 175
- “Creating a Configuration Manager” on page 176
- “Enabling a User Name Server” on page 186
- “Creating a User Name Server” on page 186
- “Using the Default Configuration wizard” on page 200
- “Using the Command Assistant wizard” on page 202
- “Verifying components” on page 204
- “Connecting components” on page 204
- “Preparing the environment for WebSphere Adapters nodes” on page 206
- “Modifying a broker” on page 207
- “Modifying a Configuration Manager” on page 210
- “Modifying a User Name Server” on page 213
- “Modifying access to the broker database” on page 215
- “Moving from WebSphere Message Broker on a distributed platform to z/OS” on page 216
- “Deleting a broker” on page 217
- “Deleting an execution group from a broker using the command line” on page 216
- “Deleting a Configuration Manager” on page 219
- “Disabling a User Name Server” on page 220
- “Deleting a User Name Server” on page 221

Creating a broker

You can create brokers on any platform supported by WebSphere Message Broker. On all platforms apart from HP-UX (Integrity platform), the broker runs as a 32-bit application, even if you subsequently create 64-bit execution groups on the broker. On HP-UX (Integrity platform), the broker is always a 64-bit application.

Create a broker using the command line on the system where the broker component is installed. On Windows and Linux (x86 platform), 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. Broker names are case-sensitive on all supported platforms, except Windows.

Follow the instructions appropriate to the platform on which you are creating the broker:

- Windows
- Linux and UNIX systems
- z/OS

Using WebSphere MQ trusted applications

Before you start:

You must complete the following tasks:

- Ensure that your user ID is a member of the **mqm** group. On HP-UX and Solaris, specify the user ID **mqm** as the service user ID when you create the broker. On Windows, use any service user ID that is a member of **mqm**. Refer to “Security requirements for administrative tasks” on page 591.
- 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 MQCONN call” in the *WebSphere MQ Application Programming Guide*, available on the WebSphere MQ library Web page.

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 using the “**mqsicreatebroker command**” on page 416, or modify an existing broker using the “**mqsichangebroker command**” on page 356.

Configuring a broker as a trusted application does not affect the operation of WebSphere MQ channel agents or listeners. For more information about running these as trusted applications, see the section “Running channels and listeners as trusted applications” in *WebSphere MQ Intercommunication*, available on the WebSphere MQ library Web page.

Take care when deploying user-defined nodes or parsers. Because a trusted application (the broker) runs in the same operating system process as the queue manager, a user-defined node or parser might compromise the integrity of the queue manager. Consider fully the restrictions that apply to your environment and test user-defined nodes and parsers in a non-trusted environment before deploying them in a trusted broker.

You can either configure a new broker to run as a trusted application, or modify an existing broker.

- To configure a new broker, on a command line run the **mqsicreatebroker** command with the **-t** flag, which specifies that the broker is created as a trusted application.

For example, enter the following command to create a broker called **WBRK_BROKER** as a trusted application:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw  
-q WBRK_QM -n WBRKBKDB -u wbrkuid -p wbrkpw -t
```

Refer to “Creating a broker” on page 163 for more detailed information about how to create a broker for your platform.

- To modify an existing broker:
 1. Run the **mqsisistop** 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 207 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, you create brokers on the command line; on Linux (x86 platform), you can also create brokers in the Message Broker Toolkit using the Command Assistant wizard.

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 164.
- Consider the security of the broker.

When you create a broker, if the 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 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 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 the databases” on page 111.

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.

Linux On Linux (zSeries platform) and Linux (POWER platform), the only supported database manager is DB2, ODBC is not used; the broker connects to the broker database directly. When you create the broker, use the DB2 alias of the database as the data source name.

For more information, see “Enabling ODBC connections to the databases” on page 113.

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 access 32 bit databases” on page 127.

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 `WBRKKBKDB`, enter the following command:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw  
-q WBRK_QM -n WBRKKBKDB -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](#).

For more information about the command options, see “`mqsicreatebroker` command” on page 416.

You have created and started a broker.

Next, you must perform the following tasks:

1. Create any other components that you need.
2. Create a WebSphere MQ infrastructure to connect the components together.
3. Add the broker to the broker domain:
 - To add the broker using the workbench, see “[Adding a broker to a broker domain](#)” on page 227.
 - To add the broker using the Configuration Manager Proxy Java API, see [Creating domain objects using the Configuration Manager Proxy](#).

When you have completed these tasks, the broker is ready to use.

Creating a broker on Windows

On Windows, you can create brokers on the command line or using the Command Assistant wizard in the Message Broker Toolkit.

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 164
- Consider the security of the broker.

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:

1. Ensure that the user ID with which the broker will 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 will connect to

the broker database using the user ID and password that you will specify in the `-i` and `-u` parameters of the `mqsicreatebroker` command when you create the broker.

For more information, see “Authorizing access to the databases” on page 111.

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 113.

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 `WBRKKBKDB`, enter the following command:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw  
-q WBRK_QM -n WBRKKBKDB -u dbuid -p dbpw
```

where:

- `wbrkuid` and `wbrkpw` are the user name and password under which the broker will run.
- `dbuid` and `dbpw` are the user name and password that the broker will use 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.

For more information about the command options, see “`mqsicreatebroker` command” on page 416.

You have now created and started a broker.

Now, perform the following tasks:

1. Create any other components that you need.
2. Create a WebSphere MQ infrastructure to connect the components together
3. Add the broker to the broker domain:
 - To add the broker using the workbench, see “Adding a broker to a broker domain” on page 227.
 - To add the broker using the Configuration Manager Proxy Java API, see Creating domain objects using the Configuration Manager Proxy

When you have completed these tasks, the broker will be ready to use.

Creating a broker on z/OS

The process for creating a broker involves the following tasks:

1. “Collecting the information required to create a broker” on page 168
2. “Creating the broker PDSE” on page 170

3. "Creating the broker directory on z/OS" on page 170
4. "Customizing the broker component data set" on page 171
5. "Customizing the broker JCL" on page 171
6. "Creating the environment file" on page 172
7. "Priming DB2" on page 173
8. "Creating the broker component" on page 174
9. "Copying the broker started task to the procedures library" on page 174

Collecting the information required to create a broker:

This is part of the larger task of creating a broker on z/OS.

You need to complete the information in each of the tables, at the following links, before continuing:

- "Installation information - broker and User Name Server"
- "DB2 information"
- "Component information - broker" on page 169

Installation information - broker and 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 551.

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	

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 551.

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 551.

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 bufferpool	++DB2BUFFERPOOL++	BP0	
DB2 index bufferpool	++DB2INDEXBP++	BP0	
DB2 LOB table bufferpool	++DB2LOBBP++	BP0	

<code>mqsicreatebroker</code> options	<code>++OPTIONS++</code>	Any additional optional parameters for the <code>mqsicreatebroker</code> 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 168 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 -l /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 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 command to set the group of the directory:

```
chgrp <group> <pathname>
```

For example:

```
chgrp WMQI /var/wmqi/MQP1BRK
```

You have to be the owner of the group or have superuser authority to use this command.

7. To give the group READ, WRITE, and EXECUTE access, use the 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 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 168.

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 547 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 548 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 168
 - “DB2 information” on page 168
 - “Component information - broker” on page 169
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. 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++ ' ' a11"
```

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 416 for further information.

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 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 171

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 172.

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 173.

If the user ID submitting the BIPCRBK 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 416 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 forBIP8071I: Successful command completion.

If you encounter any problems, delete the broker and recreate it using the following procedure. Note that you need 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 65 and “Summary of required access (z/OS)” on page 541 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

This topic describes how to add additional execution groups to a broker using the command line.

Before you start:

You must complete the following tasks:

- “Adding a broker to a broker domain” on page 227

When you create a broker, it has a default execution group. If you want additional execution groups, you must create them explicitly. 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 more details about creating an execution group in the workbench, see “Adding an execution group to a broker in the workbench” on page 232. For information about the CMP API, see Developing applications that use the Configuration Manager Proxy Java API.

For more 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. On Windows
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 Configuration Manager’s queue manager is listening.

QMGR

The name of the Configuration Manager’s queue manager.

BROKER

The name of the broker.

EG1

The name of the execution group that you want to create.

On completion of this task, you have requested that the Configuration Manager creates an execution group on the broker when it is next deployed.

Adding an execution group to a broker on z/OS

Before you start:

You must complete the following tasks:

- “Adding a broker to a broker domain” on page 227

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 232

To add an execution group to a broker on z/OS:

Configure and run the BIPCREG job to create an execution group. Note that you need to create the broker in the Configuration Manager before you run the BIPCREG job.

On completion of this task, you have requested that the Configuration Manager creates an execution group on the broker when it is next deployed.

Creating a Configuration Manager

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591.
- For platforms other than z/OS, see “Considering security for a Configuration Manager” on page 50 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.

- Create 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 creating a Configuration Manager, by running the product profile or console; refer to Setting up a command environment.
 - 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 204.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 223.

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 204.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 223.

Creating a Configuration Manager on Linux

Follow the steps detailed in this task for creating a Configuration Manager on Linux 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 **mqscreateconfigmgr** 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 204.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 223.

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:

```
mqscreateconfigmgr 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 204.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 223.

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 204.
2. Create a broker domain connection using the workbench. Refer to “Creating a domain connection” on page 223.

Creating a Configuration Manager on z/OS

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 182
3. “Creating the Configuration Manager directory on z/OS” on page 183
4. “Customizing the Configuration Manager component data set” on page 183
5. “Customizing the Configuration Manager JCL” on page 183
6. “Creating the Configuration Manager component” on page 185
7. “Copying the Configuration Manager started task to the procedures library” on page 185

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” on page 182

Installation 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 551.

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 551.

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 181
- “Creating the Configuration Manager PDSE” on page 182

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 170 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 181.

Create the Configuration Manager data set in TSO, identified by ++COMPONENTDATASET++, as instructed below:

1. Copy BIPCPRF from <hlq>.SBIPSAMP to ++COMPONENTDATASET++.
2. Copy the members specified in “Configuration Manager PDSE members originating in <hlq>.SBIPPROC” on page 550 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.”

- 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 BIPCPR0F (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 BIPCPR0F 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 181
 - “Component information - Configuration Manager” on page 182
2. Activate the renamed BIPEDIT file before you customize any other JCL files. Do this by running the following TSO command:

```
        ATTLIB 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++ ' ' a11"
```

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

Before starting this task, you must have completed “Customizing the Configuration Manager JCL” on page 183.

If the user ID submitting the BIPCRCM command has the appropriate WebSphere MQ authorities, you can ignore the optional **mqsicreateconfigmgr** parameters -1 and -2. If it is your intention to have a different administrator create the WebSphere MQ resources, you can consider using one of these optional parameters; see the “mqsicreateconfigmgr command” on page 428 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. You must run this job first, and to do this you need authority to access the Configuration Manager.
2. Edit BIPCRCM and submit the job with option -2. This job creates the WebSphere MQ queues. If you do not have the requisite authority, ask your WebSphere MQ system administrator to run the job.
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 Configuration Manager and recreate it using the following procedure. Note that you need 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.”

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 207.
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 210.

Now that you have made the required changes to the broker and Configuration Manager, you can create the User Name Server, and thus enable publish/subscribe services.

Creating a User Name Server

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591.
- 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 Windows, UNIX systems, and Linux, 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

Create 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.

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_UNQ_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_UNQ_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 204.

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_UNQ_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_UNQ_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 204.

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_UNQ_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_UNQ_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 204.

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_UNQ_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_UNQ_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 204.

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_UNQ_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_UNNS_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 204.

Creating a User Name Server on z/OS

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”
2. “Creating the User Name Server PDSE” on page 191
3. “Creating the User Name Server directory on z/OS” on page 192
4. “Creating the User Name Server runtime environment on z/OS” on page 192
5. “Customizing the User Name Server component data set” on page 193
6. “Customizing the User Name Server JCL” on page 193
7. “Creating the User Name Server component” on page 194
8. “Copying the User Name Server started task to the procedures library” on page 195

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 168
- “Component information - User Name Server” on page 191

Installation information - broker and 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 551.

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	

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 551.

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 190
- “Creating the User Name Server PDSE” on page 191

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 170 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.”

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 190.

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 549 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.”

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 168
 - “Component information - User Name Server” on page 191
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 193.

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 442 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 forBIP8071I: 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 194

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 291.

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 204.

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 197
- “Starting the WebSphere MQ channels and listeners” on page 198
- “Enabling applications to use Publish/Subscribe” on page 199
- “Enabling applications to use Publish/Subscribe security on z/OS” on page 199

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 186

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 61.

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 190

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 61.

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 197.

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:
`runmqtsr -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:
`runmqtsr -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_UNQ_QM -c WBRK_UNQ_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_UNQ_QM -c WBRK_UNQ_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:

- The Message Broker Toolkit and all runtime components must be installed on this system. You access the Default Configuration wizard through the Message Broker Toolkit, which is therefore available on Linux (x86 platform) and Windows only.
- **Windows** You must have Administrator privileges on Windows, 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 (x86 platform), 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” on page 201
- “Removing the default configuration” on page 202

The default configuration is described in more detail in the Installation Guide, which also describes how you can verify your installation using the sample programs.

Creating the default configuration

The wizard creates the following resources:

- A broker domain.
- A sample broker.
- A database to be used by the broker.

On Windows, the wizard defaults to whatever database manager is available.

The database that has been used is recorded on the Default Configuration Summary page. Details of the database manager are also written to the wizard's log file.

- A WebSphere MQ queue manager.

1. On Windows, the Default Configuration wizard starts automatically at the end of the installation of WebSphere Message Broker. You can launch the wizard manually from the Message Broker Toolkit Welcome page, which is displayed the first time you launch 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 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/wmbt/610/workspace/.metadata/DefaultConfigurationWizard.log`
- **Windows** `user_home_directory\eclipse\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 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 launch the Prepare the Samples wizard.

If you do not need to launch 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 launch the samples manually by clicking on the following sample:

- Pager samples

Alternatively, click **Help** → **Samples Gallery**, and expand **Application samples** → **Message Broker - Getting Started samples**.

You can view samples only when you use the information center that is integrated with the Message Broker Toolkit.

Removing the default configuration

1. Launch the Default Configuration wizard from the Message Broker Toolkit Welcome page, which is displayed after you launch 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/wmbt/610/workspace/.metadata/DefaultConfigurationWizard.log`
- **Windows** `user_home_directory\eclipse\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 using the wizard to remove the default configuration, you might need to remove the default configuration manually. For more information see [You experience problems with the default configuration](#).

Using the Command Assistant wizard

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

Using the wizard, you access the equivalent command line command through a graphical interface:

- “mqsicreatebroker command” on page 416, “mqsicreateconfigmgr command” on page 428, and “mqsicreateusernameserver command” on page 442
- “mqsichangebroker command” on page 356, “mqsichangeconfigmgr command” on page 366, and “mqsichangeusernameserver command” on page 404
- “mqsideletebroker command” on page 454, “mqsideleteconfigmgr command” on page 456, and “mqsideleteusernameserver command” on page 464

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 - queuemanagename  
BIP8099I: UserNameServer: UserNameServer - queuemanagename  
BIP8099I: ConfigMgr: configmgrname - queuemanagename  
BIP8071I: Successful command completion
```

Connecting components

This topic describes how to make connections between the Configuration Manager, the brokers, and the User Name Server.

Before you start:

You must complete the following tasks:

- “Creating a Configuration Manager” on page 176
- “Creating a broker” on page 163
- “Creating a User Name Server” on page 186

The steps below describe how to make connections between the Configuration Manager, the brokers, and the User Name Server.

If the components in your broker domain are supported by different queue managers, then you must establish WebSphere MQ connections between those queue managers to enable messages to be exchanged. It is important that each broker is able to exchange messages with the User Name Server that provides user name services for the broker.

If your broker domain components all run on the same system, and use a single queue manager, you do not need to create any WebSphere MQ connections between your brokers.

To achieve the required connection, complete the steps below. 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 steps below 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 have to set the `maxmsgl` attribute only on the transmission queue that sends messages from the Configuration Manager's queue manager to the broker's queue manager.

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
```

and on queue manager COMP2:

```
define qlocal('COMP1') usage(XMITQ) replace
```

2. Define the channels for the connection. Use sender-receiver pairs of channels for all two-way communications between queue managers that host WebSphere 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
```

- 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) trdtype(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) trdtype(tcp)
maxmsgl (104857600) replace
```

3. Create and start a listener for each protocol in use. Create the listener in WebSphere MQ Explorer in WebSphere MQ v6, or use the DEFINE LISTENER MQSC command. For more information see "Starting the WebSphere MQ channels and listeners" on page 198.
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 overheads by allowing the channels to stop when there is no message traffic, but ensures automatic startup when there are messages to transport.

You can set up a single receiver channel on the Configuration Manager's queue manager to support all sender channels created for the brokers. 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.

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).

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.

1. The WebSphere Adapters nodes need 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*.

Modifying a broker

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591
- “Creating a broker” on page 163
- 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 broker using the command line on the system where the broker component is installed. 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, created using the command line.

You can also modify the broker reference in the workbench, where it is possible to change broker properties, such as multicast properties.

Follow the link for the appropriate platform.

- “Modifying a broker on Linux and UNIX systems” on page 208
- “Modifying a broker on Windows” on page 208
- “Modifying a broker on z/OS” on page 209

Modifying a broker on Linux and UNIX systems

To modify a broker on Linux and UNIX systems:

1. Stop the broker using the **mqsisstop** command.
2. Enter the **mqsischangebroker** command with the parameters that you want to change: `mqsischangebroker brokername <<-i ServiceUserID> -a ServicePassword> <-p DatabaseSourcePassword> <-s UserNameServerQueueManagerName> <-j | -d> <-t | -n> <-l UserLilPath> <-g ConfigurationTimeout> <-k ConfigurationDelayTimeout> <-v StatisticsMajorInterval> <-P HttpPort>` where:

brokername

Is the broker name.

- i** Is the service user ID that is used to run the broker.
- a** Is the password for the broker user ID.
- p** Is the password for the broker's database user ID.
- s** Is the WebSphere MQ queue manger for the User Name Server
- j** Indicates that publish/subscribe access control is to be enabled for this broker.
- d** Indicates that publish/subscribe access control is to be disabled for this broker.
- t** Indicates that the broker runs as a WebSphere MQ trusted application.
- n** Indicates that the broker should cease to run as a WebSphere MQ trusted application.
- l** Indicates from where LIL (loadable implementation libraries) files are loaded.
- g** Is the maximum time (in seconds) to allow a broker to process a deployed message.
- k** Is the maximum time (in seconds) to allow a broker to process a minimum size deployed message.
- v** Is the time (in minutes) for the duration of the interval for collecting statistics archive records.
- P** Is the port that the broker HTTP listener will use.

For example, to change the user ID that is used to run the broker, enter the following command at the command prompt:

```
mqsischangebroker WBRK_BROKER -i wbrkuid -a wbrkpw
```

3. Restart the broker using the **mqsisstart** command. The broker restarts with the new properties.

If you cannot change a property using **mqsischangebroker**, delete the broker and then create a new one with the new properties.

Modifying a broker on Windows

Windows To modify a broker on Windows:

1. Stop the broker using the **mqsisstop** command.
2. Enter the **mqsischangebroker** command with the parameters that you want to change:

```
mqsichangebroker brokername <<-i ServiceUserID> -a ServicePassword>
<-p DatabaseSourcePassword> <-s UserNameServerQueueManagerName>
<-j | -d> <-t | -n> <-l UserLibPath> <-g ConfigurationTimeout>
<-k ConfigurationDelayTimeout> <-v StatisticsMajorInterval> <-P HttpPort>
```

where:

brokername

Is the broker name.

-i Is the service user ID that is used to run the broker.

-a Is the password for the broker user ID.

-p Is the password for the broker's database user ID.

-s Is the WebSphere MQ queue manager for the User Name Server

-j Indicates that publish/subscribe access control is to be enabled for this broker.

-d Indicates that publish/subscribe access control is to be disabled for this broker.

-t Indicates that the broker runs as a WebSphere MQ trusted application.

-n Indicates that the broker should cease to run as a WebSphere MQ trusted application.

-l Indicates from where LIL (loadable implementation libraries) files are loaded.

-g Is the maximum time (in seconds) to allow a broker to process a deployed message.

-k Is the maximum time (in seconds) to allow a broker to process a minimum size deployed message.

-v Is the time (in minutes) for the duration of the interval for collecting statistics archive records.

-P Is the port that the broker HTTP listener will use.

For example, to change the user ID that is used to run the broker, enter the following command at the command prompt:

```
mqsichangebroker WBRK_BROKER -i wbrkuid -a wbrkpw
```

- Restart the broker using the **mqsistart** command. The broker restarts with the new properties.

If you cannot change a property using **mqsichangebroker**, delete the broker and then create a new one with the new properties.

Modifying a broker on z/OS

Before you start:

To complete this task, you must have completed the following task:

- “Creating a broker on z/OS” on page 167

To modify a broker:

- Ensure that the broker is running
- Stop the broker components by issuing the command/F BROKERNAME, PC.
- When it has stopped, use the MVS MODIFY command with the changebroker parameters that you want to change. For example:

```
/F <BROKERNAME>,cb g=100,k=200
```

- Restart the broker components by issuing the following command/
BROKERNAME, SC

The broker now uses the changed parameters.

You cannot change all the parameters with which you created a broker. If you cannot modify a parameter that you need to change using the `changebroker` command, delete the broker and then create a new one. This will allow you to redefine all the parameters.

The parameters that you can change are:

g	ConfigurationTimeOut
k	ConfigurationDelayTimeout
s	UserNameServerQueueManagerName
l	UserLilPath
v	StatisticsArchiveInterval
P	HTTPPort

See “`mqsichangebroker` command” on page 356 for further information on these parameters.

Modifying a Configuration Manager

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591
- “Creating a Configuration Manager” on page 176
- 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 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 machine as the Configuration Manager component.

Follow the link for the appropriate platform.

- “Modifying a Configuration Manager on Linux and UNIX systems” on page 211

- “Modifying a Configuration Manager on Windows”
- “Modifying a Configuration Manager on z/OS” on page 212

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 213.

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 previous 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_UNQ_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 **mqsdeleteconfigmgr** 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 previous 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 manager 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_UNQ_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 **mqsdeleteconfigmgr** 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 181

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_UNQ_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 changeconfigmgr 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 manager for the User Name Server.
- j= Is the maximum Java virtual machine heap size, in megabytes (minimum 64).

See "mqsichangeconfigmgr command" on page 366 for further information on these parameters.

Moving the Configuration Manager to a new queue manager

The following steps show you how to move the Configuration Manager to a new queue manager that is on the same computer or on a different computer:

1. Use the **mqscreateconfigmgr** 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 **mqsdeploy** command, or the Configuration Manager Proxy. This tells all the brokers in the domain to associate themselves with the new Configuration Manager.
8. If you stopped the brokers in the domain in Step 2, start them using the **mqsistart** command as soon as deployment is initiated, so that the deployments can now be processed.
9. If it was not possible to stop the brokers in Step 2, ensure that any messages on the original Configuration Manager's queue manager's SYSTEM.BROKER.ADMIN.QUEUE are transferred manually to the new Configuration Manager's queue manager's SYSTEM.BROKER.ADMIN.QUEUE. This is the queue that brokers use to communicate their status to the Configuration Manager and if any status change event occurred between stopping the original Configuration Manager in Step 3 and the complete deployment in Step 7, any messages that report a change in status will have been sent to the old Configuration Manager.

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 591
- "Creating a User Name Server" on page 186
- 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" on page 214
- "Modifying a User Name Server on Windows" on page 214
- "Modifying a User Name Server on z/OS" on page 215

Modifying a User Name Server on Linux and UNIX systems

To modify a User Name Server on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:

1. Stop the User Name Server using the **mqsistop** command.
2. Enter the **mqsichangeusernameserver** command with the parameters that you want to change: `mqsichangeusernameserver <<-i ServiceUserID> <-a ServicePassword> <-d SecurityDomainName> <-r RefreshInterval> <-g AuthProtocolDataSource> <-j` | -o>` where:

- i Is the service user ID that is used to run the User Name Server
- a Is the password for the User Name Server user ID.
- d Is the security domain that the User Name Server uses on the Windows platform.
- r Is the number of seconds between each refresh of the User Name Server internal cache.
- j Indicates that groups and group memberships are defined in the data source for the authentication protocol, rather than being drawn from the operating system.
- o Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.
- g Is the name of the data source required by the authentication protocol.

For example, to change the number of seconds between each refresh of the User Name Server internal cache, enter the following command at the command prompt:

```
mqsichangeusernameserver -r 2000
```

3. Restart the User Name Server using the **mqsistart** command. The User Name Server restarts with the new properties.

If you cannot change a property using **mqsichangeusernameserver**, delete the User Name Server and then create a new one with the required properties.

Modifying a User Name Server on Windows

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 190

To modify a User Name Server.

1. At the command prompt issue the stopcomponent command to stop the User Name Server.
2. When it has stopped, use the MODIFY command with the changeusernameserver parameters that you want to change. For example:

```
MODIFY <usernameserver>,changeusernameserver r=2000
```
3. At the command prompt issue the startcomponent command.
The User Name Server now uses the changed parameters.

You cannot change all the parameters with which you created the User Name Server: if you cannot modify a parameter that you need to change using the changeusernameserver command, delete the User Name Server and then create a new one. This will allow you to redefine all the parameters.

Modifying access to the broker database

When you created the broker, you set the user ID through which the broker accesses its database tables using the DataSourceUserID parameter on the **mqsicreatebroker** command. You cannot change this user ID using the **mqsichangebroker** command.

To change the user ID that the broker uses to access the tables in the broker database:

1. Back up the database tables. The tables are listed in “mqsicreatebroker command” on page 416.
2. Delete the broker. For more information, see “Deleting a broker” on page 217.
3. Restore the database tables to the new location.
4. Create a new broker, specifying the new user ID. For more information, see “Creating a broker” on page 163.

The new broker accesses the database tables using the new user ID.

Moving from WebSphere Message Broker on a distributed platform to z/OS

Recommendations on how to define resources for WebSphere Message Broker for z/OS are given in the following topics:

- “z/OS customization overview” on page 138
- “Customizing the z/OS environment” on page 137
- “Creating a broker on z/OS” on page 167
- “Creating a User Name Server on z/OS” on page 190
- “Creating a Configuration Manager on z/OS” on page 181
- “Administration in z/OS” on page 537

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

Before you start:

You must complete the following tasks:

- “Adding an execution group to a broker using the command line” on page 175

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” on page 234

To delete an execution group from a broker using the command line:

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 1414 -q QMGR -b BROKER -e EG1
```

where

host Is the host name or IP address of the Configuration Manager for the domain on which the broker resides.

1414 Is the port on which the Configuration Manager’s queue manager is listening.

QMGR Is the name of the Configuration Manager’s queue manager.

BROKER Is the name of the broker.

EG1 Is the name of the execution group that you want to delete.

On completion of this task, the execution group is no longer running on the specified broker. In addition, any message flows that were running on the execution group are no longer running.

Deleting a broker

Before you start:

You must complete the following tasks:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591
- Remove the broker from the broker domain in the workbench (“Removing a broker from a broker domain” on page 230).

Delete a broker using the command line on the system where the broker 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 broker, by running the product profile or console; refer to Setting up a command environment.

You can remove the broker from the broker topology using the workbench, but the physical broker will not be deleted until the broker is physically deleted from the command line.

Follow the link for the appropriate platform.

- “Deleting a broker on Linux and UNIX systems” on page 218
- “Deleting a broker on Windows” on page 218
- “Deleting a broker on z/OS” on page 218

Deleting a broker on Linux and UNIX systems

To delete a broker on Linux and UNIX systems:

1. Stop the broker using the **mqsistop** command.
2. Enter the following command to delete the broker:

```
mqsdeletebroker 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

Windows To delete a broker:

1. Stop the broker using the **mqsistop** command.
2. Enter the following command to delete the broker:

```
mqsdeletebroker WBRK_BROKER
```

where:

WBRK_BROKER is the broker name.

On completion of this task, you have:

- Stopped the Windows service that runs the broker.
- Removed the broker's data from the database.
- Removed the record for the component in the broker registry. It is therefore removed from the list of components displayed by issuing the **mqsilist** command.

Deleting a broker on z/OS

To delete a broker:

1. Remove the broker from the broker domain, in the workbench. Refer to "Removing a broker from a broker domain" on page 230.
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 need to 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 591

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”
- “Deleting a Configuration Manager on Windows” on page 220
- “Deleting a Configuration Manager on z/OS” on page 220

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 “mqsisstop command” on page 531command.
2. Delete the Configuration Manager using the “mqsideleteconfigmgr command” on page 456 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 477 command.
- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the “mqsideleteconfigmgr command” on page 456 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 “mqsisstop command” on page 531 command.
2. Delete the Configuration Manager using the “mqsideleteconfigmgr command” on page 456 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 477 command.
- Preserved all internal data associated with the Configuration Manager, unless you specify the -n parameter on the “mqsideleteconfigmgr command” on page 456 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 **mqschangebroker** and **mqschangeconfigmgr** 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 **mqschangebroker** 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 591
- 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 220 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” on page 222
- “Deleting a User Name Server on z/OS” on page 222

Deleting a User Name Server on Linux and UNIX systems

To delete a User Name Server on Linux and UNIX systems; AIX, HP-UX, Linux (zSeries platform), Linux (x86 platform) and Solaris:

1. Stop the User Name Server using the **mqsistop** command.
2. Enter the following command to delete the User Name Server:

```
mqsdeleteusenameserver
```

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:

```
mqsdeleteusernameserver
```

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:

<i>Delete jobs</i>	<i>Description</i>
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.

Before you start:

You must have created the physical broker domain components. Refer to “Configuring broker domain components” on page 162.

You can configure a broker domain using the workbench only on Linux (x86 platform) and Windows. 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 as you require. You can launch the workbench in 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 that is added during installation, labelled 'WebSphere Message Broker Toolkit'.
- **DETAILS STILL TO BE CONFIRMED** Use the following commands in a command prompt from their location in the package group directory:
 - **Linux**

```
eclipse.bin -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”
- “Modifying domain connection properties” on page 225
- “Deleting a domain connection” on page 226
- “Adding a broker to a broker domain” on page 227
- “Modifying broker properties” on page 229
- “Removing a broker from a broker domain” on page 230
- “Removing deployed children from a broker” on page 231
- “Adding an execution group to a broker in the workbench” on page 232
- “Deleting an execution group” on page 234
- “Removing deployed children from an execution group” on page 235

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. Refer to “Changing Broker Administration preferences” on page 271.

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, or the number of changes to make to the configuration of the broker domain during the deployment.

Creating a domain connection

This topic describes how to create a domain connection.

Before you start:

Complete the following tasks:

- “Creating a Configuration Manager” on page 176.
- 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 198.

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 52. 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 **Server Project**. The Server Project is the container for your domain connection. If you have not already created a server project, you can specify the name of a new project here. The server project is created with the domain connection.
 - b. The **Connection name**. 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 server project holds the .configmgr domain connection file.

The view of the broker domain is displayed in the Domains view.

Modifying domain connection properties

Before you start:

To complete this task, you must have completed the following tasks:

- “Creating a domain connection” on page 223
- Disconnect from the broker domain. Refer to “Connecting to and disconnecting from the broker domain” on page 283.

Modify the parameters of the .configmgr domain connection file that are used to connect to the Configuration Manager.

To modify the domain connection properties.

1. Switch to the Broker Administration perspective.
2. In the Broker Administration Navigator view, expand Domain Connections, and open your server project.
3. Right-click the .configmgr domain connection file and click **Open With** → **Domain Connection Editor**. From here you can change:
 - a. The name of the WebSphere MQ queue manager that the Configuration Manager is using. This property is mandatory.
 - b. The host name or IP address of the machine on which the Configuration Manager is running. This property is mandatory.
 - c. The TCP port on which the WebSphere MQ queue manager is listening. This property must be a valid positive number.

- d. Optional: The name of the 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 Configuration Manager's queue manager by using the WebSphere MQ:

- `runmqsc` command to create a channel with options `CHLTYPE(SVRCONN)` and `TRPTYPE(TCP)`, or
- 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. You can leave this box blank if it does not apply to your domain connection.

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

- 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 52 for more information). The cipher suite field displays available cipher suites. Click **More** to configure a custom cipher suite in the Broker Administration preferences window. If a cipher suite is not specified, all other fields in the SSL section are unavailable.

4. Close the editor.
5. You are prompted to save the changes; click **Yes**. If the broker domain is connected, you are prompted to disconnect before you can save your changes. The `.configmgr` domain connection file is updated with the new parameters.

You can also change the domain connection parameters for a *disconnected* domain, from the Domains view. Right-click the disconnected domain and click **Edit Parameters**. This opens the Domain Connection editor. Follow steps 3 to 5 above to make the changes you require.

Deleting a domain connection

Before you start:

You must complete the following task:

- "Creating a domain connection" on page 223

To delete a domain connection, delete the corresponding `.configmgr` file from your server project.

The configuration of the broker domain is stored in the Configuration Manager and is not affected by deleting the domain connection. If you create a new domain connection to the same Configuration Manager, the broker domain will be configured and available for use.

The following steps show you how to delete a domain connection.

1. Switch to the Broker Administration perspective.
2. In the Broker Administration Navigator view:
 - a. Expand Domain Connections
 - b. Open the servers project (called Servers by default).
3. Right-click the domain connection file(<connection name>.configmgr) and click **Delete**.
4. Click **OK** at the prompt, to confirm that you want to delete the domain connection.

If the broker domain in the Domains view is connected, you are prompted to disconnect from the broker domain before it can be deleted. Click **Yes** to disconnect from the broker domain. Clicking **Cancel** at this prompt, cancels deletion of the domain connection.

On completion of this task:

- The `.configmgr` file is removed from the server project.
- The view of the broker domain and its hierarchy is removed from the Domains view.

Adding a broker to a broker domain

This topic describes how to add a broker to a broker domain.

Before you start:

You must complete the following tasks:

- “Creating a broker” on page 163
- “Connecting to and disconnecting from the broker domain” on page 283

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 204.
 - 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 271. 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

Before you start:

You must complete the following task:

- “Adding a broker to a broker domain” on page 227

You can copy a broker that you have previously added to a broker domain. The new broker reference can be pasted within the same broker domain only.

The new broker inherits the same short and long description as the original broker, and the same execution groups are added; 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" on page 230.
- 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 227
- Start the broker. See "Starting and stopping a broker" on page 287.

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 323.
 - 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 242.

- 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

Before you start:

You must complete the following tasks:

- “Adding a broker to a broker domain” on page 227

You might need to rename a broker (and possibly its queue manager; refer to “Modifying broker properties” on page 229) if your original attempt at creating a broker reference contained an error.

Renaming a broker is simpler than deleting and re-creating it.

The following steps show you how to rename a broker:

1. Switch to the Broker Administration perspective.
2. In the Domains view, right-click the broker and click **Rename**. The **Rename Broker** dialog is displayed.
3. In the **New name** field, type the new name of the broker.
The name must be exactly the same name as that specified on the **mqsicreatebroker** command.
4. Click **Finish** to rename the broker.

The broker’s name is updated in the Domains view, and in the Topology editor.

Removing a broker from a broker domain

Before you start:

You must complete the following task:

- “Adding a broker to a broker domain” on page 227

Removing a broker from a broker domain deletes its broker reference in the configuration repository. The broker is not deleted from the system when you perform this task; it simply marks the broker as logically deleted from the configuration repository.

If you want to move a broker from one topology to another, you need to delete and recreate the broker physically (using the **mqsdeletebroker** 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 217 for further information.

The following steps show you how to remove a broker from a broker domain.

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the broker domain to reveal the broker that you want to remove.

To remove more than one broker from the same broker domain, select each broker while holding down the Ctrl key.

3. Right-click the broker and click **Delete**.
4. Click **OK** at the prompt to confirm that you want to remove the broker from the broker domain.
5. You are, by default, prompted to deploy the updated publish/subscribe topology configuration.

You only need to deploy the topology if you are using publish/subscribe and want to share publications or subscriptions.

If you are *not* using publish/subscribe, click **None**, as a deployment is not necessary; when removing a broker from the domain, the Configuration Manager automatically requests the broker component to stop message flows that are running and to tidy up any resources being used. (Although if this operation fails for any reason, you can again request the broker to tidy up by deploying a delta topology.)

You can set user preferences so that you are not prompted to deploy the publish/subscribe topology. Instead, you can choose for either a complete or delta deployment to be performed automatically. Alternatively, if you are not using publish/subscribe, you might prefer to set the preference so that *no* automatic deployment takes place.

The broker and its execution groups are removed as components of the broker domain. Confirmation of the broker’s deletion is in two places:

- The broker is removed from the Domains view.
- The broker icon is removed from the Broker Topology editor. If the broker was connected to another broker, this connection is also removed.

To delete the physical broker after you have removed the broker from the domain, refer to “Deleting a broker” on page 217.

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 an execution group 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 toolkit and may be reused if required.

To remove the deployed children from a broker:

1. Switch to the Broker Administration perspective.

2. In the Domains view, expand the broker domain to reveal the broker with which you want to work.
3. Right-click the broker, and click **Remove Deployed Children**.
4. Click **OK** at the prompt to confirm that you want to delete all execution groups on the broker.

This will remove all message flows and message sets from all execution groups, and delete all execution groups. An automatic broker configuration deploy is immediately performed for the broker to save the changes.

A BIP08921 information message is displayed, to show that the request was received by the Configuration Manager. Verify the results of the deployment by opening the Event Log.

Adding an execution group to a broker in the workbench

This topic describes how to add additional execution groups to a broker in the workbench.

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. 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 175. For information about the CMP API, see *Developing applications that use the Configuration Manager Proxy Java API*.

For more details about why you might want to create multiple execution groups, see *Execution groups*.

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 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 as a 64-bit application.
To create a 64-bit execution group, the broker’s queue manager must also be 64-bit.

You can change the default value for this option, which is initially set to 32-bit; 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

Before you start:

You must complete the following tasks:

- “Adding an execution group to a broker in the workbench” on page 232
- “Adding a broker to a broker domain” on page 227

You can copy an existing execution group from a broker to another broker within the same broker domain. 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.

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 232

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

Before you start:

You must complete the following task:

- “Adding an execution group to a broker in the workbench” on page 232

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 216.

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

To configure a publish/subscribe topology, you need to do the following things:

1. Design and configure your broker domain.
For further information, refer to “Planning a broker domain” on page 89 and “Configuring broker domain components” on page 162
2. Define the topic trees that you require.
For further information, refer to Topics and “Adding a new topic” on page 252.
3. Decide which security options to use.
For further information, refer to “Publish/subscribe security” on page 70 and “Securing the publish/subscribe domain” on page 78.

Setting up the broker domain for publish/subscribe

Refer to the following topics:

- “Creating a broker” on page 163
- “Modifying a broker” on page 207
- “Adding a broker to a broker domain” on page 227
- “Configuring broker domain components” on page 162

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:

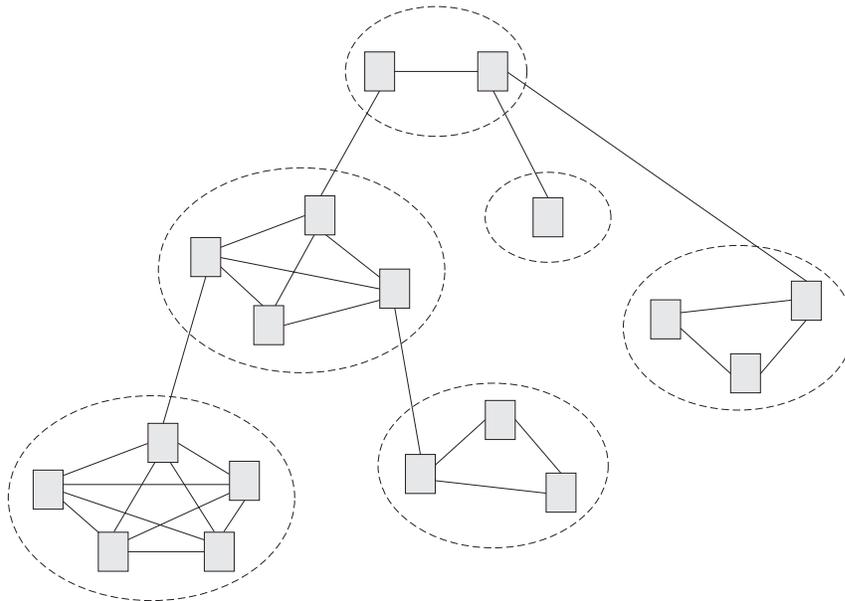
- Client applications can communicate with a nearby broker rather than with a distant broker, thereby getting better response times.
- By using more than one broker, more subscribers can be supported.

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.

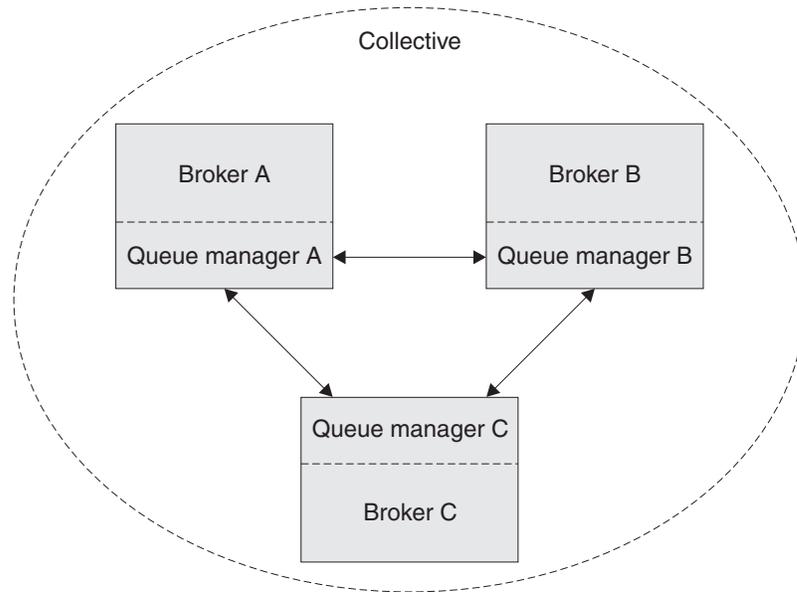


Collectives: A *collective* is a set of brokers that are fully interconnected and form part of a multi-broker network for publish/subscribe applications.

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

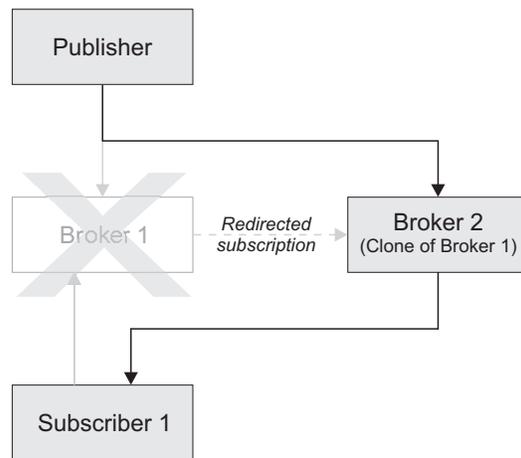
These properties apply to all Multicast messages.

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 Brokers Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class `com.ibm.broker.config.proxy.CollectiveProxy`.

The following steps show you how to 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 Brokers Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class `com.ibm.broker.config.proxy.CollectiveProxy`.

The following steps show you how to 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 Brokers Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class `com.ibm.broker.config.proxy.CollectiveProxy`.

The following steps show you how to 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 Brokers Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications using the CMP and Class `com.ibm.broker.config.proxy.CollectiveProxy`*.

The following steps show you how to remove a broker from a collective.

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double-click the Topology item to open the Broker Topology Editor.
4. Right-click the connection that you want to delete and select Delete.

Setting up a multicast broker

You can set up a multicast broker by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications that use the Configuration Manager Proxy Java API and Class `com.ibm.broker.config.proxy.BrokerProxy.MulticastParameterSet`*.

To enable a broker to be capable of handling multicast requests you must perform the following steps:

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 panel 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 ff02:0:0:0:0:0:1 through ff02:ffff:ffff:ffff:ffff:ffff:ffff

The default value is ff02:0:0:0:0:0:1

IPv6 Maximum Address

The highest IPv6 address that the broker can use for its multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:1 through ff02: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

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 248.

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 2000, 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 children, *foo* that is enabled for multicast, and *bar* that is not enabled for multicast.

The three possible settings are:

Accept

The default. 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 *foo* over multicast, but does not receive any messages published on *bar*.

Reject A multicast subscription to a topic with children that are disabled for Multicast is rejected by the broker. Subscriptions to *multicast/#* are rejected.

Revert Subscriptions to a topic that is disabled for multicast, or has children that are disabled for multicast, result in unicast transmission. A multicast subscription to *multicast/#* receives messages published on *foo* and *bar*, but the messages are sent unicast rather than multicast.

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 panel 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. But 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

Note: This property is not used in Version 6.

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 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. You must restart the broker for the changes that you have made to take effect.

Before you can use multicast, you must define the topics that are capable of being multicast. See "Making topics multicast" on page 248.

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 made using `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 "mqsichangeproperties command" on page 381.

Property name	mqsichangeproperties parameter
Multicast Enabled	multicastEnabled
IPv4 Minimum Address	multicastMinimumIPv4Address
IPv4 Maximum Address	multicastMaximumIPv4Address
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:

```
mqschangeproperties 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 also their values.

For the changes to be effective, you must 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. This might cause problems when you try to restart the broker.

The following actions might help stop this happening.

1. Tune the publish/subscribe topology:
 - a. Balance execution groups across more brokers; this means that fewer execution groups need to start at the same time and have concurrent locks for the same DB2 subsystem.
 - b. Put the brokers in publish/subscribe collectives; this reduces the number of subscriptions in a single broker table and reduces the amount of concurrent access to DB2. See “Publish/subscribe topologies” on page 236 for more information about this.
2. Increase the IRLM storage that is available:
 - a. Set the value of MAXCSA so high that the ECSA that is required by the IRLM never reaches this value. Because IRLM gets storage only when it needs it, choose a value that is higher than you expect IRLM to need.
 - b. If you are unable to choose a value of MAXCSA sufficiently high that it cannot be exceeded by the ECSA that is required by the IRLM, use the option PC=YES on the START irlmproc command. This causes the IRLM to place in its private address space the control block structures that relate to locking. There is more information about this in the IBM Redbooks publication *DB2 UDB for OS/390 Version 7 Performance Topics, SG24-5351*.

Note: There might be a slight (approximately 1 to 2 percent) performance degradation when you run with PC=YES. See *DB2 Universal Database for OS/390 and z/OS Version 7 Administration Guide, SC26-9931* for more information.

Setting up cloned brokers

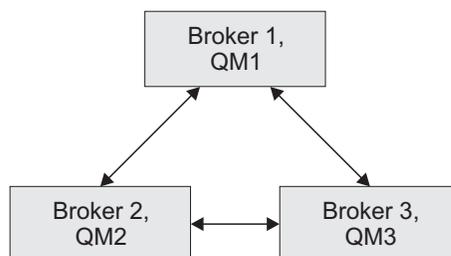
Each broker that is to be cloned with other brokers must be told which brokers are to be its clones.

- 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:

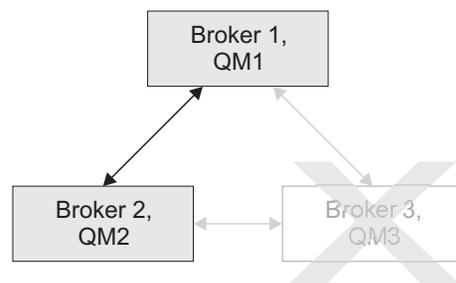
1. `mqsichangeproperties broker4 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "broker1,QM1,broker2,QM2,broker3,QM3"`
2. `mqsichangeproperties broker1 -e default -o DynamicSubscriptionEngine -n clonedPubSubBrokerList -v "+broker4,QM4"`
3. `mqsichangeproperties broker2 -e default -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” on page 252
- “Deleting a topic” on page 252
- “Querying subscriptions” on page 253

Adding a new topic

You can define a new topic explicitly by using either the Message Brokers Toolkit or the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit.

For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications using the CMP and Class com.ibm.broker.config.proxy.TopicProxy*.

You can define a new topic implicitly by sending to the message broker a Publish command that specifies the new topic.

However, to define a new topic explicitly, 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 either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications using the CMP and Class com.ibm.broker.config.proxy.TopicProxy*.

To delete a topic:

1. Switch to the Broker Administration perspective.
2. In the Domains view, expand the appropriate broker domain.
3. Double-click **Topics** to open the Topics Hierarchy Editor.
4. In the Topics Hierarchy editor, right-click the topic that you want to delete, and select **Delete**; alternatively, select the topic that you want to delete and press the Delete key, or select Delete from the Edit menu.

The topic is deleted locally, but the delete is not effective until you do a save.

Querying subscriptions

You can query a subscription either by using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see Developing applications using the CMP and Class `com.ibm.broker.config.proxy.TopicProxy`.

To query a subscription:

1. Switch to the Broker Administration perspective.
2. In the **Domains** view, click **Subscriptions** from the list of domain objects shown; the Subscriptions Query Editor opens in the workbench.

You can also open the editor by double-clicking the **Subscriptions** item in the tree, or by right-clicking the **Subscriptions** item and clicking **Open**, or by clicking the **Subscriptions** item and clicking **Enter**.

3. Fill in the fields that are required to generate your subscriptions query.
To generate your query, you might not need to fill in all the fields shown.
4. Click **Query**. The results of your query are displayed in the lower part of the edit window.

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
- Create a broker

On distributed platforms, the default behavior of the broker is to manage all message flow transactions 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 using the XA protocol standard. For more information about global coordination of transactions, see "The Transactional model" on page 254.

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 154.

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 135.
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 257
 - “Configuring global coordination with DB2 using a 64-bit queue manager” on page 259
 - “Configuring global coordination with Oracle using a 32-bit queue manager” on page 263
 - “Configuring global coordination with Oracle using a 64-bit queue manager” on page 265
 - “Configuring global coordination with Sybase” on page 268
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 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

A message flow can consist of the following constituent parts:

- An input queue
- The message flow
- One or more database tables
- One or more output queues

A typical sequence of events when a message flow processes a message is described below:

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.

Note that this sequence of events makes no distinction between accessing tables and writing output messages. Although a flow often produces some sort of output message, there is no real distinction 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=====
      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, there is no change to the state of the data; this 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 taken place, but no more changes will 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 above, the broker, queuing systems, and databases have a *transactional model*. As processing proceeds, additional data is stored 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=====
      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, there is no extra data in the system; this 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 there is no extra data in the system once more. Between these events, there is no change to the state of the extra data and this 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, so 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 following a subsequent failure will not undo the changes).

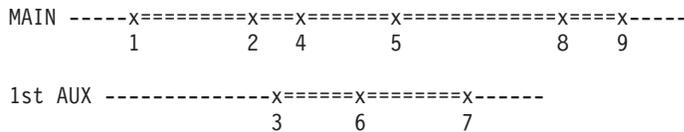
The mode of operation that is described above is known as *coordinated transaction mode*. The successful completion of a transaction is known as its *commit*. Unsuccessful completion is known as *rollback*.

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 need to be sent even when the updates to the main tables and queues are rolled back.

To satisfy such requirements, WebSphere Message Broker allows changes to queues and tables to be take place in a separate transaction. This behavior is illustrated in the following diagram:



The *MAIN* line represents the main transaction, which includes the extra data that is needed to restore the original state should the need arise. 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 need to be made under the auxiliary transaction are complete and it commits the changes.

If the message flow were to fail 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 so, if the message flow updates tables that belong to different database instances (different data source names), there is an auxiliary transaction for each database. You must commit or roll back these transactions individually. Any updates that have not been committed or rolled back when the operation completes (at time 9 in the example above) 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 other 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 above. In the case of WebSphere MQ, each individual uncoordinated read or write operation to a queue has an implied commit action so 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 above description mentions message flows but does not mention nodes. The way that a message flow is divided into nodes has no effect on transactions. For operations on databases, any number of nodes can make updates to the main transaction and to any number of auxiliary transactions without restriction.

Entirely Uncoordinated Transactions

When all database updates are done within auxiliary transactions, set the **Coordinated Transaction** attribute of the message flow to no to affect all table references outside the main transaction. This means that you do not have to specify the attribute on each database operation.

Configuring global coordination with DB2 using a 32-bit queue manager

If you want to globally coordinate message flow transactions with updates in DB2 databases, configure the broker environment. This topic describes how to configure the broker environment when the broker uses a 32-bit 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 are 32-bit. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate only message flows that are deployed 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 135.
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 as well as for the user databases.

On Linux (x86 platform) 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=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 is used by the broker to connect to the database. The user name was defined using the `-u` flag on the `mqsicreatebroker` command when creating the broker. If the `-u` parameter was not specified, the data source user name is the same as the broker’s service user name, which was defined using the `-i` flag on the `mqsicreatebroker` command.
 - `MyPassword` is the password associated with the user name.
- e. Accept the default values for all the other lines in the stanza. For example, on AIX:

```
XAResourceManager:
Name=DB2
SwitchFile=/opt/mqsi/sample/xatm/db2swit
XAOpenString=db=MYDB,uid=wbrkuid,pwd=wbrkpw,toc=t
XACloseString=
ThreadOfControl=THREAD
```

On Windows:

- a. From the Start menu, open the graphical administration tool for your version of WebSphere MQ:
 - WebSphere MQ Version 6: WebSphere MQ Explorer
- b. Open the queue manager’s Properties dialog, then open **XA resource managers** (Version 6) page.
- 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 as 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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
 - *MyPassword* is the password 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.
3. Stop then restart the queue manager to apply the changes because `qm.ini` is read only when the queue manager starts.

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.

Now, you can deploy globally coordinated message flows to the broker.

Configuring global coordination with DB2 using a 64-bit queue manager

To globally coordinate message flow transactions with updates in DB2 databases, configure the broker environment. This topic describes how to configure the broker environment when the broker uses a 64-bit queue manager.

Before you start:

- Ensure that the databases are configured for global coordination of transactions.

All WebSphere MQ Version 6 queue managers on 64-bit platforms are 64-bit. 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 be 64-bit too.

To configure your broker environment for global coordination using a 64-bit queue manager as the transaction manager:

1. On Linux and UNIX platforms, create the following symbolic links to specify the location of the ODBC database drivers and switch file that are supplied with WebSphere Message Broker:

```
ln -s install_dir/sample/xatm/db2swit /var/mqm/exits64/db2swit
```

Where `install_dir` is the location in which WebSphere Message Broker is installed

2. Follow the instructions appropriate to your execution groups:
 - “32-bit execution groups”
 - “64-bit execution groups” on page 261

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. On Linux and UNIX platforms, 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 as well as for the user databases.

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=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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker’s service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
 - *MyPassword* is the password 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
```

On Windows:

- a. From the Start menu, open the WebSphere MQ Explorer graphical administration tool.
- b. Open the queue manager's Properties dialog, then open the **XA resource managers** page.
- 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
```

- 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 as appropriate for your configuration:

- *install_dir* is the location in which the WebSphere Message Broker instance is installed.
- *MyDataSource* is the name of the data source to which you want to connect.
- *MyUserId* must be the user name that is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
- *MyPassword* is the password 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.
3. Stop then restart the queue manager to apply the changes because `qm.ini` is read only when the queue manager starts.

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.

Now, 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. 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 as well as for the user databases.

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=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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
- *MyPassword* is the password 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 when the queue manager starts.

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.

Now, you can deploy globally coordinated message flows to the broker.

Configuring global coordination with Oracle using a 32-bit queue manager

If you want to globally coordinate message flow transactions with updates in Oracle databases, configure the broker environment. This topic describes how to configure the broker environment when the broker uses a 32-bit 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 are 32-bit. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate only message flows that are deployed to 32-bit execution groups.

To configure your broker environment for global coordination using a 32-bit queue manager as the transaction manager with the DataDirect drivers:

1. On Linux (x86 and x86-64 platforms) and UNIX, run the `mqsimanagexalinks` command.
2. On Linux (x86 and x86-64 platforms), 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 as well as for the user databases.

On Linux (x86 and x86-64 platforms) 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=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA+SQLNET=MyServerName+HostName=MyHostName+PortNumber=
MyPortNumber+Sid=MySID+ACC=P/MyUserId/MyPassword+sestm=
100+threads=TRUE+DataSource=MyDataSourceName+DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```
- c. The switch file is supplied by WebSphere Message Broker and varies by platform. The following table shows the file path of the switch file for each platform:

Table 1.

Platform	Switch file path
AIX	UKor8dte22.so
HP-UX (PA-RISC platform)	UKor8dte20.sl
Linux (x86 platform)	UKor8dte22.so
Linux (x86-64 platform)	UKor8dte22.so
Solaris (SPARC platform)	UKor8dte22.so

- d. On the `XAOpenString` line, replace the following values with values that are appropriate for your configuration:

- *MyServerName* is the name of the Oracle server.
 - *MyHostName* is the name of the TCP/IP host on which the Oracle database resides.
 - *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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
 - *MyPassword* is the password associated with the user name.
 - *MyDataSourceName* is the ODBC data source name for the database.
- e. Accept the default values for all the other lines in the stanza. For example, on AIX:
- ```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc22.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
```

#### On Windows:

- a. From the Start menu, open the graphical administration tool.
- b. Open the queue manager's Properties dialog, then open the **XA resource managers** page.
- 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\UKor8dtc22.dll
```
- d. In the **XAOpenString** field, paste the following string:

```
ORACLE_XA+SQLNET=MyServerName+HostName=MyHostName+PortNumber=MyPortNumber+Sid=
MySID+ACC=P/MyUserId/MyPassword+sestm=100+threads=TRUE+DataSource=
MyDataSourceName+DB=MyDataSourceName+K=2+
```
- e. In the **XAOpenString** field, replace the values as appropriate for your configuration:
  - *MyServerName* is the name of the Oracle server.
  - *MyHostName* is the name of the TCP/IP host on which the Oracle database resides.
  - *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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
  - *MyPassword* is the password associated with the user name.
  - *MyDataSourceName* is the ODBC data source name for the database.

For example:

```
ORACLE_XA+SQLNET=diaz+HostName=diaz.hursley.ibm.com+PortNumber=1521+Sid=diaz+ACC=
P/wbrkuid/wbrkpw+sestm=100+threads=T
RUE+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 when the queue manager starts.

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.

Now, you can deploy globally coordinated message flows to the broker.

## Configuring global coordination with Oracle using a 64-bit queue manager

To globally coordinate message flow transactions with updates in Oracle databases, configure the broker environment. This topic describes how to configure the broker environment when the broker uses a 64-bit queue manager.

### Before you start:

- Ensure that the databases are configured for global coordination of transactions.

All WebSphere MQ Version 6 queue managers on 64-bit platforms are 64-bit. 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 be 64-bit too.

To configure your broker environment for global coordination using a 64-bit queue manager as the transaction manager with the DataDirect drivers:

1. On Linux and UNIX, run the `mqsimanagexalinks` command.
2. On Linux (x86-64 platform), Solaris (SPARC platform), HP-UX (PA-RISC platform), 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 as well as for the user databases.

**On Linux (except Linux (x86 platform)) 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=OracleXA
SwitchFile=SwitchFileName
XAOpenString=ORACLE_XA+SQLNET=MyServerName+HostName=MyHostName+PortNumber=
MyPortNumber+Sid=MySID+ACC=P/MyUserId/MyPassword+sestm=
100+threads=TRUE+DataSource=MyDataSourceName+DB=MyDataSourceName+K=2+
XACloseString=
ThreadOfControl=THREAD
```

c. The switch file is supplied by WebSphere Message Broker and varies by platform. The following table shows the name of the switch file for each platform:

| Platform                   | Switch file path |
|----------------------------|------------------|
| AIX                        | UKor8dtc22.so    |
| HP-UX (Integrity platform) | UKoradtc22.so    |
| HP-UX (PA-RISC platform)   | UKor8dtc20.sl    |
| Linux (POWER platform)     | UKoradtc22.so    |
| Linux (zSeries platform)   | UKoradtc22.so    |
| Solaris (SPARC platform)   | UKor8dtc22.so    |
| Solaris (x86-64 platform)  | UKoradtc22.so    |

d. On the **XAOpenString** line, replace the following values with values that are appropriate for your configuration:

- *MyServerName* is the name of the Oracle server.
- *MyHostName* is the name of the TCP/IP host on which the Oracle database resides.
- *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 is used by the broker to connect to the database. The user name was defined using the `-u` flag on the `mqsicreatebroker` command when creating the broker. If the `-u` parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the `-i` flag on the `mqsicreatebroker` command.
- *MyPassword* is the password associated with the user name.
- *MyDataSourceName* is the ODBC data source name for the database.

e. Accept the default values for all the other lines in the stanza. For example, on AIX:

```
XAResourceManager:
Name=OracleXA
SwitchFile=UKor8dtc22.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
```

#### On Windows:

- a. From the Start menu, open the WebSphere MQ Explorer graphical administration tool.
- b. Open the queue manager's Properties dialog, then open the **XA resource managers** page.
- c. In the **SwitchFile** field, enter the name of the switch file, as shown in the following example:

```
UKor8dtc22.d11
```

- d. In the **XAOpenString** field, paste the following string:

```
ORACLE_XA+SQLNET=MyServerName+HostName=MyHostName+PortNumber=MyPortNumber+Sid=MySID+ACC=
P/MyUserId/MyPassword+sestm=100+threads=TRUE+DataSource=MyDataSourceName+DB=
MyDataSourceName+K=2+
```

- e. In the **XAOpenString** field, replace the values as appropriate for your configuration:

- *MyServerName* is the name of the Oracle server.
- *MyHostName* is the name of the TCP/IP host on which the Oracle database resides.
- *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 is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
- *MyPassword* is the password associated with the user name.
- *MyDataSourceName* is the ODBC data source name for the database.

For example:

```
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+
```

- 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 when the queue manager starts.

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.

Now, you can deploy globally coordinated message flows to the broker.

## Configuring global coordination with Sybase

To globally coordinate message flow transactions with updates in Sybase databases, configure the broker environment. This topic describes how to configure the broker environment.

### 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 are 32-bit. 32-bit queue managers can coordinate transactions only in 32-bit mode and can coordinate only message flows that are deployed to 32-bit execution groups.

All WebSphere MQ Version 6 queue managers on 64-bit platforms are 64-bit. 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 be 64-bit too.

To configure your broker environment for global coordination using a WebSphere MQ queue manager as the transaction manager with the DataDirect drivers:

1. 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 as well as for the user databases.

### 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 -MyServerName,MyPortNumber -Uuid -Ppwd -K2
 XACloseString=
 ThreadOfControl=THREAD
```

- c. The switch file is supplied by WebSphere Message Broker and varies by platform. The following table shows the name of the switch file for each platform:

Table 2.

| Platform                                      | Switch file path |
|-----------------------------------------------|------------------|
| AIX                                           | UKasedtc22.so    |
| HP-UX (Integrity platform)                    | UKasetdc22.so    |
| HP-UX (PA-RISC platform)                      | UKasedtc20.sl    |
| Linux (all platforms except zSeries platform) | UKasedtc22.so    |

Table 2. (continued)

| Platform                             | Switch file path |
|--------------------------------------|------------------|
| Solaris (SPARC and x86-64 platforms) | UKasedtc22.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 on which the Sybase ASE server resides.
  - *MyPortNumber* is the TCP/IP port on which the Sybase ASE server is listening.
  - *uid* must be the user name that is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
  - *pwd* is the password 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=UKasedtc22.so
 XAOpenString=-NSYBASEDB -Adiaz,1521 -Uwbrkuid -Pwbrkpw -K2
 XACloseString=
 ThreadOfControl=THREAD
```

**On Windows:**

- a. From the Start menu, open the graphical administration tool.
- b. Open the queue manager's Properties dialog, then open the **XA resource managers** page.
- 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\UKase22.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 as appropriate for your configuration:
  - *install\_dir* is the location in which the broker is installed.
  - *MyServerName* is the name of the TCP/IP host on which the Sybase ASE server resides.
  - *MyPortNumber* is the TCP/IP port on which the Sybase ASE server is listening.
  - *uid* must be the user name that is used by the broker to connect to the database. The user name was defined using the **-u** flag on the `mqsicreatebroker` command when creating the broker. If the **-u** parameter was not specified, the data source user name is the same as the broker's service user name, which was defined using the **-i** flag on the `mqsicreatebroker` command.
  - *pwd* is the password 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 when the queue manager starts.

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.

Now, you can deploy globally coordinated message flows to the broker.

---

## Configuring the workbench

The following topics show you how to configure aspects of the workbench.

- “Changing workbench preferences”
- “Changing Broker Administration preferences” on page 271
- “Configuring CVS to run with the Message Broker Toolkit” on page 271

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.

Here 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 debug preferences
- Changing trace settings
- Enabling PDE runtime capabilities

## Changing Broker Administration preferences

You can change the following Broker Administration preferences:

- Show empty projects in the Navigator views.
- Show ACL restricted objects in the Navigator views.
- Warn before deleting alerts.
- Prompt to perform a topology deploy after changes. You can change this to never deploy, always a delta deploy, or always a complete deploy.
- Prompt to perform a topics deploy after changes. You can change this to never deploy, always a delta deploy, or always a complete deploy.

To change Broker Administration preferences:

1. Switch to the **Broker Administration perspective**.
2. Click **Window** → **Preferences**.
3. Expand the **Broker Administration** category in the left pane.
4. Make your selections.
5. Click **OK**.

## 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.

## 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 do this depends on the operating system:

- “Changing your locale on UNIX and Linux systems”
- “Changing your locale on Windows” on page 275
- “Changing your locale on z/OS” on page 275

WebSphere Message Broker uses code page converters to support character sets from different environments. “Code page converters” on page 276 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. This is used when either LC\_ALL is not set, or an applicable value for LC\_\* is not set

## NLSPATH

Delimited list of paths to search for message catalogues

## TZ

Time zone

LC\_MESSAGES and NLSPATH are the most important variables to the broker. These variables define the language and location of response messages that the broker uses. The broker profile file, mqsiprofile, sets NLSPATH. Either you, or your system must set LC\_MESSAGES. The value set in LC\_MESSAGES must be a value that the broker recognizes. LC\_CTYPE is also important to the broker because it defines the character conversion that the broker performs when interacting with the local environment.

If you use common desktop environment (CDE), use this to set the locale instead of setting LANG and LC\_ALL directly. The NLSPATH variable respects either method. Before setting the code page, check that it is one of the Supported code pages.

For example, to set WebSphere 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 276.

## 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 168 and “Installation information - Configuration Manager” on page 181 for further information.

The locale is set in the appropriate component profile (BIPBPROF for the broker, BIPCPRF 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

**Before you start:**

- Read “Code page converters,” which provides information about what a code page converter is, and about the code pages that WebSphere Message Broker supports.

If you need to support a code page that is not in the default set of code pages that WebSphere Message Broker supports, 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 as long as 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:
  - 1 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 `mqsicchangebroker` 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 `mqsicchangebroker` 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 276, 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 the previous level of the broker and WebSphere Message Broker Version 6.1 are significant, so the set of converters from the previous level has been included with WebSphere Message Broker Version 6.1.

To use one of these converters, take the following steps:

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 home directory of your WebSphere Message Broker installation.

2. Copy the `.cnv` file for the required code page to a directory that is accessible by the broker; for example, on UNIX systems `/var/mqsi/converters`, and on Windows systems `C:\Documents and Settings\All Users\Application Data\ibm\mqsi\converters`. To make sure that the copied file does not conflict with an existing converter, give the file a unique name. To ensure that the file does not cause a conflict of filenames, do not use a number that is already used in one of the supported code pages. If the converter is to be 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:
  - 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.  
`mqsichangebroker -c directory`
  - To affect all the products and the broker command-line tools using international components for Unicode (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.

Copy the entire set of converters (\*.cnv) and aliases (\*.icu) to reproduce the behavior of the previous level of the product.

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` command with the `-c` parameter to set the converter path. Use the `ICU_DATA` environment variable instead.



---

## Part 3. Administering the broker domain

|                                                                                   |            |
|-----------------------------------------------------------------------------------|------------|
| <b>Administering the broker domain . . . . .</b>                                  | <b>283</b> |
| Connecting to and disconnecting from the broker domain . . . . .                  | 283        |
| Connecting to and disconnecting from the broker domain on z/OS . . . . .          | 284        |
| Starting and stopping message flows . . . . .                                     | 286        |
| Starting and stopping a broker . . . . .                                          | 287        |
| Starting and stopping a broker on Linux and UNIX systems . . . . .                | 287        |
| Starting and stopping a broker on Windows . . . . .                               | 287        |
| Starting and stopping a broker on z/OS . . . . .                                  | 288        |
| Starting and stopping a Configuration Manager . . . . .                           | 288        |
| Starting and stopping a Configuration Manager on Linux and UNIX systems . . . . . | 288        |
| Starting and stopping a Configuration Manager on Windows. . . . .                 | 289        |
| Starting and stopping a Configuration Manager on z/OS . . . . .                   | 289        |
| Starting and stopping the User Name Server . . . . .                              | 290        |
| Starting and stopping the User Name Server on Linux and UNIX systems . . . . .    | 290        |
| Starting and stopping the User Name Server on Windows. . . . .                    | 290        |
| Starting and stopping the User Name Server on z/OS . . . . .                      | 291        |
| Starting a WebSphere MQ queue manager as a Windows service . . . . .              | 291        |
| Stopping a WebSphere MQ queue manager when you stop a component . . . . .         | 292        |
| Viewing broker domain log information . . . . .                                   | 293        |
| Refreshing broker domain log information. . . . .                                 | 293        |
| Filtering broker domain log information . . . . .                                 | 294        |
| Saving broker domain log information . . . . .                                    | 295        |
| Clearing broker domain log information . . . . .                                  | 296        |
| Changing the location of the work path for WebSphere Message Broker . . . . .     | 296        |
| Changing the location of the work path on Windows. . . . .                        | 296        |
| Changing the location of the work path on UNIX and Linux . . . . .                | 297        |
| Changing Event Log editor preferences. . . . .                                    | 298        |
| Backing up resources . . . . .                                                    | 298        |
| Backing up the broker domain on distributed systems . . . . .                     | 299        |
| Backing up the broker domain on z/OS . . . . .                                    | 300        |
| Backing up the Message Broker Toolkit workspace . . . . .                         | 301        |



---

## Administering the broker domain

Administering the broker domain includes the tasks that you operate frequently to activate and run your broker domain. These are listed below:

- “Connecting to and disconnecting from the broker domain”
- “Starting and stopping message flows” on page 286
- “Starting and stopping a broker” on page 287
- “Starting and stopping a Configuration Manager” on page 288
- “Starting and stopping the User Name Server” on page 290
- “Starting a WebSphere MQ queue manager as a Windows service” on page 291
- “Stopping a WebSphere MQ queue manager when you stop a component” on page 292
- “Viewing broker domain log information” on page 293
- “Refreshing broker domain log information” on page 293
- “Filtering broker domain log information” on page 294
- “Saving broker domain log information” on page 295
- “Clearing broker domain log information” on page 296
- “Changing Event Log editor preferences” on page 298
- “Backing up resources” on page 298

These tasks can be performed using one, or more, of the administrative techniques available with WebSphere Message Broker:

- The Message Brokers Toolkit
- The WebSphere Message Broker commands
- The Configuration Manager Proxy Java API

For each task, the administrative techniques that can be used are identified.

---

## Connecting to and disconnecting from the broker domain

### Before you start:

You must complete the following task:

- “Creating a domain connection” on page 223

You can connect to, and disconnect from, the broker domain by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy, see [Connecting to the Configuration Manager using the Configuration Manager Proxy](#).

Use a domain connection to connect to the broker domain in the workbench.

The following steps show you how to connect to the broker domain and how to disconnect from the broker domain.

1. To connect to the broker domain:
  - a. Switch to the Broker Administration perspective.

- b. In the Domains view, right-click the broker domain to which you want to connect, and click **Connect**. This starts the domain connection to the Configuration Manager.

When connected, the workbench status line is changed (for example *WBRK\_QM@localhost:1414* is connected). The Broker Topology and Topics are populated, and the broker domain and broker topology icons change to reflect the connected state.

On successful connection, the Configuration Manager name is shown in the Domains view in the form ConfigurationManagerName on QMgrName@Hostname:PortNumber.

**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

This topic describes 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 181.
- 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 198.

Ensure that your Message Brokers Toolkit machine 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 above by entering:

```
/F <configmgrname>,LA
```

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. 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 63.
  - 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.

**Note:** The **JAR File Location** is required if a Security Exit **Class** is entered.

- f. Optional: The **Cipher Suite**, **Distinguished Names**, **CRL Name List**, **Key Store**, and **Trust Store** parameters are required when enabling SSL. See “Implementing SSL authentication” on page 52. 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 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.
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:
    - a. The name of your **Server Project**. The Server Project is the container for your domain connection. If you have not already created a server project, you can specify the name of a new project here. The server project is created with the domain connection.
    - b. The **Connection name**. 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 Navigator view, under Domain Connections. The server project holds the .configmgr domain connection file.

The view of the broker domain is displayed in the Domains view.

---

## Starting and stopping message flows

You can start and stop a message flow by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy, see Navigating broker domains using the Configuration Manager Proxy.

From the workbench you can start and stop:

- All message flows in all execution groups, assigned to a specific broker.
  - All message flows in a specific execution group.
  - A single message flow.
1. To start a message flow:
    - a. Switch to the Broker Administration perspective.
    - b. In the Domains view, expand your broker domain to locate your message flow:
      - To start all message flows in all execution groups for a broker, right-click the broker and click **Start Message Flows**.
      - To start all message flows in a specific execution group, right click the execution group and click **Start Message Flows**.
      - To start a single message flow, right-click the message flow and click **Start**.

The Configuration Manager sends a message to the broker to start the specified message flows.

A BIP0892I information message is displayed to show that the Configuration Manager has received the request. Verify the results of the deployment by opening the Event Log.

There might be a short delay for the Configuration Manager to respond.

The alert Message Flow is not running is removed from the Alert Viewer.

2. To stop a message flow:
  - a. Switch to the Broker Administration perspective.
  - b. In the Domains view, expand your broker domain to locate your message flow:
    - To stop all message flows in all execution groups for a broker, right-click the broker and click **Stop Message Flows**.
    - To stop all message flows in a specific execution group, right click the execution group and click **Stop Message Flows**.
    - To stop a single message flow, right-click the message flow, and click **Stop**.
  - c. Open the Event Log for the broker domain. A BIP0892I information message is displayed to show that the Configuration Manager has received the request.

There might be a short delay for the Configuration Manager to respond.

The alert Message Flow is not running is added to the Alert Viewer.

---

## Starting and stopping a broker

### Before you start:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591.
- 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 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 288

## Starting and stopping a broker on Linux and UNIX systems

The following steps show you how to start and stop a broker.

1. Run '`<install_dir>/bin/mqsiprofile`' to source the `mqsiprofile` script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere 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 syslog to ensure that the broker has initialized successfully.
3. To stop a broker, enter the following command on the command line:  
`mqsistop WBRK_BROKER`  
Substitute your own broker name for `WBRK_BROKER`.  
You can also request that the broker’s queue manager is stopped by this command. Refer to “Stopping a WebSphere MQ queue manager when you stop a component” on page 292.

## Starting and stopping a broker on Windows

The following steps show you how to start and stop a broker.

1. Run the `<install_dir>/bin/mqsiprofile` command to set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere 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`.  
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 291.

The broker and its associated queue manager are started. The command initiates the startup of the broker's Windows service. Check the Application Log of the Windows Event Viewer to ensure that the broker has initialized successfully.

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 292.

## Starting and stopping a broker on z/OS

The following steps show you how to start and stop a broker.

1. From SDSF, start the component by using the command `/S <broker name>`. If MQP1BRK is the name of the broker, this command produces, for example, the following output:

```
+BIP9141I MQP1BRK 0 The component was started
```

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.

2. To stop a broker use the command:

```
/P <broker name>
```

---

## Starting and stopping a Configuration Manager

### Before you start:

- Ensure that your user ID has the correct authorizations to perform the task; refer to "Security requirements for administrative tasks" on page 591.
- 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 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" on page 289
- "Starting and stopping a Configuration Manager on z/OS" on page 289

## Starting and stopping a Configuration Manager on Linux and UNIX systems

Use the **mqsistart** and **mqsistop** commands from the command line.

The following steps show you how to start and stop a Configuration Manager.

1. Run `'.<install_dir>/bin/mqsiprofile'` to source the `mqsiprofile` script and set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Message Broker commands.

2. To start a Configuration Manager, enter the following command:  

```
mqsistart CMGR01
```

 Substitute your own Configuration Manager name for CMGR01. The Configuration Manager and its associated queue manager are started.
3. To stop a Configuration Manager, enter the following command on the command line:  

```
mqsistop CMGR01
```

 Substitute your own Configuration Manager name for CMGR01.  
 You can also request that the Configuration Manager's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 292.

## Starting and stopping a Configuration Manager on Windows

Use the **mqsistart** and **mqsistop** commands from the command line.

The following steps show you how to start and stop a Configuration Manager.

1. Run the `<install_dir>/bin/mqsiprofile` command to set up the environment for a single targeted runtime. You must do this before you can run any of the WebSphere Message Broker commands.
2. To start a Configuration Manager, enter the following command:  

```
mqsistart CMGR01
```

 Substitute your own Configuration Manager name for CMGR01. If you do not specify a name, the default of `configmgr` is used. The Configuration Manager and its associated queue manager are started. The command initiates the startup of the Configuration Manager's Windows service. Check the Application Log of the Windows Event Viewer to ensure that the Configuration Manager has initialized successfully.  
 You can also request that the Configuration Manager's queue manager is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service" on page 291.
3. To stop a Configuration Manager, enter the following command on the command line:  

```
mqsistop CMGR01
```

 Substitute your own Configuration Manager name for CMGR01. If you do not specify a name, the default of `configmgr` is used.  
 You can also request that the Configuration Manager's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 292.

## Starting and stopping a Configuration Manager on z/OS

The following steps show you how to start and stop a Configuration Manager.

1. From SDSF, start the component by using the command `/S <Configuration Manager name>`. If MQP1CMGR is the name of the Configuration Manager, this command produces, for example, the following output:

```
+BIP9141I MQP1CMGR 0 The component was started
```

2. To stop a Configuration Manager use the command:  
`/P <Configuration Manager name>`

---

## Starting and stopping the User Name Server

### Before you start:

- Ensure that your user ID has the correct authorizations to perform the task. Refer to “Security requirements for administrative tasks” on page 591.
- 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”
- “Starting and stopping the User Name Server on z/OS” on page 291

## 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.

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 292.

## 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.

You can also request that the User Name Server's queue manager is started as a Windows service. Refer to "Starting a WebSphere MQ queue manager as a Windows service."

3. To stop a User Name Server enter the following command on the command line:

```
mqsistop usernameserver
```

You can also request that the User Name Server's queue manager is stopped by this command. Refer to "Stopping a WebSphere MQ queue manager when you stop a component" on page 292.

## Starting and stopping the User Name Server on z/OS

The following steps show you how to start and stop a User Name Server.

1. From SDSF, start the component by using the command /S <User Name Server name>. If MQP1UNS is the name of the User Name Server, this command produces, for example, the following output:

```
+BIP9141I MQP1UNS 0 The component was started
```

2. To stop a User Name Server, use the command:  
/P <User Name Server name>

---

## Starting a WebSphere MQ queue manager as a Windows service

### Before you start:

You must complete the following task:

- Stop the queue manager for the WebSphere 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 525 starts the associated queue manager if it is not already running. When you start any of these components on Windows, it 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 the WebSphere 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 "mqsisstart command" on page 525.

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 component using the "mqsisstart command" on page 525.

---

## Stopping a WebSphere MQ queue manager when you stop a component

### Before you start:

You must complete the following task:

- If you are using a single queue manager to support more than one WebSphere 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), you are recommended to specify the `-q` flag only on the final stop command, having stopped the other components first. The `-q` flag initiates a queue manager termination regardless of any other component currently using that queue manager.

If you are preparing to stop a broker, Configuration Manager or User Name Server, you can stop the component's WebSphere MQ queue manager at the same time.

You can specify a `-q` parameter on the **mqsisstop** command to initiate a controlled shutdown of the queue manager for a WebSphere Message Broker component.

To stop a WebSphere MQ queue manager enter the following command on the command line:

```
mqsisstop WBRK_BROKER -q
```

where:

WBRK\_BROKER is the name of the component.

`-q` stops the WebSphere MQ queue manager associated with the component.

The command cannot complete until shutdown of the queue manager has completed.

---

## Viewing broker domain log information

You can view broker domain log information by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications using the CMP* and Class `com.ibm.broker.config.proxy.LogProxy`.

Broker domain log information is written to the Event Log editor in the workbench.

The Event Log contains information about events that occur within your broker domain. These events can be information, errors, or warnings and relate to your own actions. To view events for a particular broker, look for the name of the broker in the Source column.

Each event contains the following information:

- **Message:** The event number.
- **Source:** Where the event has come from (within the broker domain).
- **TimeStamp:** The date and time that the event occurred. TimeStamps are taken from the computer that is hosting the Configuration Manager.
- **Details:** What has caused the event and what action is needed to rectify it.

The following steps show how to view broker domain log information.

1. Switch to the **Broker Administration** perspective.
2. In the **Domains** view, expand the broker domain with which you want to work, to reveal the Event Log.
3. Double-click the Event Log. All broker domain log information specific to the broker domain with which you are working is displayed in the Event Log editor.

If the broker domain is not connected, you are prompted to connect to the broker domain before the Event Log is opened.

The Event Log editor has two panes called **Logs** and **Details**. The top half of the view lists all the events, in date and time order. The bottom half of the view shows the details of a specific selected event. You can maximize and minimize each pane, and toggle between them.

4. Click the event that you want to view in more detail from the top half of the Event Log view. The details of this event can then be viewed in the bottom half of the view.

When you filter information, as described in “*Filtering broker domain log information*” on page 294, a note appears next to the view label to indicate that a filter has been applied.

---

## Refreshing broker domain log information

Messages about events that occur in the broker domain are created in the Event Log editor.

To refresh the Event Log editor with new messages:

1. Switch to the **Broker Administration** perspective.

2. In the **Domains** view, expand the appropriate domain to display its components.
3. Double-click on **Event Log** to launch the Event Log editor in the editor area.
4. Right-click in the **Logs** pane and click **Revert** from the pop-up menu. **Revert** does not remove or overwrite existing messages in the Event Log editor or the configuration repository.

---

## Filtering broker domain log information

Broker domain log information is written to the Event Log editor in the workbench. The Event Log contains information about events that occur in a broker domain. 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 message brokers)
- Event timestamp (the date and time when the event occurred)

You can filter on the type, the source and the timestamp of the message, to restrict the number of log entries that are displayed in the Event Log editor. You can also filter events, event by event, to view a particular set of events in the Event log editor. The filter settings 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 ...**. This action opens the Filter Log dialog.

The Filter Log dialog has the following controls:

- A check box for each of the three message types (information, error or warning)
- A check box for each of the possible message sources for all known entries so far
- Under the source check boxes, a **Select All** button to select all sources and a **Deselect All** button to deselect all sources
- A non-editable drop-down list containing each possible timestamp of all known entries so far
- A table of check box items for each log entry, with a vertical scroll bar to access additional entries
- Under the table, a **Select All** button to select all events, and a **Deselect All** button to deselect all events from this table
- A **Restore Defaults** button

In the default view of the Log Filter, all message type check boxes are selected, indicating that messages of all three types are displayed. All message source check boxes are also selected by default, indicating that messages of all sources are displayed. The timestamp is the one for the oldest known event so far. All events are selected in the table.

To restore the default settings, click **Restore Defaults** (the defaults are: all types selected, all sources selected, timestamp set to oldest event one, all events selected).

To discard all changes, click **Cancel**.

To save the current filter settings, click **OK**. This applies the settings to the opened editor (event filtering is based on the new settings, and the editor is refreshed).

Using the Event Log Filter, you can filter broker domain log information in the following ways:

- Filter by message type, so that the log only displays messages of a particular type:
  - a. Clear the check boxes for all three event types.
  - b. Check the check box of the event type that you want to view (you can view more than one type by selecting the appropriate check boxes). All events with the selected types are selected in the filter dialog table of entries.
  - c. Click **OK** to activate the filter. All events with the selected types are displayed in the editor.
- Filter by message source, so that the log only displays messages from one source:
  - a. Deselect all sources, click the **Deselect All** button under the source check boxes, or clear the check boxes for all sources.
  - b. Check the check box of the event source that you want to view (you can view more than one source by selecting the appropriate check boxes). You can also view all sources click by clicking the **Select All** button under the source check boxes. All events with the selected sources are selected in the filter dialog table of entries.
  - c. Click **OK** to activate the filter. All events with the selected sources are displayed in the editor.
- Filter by timestamp, so that the log hides messages generated before a specified timestamp:
  - a. Open the **Hide Events generated before** drop-down list by clicking the down arrow. Select the timestamp you want to filter on. All events generated before this timestamp are deselected in the filter dialog table of entries.
  - b. Click **OK** to activate the filter. All events generated before the reference timestamp are hidden in the editor. All other events are displayed.

You can combine filtering options (type, source, time stamp). The combinations you select are identified in the table (entries are automatically ticked or non-ticked to indicate the choices you make). The editor is also updated based on the events tables which combines all filtering options. Ticked entries are displayed, non-ticked entries are hidden. For incoming new events, the filtering options based on type, source, timestamp are applied.

---

## Saving broker domain log information

Broker domain log information is written to the Event Log Editor in the workbench.

Event Log information is deleted automatically from the Configuration Manager after 72 hours.

To save broker domain log information:

1. Switch to the Broker Administration perspective.
2. Open the Event Log for the appropriate broker domain.

3. Right-click in the Event Log view and click **Save Log As...**
4. You are prompted to save the log information to an appropriate directory.  
By default the log is saved as log.txt. However, you can change the name of this text file.  
The file can also be saved using XML format, with .xml file extension.

Each message recorded in the event log is written to the text file with the same information that is detailed in the event log itself.

To view the log, open the log.txt file in an appropriate text editor.

---

## Clearing broker domain log information

You can clear broker domain log information by either using the Message Brokers Toolkit or by using the Configuration Manager Proxy Java API. This topic describes how to use the Message Brokers Toolkit. For information about how to use the Configuration Manager Proxy (CMP), see *Developing applications using the CMP and Class com.ibm.broker.config.proxy.LogProxy*.

To clear all the broker domain log information from the Event Log:

1. Switch to the Broker Administration perspective.
2. Open the Event Log for the appropriate broker domain.
3. In the Event Log editor menu, click **Event Log Editor** → **Clear Log ...**  
If you have set user preference to *warn before deleting events*, a prompt asks you to confirm deletion. Click **OK**.  
If you have not set user preferences to *warn before deleting events*, the event log is cleared automatically.  
When you clear the event log, all recorded events that are in view are deleted from the repository.

---

## Changing the location of the work path for WebSphere Message Broker

The work path directory is the location in which WebSphere Message Broker 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

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 by running the following command:

```
mqservice component_name
```

where *component\_name* is the name of one of the components (for example, a broker) in the WebSphere Message Broker installation.

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-hand pane of the Computer Management dialog, click **Disk Management**. The new partition that you added and any existing partitions are listed in the right-hand 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 UNIX and Linux

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 (x86 platform):

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 by running the following command:

```
mqservice component_name
```

where *component\_name* is the name of one of the components (for example, a broker) in the WebSphere Message Broker installation.

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 the following preferences for the Event Log editor:

- Choose not to display a warning before deleting log events. The default is to display a warning.
- Change the color for each type of event (Warning, Information, and Error). You can choose from a palette of basic colors or define custom colors. The default color for all events is black.
- Define the style and size of the font used for event details. The default is Tahoma, regular, 8 point.

To change preferences:

1. Switch to the **Broker Administration perspective**.
2. Click **Window>Preferences**.
3. Expand the **Broker Administration** category in the left pane.
4. Click **Event Log Editor** within the expanded **Broker Administration** category to open the Event Log editor preferences page.
5. Make your selections.
6. Click **OK**.

---

## Backing up resources

Brokers rely on a database manager to maintain and control their configuration data. Brokers, the Configuration Manager, and the User Name Server rely on WebSphere MQ to transport and guarantee messages between components. You must establish a backup process that includes these sources of information to preserve the integrity and consistency of your broker domain.

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 you are using for details of the relevant database backup procedures.

Consult your database administrator and agree upon the:

- Frequency of backups
- Quiesce backup points to take

Depending upon your workflow, this can be hourly or daily or weekly.

You should aim to always be 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 installing 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” on page 299

- “Backing up the broker domain on z/OS” on page 300
- “Backing up the Message Broker Toolkit workspace” on page 301

## Backing up the broker domain on distributed systems

These steps tell you how to back up a broker domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. The backup and restoration of a broker needs to extend to every broker that is deployed to by the Configuration Manager. For more information about carrying out these steps, see the links at the end of this topic.

To back up the components:

1. Stop each broker.
2. Stop the Configuration Manager.
3. Back up the Configuration Manager data repository using the **mqsibackupconfigmgr** command.
4. Back up each broker database.

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:

- On Windows the directory is:  
C:\Documents and Settings\All Users\Application Data\IBM\MQSI
- On the other distributed platforms the directory is:  
/var/mqsi

and any broker-specific work paths. These paths are the ones specified by the **-w** flag on the **mqsicreatebroker** command.

To restore the components:

1. **Stop and remove the existing components in the Configuration Manager domain.**
  - a. Disconnect from the domain on the Message Broker Toolkit.
  - b. Stop each broker.
  - c. Stop the Configuration Manager.
  - d. Delete each broker using the **mqsdeletebroker** 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 **mqsdeleteconfigmgr** 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 on the Message Broker Toolkit. This re-imports the broker topology, excluding execution groups and flows, from the Configuration Manager.
- g. Deploy the topology configuration on the Message Broker Toolkit. This 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

The following instructions describe how to back up a broker domain so that it can be restored for migration purposes or in the event of an unrecoverable failure. You should back up, and plan for restoration, on every system on which there is 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. Note 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:

- On Windows the directory is:  
`C:\Documents and Settings\All Users\Application Data\IBM\MQSI`
- On the other distributed platforms the directory is:  
`/var/mqsi`

This needs to be kept 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 using the JCL job BIPBUDB.

### Restoring 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.
    - 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 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 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.

- **Windows** On Windows, the default workspace directory is created at `C:\Documents and Settings\user\IBM\wmbt61\workspace`, where *user* is the user name with which you are logged on.
- **Linux** On Linux, the default workspace directory is created at `/home/user/IBM/wmqi61/workspace`, where *user* is the user name with which you are logged on.

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.

The workspace directory contains a directory called `.metadata`, which contains your personal settings and preferences for the Message Broker Toolkit. If the `.metadata` directory gets corrupted, you lose these settings and the Message Broker Toolkit 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, just take a copy of the directory.

In the workspace, a directory exists for each project (message flow project, message set project, or server project) that you have created in the Message Broker Toolkit. These directories contain your data, which must be backed up.

Use one of the following methods to back up the data in your workspace:

- Export the projects from within the Message Broker Toolkit. 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 the **Include source files** check box 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.

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

|                                                                            |     |
|----------------------------------------------------------------------------|-----|
| <b>Databases</b> . . . . .                                                 | 305 |
| odbc32.ini sample file . . . . .                                           | 305 |
| odbc64.ini sample file. . . . .                                            | 311 |
| <br>                                                                       |     |
| <b>Operations</b> . . . . .                                                | 323 |
| Broker properties . . . . .                                                | 323 |
| Commands . . . . .                                                         | 324 |
| Summary of commands on Windows, z/OS,<br>Linux, and UNIX systems . . . . . | 326 |
| Syntax diagrams: available types . . . . .                                 | 329 |
| Characters allowed in commands. . . . .                                    | 333 |
| Rules for using commands . . . . .                                         | 334 |
| Responses to commands. . . . .                                             | 335 |
| Runtime and workbench commands (common)                                    | 335 |
| WebSphere Message Broker workbench<br>commands . . . . .                   | 339 |
| Runtime commands . . . . .                                                 | 354 |
| z/OS specific information . . . . .                                        | 537 |
| Administration in z/OS . . . . .                                           | 537 |
| z/OS customization . . . . .                                               | 539 |
| z/OS JCL variables . . . . .                                               | 551 |
| z/OS sample files supplied. . . . .                                        | 555 |
| <br>                                                                       |     |
| <b>Security requirements for administrative tasks</b>                      | 591 |
| ACL permissions . . . . .                                                  | 591 |
| Security requirements for Linux and UNIX<br>platforms. . . . .             | 592 |
| Security requirements for Windows platforms . . . . .                      | 593 |
| Security requirements for z/OS . . . . .                                   | 596 |



---

## Databases

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 311

---

### odbc32.ini sample file

#### 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: 64-bit considerations” on page 122.

#### 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 5.2 Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 SQL Server Wire Protocol
IFORMIXDB=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/V5.2/lib/UKor822.so
Description=DataDirect 5.2 Oracle Wire Protocol
ApplicationsUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle server name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
ApplicationsUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1

Sybase Stanza for a UTF8 datasource
[SYBASEBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 Sybase Wire Protocol
Database=<Your Database Name>
ServerName=<Your Sybase Server Name>
ApplicationsUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
SelectMethod=0
NetworkAddress=<Your Sybase Server Name>,<Your Sybase Port Number>
SelectUserName=1
ColumnSizeAsCharacter=1
Charset=UTF8

UNIX to SQL Stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 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]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## HP-UX (PA-RISC platform)

```
#####
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 5.0 Oracle Wire Protocol
SYBASEDB=DataDirect 5.0 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.0 Sybase UTF8 Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver
#####
Individual data source stanzas
#####
DB2 version 8 stanza
[DB2V8DB]
Driver=<Your DB2 v8 install directory>/lib/libdb2.s1
Description=DB2V8DB DB2 ODBC Database
Database=DB2V8DB

DB2 version 9 stanza
[DB2V9DB]
Driver=<Your DB2 v9 install directory>/lib32/libdb2.s1
Description=DB2V9DB DB2 ODBC Database
Database=DB2V9DB

Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC32/V5.0/lib/UKor820.s1
Description=DataDirect 5.0 Oracle
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Server Name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.0/lib/UKase20.s1
Description=DataDirect 5.0 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
```

```

Sybase Stanza for a UTF8 datasource
[SYBASEDUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.0/lib/UKase20.s1
Description=DataDirect 5.0 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
Charset=UTF8

[INFORMIXDB]
Driver=<Your Informix Client Directory>/lib/cli/iclit09b.s1
Description=IBM Informix ODBC Driver
ServerName=<YourServerName>
Database=<Your Database Name>

#####
Mandatory information stanza
#####

[ODBC]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.0/lib/odbc trac.s1
InstallDir=<Your Broker install directory>/ODBC32/V5.0
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## Linux (x86 platform and x86-64 platform)

```

#####
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 5.2 Oracle
SYBASEDB=DataDirect 5.2 Sybase Wire Protocol
SYBASEDUTF8=DataDirect 5.2 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 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/V5.2/lib/UKor822.so
Description=DataDirect 5.2 Oracle
EnableDescribeParam=1
OptimizePrepare=1
ServerName=Your Oracle Server Name
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

#Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

#Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

Unix to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Server Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1

[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]
```

```

Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## Solaris (SPARC platform)

```

#####
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 5.2 Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 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/V5.2/lib/UKor822.so
Description=DataDirect 5.2 Oracle
EnableDescribeParam=1
OptimizePrepare=1
ServerName=<Your Oracle Server Name>
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

Sybase Stanza for a UTF8 datasource
[SYBASEDUTF8]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

UNIX to SQLServer Stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC32/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 SQL Server Wire Protocol
Address=<Your SQLServer Server Name>,<Your SQLServer Port Number>
Database=<Your Database Name>
AnsiNPW=Yes
QuoteId=No
ColumnSizeAsCharacter=1

[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]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace32.out
TraceDll=<Your Broker install directory>/ODBC32/V5.2/lib/odbc trac.so
InstallDir=<Your Broker install directory>/ODBC32/V5.2
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: 32-bit considerations” on page 118.

### 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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

Unix to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 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]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## HP-UX (PA-RISC platform)

```

#####
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 Wire Protocol
INFORMIXDB=IBM Informix ODBC Driver

#####
Individual data source stanzas
#####

```

```

DB2 stanza
[DB2DB]
DRIVER=libdb2Wrapper64.s1
Description=DB2DB DB2 ODBC Database
Database=DB2DB

Oracle stanza
[ORACLEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKora22.s1
Description=DataDirect 5.2 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

Sybase Stanza
[SYBASEDB]
Driver=<Your install directory>/ODBC64/V5.2/lib/UKase22.s1
Description=DataDirect 5.2 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

Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your install directory>/ODBC64/V5.2/lib/UKase22.s1
Description=DataDirect 5.2 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
Charset=UTF8

Informix Stanza
[INFORMIXDB]
Driver=libinfWrapper64.s1
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

#####
Mandatory information stanza
#####

[ODBC]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.s1
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## HP-UX (Integrity platform) and Solaris (x86-64 platform)

```
#####
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 Wire Protocol
#####
Individual data source stanzas
#####
DB2 stanza
[DB2DB]
Driver=libdb2Wrapper64.so
Description=DB2DB DB2 ODBC Database
Database=DB2DB
[ORACLEDB]
Driver=<Your install directory>/ODBC64/V5.2/lib/UKora22.so
Description=DataDirect 5.2 Oracle Wire Protocol
HostName=<Your Oracle Server Machine Name>
PortNumber=<Port on which Oracle is listening on HostName>
SID=<Your Oracle SID>
CatalogOptions=0
ProcedureRetResults=1
EnableStaticCursorsForLongData=0
ApplicationUsingThreads=1
EnableDescribeParam=1
OptimizePrepare=1
WorkArounds=536870912
ColumnSizeAsCharacter=1
Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

#####
Mandatory information stanza
#####

[ODBC]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4

```

## Linux (POWER platform)

```

#####
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
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]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

## Linux (x86-64 platform)

```

#####
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

[INFORMIXDB]
Driver=libinfWrapper64.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>

```

```

Mandatory information stanza #####
#####
```

```
[ODBC]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8
```

## Linux (zSeries platform)

```

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 5.2 64bit Oracle 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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
WorkArrounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1
```

```

[INFORMIXDB]
Driver=libinfWrapper.so
Description=IBM Informix ODBC Driver
ServerName=<Your Informix Server Name>
Database=<Your Database Name>
#####
Mandatory information stanza
#####

[ODBC]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```

### Solaris (SPARC platform)

```

#####
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 5.2 64bit Oracle Wire Protocol
SYBASEDB=DataDirect 5.2 64bit Sybase Wire Protocol
SYBASEDBUTF8=DataDirect 5.2 64bit Sybase UTF8 Wire Protocol
SQLSERVERDB=DataDirect 5.2 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/V5.2/lib/UKora22.so
Description=DataDirect 5.2 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
ApplicationsUsingThreads=1
EnableDescribeParam=1

```

```

OptimizePrepare=1
WorkArounds=536870912
ProcedureRetResults=1
ColumnSizeAsCharacter=1

Sybase Stanza
[SYBASEDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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

Sybase Stanza for a UTF8 datasource
[SYBASEDBUTF8]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKase22.so
Description=DataDirect 5.2 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
Charset=UTF8

UNIX to SQLServer stanza
[SQLSERVERDB]
Driver=<Your Broker install directory>/ODBC64/V5.2/lib/UKmsss22.so
Description=DataDirect 5.2 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]
Trace=0 # To turn on ODBC trace set = 1
TraceFile=<A Directory with plenty of free space to hold trace output>/odbctrace64.out
TraceDll=<Your Broker install directory>/ODBC64/V5.2/lib/odbctrac.so
InstallDir=<Your Broker install directory>/ODBC64/V5.2
UseCursorLib=0
IANAAppCodePage=4
UNICODE=UTF-8

```



---

## Operations

Follow the links below for more information:

- “Broker properties”
- “Commands” on page 324
- “z/OS specific information” on page 537

---

### Broker properties

The following table is a reference for the broker properties.

| Property                     | Meaning                                                                                                                                                                                                                                                                                                       |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Queue Manager Name           | The name of the WebSphere MQ queue manager being used by the broker. The name must be exactly the same name specified for this broker’s queue manager on the <b>mqsicreatebroker</b> 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:<br><b>P</b> Password in the clear<br><b>M</b> Mutual Challenge and response<br><b>S</b> Asymmetric SSL<br><b>R</b> Symmetric SSL                                                                              |
| SSL Key Ring File Name       | The filename (including path) to the SSL Key Ring file. This is required for authentication when using the Asymmetric and Symmetric (S and R) authentication protocols.<br><br>The file name and path refer to the path accessible from the broker and not necessarily the tooling, if on different machines. |
| SSL Password File Name       | The SSL key ring file is encrypted and requires a passphrase to decode it. This field is used to specify the filename containing the passphrase required.<br><br>The file name and path refer to the path accessible from the broker and not necessarily the tooling, if on different machines.               |
| SSL HTTP Listener            | Use the “mqsicreateusernameeserver command” on page 442 or “mqsichangeusernameeserver command” on page 404 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 381. See “Implementing quality of protection” on page 64 for details on setting these for temporary topics.

---

## Commands

This topic lists the Message Broker Toolkit commands, followed by the other commands grouped by function, provided by WebSphere Message Broker on distributed systems.

The “Summary of commands on Windows, z/OS, Linux, and UNIX systems” on page 326 lists the comparable commands on WebSphere Message Broker for z/OS.

### WebSphere Message Broker Toolkit commands

|                                  |                                                 |
|----------------------------------|-------------------------------------------------|
| <b>mqsipplybaroverride</b>       | “mqsipplybaroverride command” on page 335       |
| <b>mqsicreatebar</b>             | “mqsicreatebar command” on page 339             |
| <b>mqsicreatemsgdefs</b>         | “mqsicreatemsgdefs command” on page 342         |
| <b>mqsicreatemsgdefsfromwsdl</b> | “mqsicreatemsgdefsfromwsdl command” on page 350 |
| <b>mqsimigratemfmaps</b>         | “mqsimigratemfmaps command” on page 352         |
| <b>mqsireadbar</b>               | “mqsireadbar command” on page 338               |

### WebSphere Message Broker commands

#### Broker commands

|                                 |                                                |
|---------------------------------|------------------------------------------------|
| <b>mqsicreatebroker</b>         | “mqsicreatebroker command” on page 416         |
| <b>mqsichangebroker</b>         | “mqsichangebroker command” on page 356         |
| <b>mqsideletebroker</b>         | “mqsideletebroker command” on page 454         |
| <b>mqsicreateexecutiongroup</b> | “mqsicreateexecutiongroup command” on page 440 |
| <b>mqsideleteexecutiongroup</b> | “mqsideleteexecutiongroup command” on page 462 |
| <b>mqsireload</b>               | “mqsireload command” on page 500               |
| <b>mqsireloadsecurity</b>       | “mqsireloadsecurity command” on page 501       |
| <b>mqsicbrreport</b>            | “mqsicbrreport command” on page 356            |
| <b>mqsipplybaroverride</b>      | “mqsipplybaroverride command” on page 335      |
| <b>mqsireadbar</b>              | “mqsireadbar command” on page 338              |

#### Database commands

|                          |                                         |
|--------------------------|-----------------------------------------|
| <b>mqsicreatedb</b>      | “mqsicreatedb command” on page 438      |
| <b>mqsichangedbimgr</b>  | “mqsichangedbimgr command” on page 371  |
| <b>mqsidedetedb</b>      | “mqsidedetedb command” on page 461      |
| <b>mqsisetdbparms</b>    | “mqsisetdbparms command” on page 520    |
| <b>mqs_setupdatabase</b> | “mqs_setupdatabase command” on page 525 |
| <b>mqsimanagexalinks</b> | “mqsimanagexalinks command” on page 488 |

#### Configuration Manager commands

|                             |                                            |
|-----------------------------|--------------------------------------------|
| <b>mqsicreateconfigmgr</b>  | “mqsicreateconfigmgr command” on page 428  |
| <b>mqsichangeconfigmgr</b>  | “mqsichangeconfigmgr command” on page 366  |
| <b>mqsideleteconfigmgr</b>  | “mqsideleteconfigmgr command” on page 456  |
| <b>mqsibackupconfigmgr</b>  | “mqsibackupconfigmgr command” on page 354  |
| <b>mqsirestoreconfigmgr</b> | “mqsirestoreconfigmgr command” on page 519 |

#### User Name Server commands

|                                                          |                                |                                                      |
|----------------------------------------------------------|--------------------------------|------------------------------------------------------|
|                                                          | <b>mqsicreateusername</b>      | “mqsicreateusername command” on page 442             |
|                                                          | <b>mqsichangeusername</b>      | “mqsichangeusername command” on page 404             |
|                                                          | <b>mqsdeleteusername</b>       | “mqsdeleteusername command” on page 464              |
| Start and stop commands                                  |                                |                                                      |
|                                                          | <b>mqsistart</b>               | “mqsistart command” on page 525                      |
|                                                          | <b>mqsistop</b>                | “mqsistop command” on page 531                       |
|                                                          | mqsistartmsgflow               | “mqsistartmsgflow command” on page 528               |
|                                                          | mqsistopmsgflow                | “mqsistopmsgflow command” on page 534                |
| List and trace commands                                  |                                |                                                      |
|                                                          | <b>mqsilist</b>                | “mqsilist (list resources) command” on page 477      |
|                                                          | <b>mqsichangetrace</b>         | “mqsichangetrace command” on page 397                |
|                                                          | <b>mqsiformatlog</b>           | “mqsiformatlog command” on page 474                  |
|                                                          | <b>mqsireadlog</b>             | “mqsireadlog command” on page 496                    |
|                                                          | <b>mqsireporttrace</b>         | “mqsireporttrace command” on page 516                |
| WebSphere MQ Publish/Subscribe interoperability commands |                                |                                                      |
|                                                          | <b>mqsiclearmqpubsub</b>       | “mqsiclearmqpubsub command” on page 408              |
|                                                          | <b>mqsijoinmqpubsub</b>        | “mqsijoinmqpubsub command” on page 475               |
|                                                          | <b>mqsilistmqpubsub</b>        | “mqsilistmqpubsub command” on page 486               |
| Migration commands                                       |                                |                                                      |
|                                                          | mqsimigratecomponents          | “mqsimigratecomponents command” on page 491          |
| Properties commands                                      |                                |                                                      |
|                                                          | <b>mqsichangeproperties</b>    | “mqsichangeproperties command” on page 381           |
|                                                          | <b>mqsireportproperties</b>    | “mqsireportproperties command” on page 509           |
| Statistics commands                                      |                                |                                                      |
|                                                          | <b>mqsichangeflowstats</b>     | “mqsichangeflowstats command” on page 372            |
|                                                          | <b>mqsireportflowstats</b>     | “mqsireportflowstats command” on page 504            |
| Miscellaneous commands                                   |                                |                                                      |
|                                                          | <b>mqsichangeflowuserexits</b> | “mqsichangeflowuserexits command” on page 377        |
|                                                          | <b>mqsicreateaclentry</b>      | “mqsicreateaclentry command” on page 409             |
|                                                          | mqsicreateconfigurable         | “mqsicreateconfigurable service command” on page 435 |
|                                                          | mqsicreateexecutiongroup       | “mqsicreateexecutiongroup command” on page 440       |
|                                                          | <b>mqsdeleteaclentry</b>       | “mqsdeleteaclentry command” on page 448              |
|                                                          | mqsdeleteconfigurable          | “mqsdeleteconfigurable service command” on page 460  |
|                                                          | mqsdeleteexecutiongroup        | “mqsdeleteexecutiongroup command” on page 462        |
|                                                          | <b>mqsdeploy</b>               | “mqsdeploy command” on page 465                      |
|                                                          | <b>mqsilistaclentry</b>        | “mqsilistaclentry command” on page 479               |

**mqsireportflowuserexits** “mqsireportflowuserexits command” on page 507

**mqsisetsecurity** “mqsisetsecurity command” on page 524

## Summary of commands on Windows, z/OS, Linux, and UNIX systems

The following table summarizes the *runtime* commands available on Windows platforms, Linux, and UNIX systems and provides the z/OS equivalent, where it is available.

| Command on Windows platforms, Linux, and UNIX systems | z/OS equivalent: type                             | z/OS equivalent                                 | z/OS References                                                                                             |
|-------------------------------------------------------|---------------------------------------------------|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------|
| mqsiaapplybaroverride                                 | Utility JCL                                       | BIPOBAR                                         | “Contents of the broker PDSE” on page 547                                                                   |
| mqsibackupconfigmgr                                   | Utility JCL                                       | BIPBUM                                          | “Contents of the Configuration Manager PDSE” on page 550                                                    |
| mqsibrreport                                          | N/A.                                              | -                                               | -                                                                                                           |
| mqsichangebroker                                      | 1. Console command: modify<br>2. Utility JCL      | 1. changebroker<br>2. BIPCHBK                   | 1. “mqsichangebroker command” on page 356<br>2. “Contents of the broker PDSE” on page 547                   |
| mqsichangeconfigmgr                                   | 1. Console command: modify<br>2. Utility JCL      | 1. changeconfigmgr<br>2. BIPCHCM                | 1. “mqsichangeconfigmgr command” on page 366<br>2. “Contents of the Configuration Manager PDSE” on page 550 |
| mqsichangedbimgr                                      | N/A                                               | -                                               | -                                                                                                           |
| mqsichangeflowstats                                   | 1. Console command: modify<br>2. Utility JCL      | 1. changeflowstats<br>2. BIPCHMS                | 1. “mqsichangeflowstats command” on page 372<br>2. “Contents of the broker PDSE” on page 547                |
| mqsichangeflowuserexits                               | 1. UNIX System Services command<br>2. Utility JCL | 1. <b>mqsichangeflowuserexits</b><br>2. BIPCHUE | 1. “mqsichangeflowuserexits command” on page 377<br>2. “Contents of the broker PDSE” on page 547            |
| mqsichangeproperties                                  | Utility JCL                                       | BIPCHPR                                         | “Contents of the broker PDSE” on page 547                                                                   |
| mqsichangetrace                                       | console command: modify                           | changetrace                                     | “mqsichangetrace command” on page 397                                                                       |
| mqsichangeusernameserver                              | 1. Console command: modify<br>2. Utility JCL      | 1. changeusernameserver<br>2. BIPCHUN           | 1. “mqsichangeusernameserver command” on page 404<br>2. “Contents of the User Name Server PDSE” on page 549 |
| mqsiclearmqpubsub                                     | Utility JCL                                       | BIPCLMP                                         | “Contents of the broker PDSE” on page 547                                                                   |

|                              |                                                 |                                                |                                                                                                                   |
|------------------------------|-------------------------------------------------|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| mqscreateaclentry            | 1. Console command:<br>modify<br>2. Utility JCL | 1. createaclentry<br>2. BIPCRACL               | 1. "mqscreateaclentry command" on page 409<br>2. "Contents of the Configuration Manager PDSE" on page 550         |
| mqscreatebroker              | UNIX System Services command                    | "mqscreatebroker command" on page 416          | "mqscreatebroker command" on page 416                                                                             |
| mqscreateconfigmgr           | Utility JCL                                     | BIPRCRM                                        | "Contents of the Configuration Manager PDSE" on page 550                                                          |
| mqscreateconfigurableservice | Utility JCL                                     | BIPJADPR                                       | "Contents of the broker PDSE" on page 547                                                                         |
| mqscreateexecutiongroup      | Utility JCL                                     | BIPCREG                                        | "Contents of the Configuration Manager PDSE" on page 550                                                          |
| mqscreateusernameeserver     | UNIX System Services command                    | "mqscreateusernameeserver command" on page 442 | "mqscreateusernameeserver command" on page 442                                                                    |
| mqsdeleteaclentry            | 1. Console command:<br>modify<br>2. Utility JCL | 1. deleteaclentry<br>2. BIPDLACL               | 1. "mqsdeleteaclentry command" on page 448<br>2. "Contents of the Configuration Manager PDSE" on page 550         |
| mqsdeletebroker              | Utility JCL                                     | BIPDLBK                                        | "Contents of the broker PDSE" on page 547                                                                         |
| mqsdeleteconfigmgr           | Utility JCL                                     | BIPDLCM                                        | "Contents of the Configuration Manager PDSE" on page 550                                                          |
| mqsdeleteconfigurableservice | Utility JCL                                     | BIPJADPR                                       | "Contents of the broker PDSE" on page 547                                                                         |
| mqsdeleteexecutiongroup      | Utility JCL                                     | BIPDLEG                                        | "Contents of the Configuration Manager PDSE" on page 550                                                          |
| mqsdeleteusernameeserver     | Utility JCL                                     | BIPDLUN                                        | "Contents of the User Name Server PDSE" on page 549                                                               |
| mqsdeploy                    | 1. Console command:<br>modify<br>2. Utility JCL | 1. deploy<br>2. BIPDPLY                        | 1. "mqsdeploy command" on page 465<br>2. "Contents of the broker PDSE" on page 547                                |
| mqsiformatlog                | Utility JCL                                     | BIPFMLG                                        | "Contents of the broker PDSE" on page 547                                                                         |
| mqsijoinmqpubsub             | Utility JCL                                     | BIPJNMP                                        | "Contents of the broker PDSE" on page 547                                                                         |
| mqsilist                     | 1. Console command:<br>modify<br>2. Utility JCL | 1. list<br>2. BIPLIST                          | 1. "mqsilist (list resources) command" on page 477<br>2. "Contents of the Configuration Manager PDSE" on page 550 |

|                         |                                                         |                                                      |                                                                                                          |
|-------------------------|---------------------------------------------------------|------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| mqsilistaclentry        | 1. Console command: modify<br>2. Utility JCL            | 1. listaclentry<br>2. BIPLIACL                       | 1. "mqsilistaclentry command" on page 479<br>2. "Contents of the Configuration Manager PDSE" on page 550 |
| mqsilistmqpubsub        | Utility JCL                                             | BIPLSMP                                              | "Contents of the broker PDSE" on page 547                                                                |
| mqsimigratecomponents   | Utility JCL                                             | BIPMGCMP                                             | "mqsimigratecomponents command" on page 491                                                              |
| mqsireadbar             | Utility JCL                                             | BIPRBAR                                              | "Contents of the broker PDSE" on page 547                                                                |
| mqsireadlog             | Utility JCL                                             | BIPRELG                                              | "Contents of the broker PDSE" on page 547                                                                |
| mqsireload              | console command: modify                                 | reload                                               | "mqsireload command" on page 500                                                                         |
| mqsireloadsecurity      | Utility JCL                                             | BIPRLSEC                                             | "Contents of the Configuration Manager PDSE" on page 550                                                 |
| mqsireportflowstats     | 1. Console command: modify<br>2. Utility JCL            | 1. reportflowstats<br>2. BIPRPMS                     | 1. "mqsireportflowstats command" on page 504<br>2. "Contents of the broker PDSE" on page 547             |
| mqsireportflowuserexits | 1. UNIX System Services command<br>2. Utility JCL       | 1. <b>mqsireportflowuserexits</b><br>2. BIPRPUE      | 1. "mqsireportflowuserexits command" on page 507<br>2. "Contents of the broker PDSE" on page 547         |
| mqsireportproperties    | Utility JCL                                             | BIPRPPR                                              | "Contents of the broker PDSE" on page 547                                                                |
| mqsireporttrace         | console command: modify                                 | reporttrace                                          | "mqsireporttrace command" on page 516                                                                    |
| mqsirestoreconfigmgr    | Utility JCL                                             | BIPRSCM                                              | "Contents of the Configuration Manager PDSE" on page 550                                                 |
| mqsisetdbparms          | Utility JCL                                             | BIPSDBP                                              | "mqsisetdbparms command" on page 520                                                                     |
| mqsisetsecurity         | N/A                                                     | -                                                    | -                                                                                                        |
| mqsistart               | 1. Console command: start<br>2. console command: modify | 1. standard MVS start command<br>2. startcomponent   | 1. -<br>2. "mqsistart command" on page 525                                                               |
| mqsistartmsgflow        | Utility JCL                                             | BIPSTMF                                              | "Contents of the Configuration Manager PDSE" on page 550                                                 |
| mqsistop                | 1. console command: stop<br>2. console command: modify  | 1. standard MVS stop command<br>2. 'p' stopcomponent | 1. -<br>2. "mqsistop command" on page 531                                                                |

|                 |             |         |                                                          |
|-----------------|-------------|---------|----------------------------------------------------------|
| mqsistopmsgflow | Utility JCL | BIPSPMF | "Contents of the Configuration Manager PDSE" on page 550 |
|-----------------|-------------|---------|----------------------------------------------------------|

## 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:

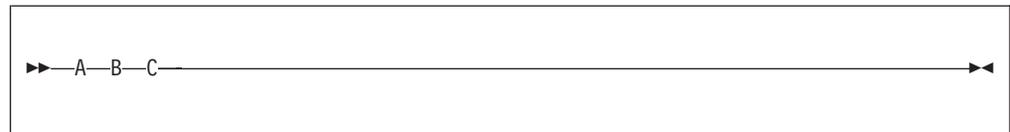
- "How to read railroad diagrams"
- "How to read dotted decimal diagrams" on page 331

### How to read railroad diagrams

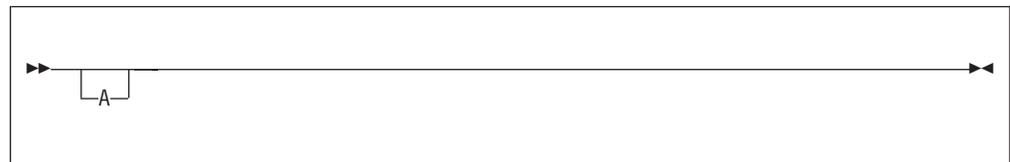
Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

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:



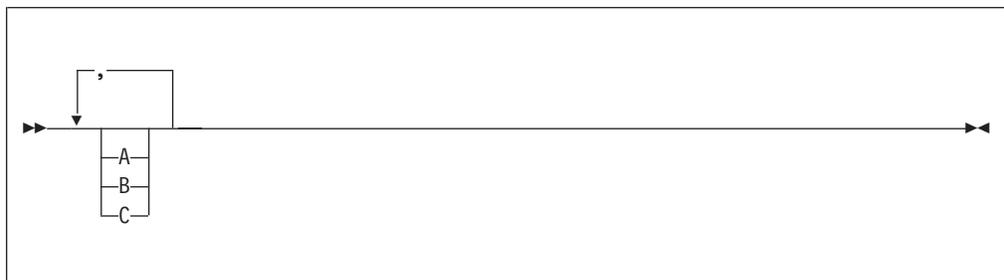
The next example specifies that values A, B, and C are options, one of which you must specify:



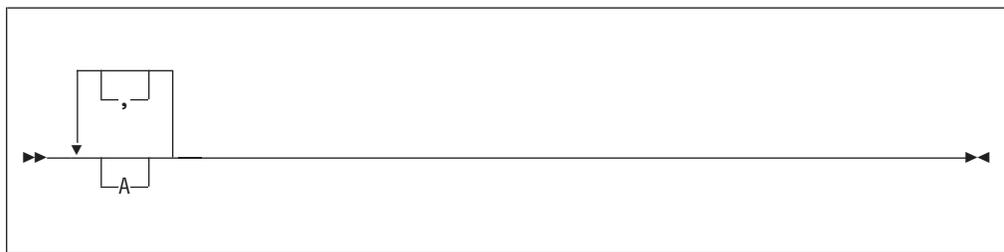
Values A, B, and C are options in this example, one of which you can specify:



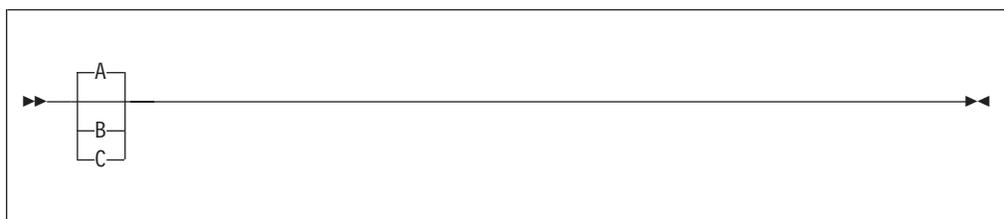
The next example shows that you can specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow:



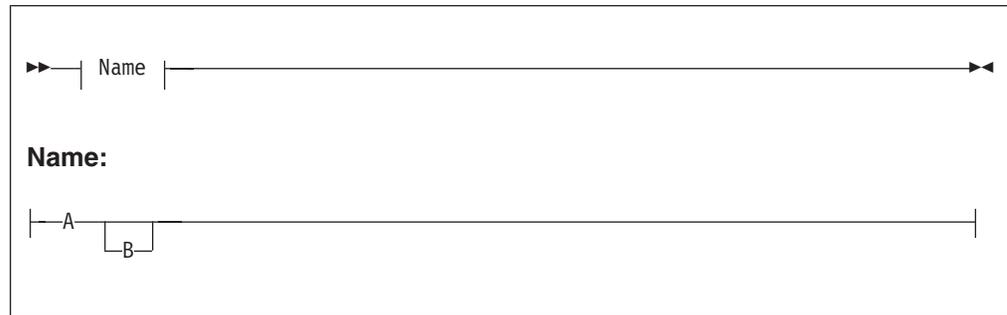
In this example, you can specify value A multiple times. The separator in this example is optional:



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 (User Name Server, message sets, message flows, 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 324 .

- 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)

This topic describes the commands that are common to both runtime and toolkit 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 338

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**

#### **Supported platforms:**

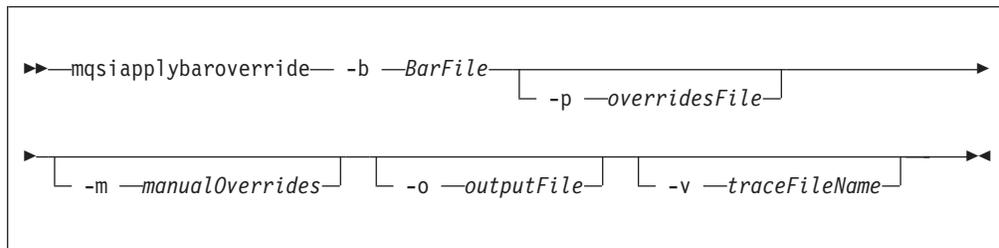
- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPOBAR.

#### **Purpose:**

With the `mqsiaapplybaroverride` command, you can replace configurable values in the broker archive (`bar`) deployment descriptor with new values that you specify in a properties file.

Write scripts to create broker archive files and apply different override values in the broker deployment descriptor archive file using the `mqsiaapplybaroverride` 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:

- A broker archive containing the deployment descriptor that will be 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, the list must be enclosed by 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 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 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 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 all platforms, 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 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 using the deployment descriptor contained in c:\previous.bar:

```
mqsiapplybaroverride mqsiapplybaroverride -b c:\test.bar -p c:\previous.bar
```

Override the deployment descriptor in c:\test.bar using the deployment descriptor contained in c:\broker.xml:

```
mqsipplybaroverride mqsipplybaroverride -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:

```
mqsipplybaroverride mqsipplybaroverride -b c:\test.bar -o OLDA=NEWA:OLDB=NEWB
```

Override the value of the property name sampleFlow#MQInput.queueName to NEWC:

```
mqsipplybaroverride mqsipplybaroverride -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

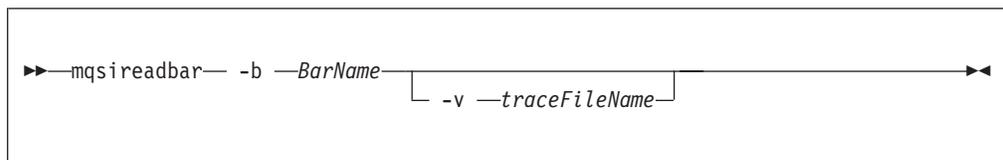
### Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPRBAR.

### Purpose:

The **mqsireadbar** command lists to the console 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 is an internal debug trace, which is to be used only at the request of an IBM service representative.

### Authorization:

The user ID used to invoke 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 toolkit or runtime), before it displays anything else. For example:

```
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 any deployment descriptor 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 displays defined keywords within the given file to the console

```
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 324 for a list of all the WebSphere Message Broker commands.

### mqsicreatebar command

#### Supported platforms:

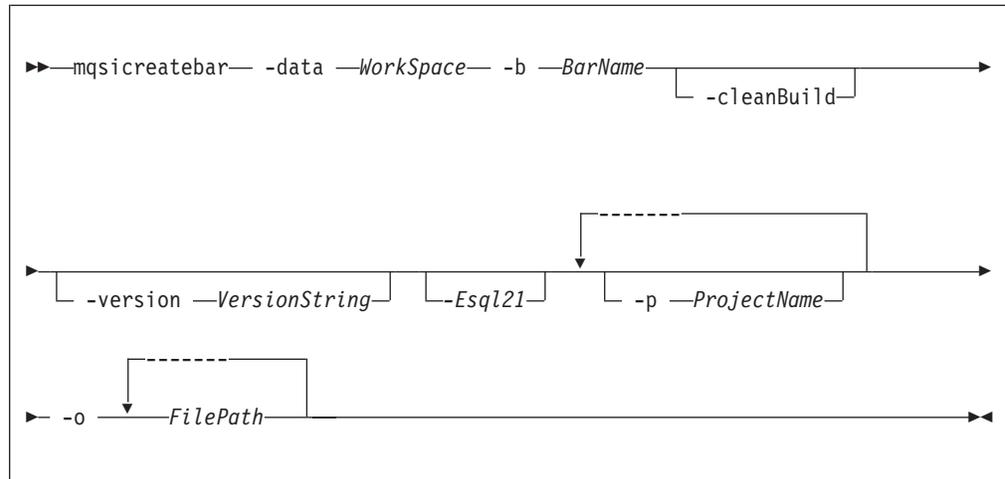
- Windows
- Linux (x86 platform)

## Purpose:

The **mqsicreatebar** command provides a command line compiler that creates deployable broker archive files containing message flows and dictionaries.

Although you do not have to use a repository, if you do 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 (zip format) archive file where the result is stored. The bar is replaced if it already exists and the META-INF/broker.xml file is created.

### **-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, before the file extension.

### **-Esq21**

(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 plug-in 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.

You can add more than one deployable file to this command by using the following format: `-o FilePath1 FilePath2 .... FilePath'n'`

**Authorization:**

On Windows, the user ID used to invoke this command must have Administrator authority on the local system. On Linux, 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 on Windows is C:\Program Files\IBM\WMBT610.

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

This command can be used to create message definition files.

### Supported platforms:

- Windows
- Linux (x86 platform)

### Purpose:

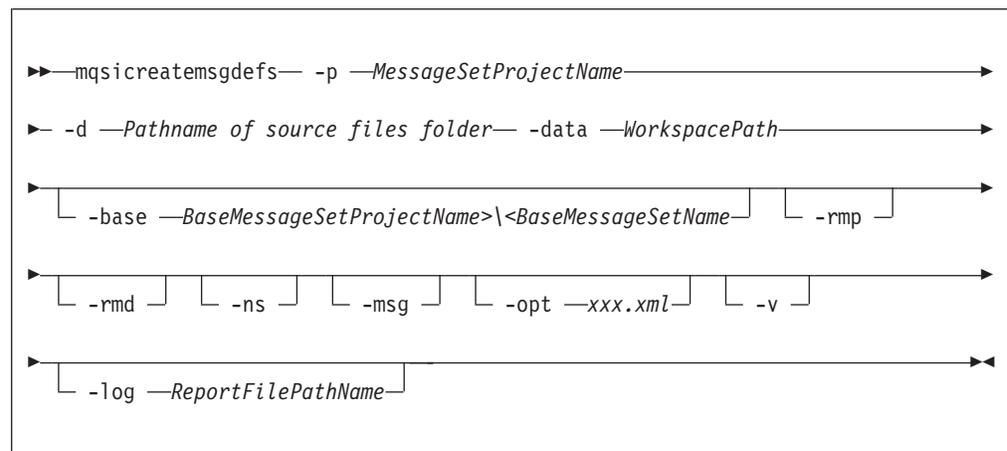
The `mqsicreatemsgdefs` command generates message definition files (\*.mxsd), according to a set of import options that are specified in an option file. The generated files are placed in the specified message set folder.

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 invokes the appropriate operation based on the extensions to the files.

### Note:

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. This might include files that are not related to the message definitions that you are trying to create.
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**  
(Optional) If a new message set is to be created, this is 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.
- Note:**
1. If this flag is omitted, and a message definition file of the same name exists, you receive a warning.
  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” on page 344
    - **COBOL language** - “COBOL options file” on page 345
    - **XSD\_NO\_NS** - “XSD options file” on page 347
 If you do not specify an option, the default options file (`mqsicreatemsgdefs.xml`) is used; see “Default options file” on page 348.  
  
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 omitted, 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.

**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:**

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” on page 348 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” on page 348.

For further information on using these options, see *Importing from C*.

| <C> element           | Possible values                                                                                                                                                                                            |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| COMPILER_NAME         | <ul style="list-style-type: none"> <li>• Msvc (Default)</li> <li>• Icc</li> <li>• AIXgcc</li> <li>• AIXxlc</li> <li>• OS390</li> </ul>                                                                     |
| CODEPAGE              | <ul style="list-style-type: none"> <li>• SO8859-1</li> <li>• Cp037</li> <li>• Cp1252 (Default)</li> </ul>                                                                                                  |
| FLOATING_POINT_FORMAT | <ul style="list-style-type: none"> <li>• IEEE Extended INTEL (Default)</li> <li>• IEEE Extended AIX</li> <li>• IEEE Extended OS/390</li> <li>• IEEE Non-Extended</li> <li>• IBM 390 Hexadecimal</li> </ul> |
| INCLUDE_PATH          | Absolute path names of other header files, or an empty string (this is the default value).                                                                                                                 |
| BYTE_ORDER            | <ul style="list-style-type: none"> <li>• Little Endian (Default)</li> <li>• Big Endian</li> </ul>                                                                                                          |
| ADDRESS_SIZE          | <ul style="list-style-type: none"> <li>• 32 (Default)</li> <li>• 64</li> </ul>                                                                                                                             |
| SIZE_OF_LONG_DOUBLE   | <ul style="list-style-type: none"> <li>• 64 (Default)</li> <li>• 128 (Not supported)</li> </ul>                                                                                                            |

| <C> element                     | Possible values                                                                                                                                                  |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PACK_LEVEL                      | <ul style="list-style-type: none"> <li>• 1</li> <li>• 2</li> <li>• 4</li> <li>• 8 (Default)</li> <li>• 16</li> </ul>                                             |
| SIZE_OF_ENUM                    | <ul style="list-style-type: none"> <li>• 1</li> <li>• 2</li> <li>• 4</li> <li>• 5 (Default)</li> </ul>                                                           |
| PRESERVE_CASE_IN_VARIABLE_NAMES | <ul style="list-style-type: none"> <li>• True (Default)</li> <li>• False</li> </ul>                                                                              |
| STRING_ENCODING                 | <ul style="list-style-type: none"> <li>• SPACE - Fixed-length strings (Default)</li> <li>• NULL - Null-terminated strings</li> </ul>                             |
| STRING_PADDING_CHARACTER        | <ul style="list-style-type: none"> <li>• SPACE (Default)</li> <li>• NUL</li> <li>• 'c'</li> <li>• "c"</li> <li>• 0xYY</li> <li>• YY</li> <li>• U+xxxx</li> </ul> |
| 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_.                                                                                       |
| PRE_PROCESSING_OPTION           | <ul style="list-style-type: none"> <li>• none (default)</li> <li>• ale_idoc</li> <li>• file_idoc</li> </ul>                                                      |

**Note:** If COMPILER\_NAME is set to AIXxlc, the value of PACK\_LEVEL is not used.

**Note:** In INCLUDE\_PATH, paths should be separated by the system-dependent path separator character.

#### **COBOL options file:**

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” on page 348 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” on page 348.

For further information about using these options, see Importing from COBOL copybooks.

| <COBOL> element                           | Possible values                                                                                                                                                  |
|-------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| PLATFORM_SELECTION                        | <ul style="list-style-type: none"> <li>• 0 (Win32) (Default)</li> <li>• 1 (AIX)</li> <li>• 2 (z/OS)</li> </ul>                                                   |
| CODEPAGE                                  | <ul style="list-style-type: none"> <li>• ISO8859_1 (Default)</li> <li>• 037</li> </ul>                                                                           |
| FLOATING_POINT_FORMAT                     | <ul style="list-style-type: none"> <li>• IEEE Non-Extended (Default)</li> <li>• IBM 390 Hexadecimal</li> </ul>                                                   |
| ENDIAN                                    | <ul style="list-style-type: none"> <li>• Big</li> <li>• Little (Default)</li> </ul>                                                                              |
| EXT_DECIMAL_SIGN                          | <ul style="list-style-type: none"> <li>• ASCII (Default)</li> <li>• EBCDIC</li> <li>• EBCDIC Custom</li> </ul>                                                   |
| TRUNC                                     | <ul style="list-style-type: none"> <li>• STD (Default)</li> <li>• OPT</li> <li>• BIN</li> </ul>                                                                  |
| NSYMBOL                                   | <ul style="list-style-type: none"> <li>• DBCS</li> <li>• NATIONAL (Default)</li> </ul>                                                                           |
| QUOTE                                     | <ul style="list-style-type: none"> <li>• SINGLE</li> <li>• DOUBLE (Default)</li> </ul>                                                                           |
| CREATE_DEFAULT_VALUES FROM_INITIAL_VALUES | <ul style="list-style-type: none"> <li>• True</li> <li>• False (Default)</li> </ul>                                                                              |
| CREATE_FACETS_FROM LEVEL_88_VALUE_CLAUSES | <ul style="list-style-type: none"> <li>• True</li> <li>• False (Default)</li> </ul>                                                                              |
| PRESERVE_CASE_IN VARIABLE_NAMES           | <ul style="list-style-type: none"> <li>• True (Default)</li> <li>• False</li> </ul>                                                                              |
| CREATE_NULL_VALUES_FOR_FIELDS             | <ul style="list-style-type: none"> <li>• True</li> <li>• False (Default)</li> </ul>                                                                              |
| NULL_CHARACTER                            | <ul style="list-style-type: none"> <li>• SPACE (Default)</li> <li>• NUL</li> <li>• 'c'</li> <li>• "c"</li> <li>• 0xYY</li> <li>• YY</li> <li>• U+xxxx</li> </ul> |

| <COBOL> element             | Possible values                                                                                                                                                  |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STRING_PADDING_CHARACTER    | <ul style="list-style-type: none"> <li>• SPACE (Default)</li> <li>• NUL</li> <li>• 'c'</li> <li>• "c"</li> <li>• 0xYY</li> <li>• YY</li> <li>• U+xxxx</li> </ul> |
| 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:

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” on page 348.

For further information about using these options, see Importing from XML Schema.

| <XSD> element | Possible values                                                                                         |
|---------------|---------------------------------------------------------------------------------------------------------|
| MSG           | <ul style="list-style-type: none"> <li>• elements (default)</li> <li>• types</li> <li>• both</li> </ul> |

| <XSD_NO_NS> element | Possible values                                                                                          |
|---------------------|----------------------------------------------------------------------------------------------------------|
| IMPORT              | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> </ul>                   |
| REDEFINE            | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> <li>• accept</li> </ul> |
| LIST                | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> <li>• accept</li> </ul> |

| <XSD_NO_NS> element                                                                       | Possible values                                                                                                                 |
|-------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| UNION                                                                                     | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> <li>• accept</li> </ul>                        |
| ABSTRACT_CT                                                                               | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> <li>• accept</li> </ul>                        |
| ABSTRACT_ELEMENT                                                                          | <ul style="list-style-type: none"> <li>• modify (default)</li> <li>• reject</li> <li>• accept</li> </ul>                        |
| XSD_PREFIX                                                                                | <ul style="list-style-type: none"> <li>• xsi (default)</li> <li>• &lt;any other prefix&gt;</li> </ul>                           |
| URL_PREFIX_PAIRS<br><b>Note:</b> You can specify zero, or more URL_PREFIX_PAIRS elements. | Attribute pair <ul style="list-style-type: none"> <li>• uri=&lt;uri_value&gt;</li> <li>• prefix=&lt;prefix_value&gt;</li> </ul> |

### Default options file:

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.

```
<?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

### Supported platforms:

- Windows
- Linux (x86 platform)

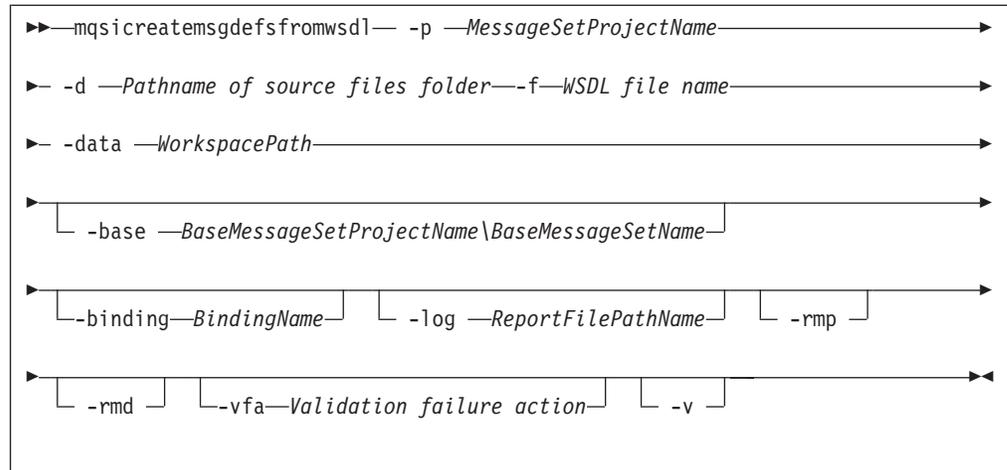
### Purpose:

The mqsicreatemsgdefsfromwsdl command can be used to import a single WSDL definition. 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.

### Note:

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. This could include any files not associated to that WSDL definition.
2. If the WSDL definition uses a relative path that includes files outside of the directory or subdirectory structure specified, these files must be imported into the workspace prior to running the command. Care must be taken to ensure that the relative paths are still valid after importing these files into the workspace
3. Message sets that are created will be 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 run time 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, this is 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 this flag is omitted, and a message definition file of the same name exists, you receive a warning.
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 will stop and errors will be written to the log file.
- warn: If the WSDL definition is not WS-I compliant, the import process will write 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 will be written to the logfile.

**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**

The mqsimigratemfmaps command migrates mapping and database definitions to Version 6.1.

**Supported platforms:**

- Windows
- Linux (x86 platform)

**Purpose:**

Use the mqsimigratemfmaps command to migrate resources to WebSphere Message Broker Version 6.1 from WebSphere Business Integration Message Broker Version 5.0:

- 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.

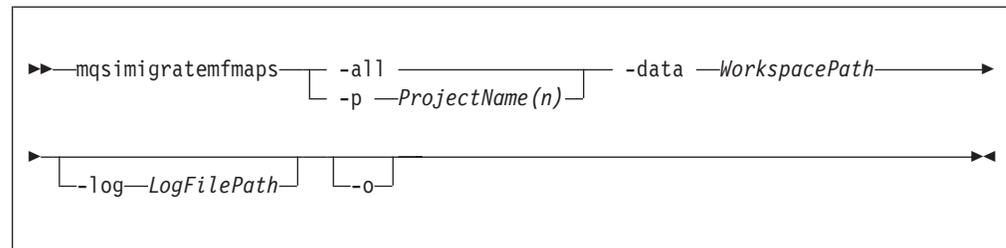
This command is valid only for migration of Version 5.0 resources. You do not need to migrate Version 6.0 maps which are compatible with Version 6.1. If you have data definitions in your Version 6.0 projects, you must migrate these in the workbench; you cannot use this command.

**Notes:**

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 in Version 6.1; 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 (x86 platform) and C:\Program Files\IBM\WMBT610 on Windows.

**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:**

No specific authority is required to invoke 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 324 for a list of all the WebSphere Message Broker commands.

### **mqsibackupconfigmgr command**

Command to back up the Configuration Manager repository online and use the Configuration Manager Proxy to communicate with the Configuration Manager.

#### **Supported platforms:**

- Windows
- Linux and UNIX systems

- z/OS using BIPBUCM; see “Contents of the Configuration Manager PDSE” on page 550

**Purpose:**

This command backs up the Configuration Manager repository online and uses the Configuration Manager Proxy to communicate with the Configuration Manager.

**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:

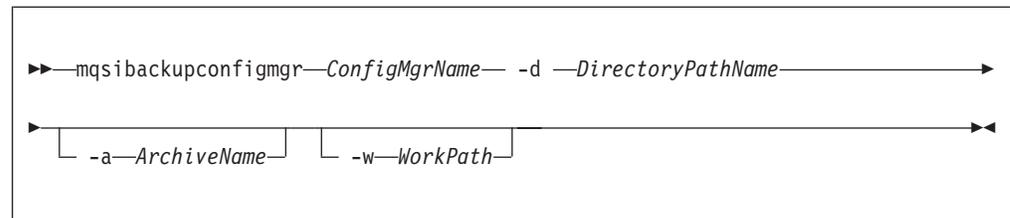
<Configuration Manager directory>/components/<component name>/<directory name>/service.properties

You must keep this file with the .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 (work path location) option on the mqsicreateconfigmgr command when you created the Configuration Manager, you must specify the -w (work path 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 work path 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:**

The user ID used to invoke this command 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. In addition, for Linux and UNIX systems, the user ID must be a member of mqbrkrs.

**Examples:**

```
mqsibackupconfigmgr ConfigMgr -d c:\mqstest\db -a archive1
```

## mqsicbrreport command

**Supported platforms:**

- Windows
- Linux and UNIX systems

**Purpose:**

Use the **mqsicbrreport** command to help identify applications that use a content-based routing filter. The program inspects a broker's subscriptions table, and reports any filters it finds that might cause incompatible behavior.

You must invoke on the system on which the program is running and you must be logged on as the service user ID for the broker.

**Syntax:**

```
►►mqsicbrreport—BrokerName—◄◄
```

**Parameters:**

**BrokerName**

(Required) The name of the broker to inspect. The output is written to stdout and you must review the report generated so that you can make any necessary changes to your programs.

**Note:** This utility only identifies and reports possible incompatibilities.

**Authorization:**

On Windows platforms, the user ID used to invoke this command must have **Administrator** authority on the local system.

On Linux and UNIX systems, the user ID used to invoke this command must be **root**. It must also be a member of the **mqbrkrs** group.

## mqsichangebroker command

**Supported 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 BIPCHBK; see "Contents of the broker PDSE" on page 547

**Purpose:**

Use the **mqsichangebroker** command to change some of the startup parameters of 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 of a broker.

You must stop the broker before you can 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 the broker.
- On z/OS, you must have started the original broker control process using the `/S` option. You must stop the broker components using the `/F` broker, `PC` option and start the broker components again using the `/S` broker, `SC` option.

See “**mqsistop** command” on page 531 and “**mqsistart** command” on page 525 for more information.

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 362

**Authorization:**

On Windows systems, the user ID used to invoke this command must have **Administrator** authority on the local system.

On Linux and UNIX systems, the user ID used to invoke this command must be a member of the **mqbrkrs** group.

On z/OS systems, the user ID used to invoke this command must be a member of a group that has `READ` and `WRITE` access to the component directory.

**Using LDAP:** Ensure that the registry is appropriately secured to prevent unauthorized access. The setting of `LdapPrincipal` and `LdapCredentials` options on **mqsichangebroker** is not required for correct operation of the broker. The password is not stored in clear text in the file system.

**mqsichangebroker** command - Windows, Linux and UNIX systems:

*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.

The user ID under which components run. You can specify the *ServiceUserID* in any valid user name syntax. On Windows systems, valid formats are:

- domain\username
- \\server\username
- .\username
- username

On Linux and UNIX systems, only the last format, username, is valid.

If you use the unqualified form for this user ID (username) on Windows 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 593, and in “Security requirements for Linux and UNIX platforms” on page 592.

**-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, as well as its database access, update both instances at the same time. (See the description of the **-a** parameter for further details.)

**-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, "").

- 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 *WebSphere MQ Intercommunication*.
- n (Optional) The broker ceases to run as a WebSphere MQ trusted application.
- l *UserLibPath*  
(Optional) A list of paths (directories) from which the broker loads LILs (loadable implementation libraries) for user-defined message processing nodes.  
  
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: semicolon (;) on Windows systems; colon (:) on Linux and UNIX systems.
- g *ConfigurationTimeout*  
(Optional) This parameter defines the length of time (in seconds) that an execution group in the broker can take to apply a change in configuration (for example, an update that you have deployed from the workbench).  
  
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.  
  
The value that you set for this timeout depends on the system load (including CPU utilization), and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives an indication of the minimum value to set.  
  
Specify the value in seconds, in the range 10 to 3600. The default is 300.  
  
The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response.
- k *ConfigurationDelayTimeout*  
(Optional) The length of time (in seconds) that a broker can take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).  
  
This parameter represents the time that it takes for a broker and its execution groups to process a minimal deployed configuration message; it depends on queue manager network delays, the load on the broker's queue manager, and system load.  
  
You can estimate this value by issuing a command to request a simple configuration change, for example:  

```
mqsiereporttrace brokerName -e "Execution Group Name" -u
```

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.

If the broker is on a production system, increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

**-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 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 timer interval (in minutes) at which WebSphere Message Broker statistics and accounting is told to output archive records. For internal accounting, the valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the operating system has an external method of notification 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. 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.

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 "Administering the broker domain" on page 283.

*Examples:*

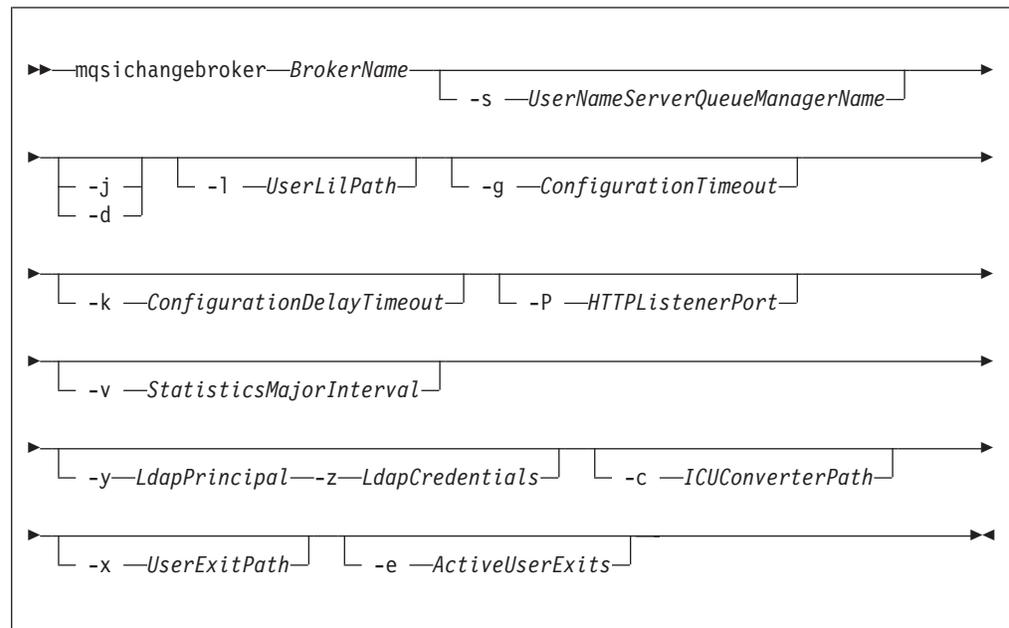
Windows, Linux, and UNIX systems:

```
mqsichangebroker WBRK_BROKER -s WBRK_UNQ_QM
mqsichangebroker WBRK_BROKER -s ""
mqsichangebroker WBRK_BROKER -x /opt/3rdparty/wmbexit
```

**mqsichangebroker command - z/OS:**

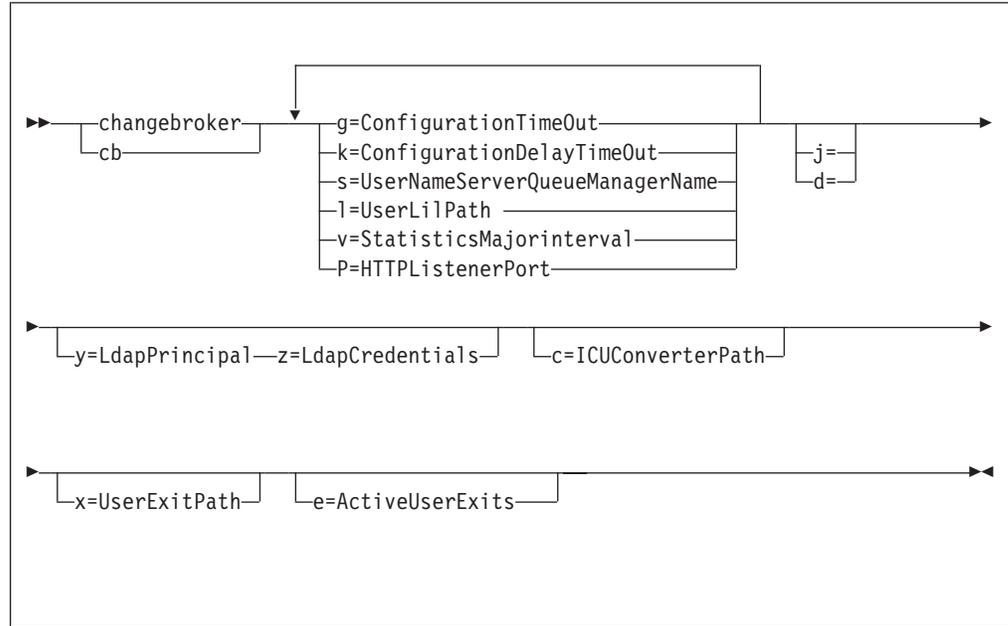
*Syntax:*

*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 LILs (loadable implementation libraries) for user-defined message processing nodes.

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: semicolon (;) on Windows systems; colon (:) on Linux and UNIX systems.

**-g** *ConfigurationTimeout*

(Optional) This parameter defines the length of time (in seconds) that an execution group in the broker can take to apply a change in configuration (for example, an update that you have deployed from the workbench).

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.

The value that you set for this timeout depends on the system load (including CPU utilization), and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives an indication of the minimum value to set.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response.

**-k** *ConfigurationDelayTimeout*

(Optional) The length of time (in seconds) that a broker can take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).

This parameter represents the time that it takes for a broker and its execution groups to process a minimal deployed configuration message; it depends on queue manager network delays, the load on the broker's queue manager, and system load.

You can estimate this value by issuing a command to request a simple configuration change, for example:

```
F MQP1BRK,reporttrace u=yes,e='exgrp1'
```

The name is case sensitive; enclose names in single quotation marks if they are in mixed case.

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.

If the broker is on a production system, increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

**-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 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 WebSphere Message Broker statistics and accounting is notified that archive records are to be output. The valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the operating system has an external method of notification 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.

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 "Administering the broker domain" on page 283.

*Examples:*

z/OS:

F MQP1BRK,cb g=100,k=200

You must use a comma between each command option. The following command illustrates this requirement and uses both the **u** and **e** parameters:

```
/f MA05BRK,cb x='/u/test/wbi/MsgFlowTrackingUserExit/zOS',e='MqsiStrUserExit02:MqsiStrUserExit03'
```

## **mqsichangeconfigmgr 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 BIPCHCM; see “Contents of the Configuration Manager PDSE” on page 550

### **Purpose:**

Use the **mqsichangeconfigmgr** command to change some of the properties of the Configuration Manager.

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 /S Configuration Manager, SC option.

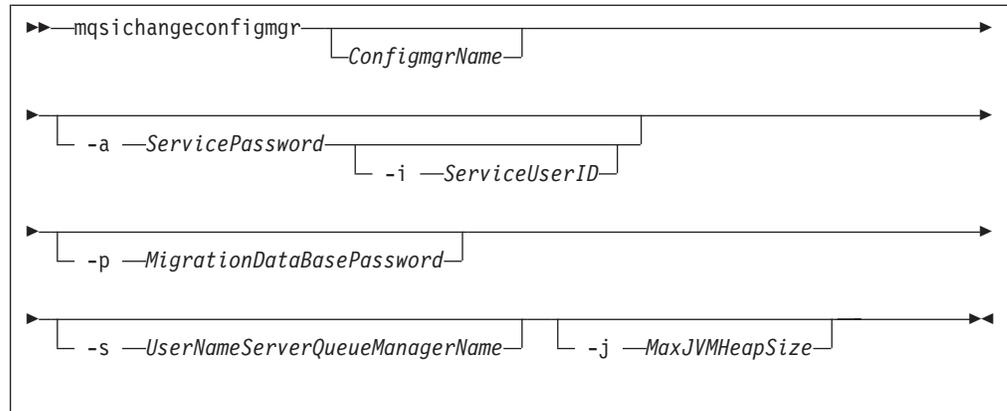
See “mqsistop command” on page 531 and “mqsistart command” on page 525 for more information.

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 368
- “mqsichangeconfigmgr command - z/OS” on page 370

### **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 593.

**-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 128 MB is used.

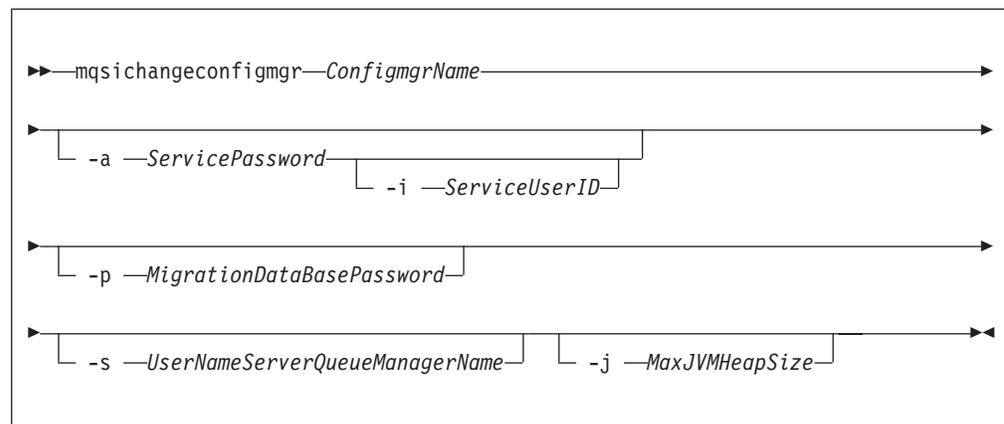
If you want to change other properties, you must delete and recreate the Configuration Manager. If you want to change the `DataBaseUserID`, see “Modifying access to the broker database” on page 215 for instructions.

*Examples:*

```
mqschangeconfigmgr CMGR01 -i ADMIN
mqschangeconfigmgr CMGR01 -s ""
```

**mqschangeconfigmgr 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 592.

**-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 128 MB is used.

If you want to change other properties, you must delete and recreate the Configuration Manager. If you want to change the DataBaseUserID, see “Modifying access to the broker database” on page 215 for instructions.

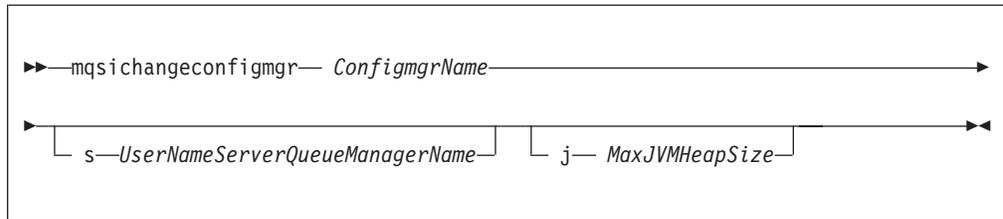
*Examples:*

```
mqsichangeconfigmgr CMGR01 -i ADMIN
mqsichangeconfigmgr CMGR01 -s ""
```

## mqschangeconfigmgr command - z/OS:

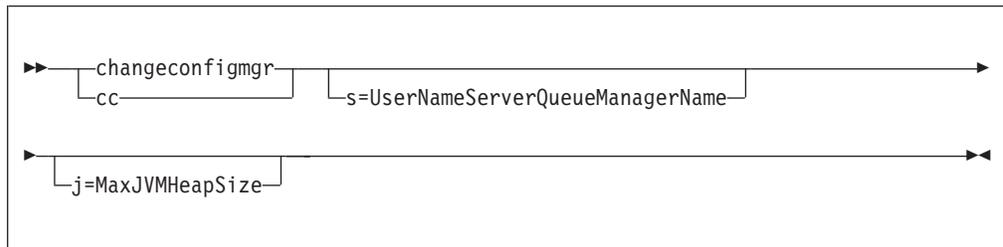
*Syntax:*

*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 128 MB is used.

If you want to change other properties, you must delete and recreate the Configuration Manager. If you want to change the `DataBaseUserID`, see "Modifying access to the broker database" on page 215 for instructions.

*Examples:*

```
mqschangeconfigmgr CMGR01 -s ""
```

## mqsichangedbimgr command

### Supported platforms:

- Windows

### 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:

```
►► mqsichangedbimgr -a —ServicePassword [-i —ServiceUserID] ◄◄
```

### Parameters:

#### -a *ServicePassword*

(Required) The new password for the ServiceUserID. To complete a password change successfully, you must:

- Stop all the broker components using the **mqsistop** command.
- Change the password using the appropriate Windows facilities.
- Use the appropriate mqsichange commands to update all the components that reference this same ServiceUserID and password.
- Restart all the broker components using the **mqsistart** command.

#### -i *ServiceUserID*

(Optional) The user ID under which the default database runs. This can be specified in any valid username syntax. On Windows systems, these are:

- domain\username
- \\server\username
- .\username
- username

If you use the unqualified form for this user ID (username) on Windows systems, the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete. The ServiceUserID specified must be a member of the local group **mqbrkrs**. On Windows systems, it 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). The security requirements for the ServiceUserID are detailed in “Security requirements for Windows platforms” on page 593.

### Authorization:

The user ID used to invoke 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 IBM WebSphere Message Broker DatabaseInstanceMgr service:

```
mqsichangedbimgr -a wbrkpw1
```

## 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 BIPCHMS; see “Contents of the broker PDSE” on page 547

### 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.

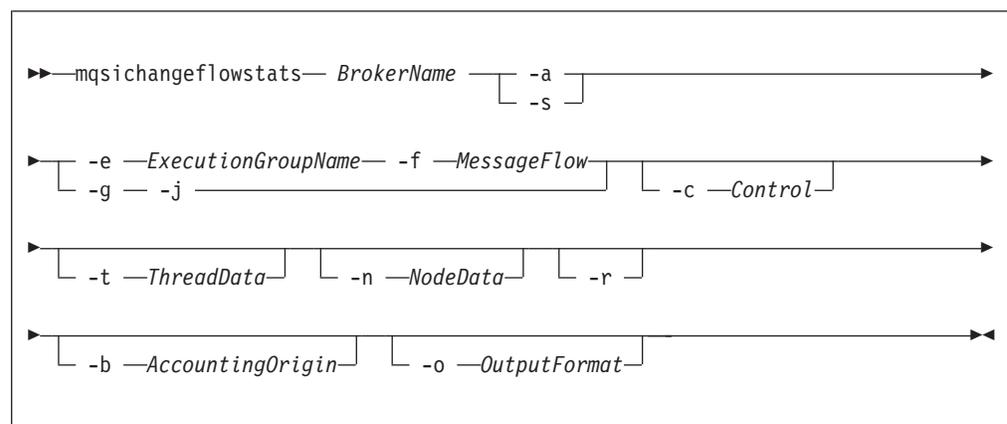
The options set using this command remain active until modified by a subsequent **mqsichangeflowstats** command.

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 374

### mqsichangeflowstats command - Windows, Linux and UNIX systems:

*Syntax:*



*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
```

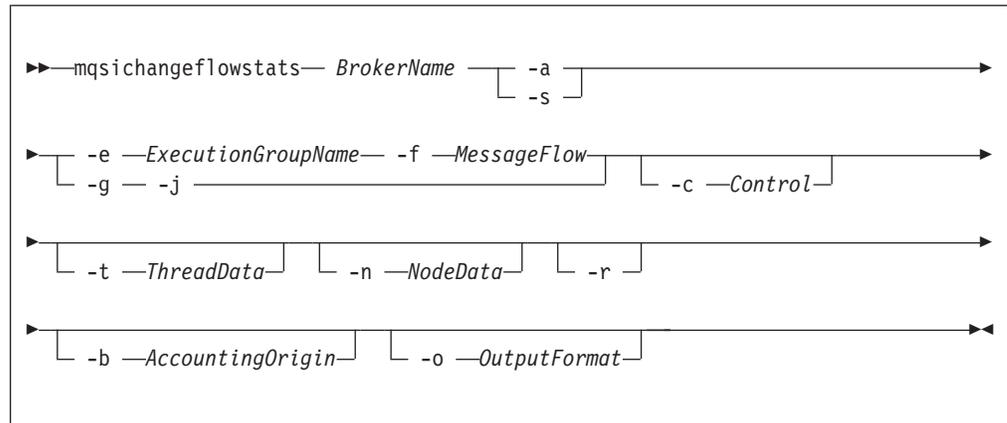
Turn off snapshot data for all message flows in all execution groups for Broker A.

```
mqsichangeflowstats BrokerA -s -g -j -c inactive
```

**mqsichangeflowstats command - z/OS:**

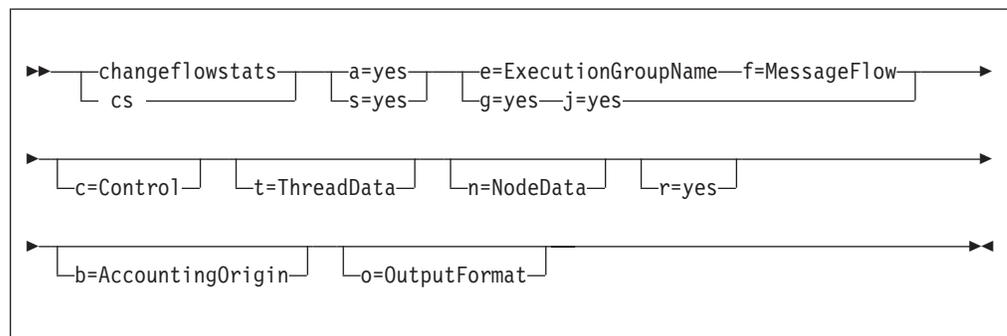
*Syntax:*

z/OS command - BIPCHMS:



z/OS console command:

**Synonym:** cs



*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:

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 -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 -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

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 547

Purpose:

Use the **mqsichangeflowuserexits** command to set the list of active or inactive user exits. There is a list of active and a list of inactive user exits for each execution group and message flow. The effective state of user exits for a given flow is decided when the flow starts.

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, then its state is taken from the execution group setting. If its state is not set for the message flow or execution group, then 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 that 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 are invoked in the order that 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 that 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 any user exits specified become invalid then, depending on which list the user exit appeared in, the following action is taken:

- If the user exit was specified in the message flow's active or inactive list, then 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, then the execution group fails to start and a BIP2314 message is written to system log.

This could happen for one of the following 3 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 broker is unable to load it.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

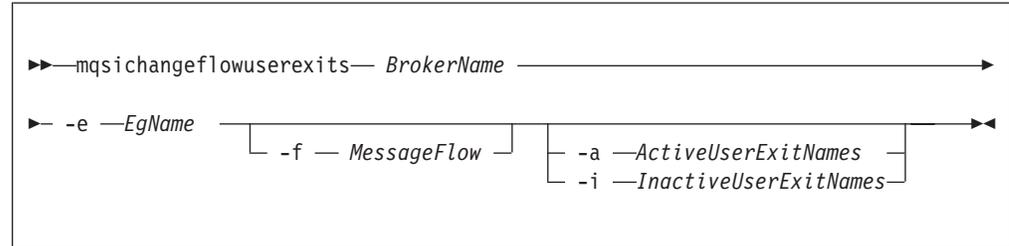
- “mqsichangeflowuserexits command - Windows, Linux, and UNIX systems” on page 379
- “mqsichangeflowuserexits command - z/OS” on page 380

Authorization:

On Windows, Linux and UNIX systems, the user ID that is used to invoke this command must have **mqbrkrs** group authority.

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 -i 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)

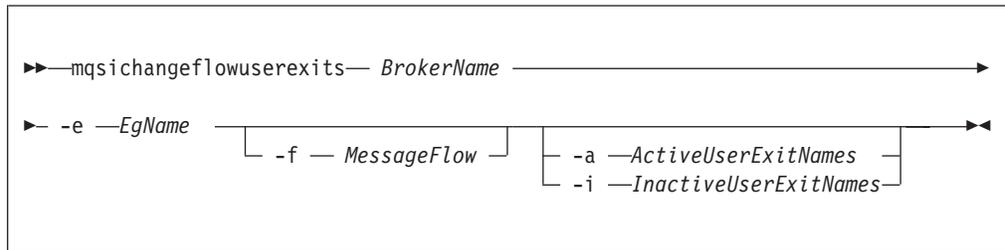
```
mqschangeuserexits WBRK_BROKER -e default -f myFlow -i exit1 -a exit2
```

BIP8071I: Successful command completion.

mqschangeuserexits command - z/OS:

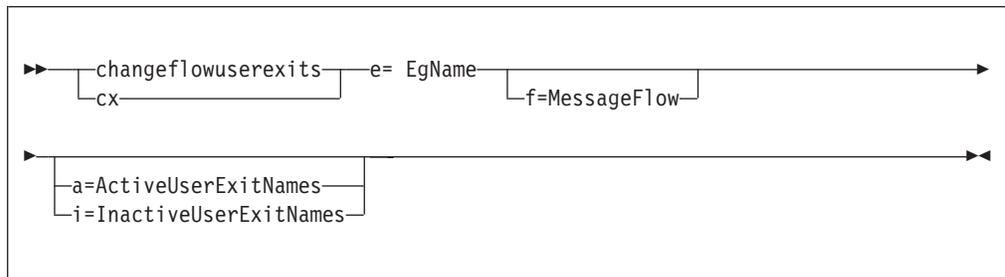
Syntax:

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 -i 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 547.

Purpose:

Use the mqsichangeproperties command to change properties that relate to the broker and the configurable services that have been supplied, or that you have created. These services include:

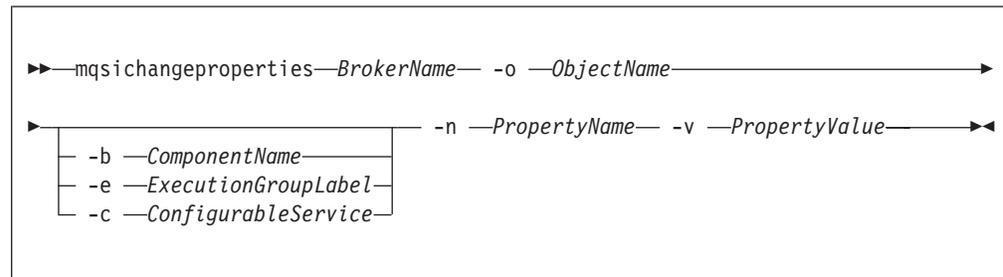
- Inter-broker communications
- The securitycache component
- The httplistener component, which includes the HTTP and HTTPS (SSL) support for the HTTPInput and HTTPReply nodes
- The httpConnectorcomponent, which includes the HTTP and HTTPS (SSL) support for the SOAPInput and SOAPReply nodes
- JMSProviders
- JDBCProviders

You can also change properties using the Configuration Manager Proxy API.

Usage notes:

- Before you run this command, ensure that the broker is running.
- If you change any value, stop and restart the broker for the change to take effect.

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*.

-e *ExecutionGroupLabel*

(Optional) The label of the execution group for which settings are to be viewed or changed.

The special name *AllExecutionGroups* is allowed, and is used to view settings for the Connector object name; this label allows you to view broker-wide settings that affect all execution groups.

-c *ConfigurableService*

(Optional) The type of configurable service type, such as *JMSProviders*, *JDBCProviders*, *EISProviders*, *SecurityProfiles*, *PolicySets*, *PolicySetBindings*, and *FtpServer*. Use the *mqsireportproperties* command to view the list of valid values.

-o *ObjectName*

(Required) The name of the object for which you want to change the properties.

The values:

- *SAP*, *Siebel*, and *PeopleSoft* are accepted for WebSphere Adapters components.
- *HTTPConnector*, and *HTTPSConnector* are accepted for all execution groups, and allowed if you select the value *httplistener* for *ComponentName*.
- *SecurityCache*
- *BrokerRegistry* is used without the need to specify *ComponentName*, *ExecutionGroupLabel*, or *ConfigurableService*.
- *SecurityProfileName* to be changed.

-n *PropertyName*

(Required) The name of the property to be changed.

For a JMSProviders or EISProviders resource, valid property names are:

- jarsURL
- nativeLibs

For a JDBCProviders resource, valid property names are:

- databaseType
- databaseVersion
- type4DriverClassName
- connectionUrlFormat
- connectionUrlFormat Attr1-5
- databaseName
- description
- jarsURL
- portNumber
- serverName
- securityIdentity

For a SecurityProfiles resource, valid property names are:

- authentication = {NONE, LDAP,TFIM}
- authenticationConfig = string
- mapping = {NONE, TFIM}
- mappingConfig = string
- authorization = {NONE, LDAP, TFIM}
- authorizationConfig = string
- propagation = {TRUE, FALSE}
- passwordValue = {PLAIN, MASK, OBFUSCATE}

For an FtpServer resource, valid property names are:

- serverName
- securityIdentity
- accountInfo
- remoteDirectory
- connectionType
- scanDelay
- transferMode

For a BrokerRegistry resource, valid property names are:

- brokerKeystoreType
- brokerKeystoreFile
- brokerKeystorePass
- brokerTruststoreType
- brokerTruststoreFile
- brokerTruststorePass
- httpConnectorPortRange
- httpsConnectorPortRange

-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 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:

```
mqschangeproperties WBRK_BROKER -c JDBCProviders -o DB2EXTRA -n connectionUrlFormat  
-v "jdbc:db2://[serverName]:[portNumber]/[databaseName]:user=[user];password=[password];"
```

If you set the `-c` parameter to `JMSProviders` or `EISProviders`, and the `-n` parameter to `jarsURL`, the expected value is a URL that specifies the file location of the JMS or EIS provider JAR files, while omitting the `file://` of the URL.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `authentication`, the property value can be `NONE`, `LDAP`, `TFIM`, or a user-defined value to specify an alternative.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `authenticationConfig`, the expected property value is a provider-specific configuration string.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `mapping`, the property value can be `NONE`, `TFIM`, or a user-defined value to specify an alternative.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `mappingConfig`, the expected property value is a provider-specific configuration string.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `authorization`, the property value can be `NONE`, `LDAP`, `TFIM`, or a user-defined value to specify an alternative.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `authorizationConfig`, the expected value is a provider-specific configuration string.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `propagation`, the property value can be either `TRUE` or `FALSE`.

If you set the `-c` parameter to `SecurityProfiles`, and the `-n` parameter to `passwordValue`, the property value can be `PLAIN`, `MASK`, or `OBFUSCATE`.

For Inter-broker communications, the following *ObjectName* value is available:

- *DynamicSubscriptionEngine*

For the `httpListener` component, when using the `HTTPInput` and `HTTPReply` nodes, the following *ObjectName* value is available:

- *HTTPListener* for controlling the `HTTPListener` process.

For the `httpListener` component, when using the `SOAPInput` and `SOAPReply` nodes, the following *ObjectName* values are available:

- *HTTPConnector* for controlling HTTP communication.
- *HTTPSConnector* for controlling HTTPS communication.

For compatibility mode with previous versions the following *ObjectName* value is available:

- *ComIbmXmlParserFactory*

For the securitycache component, the following *ObjectName* value is available:

- SecurityCache

httplistener component parameter values

Valid property names for the httplistener component:

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

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

Valid property names for the *HTTPConnectorObjectName* (Listener):

address

For servers with more than one IP address, this value specifies which address is used for listening on the specified port. By default, this port is used on all IP addresses associated with the server. If specified, only one address can be used.

- Value type - string
- Initial value - null

port The TCP port number on which this Connector creates a server socket and awaits incoming connections.

- Value type - integer
- Initial value - 7800

Valid property names for the *HTTPConnectorObjectName* (Execution groups):

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

explicitlySetPortNumber

The TCP port number on which this Connector creates a server socket and awaits incoming connections.

You can set this value in the range zero through 65 536.

- Value type - integer
- Initial value - 7800

Similarly for the *HTTPSConnector ObjectName*:

- Port applies if you are using the listener
- `explicitlySetPortNumber` applies if you are using execution groups

Valid property names for the `HTTPSCollectorObjectName`:

address

For servers with more than one IP address, this value specifies which address is used for listening on the specified port. By default, this port is used on all IP addresses associated with the server. If specified, only one address can be used.

- Value type - string
- Initial value - null

port The TCP port number on which this Connector creates a server socket and awaits incoming connections.

- Value type - integer
- Initial value - 7800

explicitlySetPortNumber

The TCP port number on which this Connector creates a server socket and awaits incoming connections.

You can set this value in the range zero through 65 536.

- Value type - integer
- Initial value - 7800

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 platforms (AIX, z/OS, Linux, Windows)

clientAuth

Set to true if the SSL stack requires a valid certificate chain from the client before accepting a connection. A false value (which is the default) does not require a certificate chain unless the client requests a resource protected by a security constraint that uses CLIENT-CERT authentication.

- Value type - string
- Initial value - false

keystoreFile

The path to the keystore file where the server certificate, which is to be loaded, has been stored. By default the http listener will expect a file called `.keystore` in the home directory of the user who started the broker.

- Value type - string
- Initial value - *default value* (described 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 possible values for *keystoreType*, *sslProtocol*, and *ciphers* are JSSE-implementation specific, and these values are in the JSSE provider documentation.

Inter-broker communications parameter values**Valid property names for inter-broker communications:****brokerInputQueues**

Specifies the maximum number of dispatch queues which are to be used when processing messages from an interbroker connection. Increasing the value might increase the rate at which messages can be transmitted across an interbroker connection:

- Value type - Integer
- Initial value - 1

brokerInputQueueLength

Defines the maximum number of messages that can be stored in each input queue; the higher the value, the higher the number of input messages that can be stored in each input queue. Note that the higher the value of this property, the larger the amount of memory that the broker requires for each queue:

- Value type - Integer
- Initial value - 99

brokerPingInterval

Time in milliseconds between broker initiated ping messages on broker-broker connections. Ping messages ensure that the communications are still open between both sides of the connection, and are generated internally. If the value is 0, the broker does not initiate pings.

- Value type - Integer
- Initial value - 5000

clientPingInterval

Time in milliseconds between broker initiated ping messages on broker-client connections. Ping messages ensure that the communications are still open between both sides of the connection, and are generated internally. If the value is 0, the broker does not initiate pings.

- Value type - Integer
- Initial value - 30000

clonedPubSubBrokerList

The list of brokers in which <brokername> registers to be a clone.

This list takes the format

BrokerName1,BrokerQMName1,BrokerName2,BrokerQMName2. Therefore, if you require to clone three brokers BRKA, BRKB, and BRKC, you must run the following commands:

```
mqsichangeproperties BRKA -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"BRKB,BRKBQM,BRKC,BRKCQM\"
mqsichangeproperties BRKB -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"BRKA,BRKAQM,BRKC,BRKCQM\"
mqsichangeproperties BRKC -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"BRKA,BRKAQM,BRKB,BRKBQM\"
```

It is also possible to remove brokers from a cloned environment. For example, if BRKC is to be removed from the clone use the following commands:

```
mqsichangeproperties BRKA -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"-BRKC\"
mqsichangeproperties BRKB -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"-BRKC\"
mqsichangeproperties BRKC -e default -o DynamicSubscriptionEngine
-n clonedPubSubBrokerList -v \"\"
```

Note the minus symbol in the list for BRKA and BRKB. BRKC does not specify any brokers and is, therefore, not a clone. However, BRKA and BRKB remain clones of each other.

- Value type - String
- Initial value - none

enableClientDiscOnQueueOverflow

If true, and if after deleting all possible messages the maxClientQueueSize is still exceeded, the broker disconnects the client.

- Value type - Boolean
- Initial value - False

enableQopSecurity

Enables the level of quality of protection for messages.

By default, Quality of Protection is enabled if either the **isysQopLevel** or **sysQopLevel** value has been changed from the default value of none.

- Value type - string
- Initial value - none

interbrokerHost

Specifies the IP host name of the broker. A single broker configuration can be left to default as null.

```
mqsichangeproperties <broker> -o DynamicSubscriptionEngine -n
interbrokerHost -v <IP host name>
```

- Value type - String
- Initial value - null

If you change the value, the broker must be stopped and restarted. Then you must redeploy the full topology.

interbrokerPort

Port number on which the Broker listens for incoming interbroker

connections. If 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, the broker must be stopped and restarted. Then you must redeploy the full topology.

isysQopLevel

Applies to the system and allows brokers only to publish and subscribe.

- Value type - string
- Initial value - none

jvmMaxHeapSize

Specifies the size of the Java Virtual Machine (JVM) heap size used with the JVMManager for your Java plug in nodes.

This value must be in the range 16 777 216 through 8 589 934 592.

- Value type - Integer
- Initial value - 134 217 728

maxBrokerQueueSize

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

maxClientQueueSize

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.

maxConnections

The maximum number of concurrently connected clients which the broker permits. If this limit is reached, the broker denies new connection requests from clients. If this value is less than 0, the number of clients is unlimited.

- Value type - Integer
- Initial value - 100

maxHopCount

Specifies that maximum number of multibroker links over which a

message is sent, to ensure that messages never loop in a multibroker network. Set this value large enough to ensure that messages can travel the entire multibroker network.

- Value type - Integer
- Initial value - 20

maxMessageSize

Maximum allowed message size in bytes. If a message exceeding this maximum size is received from a client, that client is disconnected.

- Value type - Integer
- Initial value - 100000

The value of this property **must** be less than, or equal to, the *maxClientQueueSize* value.

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 through 239.255.255.255

- Value type - String
- Initial value - 239.255.255.255

multicastMaximumIPv6Address

The highest IPv6 address that the broker can use for its Multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:1 through ff02:fff:fff:fff:fff:fff:fff:fff

- Value type - String
- Initial value - ff02:fff:fff:fff:fff:fff:fff:fff

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 through 239.255.255.255

- Value type - String
- Initial value - 224.0.0.2

multicastMinimumIPv6Address

The lowest IPv6 address that the broker can use for its Multicast transmissions.

This address must be in the range ff02:0:0:0:0:0:1 through ff02:fff:fff:fff:fff:fff:fff:fff

- Value type - String
- Initial value - ff02:0:0:0:0:0:1

multicastBackoffTimeMillis

The maximum time, in milliseconds, that a client listens for another's NACKs before sending its own NACK. This value can be in the range 0 through 1000

- Value type - Integer
- Initial value - 100

multicastDataPort

The UDP data port through which Multicast packets are sent and received:

- Value type - Integer
- Initial value - 34343

multicastEnabled

Indicates whether the topics that are defined in the `multicastTopicsConfigFile` are delivered Multicast. If the value is true, the topics in the `multicastTopicsConfigFile` are delivered Multicast.

- Value type - Boolean
- Initial value - false

multicastHeartbeatTimeoutSec

Indicates the time in seconds between the arrival of control packets at each client. If a control packet does not arrive within the number, defined as twice the value specified by this property, of seconds of the previous control packet's arrival an error can be suspected.

- Value type - Integer
- Initial value - 20

multicastLimitTransRate

This property can be used in conjunction with the `multicastTransRateLimitKbps` property to control network congestion. Possible values are:

Disabled

Multicast data is transmitted as fast as possible

Static The transmission rate is limited by the value specified in `multicastTransRateLimitKbps`

Dynamic

The transmission rate can vary throughout the process, but never exceeds the value specified in `multicastTransRateLimitKbps`

- Value type - String
- Initial value - Disabled

multicastMaxKeyAge

Maximum age, in minutes, of a topic encryption key before it must be redefined.

- Value type - String
- Initial value - 360

multicastMaxMemoryAllowedKBytes

Maximum memory consumption by client reception buffers, measured in kilobytes.

- Value type - Integer
- Initial value - 262144

This parameter is available *only* if a Pragmatic General Multicast (PGM) protocol is selected.

multicastMCastSocketTTL

The maximum number of hops that a Multicast packet can make between the client and the broker. This value is one more than the maximum number of routers that there can be between the client and the broker.

A value of 1 indicates that the packet reaches all local nodes, but cannot be relayed by routers. The maximum value is 255

- Value type - Integer
- Initial value - 1

multicastMinimalHistoryKBytes

The minimum size, in kilobytes, of a buffer that is allocated as an archive

for all transmitted packets. This buffer is shared by all reliable topics, and can be used to recover lost packets. This value must be in the range 1000 through 1000000

- Value type - Integer
- Initial value - 60000

multicastMulticastInterface

Indicates what interface to use for Multicast transmissions. You can specify a host name or an IP address. A value of None causes the network interface to be operating system dependent. *IPv4 Broker Network Interface* and *IPv6 Broker Network Interface* use the same command; use a semi-colon (;) to separate the IPv4 and IPv6 addresses.

- Value type - String
- Initial value - None

Note: If you have only one network card, the default value of none works because the operating system uses the localhost value. However, if you have more than one network card you **must** set this parameter to ensure that the correct card is used.

multicastNACKAccumulationTimeMillis

The time, in milliseconds, that NACKs are aggregated in the broker, before recovered packets are sent. This value must be in the range 50 through 1000

- Value type - Integer
- Initial value - 300

multicastNACKCheckPeriodMillis

The time, in milliseconds, between periodic checks of reception status and sequence gap detection for NACK building. This value must be in the range 10 through 1000

- Value type - Integer
- Initial value - 500

multicastOverlappingTopicBehavior

This property is used to control the behavior of the broker when a client requests a Multicast subscription for a topic, that is part of a topic hierarchy containing topics, explicitly excluded for Multicast. 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

multicastPacketBuffers

The number of memory buffers that are created at startup for packet reception. Having a high number of buffers available improves the reception performance and minimizes packet loss at high delivery rates, at the price of increased memory utilization. Each buffer is 33KB and this value can be in the range 1 through 5000

- Value type - Integer

- Initial value - 500

multicastPacketSizeBytes

Indicates the size, in bytes, of Multicast packets. This value must be in the range 500 through 32000

- Value type - Integer
- Initial value - 7000

multicastProtocolType

Specifies the protocol type. Valid values are:

- PTL
- PGM/IP
- PGM/UDP
- Value type - String
- Initial value - PTL

multicastSocketBufferSizeKbytes

The size, in kilobytes, of the client's socket receiver buffer. Increasing it leads to lower loss rates. This value can be in the range 65 through 10000

- Value type - Integer
- Initial value - 3000

multicastTransRateLimitKbps

Limits the overall transmission rate in Kb (kilobits) per second.

This property is only effective if the **multicastLimitTransRate** property is not Disabled. The value of this property must not be set higher than the maximum data transmission rate of the system or the network and must be in the range 10 through 1000000

- Value type - Integer
- Initial value - 9500

nonDurableSubscriptionEvents

Indicates whether the user requires event messages when a non-durable subscriber is created or deleted. A true value causes an event publication to be created, false indicates that no event publications are made.

- Value type - Boolean
- Initial value - False

pingTimeoutMultiple

Number of consecutive clientPngIntervals or brokerPngIntervals without a response that the broker waits before disconnecting a client or broker.

- Value type - Integer
- Initial value - 3

statsInterval

The value is the number of milliseconds between statistics publications. If set to 0, statistics publications are not generated. You do not need to restart the broker after changing this property, however it can take the broker up to a minute to start producing statistics after the value is changed.

This value must be in the range 0 through 1000

- Value type - Integer
- Initial value - 0

Note: This value refers to the publish/subscribe statistics interval only.

sysQopLevel

Applies to the system and allows brokers only to publish.

- Value type - string
- Initial value - none

publish subscribe component parameter values

Valid property names for the publish subscribe component:

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

securitycache component parameter values

Valid property names for the securitycache component:

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

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

broker registry component parameter values

Valid property names for the brokerregistry component:

brokerKeystorePass

The password used to access the server certificate from the specified keystore file.

Set the password using `mqsisetdbparms` when the broker is stopped.

- Value type - string
- Initial value - `brokerKeystore::password`

brokerKeystoreFile

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)

brokerKeystoreType

The type of keystore file to be used for the server certificate.

- Value type - string
- Initial value - JKS

brokerTruststorePass

The password used to access the server certificate from the specified keystore file.

Set the password using `mqsisetdbparms` when the broker is stopped.

- Value type - string
- Initial value - `brokerKeystore::password`

brokerTruststoreFile

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)

brokerTruststoreType

The type of keystore file to be used for the server certificate.

- Value type - string
- Initial value - JKS

httpConnectorPortRange

The numeric range of ports available to the `httpConnector`.

- Value type - integer
- Initial value - 7800,7842

httpsConnectorPortRange

The numeric range of ports available to the `httpsConnector`.

- Value type - integer
- Initial value - 7800,7842

Authorization

Windows On Windows systems, the user ID used to invoke this command must have **Administrator** authority on the local system.

UNIX On UNIX systems, the user ID used to invoke this command must be a member of the `mqbrkrs` group.

Additionally, the authority required by the broker for the different protocols is as follows:

PGM/IP

The broker requires:

- **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:

Change the clientPingInterval to 200

```
mqschangeproperties TEST -o DynamicSubscriptionEngine -n clientPingInterval -v 200
```

Enable multicast

```
mqschangeproperties TEST -o DynamicSubScriptionEngine -n multicastEnabled -v true
```

Change the default ssl protocol from SSLv3 to TLS

```
mqschangeproperties TEST -b httpListener -o HTTPSConnector -n sslProtocol -v TLS
```

Set the port number when changing properties using execution groups

```
mqschangeproperties TEST -e exgroup1 -o HTTPSConnector -n explicitlySetPortNumber -v 7777
```

Set the HTTPSConnector Port Range on the broker registry

```
mqschangeproperties -o Brokerregistry -n HTTPSConnectorPortRange -v 7777,8888
```

Enable the HTTPS Connector

```
mqschangeproperties TEST -b httpListener -o HTTPListener -n enableSSLConnector -v true
```

Change the location of the JAR files for the IBM SIB JMS client

```
mqschangeproperties WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_Default_IBM_SIB -n jarsURL  
-v file://D:\SIBClient\Java
```

Enter the command on a single line; a line break has been added to enhance readability.

Enable TFIM identity mapping

```
mqschangeproperties WBRK6_DEFAULT_BROKER -c SecurityProfiles -o MyFirstSecurityProfile  
-n mapping,mappingConfig -v TFIM,http://tfimhost1.ibm.com
```

Enter the command on a single line; a line break has been added to enhance readability.

Import a policy set to a broker from a file:

```
mqschangeproperties 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. Enter the command on a single line; a line break has been added to enhance readability.

Import a policy set bindings to a broker from a file:

```
mqschangeproperties 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. Enter the command on a single line; a line break has been added to enhance readability.

Make the jar files and shared libraries available to the Websphere Adapter for SAP:

```
mqschangeproperties WBRK6_DEFAULT_BROKER -c EISProviders -o SAP -n jarsURL,nativeLibs  
-v c:\sapjco\jars,c:\sapjco\bin
```

Enter the command on a single line; a line break has been added to enhance readability.

Update the security identity for the JDBCProvider service for Oracle:

```
mqschangeproperties WBRK6_DEFAULT_BROKER -c JDBCProviders -o Oracle -n securityIdentity  
-v OracleDSN
```

OracleDSN is the DSN with which you have associated a userID and password using the mqsisetdbparms command. Enter the command on a single line; a line break has been added to enhance readability.

mqsichangetrace command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS - as a console command

Purpose:

Use the **mqsichangetrace** command to set the tracing characteristics for a component. This command is valid for:

- User trace. Specify the **-u** option.
- Service trace. Specify the **-t** option. You are recommended to use this option only if directed to do so by the action described in a BIPxxxx message, or by your IBM Support Center.
- 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 terminate user tracing for a broker, or initiate, modify, or terminate service tracing for a broker, a Configuration Manager, or the User Name Server (identified by component name). You can switch Trace nodes for a broker on and off. 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 mqsireadlog to retrieve the log as an XML format file. Use either mqsiformatlog (to produce a formatted file) or an XML browser to view the XML records.

When you set tracing on, you cause additional processing to be executed for every activity in the component you are tracing. You must expect to see some impact on performance when trace is active.

If you want to trace the command 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.

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 401

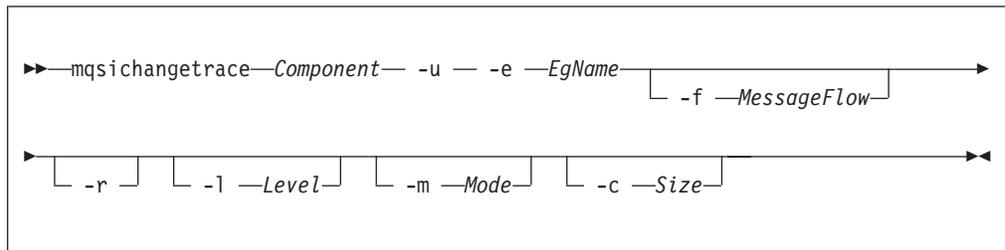
Authorization:

On Windows, Linux, and UNIX systems the user ID used to issue the command must have **mqbrkrs** authority.

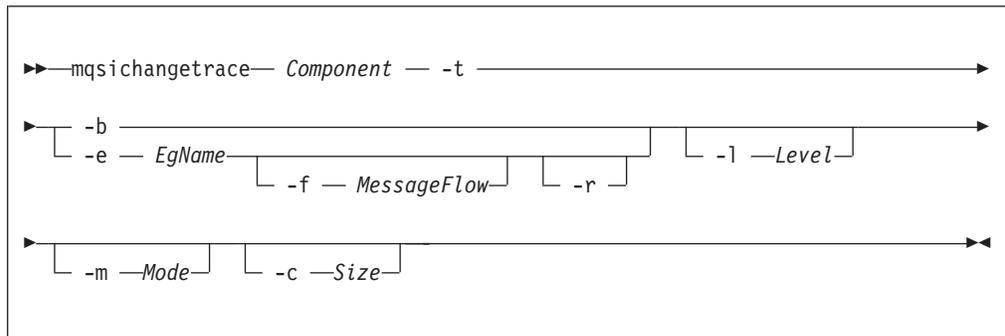
mqsichangetrace command - Windows, Linux, and UNIX systems:

Syntax:

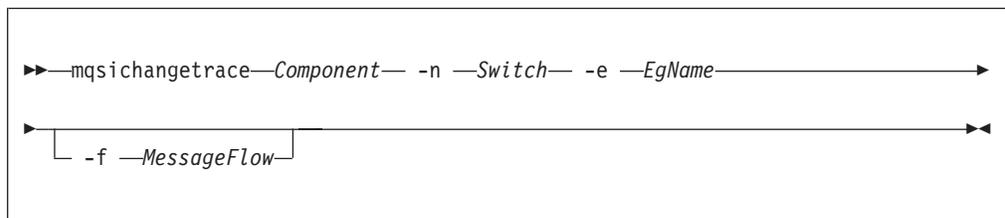
User trace:



Service trace:



Trace nodes:



Parameters:

Component

(Required) This can either be the name of a broker, or of a Configuration Manager, or the fixed value, `UserNameServer` (all are case sensitive on UNIX systems and on Linux).

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 only valid 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 only valid 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 only valid if you have specified an execution group (flag **-e**).

-l *Level*

(Optional) Set the level of the trace. This must be one of:

- `normal`. This provides a basic level of trace information.
- `none`. This switches tracing off.
- `debug`. This provides a more comprehensive trace.

This is valid for all components and each component is created with a default value of `none`. If you do not specify this parameter, the current value is unchanged. Once you successfully change this value, it is persistent.

-m *Mode*

(Optional) Indicate the way trace information is to be buffered:

- `safe`. This mode causes trace entries to be written to file when they are generated.
- `fast`. This mode causes trace entries to be buffered, and only written to file in batches.

Each component starts with a default value of `safe`. If you do not specify this parameter, the current value is unchanged.

This 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, once 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, 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:

```
mqsischangetrace WBRK_BROKER -t -b -m -l normal
```

To switch off Trace nodes in the default execution group, use the command:

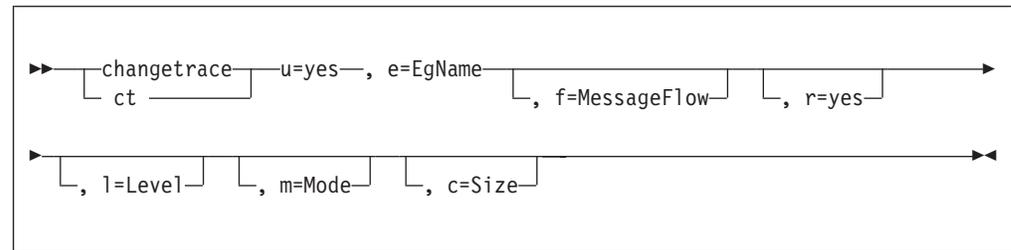
```
mqsischangetrace WBRK_BROKER -n off -e default
```

mqsischangetrace command - z/OS:

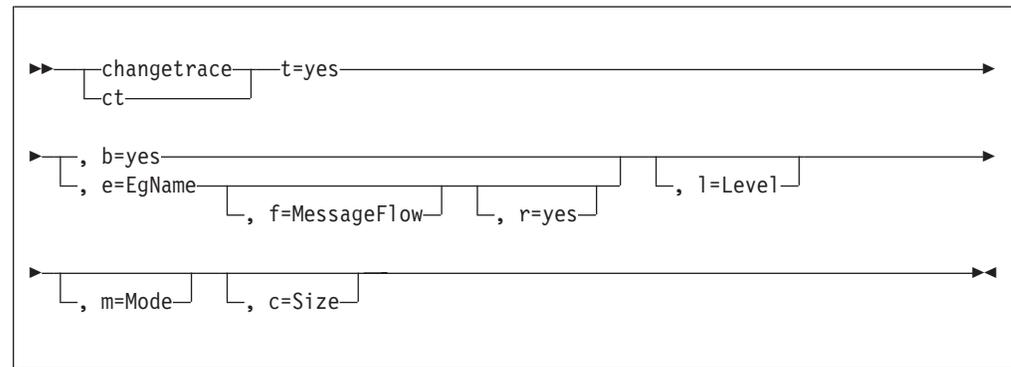
Syntax:

z/OS console command:

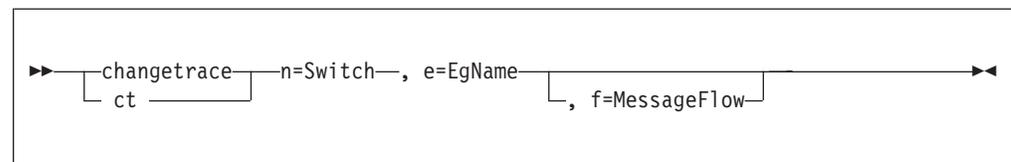
User trace



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 only valid 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 and you must include the names in single quotes if they contain mixed case characters.

-f *MessageFlow*

(Optional) Identifies the message flow for which trace options are to be modified. This option is only valid if you have specified an execution group (flag **-e**).

This name is case sensitive and you must include the names in single quotes if they contain mixed case characters.

-r

(Optional) This option requests that the trace log is reset: that is, all current records are discarded. Use this option when you start a new trace to ensure that all records in the log are unique to the new trace.

This option is only valid if you have specified an execution group (flag **-e**).

-l *Level*

(Optional) Set the level of the trace. This must be one of:

- normal. This provides a basic level of trace information.
- none. This switches tracing off.
- debug. This provides a more comprehensive trace.

This is valid for all components and each component is created with a default value of none. If you do not specify this parameter, the current value is unchanged. Once you successfully change this value, it is persistent.

-m *Mode*

(Optional) Indicate the way trace information is to be buffered:

- safe. This mode causes trace entries to be written to file when they are generated.
- fast. This mode causes trace entries to be buffered, and only written to file in batches.

Each component starts with a default value of safe. If you do not specify this parameter, the current value is unchanged.

This 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, once 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, 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 (mqsisformatlog) 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 **mqsisreadlog** 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 **mqsisreadlog** 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

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 549

Purpose:

Use the **mqsichangeusernameserver** command on Windows platforms, Linux, and UNIX systems to change some of the properties of the User Name Server.

Use the **mqsichangeusernameserver** command on z/OS to change 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:

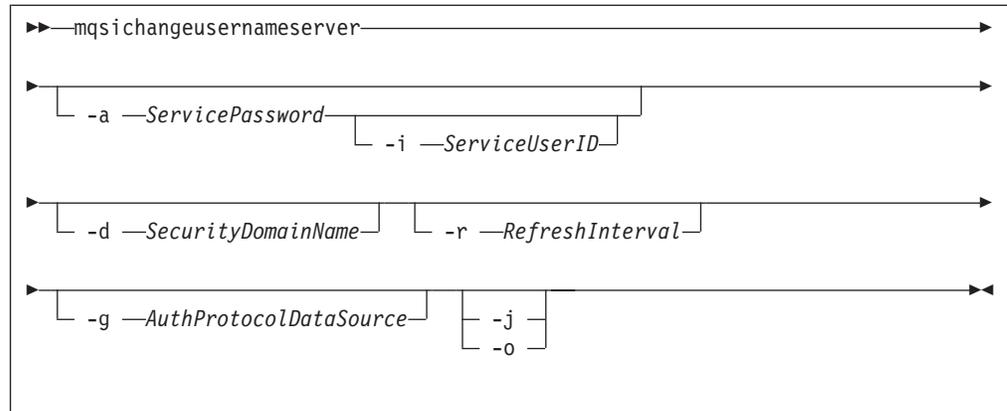
- On Windows, Linux, and UNIX systems, use the **mqsistop** and **mqsistart** commands.
- On z/OS, you must have started the original User Name Server control process using the /S option. You must stop the User Name Server components using the /F User Name Server, PC option and start the User Name Server components again using the /S User Name Server, SC option.

See “mqsistop command” on page 531 and “mqsistart command” on page 525 for more information.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsichangeusernameserver command - Windows”
- “mqsichangeusernameserver command - Linux and UNIX systems” on page 406
- “mqsichangeusernameserver command - z/OS” on page 407

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 593.

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.

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.

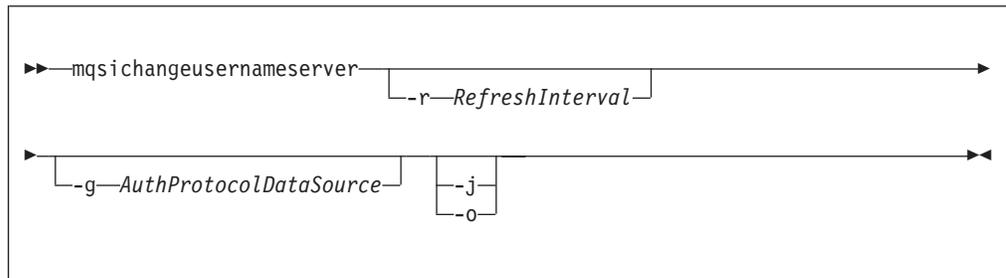
Examples:

`mqsichangeusernameserver -r 2000`

mqsichangeusernameserver command - z/OS:

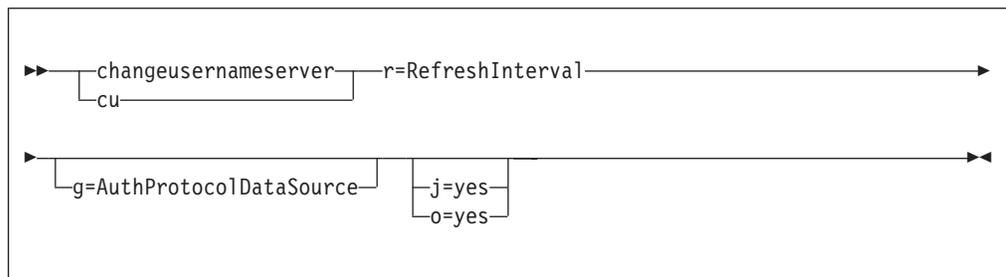
Syntax:

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

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 547

Purpose:

Use the **mqsiclearmqpubsub** command to remove an WebSphere MQ Publish/Subscribe broker as a neighbor of this WebSphere Message Broker broker.

This command removes knowledge of the WebSphere MQ Publish/Subscribe broker from the WebSphere Message Broker broker identified on this command. To complete this action you must also issue the WebSphere MQ Publish/Subscribe command **clrmqbrk** against the WebSphere MQ Publish/Subscribe broker. When both clear commands have completed, all publish/subscribe traffic between the two brokers ceases.

Use this command only if you are integrating this WebSphere 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:

```
►►—mqsiclearmqpubsub—BrokerName— -n —NeighborQueueManagerName—►►
```

Parameters:

BrokerName

(Required) The name of the broker from which knowledge of an WebSphere MQ Publish/Subscribe neighbor broker is to be removed.

-n *NeighborQueueManagerName*

(Required) The name of the queue manager that hosts the WebSphere MQ Publish/Subscribe broker for which the association as a neighbor is being removed.

Authorization:

The user ID used to invoke this command must have put and inq authority to the queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Examples:

```
mqsiclearnmqpubsub WBRK_BROKER -n MQBroker1
```

mqsicreateaclentry command

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 550

Purpose:

Use the **mqsicreateaclentry** command to create or modify the Configuration Manager data relating to the group or user access control lists that you have defined.

If you create or modify an access control group you must stop and restart the Configuration Manager for the change to take effect.

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 so that 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” on page 410
- “mqsicreateaclentry command - Linux and UNIX systems” on page 412
- “mqsicreateaclentry command - z/OS” on page 414

Authorization:

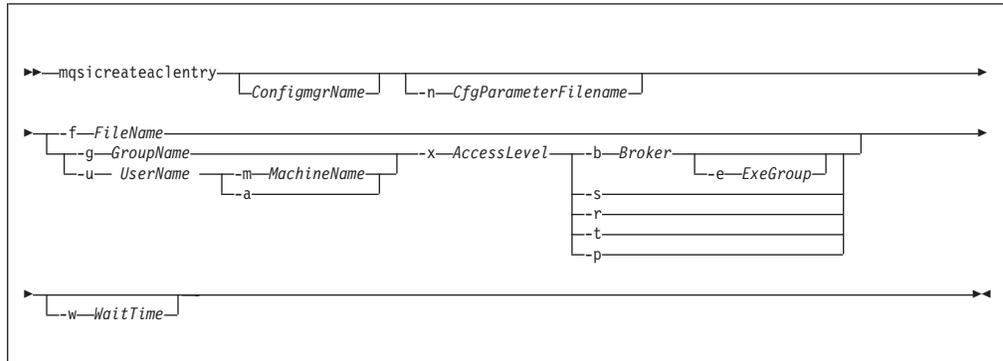
The user ID used to invoke this command must have full control permissions for the object being changed; see “ACL permissions” on page 591 for more information. In addition, for Linux and UNIX systems, the user ID must be a member of **mqbrkrs**.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. This means that the commands inherit a Full Control root ACL and you can carry out any operation.

If you submit a console command to the Configuration Manager you can change any ACL for that Configuration Manager.

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*

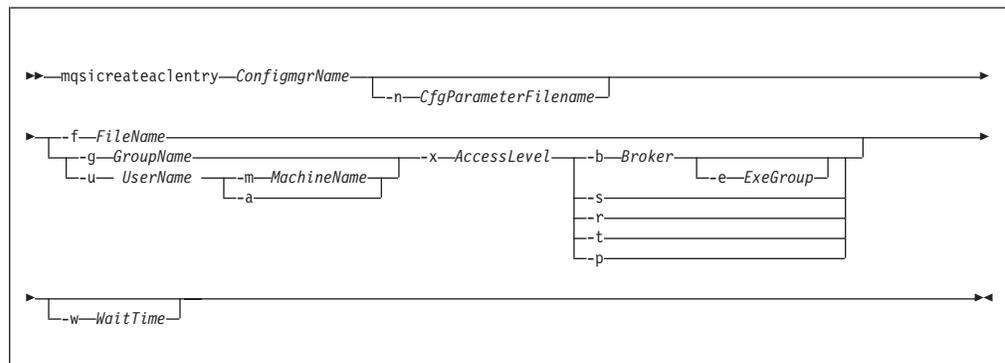
(Optional) The time in seconds that the command waits for a response from the Configuration Manager. If you do not supply a value, the command waits for 30 seconds.

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.

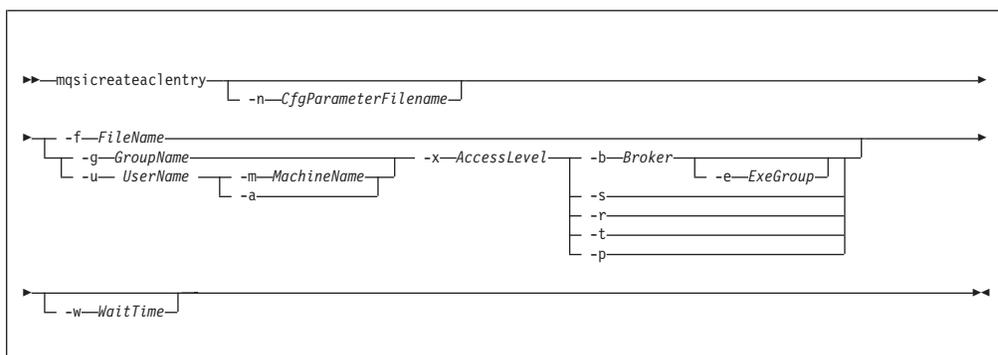
Examples:

```
mqsicreateaclentry CMGR01 -f c:\test\mylist
mqsicreateaclentry CMGR01 -g GROUPA -x F -b MYBROKER
```

mqsicreateaclentry command - z/OS:

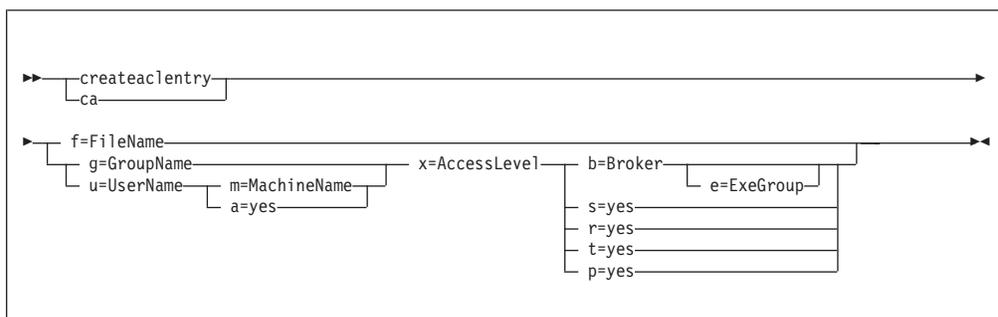
Syntax:

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 as:

```
<?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`, `mqsdeleteaclentry`, and `mqslistaclentry` 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.

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 410:

```
/f CMGR01,ca g='GROUPA' ,x='F' ,b='MYBROKER'
```

mqsicreatebroker command

Supported platforms:

- **Windows** Windows
- **Linux** **UNIX** Linux and UNIX systems
- **z/OS** z/OS. Run this command by customizing and submitting BIPCRBK; see "Contents of the broker PDSE" on page 547

Purpose:

The **mqsicreatebroker** command:

- **Windows** **Linux** **UNIX** Creates a WebSphere MQ queue manager, except on z/OS, if one does not already exist.
 1. If a WebSphere MQ queue manager is created as a result of using the `mqsicreatebroker` command, the default dead letter queue (DLQ) provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is automatically enabled. The security settings are the same as those of other broker-specific WebSphere MQ queues.

If you want to create the queue manager separately, you must set up a DLQ. The DLQ is referenced by WebSphere Message Broker when errors occur processing messages in message flows.

If a message, in either a user-defined message flow, or in the publish/subscribe model, cannot be processed, it is routed to this DLQ as a last resort. If you want to back out the message onto the input queue, effectively halting the message flow until the problem is resolved, disable the DLQ.

The `mqsdeletebroker` command does not delete the default DLQ (unless the queue manager is deleted).

2.

If you use a WebSphere MQ queue manager that has been created independently of the `mqscreatebroker` command, you might want to define clusters, which simplifies your configuration in most cases.

- **Windows** **Linux** **UNIX** Starts the WebSphere MQ queue manager, except on z/OS, if the WebSphere MQ queue manager is not already running. If the queue manager is created by using this command, it is not started as a Windows service; therefore the queue manager will stop if you log off. You must, 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.
- Creates the broker-specific WebSphere MQ queues, if they do not already exist.
- Creates database tables for the broker, if they do not already exist, or adds rows specific to this broker to existing database tables.
- **Windows** Installs a service under which the broker runs (on Windows platforms only).
- Creates a record for the component in the broker registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- **Windows** **Linux** **UNIX** “`mqscreatebroker` command - Windows, Linux, and UNIX systems” on page 419
- **z/OS** “`mqscreatebroker` command - z/OS” on page 424

Authorization:

Windows On Windows systems, the user ID used to start this command must have **Administrator** authority on the local system.

Linux **UNIX** On Linux and UNIX systems, the user ID used to start this command must be a member of both the **mqbrkrs** group and the **mqm** group.

z/OS On z/OS systems, the user ID used to start 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. The setting of `LdapPrincipal` and `LdapCredentials` options on `mqscreatebroker` is not required for correct operation of the broker. The password is not stored in clear text in the file system.

WebSphere MQ queues created:

- SYSTEM.BROKER.ADMIN.QUEUE

- SYSTEM.BROKER.CONTROL.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.QUEUE
- SYSTEM.BROKER.EXECUTIONGROUP.REPLY
- SYSTEM.BROKER.INTERBROKER.QUEUE
- SYSTEM.BROKER.INTERBROKER.MODEL.QUEUE
- SYSTEM.BROKER.MODEL.QUEUE
- SYSTEM.BROKER.WS.INPUT
- SYSTEM.BROKER.WS.REPLY
- SYSTEM.BROKER.WS.ACK

Access authority is granted for the WebSphere 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.

- **Windows** 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 recreate 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:\sql1lib\bnd\@db2cli.lst grant public
db2 connect reset
```

where X: is the drive on which DB2 is installed.

- **Linux** **UNIX** 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>/sql1lib/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.”
- 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 any 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:

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 333.

-i serviceUserId

The user ID under which components run. You can specify the *ServiceUserID* in any valid user name syntax. On Windows systems, valid formats are:

- domain\username
- \\server\username
- .\username
- username

On Linux and UNIX systems, only the last format, username, is valid.

If you use the unqualified form for this user ID (username) on Windows 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 593, and in “Security requirements for Linux and UNIX platforms” on page 592.

-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 either the WebSphere MQ Explorer, or the WebSphere MQ Services snap-in, depending on which version of WebSphere MQ you are using.

The queue manager attribute **MAXMSGLN** (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 333.

-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 Linux and UNIX platforms” on page 592
- “Security requirements for Windows platforms” on page 593

-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, DB2 grants WebSphere Message Broker the privileges of the *serviceUserId*, which results in a database connection as “already verified”. If you specify **-a** as an empty string as well as **-u** and **-p**, WebSphere Message Broker stores no passwords.

-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.

-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 is 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 terminates 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 re-create the broker.

- t (Optional) The broker runs as a WebSphere MQ trusted application.

If you specify this parameter on Windows systems, add the `ServiceUserID` (identified by `-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 *WebSphere MQ Intercommunication*.

- 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 LILs (loadable implementation libraries) for user-defined message processing nodes.

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 *configurationTimeout*

(Optional) This parameter defines the length of time (in seconds) that an execution group in the broker can take to apply a change in configuration (for example, an update that you have deployed from the workbench).

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.

The value that you set for this timeout depends on the system load (including CPU utilization), and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives an indication of the minimum value to set.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response.

-k *configurationDelayTimeout*

(Optional) The length of time (in seconds) that a broker can take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).

This parameter represents the time that it takes for a broker and its execution groups to process a minimal deployed configuration message; it depends on queue manager network delays, the load on the broker's queue manager, and system load.

You can estimate this value by issuing a command to request a simple configuration change, for example:

```
mqsi reporttrace brokerName -e "Execution Group Name" -u
```

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.

If the broker is on a production system, increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

-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 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 WebSphere Message Broker statistics and accounting is notified that archive records are to be output. The valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the operating system has an external method of notification 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. 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.

Examples:

```
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw -q WBRK_QM -s WBRK_UNQ_QM -n WBRKKBDB
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw -q WBRK_QM -n WBRKKBDB -t
mqsicreatebroker WBRK_BROKER -i wbrkuid -a wbrkpw -q WBRK_QM -n WBRKKBDB -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 333.

-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 either the WebSphere MQ Explorer, or the WebSphere MQ Services snap-in, depending on which version of WebSphere MQ you are using.

The queue manager attribute MAXMSGLN (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 333.

-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 LILs (loadable implementation libraries) for user-defined message processing nodes.

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 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 *ConfigurationTimeout*

(Optional) This parameter defines the length of time (in seconds) that an execution group in the broker can take to apply a change in configuration (for example, an update that you have deployed from the workbench).

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.

The value that you set for this timeout depends on the system load (including CPU utilization), and on each execution group's load. You can make an initial estimate by deploying the broker's entire configuration. The time taken for this to complete successfully gives an indication of the minimum value to set.

Specify the value in seconds, in the range 10 to 3600. The default is 300.

The sum of the *ConfigurationTimeout* and the *ConfigurationDelayTimeout* represents the maximum length of time that a broker can take to process a deployed configuration message before it generates a negative response.

-k *ConfigurationDelayTimeout*

(Optional) The length of time (in seconds) that a broker can take to process a minimal change in configuration (for example, an update that you have deployed from the workbench).

This parameter represents the time that it takes for a broker and its execution groups to process a minimal deployed configuration message; it depends on queue manager network delays, the load on the broker's queue manager, and system load.

You can estimate this value by issuing a command to request a simple configuration change, for example:

```
F MQP1BRK,reporttrace u=yes,e='exgrp1'
```

The name is case sensitive; enclose names in single quotation marks if they are in mixed case.

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.

If the broker is on a production system, increase the values for both *ConfigurationTimeout* and *ConfigurationDelayTimeout* to allow for application messages that are currently being processed by message flows to be completed before the configuration change is applied.

If the broker is on a development or test system, you might want to reduce time-outs (in particular, the *ConfigurationTimeout*) to improve perceived response times, and to force a response from a broker that is not showing expected behavior. However, reducing the timeout values decreases the probability of deploying a configuration change successfully.

-v *StatisticsMajorInterval*

(Optional) Specify the interval (in minutes) at which WebSphere Message Broker statistics and accounting is notified that archive records are to be output. The valid range is from 10 to 14400 minutes.

An interval of zero minutes indicates that the operating system has an external method of notification 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 an entire broker on z/OS:

```
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

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 550

Purpose:

This command completes the following actions:

- Creates a WebSphere MQ queue manager, except on z/OS, if one does not already exist.

Notes:

1. If a WebSphere MQ queue manager is created as a result of using the **mqsicreateconfigmgr** command, the default dead letter queue (DLQ) provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is automatically enabled. The security settings are the same as those of other broker-specific WebSphere MQ queues.

If you choose to create the queue manager separately, set up a dead-letter queue (DLQ); the DLQ is referenced by WebSphere Message Broker when errors occur processing messages in message flows.

If a message in either a user-defined message flow or in the publish/subscribe model cannot be processed, it is routed to this DLQ as a last resort. If you would prefer the message to be backed out onto the input queue, effectively halting the message flow until the problem is resolved, disable the DLQ.

The **mqsdeleteconfigmgr** command does not delete this queue (unless the queue manager is deleted).

2. If you are using a WebSphere MQ queue manager that has been created independently of the **mqsicreateconfigmgr** command, you can define clusters if you choose. This option simplifies your configuration.

- Starts the WebSphere MQ queue manager, if it is not already running.
- Creates the WebSphere MQ queues and channel that are specific to the Configuration Manager, if they do not already exist.
- Creates database tables for the Configuration Manager in its internal repository. If you need to transfer data from the configuration repository of an earlier release, you can use the *db2DatabaseToMigrate*, *migrationDatabaseUserId*, and *migrationDatabasePassword* parameters.
- If you run this command using the *-n* parameter and then delete the Configuration Manager using **mqsdeleteconfigmgr**, 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.

- (Windows only) - installs a Windows service, under which the Configuration Manager runs.
- Creates a record for the component in the broker registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsicreateconfigmgr command - Windows”
- “mqsicreateconfigmgr command - Linux and UNIX systems” on page 431
- “mqsicreateconfigmgr command - z/OS” on page 433

Authorization:

This command changes security privileges for the ServiceUserID; the user ID used to invoke this command must be a member of the Windows **Administrators** group on this local system.

On UNIX systems, the user ID used to invoke this command must be a member of the mqbrkrs group.

On z/OS systems, the user ID used to invoke this command must be a member of a group that has READ and WRITE access to the component directory 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:

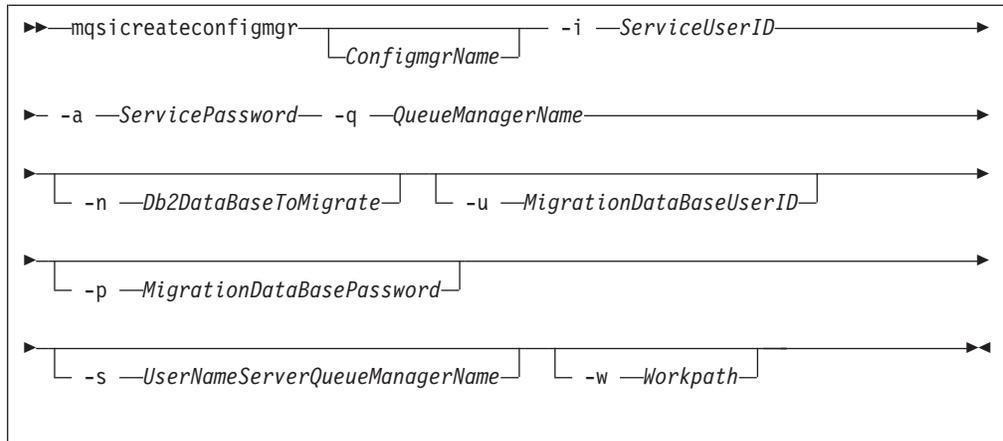
- SYSTEM.BKR.CONFIG

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 593.

-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*

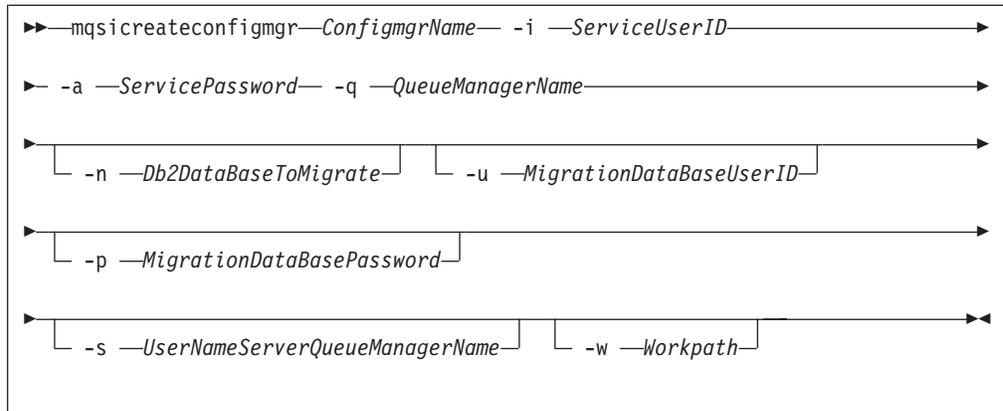
(Optional) The directory in which working files for the Configuration Manager are stored. If not specified, the default directory specified when the product was installed is used.

Examples:

```
mqscreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_CONFIG_QM
```

mqscreateconfigmgr 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 592.

-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.

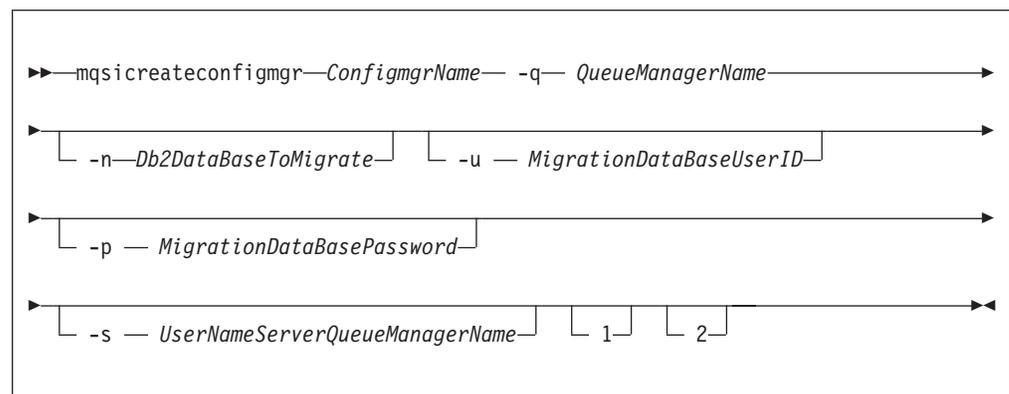
Examples:

```
mqsicreateconfigmgr CMGR01 -i wbrkuid -a wbrkpw -q WBRK_CONFIG_QM
```

mqsicreateconfigmgr command - z/OS:

Syntax:

z/OS command - BIPRCM:



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, or FTP server.

Supported platforms:

- Windows systems
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPCRCS; see “Contents of the broker PDSE” on page 547

Purpose:

Use this command to add a predefined IBM resource type.

For configurable services that you add using the `mqsicreateconfigurableservice` command:

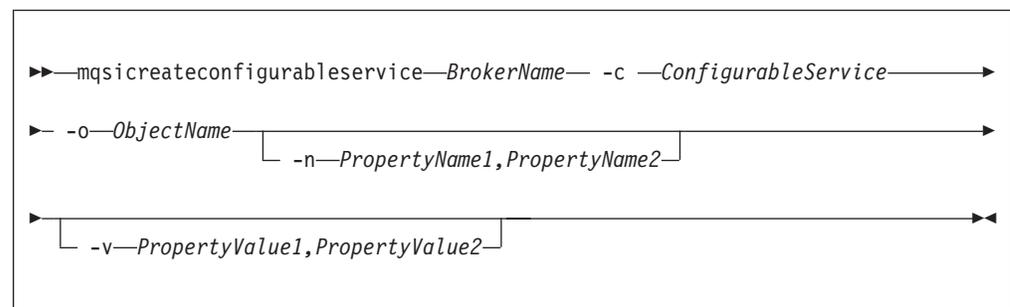
- Use the `mqsireportproperties` command to view the configurable services.
- Use the `mqsichangeproperties` command to modify the configurable services.
- Use the `mqsideleteconfigurableservice` command to delete configurable services.

You do not need to use the `mqsicreateconfigurableservice` command to create EIS providers, because they are already defined. Use the `mqsichangeproperties` command to modify them.

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:

- JMSProviders
- JDBCProviders
- SecurityProfiles
- FtpServer

-o *ObjectName*

(Required) The name of the object whose properties you want to change.

If the **-c** parameter is set to JMSProviders, the expected object name is either an IBM-defined JMS provider name, or a user-defined JMS provider name.

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.

-n *PropertyName*

(Optional) The name of the property that is being changed. For a JMSProviders resource, valid property names are:

- jarsURL
- nativeLibs

For a JDBCProviders resource, valid property names are:

- databaseType
- databaseVersion
- type4DriverClassName
- connectionUrlFormat
- connectionUrlFormat Attr1-5
- databaseName
- description
- jarsURL
- portNumber
- serverName
- securityIdentity

For a SecurityProfiles resource, valid property names are:

- authentication = {NONE, LDAP, TFIM}
- authenticationConfig = string
- mapping = {NONE, TFIM}
- mappingConfig = string
- authorization = {NONE, LDAP, TFIM}
- authorizationConfig = string
- propagation = {TRUE, FALSE}
- passwordValue = {PLAIN, MASK, OBFUSCATE}

For an FtpServer resource, valid property names are:

- serverName (this is required if the **-c** parameter is set to FtpServer)
- securityIdentity
- accountInfo
- remoteDirectory
- connectionType
- scanDelay
- transferMode

See FtpServer configurable service properties for details about these property names.

-v *PropertyValue*

(Optional, but required if **-n** 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];"
```

If you set the **-c** parameter to JMSProviders, and the **-n** parameter to jarsURL, the expected value is a URL that specifies the file location of the JMS provider JAR files, while omitting the file:// of the URL. If you do not specify the **-n** parameter, the default location for the JMS provider jar files is the broker's shared-classes directory.

If you set the **-c** parameter to JMSProviders, and the **-n** parameter to nativeLibs, the expected value is a file location of any JMS provider native libraries. If you do not specify the **-n** parameter, the default location for any JMS provider native libraries is the broker's LibPath.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to authentication, the property value can be NONE, LDAP, TFIM, or a user-defined value to specify an alternative.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to authenticationConfig, the expected property value is a provider-specific configuration string.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to mapping, the property value can be NONE, TFIM, or a user-defined value to specify an alternative.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to mappingConfig, the expected property value is a provider-specific configuration string.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to authorization, the property value can be NONE, LDAP, TFIM, or a user-defined value to specify an alternative.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to authorizationConfig, the expected value is a provider-specific configuration string.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to propagation, the property value can be either TRUE or FALSE.

If you set the **-c** parameter to SecurityProfiles, and the **-n** parameter to passwordValue, the property value can be PLAIN, MASK, or OBFUSCATE.

See FtpServer configurable service properties for details about the values that you can set for the properties of an FtpServer configurable service.

Authorization:

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:

Add a JMS provider called MyProviderXYZ for broker WBRK6_DEFAULT_BROKER:

```
mqsicreateconfigurableservice WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_MyProviderXYZ
```

Add a JMS provider called ProviderABC for broker WBRK6_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 WBRK6_DEFAULT_BROKER specifying a location for the provider's jar files, and a library path for the native 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
```

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 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
```

mqsicreatedb command

Supported platforms:

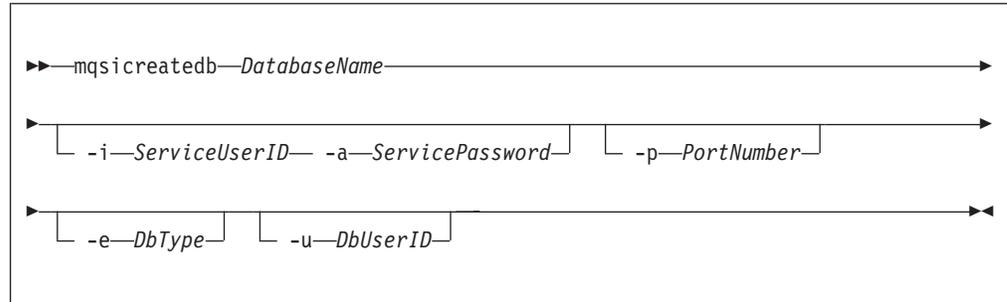
- Windows

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. This must be the first parameter, and is case sensitive. Restrictions might be placed on the permissible length of the database name by the database engine. For restrictions on the character set that can be used, see “Characters allowed in commands” on page 333.

-i ServiceUserID

(Optional, Derby only) The user ID under which the DatabaseInstanceMgr service runs.

This can be specified in any valid username syntax:

- domain\username
- \\server\username
- .\username
- username

If you use the unqualified form for this user ID (username), the operating system searches for the user ID throughout its domain, starting with the local system. This search might take some time to complete.

The ServiceUserID specified must be a member of the local group mqbrkrs. The ID can be a direct or indirect member of the group. The ServiceUserID must also be authorized to access the home directory (where WebSphere Message Broker is installed).

This parameter is ignored if the database engine specified or defaulted for the command is DB2. A ServiceUserID is required for Derby, but only for the first invocation of this command. Subsequent invocations are associated with the existing Windows DatabaseInstanceMgr service that runs under the ServiceUserID specified on the earlier command

The security requirements for the ServiceUserID are detailed in “Security requirements for Windows platforms” on page 593.

-a ServicePassword

(Optional, Derby only) The password for the ServiceUserID. Specify this only if you specify ServiceUserID.

- p** *PortNumber*
(Optional) The TCP/IP port number that this component will use on the local machine. If not specified, the default value 1527 is used.
- e** *DbType*
(Optional) The database engine to be used to create and run the database. Currently supported values are DB2 and Derby. If you do not specify this option, and only one database engine is available, that engine is used. If both are available, the default engine is DB2.
- u** *DbUserID*
(Optional, DB2 only) An additional user name that requires access to the database that is created by this command.

Authorization:

The user ID used to invoke this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Examples:

The following example sets up a database with the name brokerdb on port 1600:
mqsicreatedb brokerdb -p 1600

The following example sets up a Derby database with the name derbydb, using port number 1527:
mqsicreatedb derbydb -i wbrkuid -a wbrkpw -e Derby -p 1527

mqsicreateexecutiongroup command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPCREG; see “Contents of the Configuration Manager PDSE” on page 550

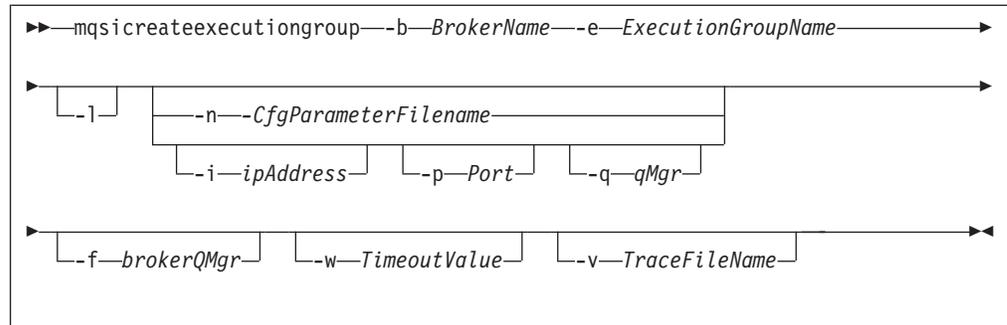
Purpose:

Use the **mqsicreateexecutiongroup** command to add a new execution group to a broker.

You must start the Configuration Manager before you can issue this command.

The broker must be defined in the Configuration Manager before this command can be used.

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.

-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 cr1NameList="" 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 as:

```

<?xml version="1.0"?>
<configmgr cr1NameList="" 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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *Port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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 platforms, Linux, and UNIX systems, you need to be a member of the 'mqm' group and the command only succeeds if the user ID running the command has the correct authority defined in the Configuration Manager's access control list.

In order to create an execution group, full control authority is required over the broker object; see "ACL permissions" on page 591 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

- 0 (Success) States that the request completed successfully and the execution group has been created successfully in the Configuration Manager's repository. The next time the broker is deployed, the new execution group list is initialized on the broker.
- 2 (Failure) States that the execution group could not be created for any reason.
- 98 States that the Configuration Manager cannot be reached.
- 99 States that the supplied arguments to the utility are not valid.

Examples:

On the domain controlled by the Configuration Manager whose queue manager is called QMGR and is listening on fred.abc.com:1414, create an execution group called 'EG1' on broker 'BROKER'.

```
mqsicreateexecutiongroup -i fred.abc.com -p 1414 -q QMGR -b BROKER -e EG1
```

On the domain specified by the file domain1.configmgr, create an execution group called 'EG2' on broker 'BROKER'.

```
mqsicreateexecutiongroup -n domain1.configmgr -b BROKER -e EG2
```

On the domain specified by the file domain2.configmgr, create an execution group 'EG3' on broker 'FRED'. Wait five minutes for the Configuration Manager to respond and send output to trace.txt.

```
mqsicreateexecutiongroup -n domain2.configmgr -b FRED -e EG3 -w 300 -v trace.txt
```

mqsicreateusername server command

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 549

Purpose:

The **mqsicreateusernameserver** command:

- Creates a WebSphere MQ queue manager, except on z/OS, if one does not already exist.
 1. If a WebSphere MQ queue manager is created as a result of using the **mqsicreateusernameserver** command, the default DLQ provided by WebSphere MQ (SYSTEM.DEAD.LETTER.QUEUE) is automatically enabled. The security settings are the same as those of other broker-specific WebSphere MQ queues.

If you choose to create the queue manager separately, set up a dead letter queue (DLQ). The DLQ is referenced by WebSphere Message Broker when errors occur processing messages in message flows.

If a message in either a user-defined message flow or in the publish/subscribe model cannot be processed, it is routed to this DLQ as a last resort. If you would prefer the message to be backed out onto the input queue, effectively halting the message flow until the problem is resolved, disable the DLQ.

The **mqsdeleteusernameserver** command does not delete this queue (unless the queue manager is deleted).

2.

If you are using a WebSphere MQ queue manager that has been created independently of the **mqsicreateusernameserver** command, you can define clusters. This simplifies your configuration.

- Starts the WebSphere MQ queue manager, if this is not already running.

If the queue manager is created by this command, it is not started as a Windows service; it stops if you log off. To avoid this happening, either remain logged on, or change the start up status of the queue manager service. (If you lock your workstation, the WebSphere MQ queue manager does not stop).
- Creates the User Name Server-specific WebSphere MQ queues, if these do not already exist.
- On Windows, installs a service under which the User Name Server runs.
- Creates a record for the component in the broker registry.

On z/OS, this command:

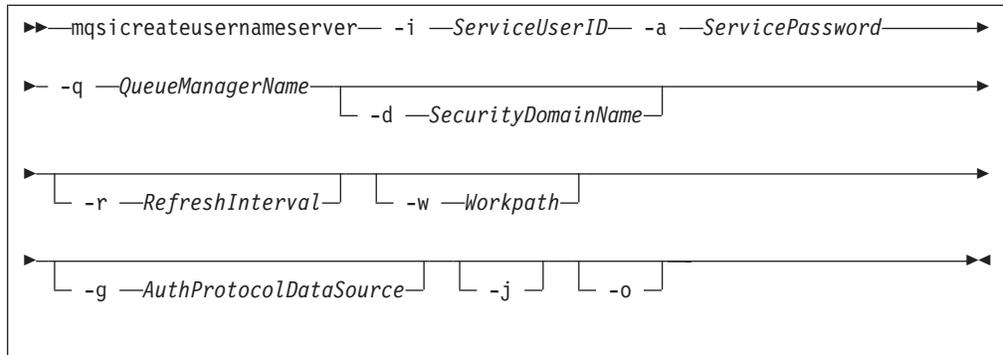
- Creates a WebSphere MQ queue manager, if one does not already exist.
- Creates the User Name Server-specific WebSphere MQ queues, if these do not already exist.
- Creates a record for the component in the broker registry.

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 445
- “mqsicreateusernameserver command - z/OS” on page 447

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 593.

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.

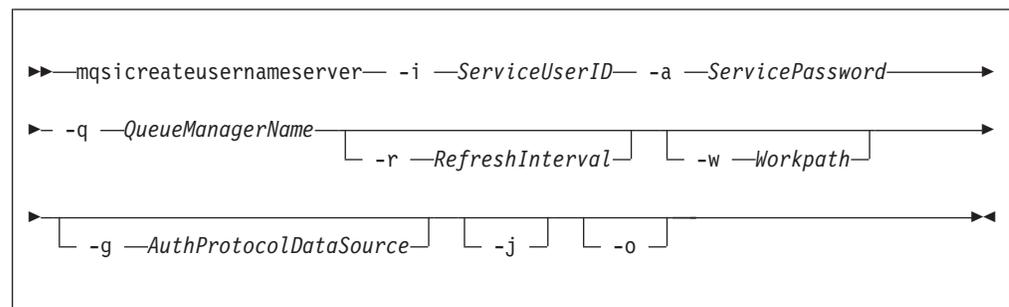
-o (Optional) Indicates that groups and group memberships are drawn from the operating system, rather than being defined in the data source for the authentication protocol.

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 592. 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.

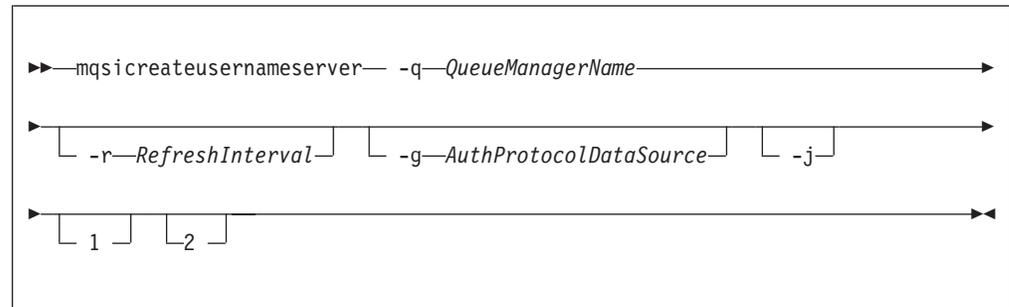
Examples:

```
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:

```
mqsicreateusername -i wbrkuid -a wbrkpw  
-q WBRK_QM -r 1000
```

mqsdeleteaclentry command

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 550

Purpose:

Use the **mqsdeleteaclentry** command to delete a Configuration Manager access control list entry that you have defined.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsdeleteaclentry command - Windows”
- “mqsdeleteaclentry command - Linux and UNIX systems” on page 450
- “mqsdeleteaclentry command - z/OS” on page 452

Authorization:

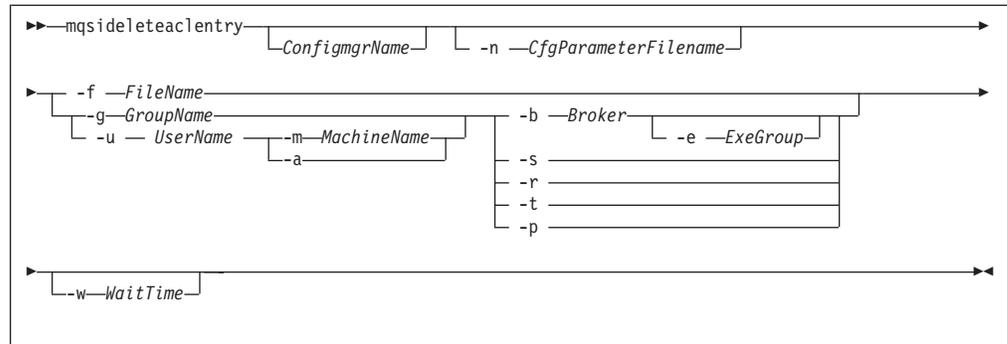
The user ID used to invoke this command must have full control permissions for the object being changed; see “ACL permissions” on page 591 for more information. In addition, for Linux and UNIX systems, the user ID must be a member of mqbrkrs.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. This means that the commands inherit a Full Control root ACL and you can carry out any operation.

If you submit a console command to the Configuration Manager you can change any ACL for that Configuration Manager.

mqsdeleteaclentry 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.

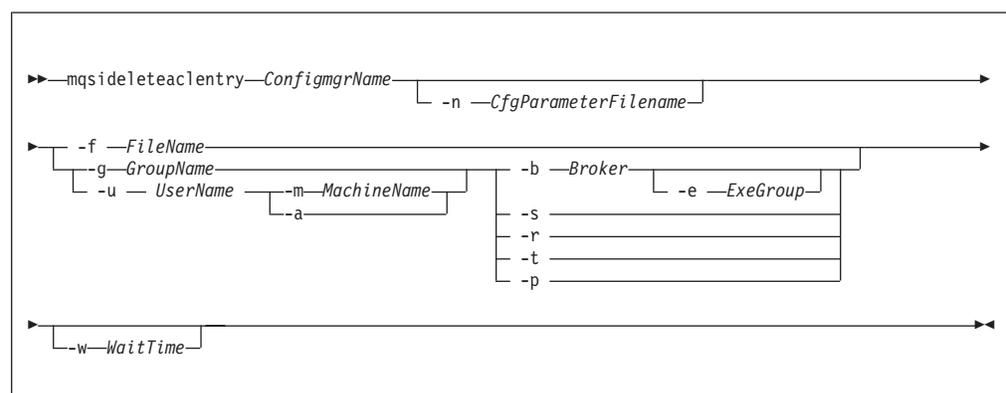
- m** *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.
- a** (Optional) The specified user name can be on any machine. This option can not be used with **-m**.
- b** *Broker*
(Optional) The object is a broker object, and 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, mqsdeleteaclentry, 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.

Examples:

```
mqsdeleteaclentry CMGR01 -f c:\test\mylist
mqsdeleteaclentry CMGR01 -g GROUPA -b MYBROKER
```

mqsdeleteaclentry 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.

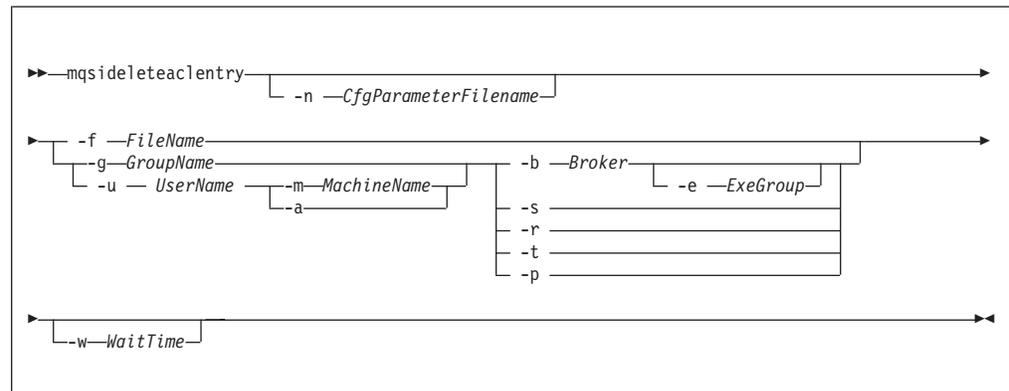
Examples:

```
mqsdeleteaclentry CMGR01 -f c:\test\mylist
mqsdeleteaclentry CMGR01 -g GROUPE -b MYBROKER
```

mqsdeleteaclentry command - z/OS:

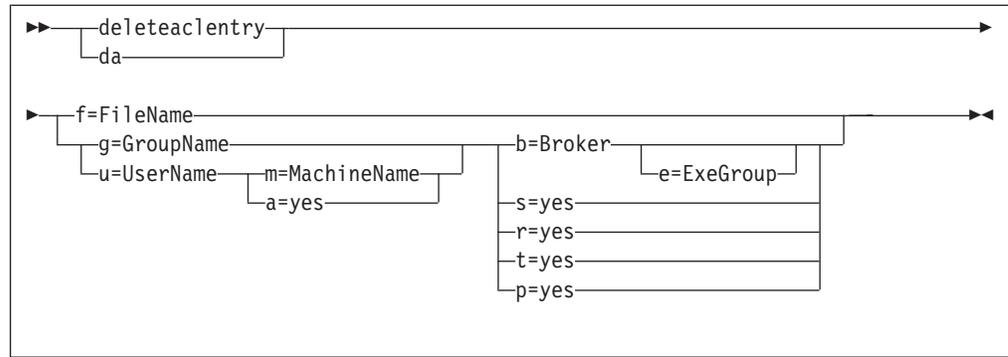
Syntax:

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 as:

```
<?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.

Examples:

```
mqsdeleteaclentry 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'
```

mqsdeletebroker command

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 547

Purpose:

Use the **mqsdeletebroker** command to delete a named broker. The command also deletes the queues on the broker's local queue manager (created when the broker was created), and its data in the broker database. You can also specify that the queue manager is to be deleted.

Even though this command deletes all the data related to this broker from the broker database tables, it does not check if the tables are empty, and it does not delete the tables.

This command:

- On Windows platforms, stops the service that runs the broker.
- Stops and deletes the WebSphere MQ queue manager for the broker, if requested.
- Removes the broker's data from the database.
- Removes the record for the component in the broker registry.

If you delete a broker that has WebSphere MQ publish/subscribe broker neighbors, you must also invoke the runmqsc command **clrmqbrk** at each of these neighbors. You also need to specify the WebSphere Message Broker broker that you are deleting with this command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

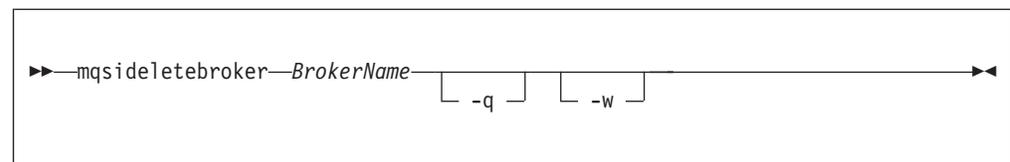
- "mqsdeletebroker command - Windows, Linux and UNIX systems"
- "mqsdeletebroker command - z/OS" on page 456

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.

mqsdeletebroker command - Windows, Linux and UNIX systems:

Syntax:



Parameters:

BrokerName

(Required) The name of the broker that you want to delete. This must be the first parameter.

- q (Optional) Specifies that the broker's queue manager is deleted. (If this option is not specified, 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.

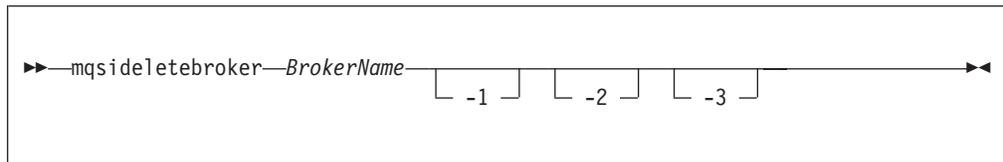
Examples:

```
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:

To delete the broker registry on z/OS for the broker CSQ1BRK, use the following command:

```
mqsideletebroker CSQ1BRK -1
```

mqsideleteconfigmgr command

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 550

Purpose:

This command:

- On the Windows platform, stops the service that runs the Configuration Manager.
- Stops and deletes the WebSphere MQ queue manager for the Configuration Manager, if requested.
- Removes the tables from the configuration repository, if requested.
- Removes the record for the component in the broker registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

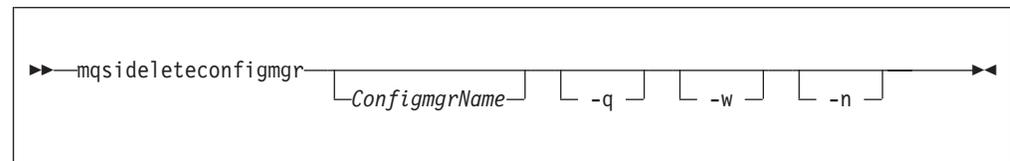
- “mqsideleteconfigmgr command - Windows”
- “mqsideleteconfigmgr command - Linux and UNIX systems” on page 458
- “mqsideleteconfigmgr command - z/OS” on page 459

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.

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:

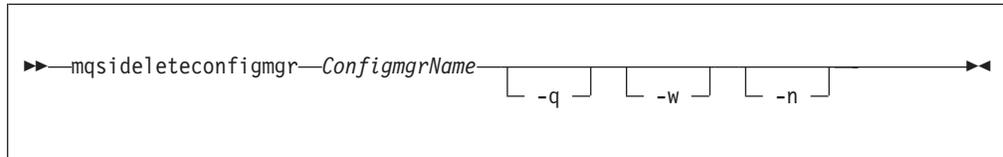
1. Locate the Configuration Manager's working directory. The default location is C:\Documents and Settings\All Users\Application Data\IBM\MQSI
2. Locate the components directory. Delete the directory with the same name as the Configuration Manager.

Examples:

```
mqsdeleteconfigmgr CMGR01 -q
```

mqsdeleteconfigmgr 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.

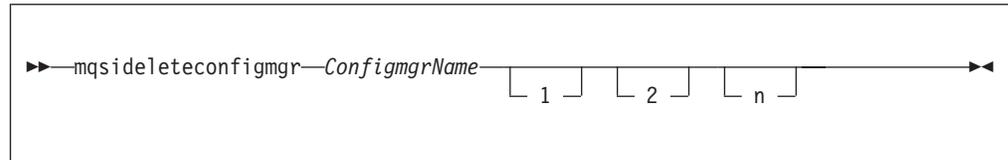
Examples:

```
mqsdeleteconfigmgr CMGR01 -q
```

mqsdeleteconfigmgr 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:

```
mqsdeleteconfigmgr CMGR01 -q
```

mqsideleteconfigurable service command

The `mqsideleteconfigurable service` command deletes a configurable service, such as a JMS provider, JDBC provider, or FTP server, that was created using the `mqsicreateconfigurable service` 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 547

Purpose:

Use the `mqsideleteconfigurable service` command to view the configurable services that you can delete.

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:

```
▶▶mqsideleteconfigurable service—BrokerName— -c —ConfigurableService—▶▶
▶▶-o—ObjectName—▶▶▶▶
```

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 type, such as JMSProviders, JDBCProviders, SecurityProfiles, or FtpServer. Use the `mqsideleteconfigurable service` command to view the list of valid values.

-o ObjectName

(Required) The name of the object for which you want to delete the properties. Use the `mqsideleteconfigurable service` command to view the list of properties that you can delete.

Authorization:

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 a JMS provider configurable service called MyProviderXYZ for broker WBRK6_DEFAULT_BROKER:

```
mqsdeleteconfigurable service WBRK6_DEFAULT_BROKER -c JMSProviders -o JMS_MyProviderXYZ
```

Delete a security profile for LDAP use:

```
mqsdeleteconfigurable service WBRK6_DEFAULT_BROKER -c SecurityProfiles -o MyLDAPProfile
```

Delete an FtpServer configurable service:

```
mqsdeleteconfigurable service WBRK6_DEFAULT_BROKER -c FtpServer -o Server01
```

mqsdeletedb command

Supported platforms:

- Windows

Purpose:

The **mqsdeletedb** removes a database that has been created by **mqscreatedb**. This includes deleting the database and removing the ODBC data source name. If this is the last 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. **Warning:** all the data stored in this database is permanently deleted.

-d DeleteAllDatabases

This must be the only parameter. Specify this option to delete all the databases that have been created by **mqscreatedb**. **Warning:** all the data stored in ALL databases is permanently deleted.

Authorization:

The user ID used to invoke this command must have Administrator authority on the local system and be part of the mqbrkrs group.

Examples:

The following example deletes the database brokerdb:

```
mqsdeletedb brokerdb
```

mqsdeleteexecutiongroup command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPDLEG; see “Contents of the Configuration Manager PDSE” on page 550

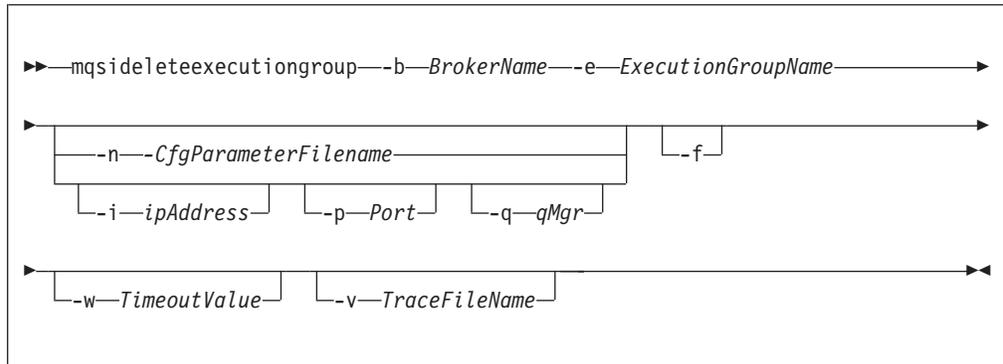
Purpose:

Use the **mqsdeleteexecutiongroup** command to delete an execution group from a broker.

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 as:

```
<?xml version="1.0"?>
<configmgr cr1NameList="" 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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *Port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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 platforms, Linux, and UNIX systems, you need to be a member of the 'mqm' group and the command only succeeds if the user ID running the command has the correct authority defined in the Configuration Manager's access control list.

In order to delete an execution group, full control authority is required over the broker object; see "ACL permissions" on page 591 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

- 0 (Success) States that the request completed successfully and the execution group has been deleted successfully. If the command was to delete an execution group to which a deployment has previously been made, this return code means that the broker has stopped and 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 any resources and send output to trace.txt.

```
mqsideleteexecutiongroup -n domain2.configmgr -b FRED -e EG3 -w 300 -v trace.txt
```

mqsideleteusernameeserver command

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 549

Purpose:

This command:

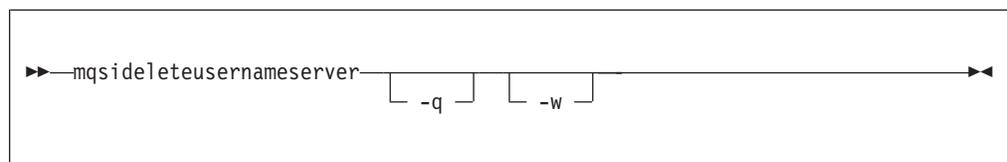
- Stops the service that runs the User Name Server.
- Stops and deletes the WebSphere MQ queue manager for the User Name Server, if requested.
- Removes the record for the component in the broker registry.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqsideleteusernameeserver command - Windows, Linux and UNIX systems”
- “mqsideleteusernameeserver command - z/OS” on page 465

mqsideleteusernameeserver 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.

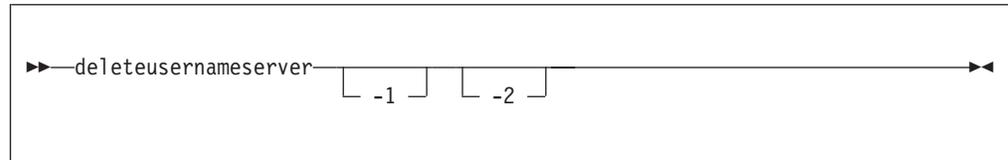
Examples:

```
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 550.

Purpose:

This command allows you to make the various flavors of deployment requests 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.

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 466
- “mqsideploy command - z/OS” on page 470

Authorization:

In order to successfully deploy, the user ID issuing the command needs to have sufficient authority defined in the Configuration Manager. The permissions required are the same as the permission required to do the equivalent function in the Message Broker Toolkit. For a list of permissions that can be defined in the Configuration Manager see “ACL permissions” on page 591.

UNIX If you are using UNIX you must have mqbrkr membership to run this command. If you do not have mqbrkr membership the command will fail 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'.
```

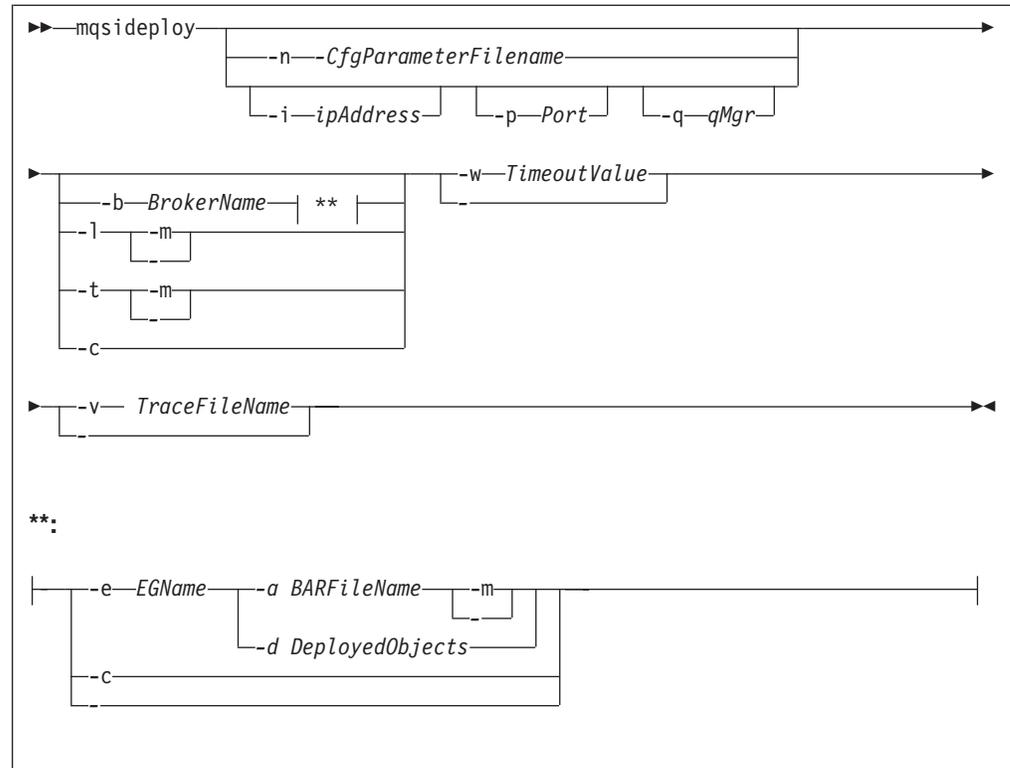
Responses:

This command returns the following responses:

- 0 (Success) States that the Configuration Manager issued the deployment request and all of the relevant brokers responded successfully before the timeout period expired.
- 2 (Failure) States that the Configuration Manager issued the deployment request and at least one broker responded negatively. See the messages output from the utility (or the Configuration Manager’s event log) for more information.
- 3 (Initiated) States that the Configuration Manager has replied, stating that deployment has started, but that no broker responses were received before the timeout occurred.
- 5 (Submitted) States that the deployment message was sent to the Configuration Manager , but that no response was received before the timeout occurred.
- 6 (SuccessSoFar) States that the Configuration Manager issued the deployment request and some, but not all, of the relevant brokers responded successfully before the timeout period expired; no brokers responded negatively.
- 98 States that the Configuration Manager cannot be reached.
- 99 States that the supplied arguments to the utility are not valid.

mqsideploy command - Windows, Linux, and UNIX systems:

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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *Port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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 *EGName*

(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 bar (broker archive) file that is to be used for message flow or message set deployment. 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 in conjunction 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 only changes 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 the **-m** parameter is also set; 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 there is no possibility that 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 utility waits for the broker to reply before returning control to the command line. The **mqsideploy** command polls the Configuration Manager log records, looking for the results of the 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 to wait before timing out, in the range 1 - 2 145 336 164. If no *timeoutValue* is provided, or a value less than 1 or greater than 2 145 336 164 is specified, an error is returned

-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 **-m** is not set, 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 **-m** is not set, 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 **-m** is not set, only changes to the topic tree are deployed to all brokers.
- For a *remove message flow or remove message set operation*, the **-m** parameter is ignored.

Examples:

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:

```
mqsidedeploy -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:

```
mqsidedeploy -i localhost -p 1414 -q QMNAME -m -w 600
```

Note that 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:

```
mqsidedeploy -n cm1.configmgr -t -m -w 600
```

Deploy a bar file using a connection file whose parameters are described in the file `cm1.configmgr`, to the specified broker, allow 10 minutes for the broker to reply, and remove all currently-deployed message flows and message sets from the execution group as part of the deployment:

```
mqsidedeploy -n cm1.configmgr -b broker1 -e default -a mybar.bar -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:

```
mqsidedeploy -n cm1.configmgr -b broker1 -w 900
```

Attempt to remove the message flow top and the dictionary bar from the execution group default on broker b1, using a connection file whose 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.)

```
mqsidedeploy -n cm1.configmgr -b B1 -e default -d top.cmf:bar.dictionary
```

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.

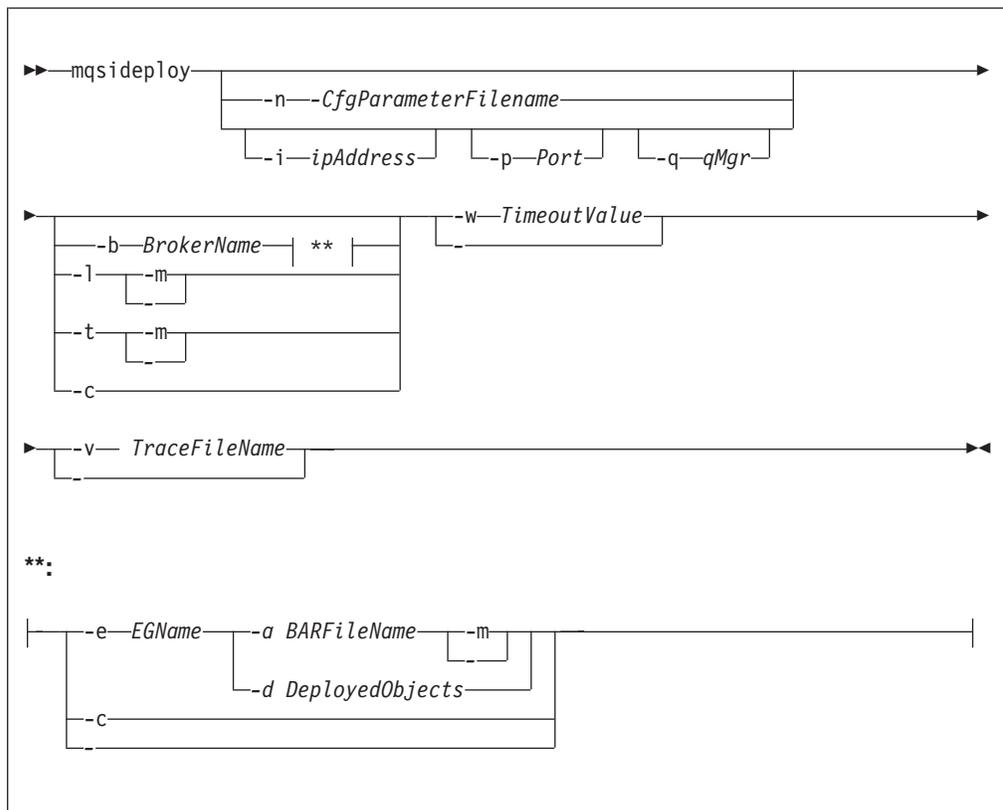
```
mqsidedeploy -n cm1.configmgr -c -w 900
```

mqsidedeploy command - z/OS:

BIPDPLY is used to run **mqsidedeploy**, and on z/OS the command does this from JAVA bindings; see "Usage note" on page 473.

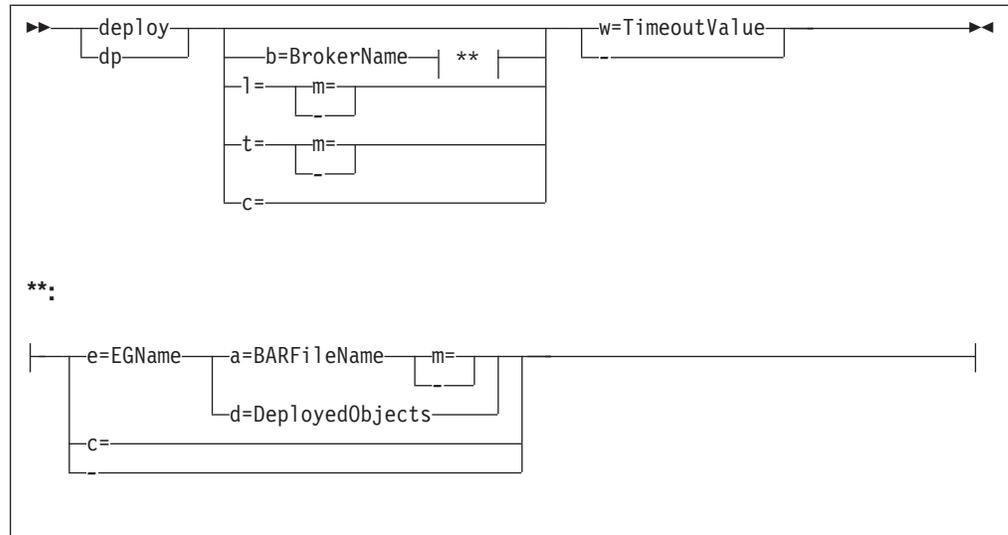
Syntax:

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 as:

```

<?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 this parameter is not specified, the default value "" 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 this parameter is not specified, 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 this parameter is not specified, the default value "" 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 *EGName*

(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 bar (broker archive) file that is to be used for message flow or message set deployment. You must also specify the **-b** and **-e** parameters with this option.

The bar file must be in the file system. The bar file can be anywhere in the file system, as long as the user ID of the person or 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 in conjunction 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 only changes 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 the **-m** parameter is also set; 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 there is no possibility that 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 utility waits for the broker to reply before returning control to the command line. The **mqsideploy** command polls the Configuration Manager log records, looking for the results of the 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 to wait before timing out, in the range 1 - 2 145 336 164. If no *timeoutValue* is provided, or a value less than 1 or greater than 2 145 336 164 is specified, an error is returned

-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 **-m** is not set, 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 **-m** is not set, 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 **-m** is not set, only changes to the topic tree are deployed to all brokers.
- For a *remove message flow or remove message set operation*, the **-m** parameter is ignored.

Usage note:

If you enter:

```
mqsideploy -i cm_name -p port -q cm_qm -l -m
```

the command attempts to run the **mqsideploy** command using WebSphere MQ Java client code, which is not allowed on z/OS. Connecting to the local Configuration Manager with the parameters **-i** and **-p** forces the command into a local mode, and the following error occurs:

```
BIP1046E: Unable to connect with the Configuration Manager's queue manager
```

Depending on the version of WebSphere MQ that you are using, the reported reason codes differ:

- On WebSphere MQ V5.3.1, you receive:
2012 0x000007dc MQRC_ENVIRONMENT_ERROR
- On WebSphere MQ V6.0, you receive:
2298 0x000008fa MQRC_FUNCTION_NOT_SUPPORTED

Examples:

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
```

Note that 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:

```
mqsidedeploy -n cm1.configmgr -t -m -w 600
```

Deploy a bar file using a connection file whose parameters are described in the file `cm1.configmgr`, to the specified broker, allow 10 minutes for the broker to reply, and remove all currently-deployed message flows and message sets from the execution group as part of the deployment:

```
mqsidedeploy -n cm1.configmgr -b broker1 -e default -a mybar.bar -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:

```
mqsidedeploy -n cm1.configmgr -b broker1 -w 900
```

Attempt to remove the message flow `top` and the dictionary `bar` from the execution group `default` on broker `b1`, using a connection file whose 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`).

```
mqsidedeploy -n cm1.configmgr -b B1 -e default -d top.cmf:bar.dictionary
```

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.

```
mqsidedeploy -n cm1.configmgr -c -w 900
```

mqsiformatlog 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 547

Purpose:

Use the **mqsiformatlog** command to process the XML log created by **mqsireadlog**. The command retrieves and formats any messages that the XML log contains into a form suitable for the locale of the user invoking the command.

This command interprets an input log file that has been created on any system in a platform-independent code page, `utf-8`. Use it to produce formatted output from input log files transferred from other systems to the system on which you issue the command. If you use this facility, ensure that you use a file transfer program that does not convert the data (for example, by specifying a binary transfer option).

You can direct the output to a file, or to the command shell.

Syntax:

```
▶▶ mqsiformatlog -i Inputfilename -o Outputfilename ▶▶
```

Parameters:

-i *Inputfilename*

(Required) The filename of the XML log file that is to be formatted. This file is created by the **mqsireadlog** command; it is encoded in utf-8.

-o *Outputfilename*

(Optional) The filename of the file into which the formatted log output is to be written. If this is not specified, the formatted log data is written to stdout.

Output written by this command (to file or stdout) is written in a code page suitable for the current user locale.

Authorization:

The user ID used to invoke this command must have read access to the input file, and write access to the output file.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Examples:

```
mqsiformatlog -i trace.xml -o formattrace.log
```

The following extract illustrates the output generated by this command:

Timestamps are formatted in local time, local time is GMT.

```
.  
. .  
2003-02-12 12:57:21.895999 388 UserTrace BIP2638E:  
MQPUT to queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager  
'WBRK_QM': MQCC=0, MQRC=0; node ConfigurationMessageFlow.outputNode'.  
The node 'ConfigurationMessageFlow.outputNode' attempted  
to write a message to the specified queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY'  
connected to queue manager 'WBRK_QM'.  
The MQCC was 0 and the MQRC was 0.  
No user action required.  
  
2003-02-12 12:57:21.895999 388 UserTrace BIP2622I:  
Message successfully output by output node 'ConfigurationMessageFlow.outputNode'  
to queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager  
'WBRK_QM'. The WebSphere MQ output node ConfigurationMessageFlow.outputNode'  
successfully wrote an output message to the specified queue  
SYSTEM.BROKER.EXECUTIONGROUP.REPLY connected to queue manager WBRK_QM.  
No user action required.  
. .  
Threads encountered in this trace: 335 388
```

mqsijoinmqpubsub command

Supported platforms:

- Windows
- Linux and UNIX systems

- z/OS. Run this command by customizing and submitting BIPJNMP; see “Contents of the broker PDSE” on page 547

Purpose:

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.

This is an asynchronous command. 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 **mqsilistmqpubsub** command to monitor the status of the asynchronous actions that result from this command.

Use this command only if you are integrating this WebSphere 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:

```

▶▶—mqsijoinmqpubsub—BrokerName— -p —ParentQueueManagerName————▶▶

```

Parameters:

BrokerName

(Required) The name of the broker that is to be joined to an WebSphere MQ Publish/Subscribe broker.

-p *ParentQueueManagerName*

(Required) The name of the queue manager that hosts the WebSphere MQ Publish/Subscribe broker to which this WebSphere Message Broker broker is to be joined.

Authorization:

The user ID used to invoke this command must have put and inq authority to the queue SYSTEM.BROKER.INTER.BROKER.COMMUNICATIONS.

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

Examples:

```

mqsijoinmqpubsub WBRK_BROKER -p MQBroker1

```

mqsilist (list resources) 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 BIPLIST; see “Contents of the Configuration Manager PDSE” on page 550

Purpose:

On Windows platforms, Linux, and UNIX systems, use the **mqsilist** command to list all the components installed on the system, all the execution groups defined to a specific broker, or all the message flows contained in a named execution group on a specific broker.

On Windows platforms, Linux, and UNIX systems, the output is directed to stdout.

From a z/OS console, use the **list** command to list all the execution groups defined to a specific broker, or all the message flows contained in a named execution group on a specific broker.

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 478

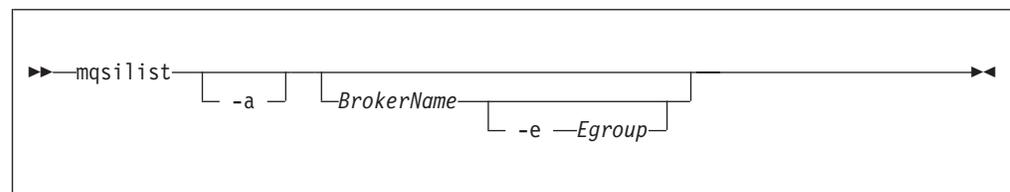
Authorization:

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

On Windows platforms, if you have specified a broker name and the **-e** flag, the user ID used to invoke this command must have **mqbrkrs** group membership.

mqsilist (list resources) command - Windows, Linux and UNIX systems:

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 - queuemanagename  
BIP8099I: ConfigMgr: configmgrname - queuemanagename  
BIP8099I: UserNameServer: UserNameServer - queuemanagename  
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 obtain the following message:

```
BIP8221I: <Component>: <ComponentName> (<Version>) - <Queue Manager>
```

BrokerName

(Optional) The name of the broker for which you want to list resources. This must be a deployed broker. A list of execution groups configured on this broker and the process ID (pid) of each is displayed.

-e Egroup

(Optional) Selects an execution group within a broker. Specify the label of the execution group for which you want to list message flows. The command returns a list of message flows assigned to the specified execution group within the broker.

The broker specified must be active for any message flow information to be returned.

Examples:

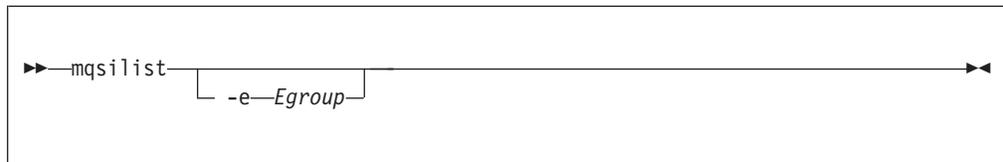
```
mqsilist WBRK_BROKER -e DefaultEG
```

```
mqsilist DatabaseInstanceMgr
```

mqsilist (list resources) command - z/OS:

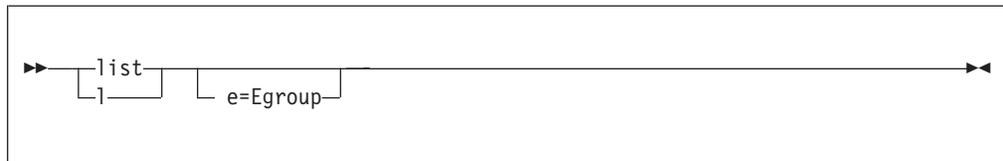
Syntax:

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 Egroup

(Optional) Selects an execution group within a broker. Specify the label of the execution group for which you want to list message flows. The command returns a list of message flows assigned to the specified execution group within the broker.

The broker specified must be active for any message flow information to be returned.

Examples:

You issue the **list** command only against a broker task name. For example:

```
F MQP1BRK,list
```

The output is the list of execution groups and process IDs in the form:

```
BIP8130I: Execution Group: <name> -<process ID>  
BIP8071I: Successful command completion
```

If you specify an execution group, for example:

```
F MQP1BRK, list e='exgrp1'
```

the output is a list of message flows in the form:

```
BIP8131I: Messageflow: <MessageFlowName>  
BIP8071I: Successful command completion
```

mqsilistaclentry command

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 550

Purpose:

Use the **mqsilistaclentry** command to view or list the currently defined:

- User groups
- Users
- Objects
- Access control lists

If you do not specify any parameters, all the groups, users, and objects are listed.

If you specify *GroupName*, only those access control lists relating to that group are listed.

If you specify *UserName*, only those access control lists relating to that specific user are listed, including any access control lists to which they belong.

If you specify *Broker*, only those groups, users, or access control lists relating to that broker are listed.

The output from this command is a description of the access rights that match the criteria specified in the command line arguments; each line takes the following form:

```
<principal> - <principaltype> - <accesstype> - <objectname> - <objecttype>
```

where

- <principal> is the name of the user or group for which a policy has been defined.
- <principal type> 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.
- <object type> 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" on page 481
- "mqsilistaclentry command - Linux and UNIX systems" on page 482
- "mqsilistaclentry command - z/OS" on page 484

Authorization:

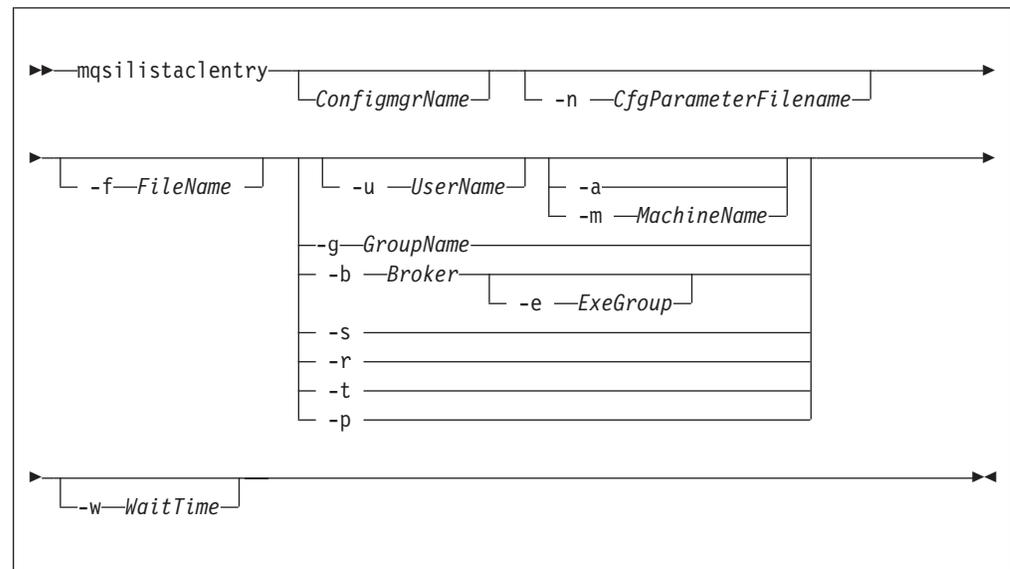
The user ID used to invoke this command must have full control permissions for the object being changed; see "ACL permissions" on page 591 for more information. In addition, for Linux and UNIX systems, the user ID must be a member of mqbrks.

When z/OS commands are run through the console, they effectively run as the Configuration Manager started-task ID. This means that the commands inherit a Full Control root ACL and you can carry out any operation.

If you submit a console command to the Configuration Manager you can change any ACL 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.

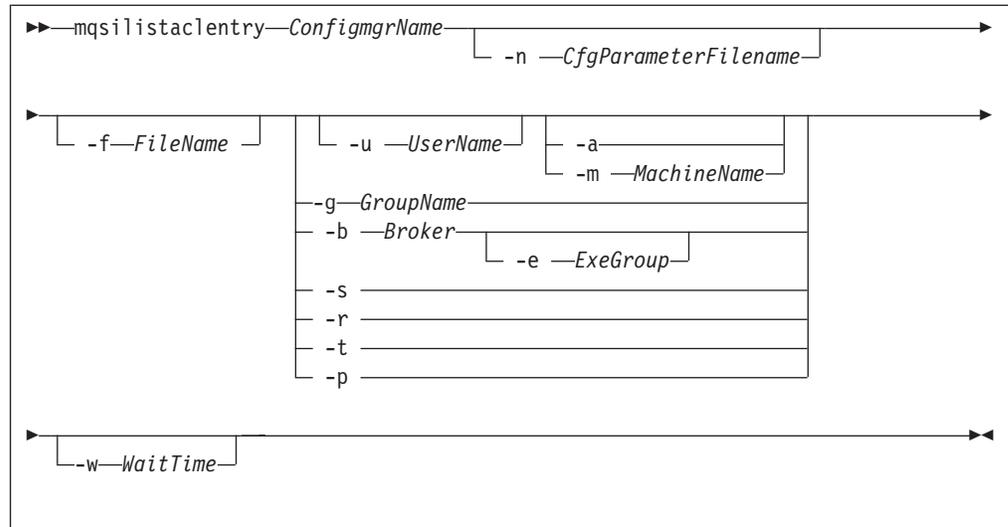
- u** *UserName*
(Optional) The user name to which this entry refers, for example, TEST\ANOTHER.
- m** *MachineName*
(Optional) The name of the machine from which a specified user can connect. This option can not be used with **-a**.
- a** (Optional) The specified user may connect to all machines. This 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.

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.

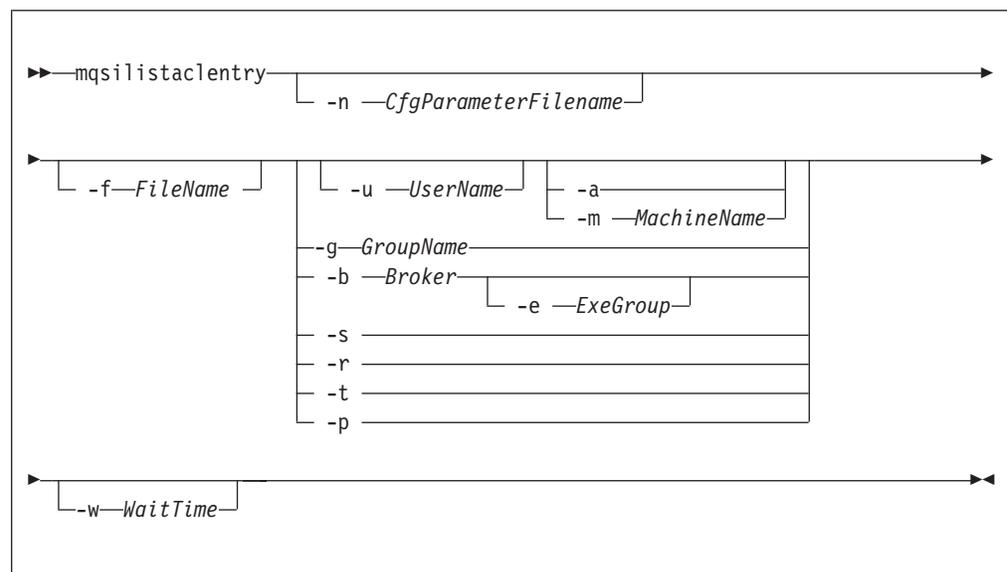
Examples:

```
mqsilistaclentry CMGR01 -b BROKER01
mqsilistaclentry CMGR01 -e BROKER01\ExeGrp01
mqsilistaclentry CMGR01 -g GROUPA
```

mqsilistaclentry command - z/OS:

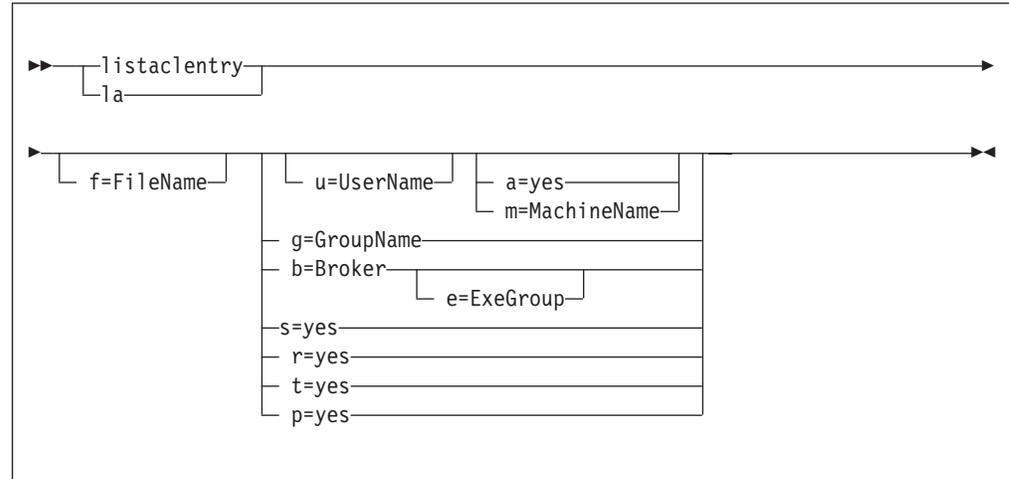
Syntax:

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 as:

```
<?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.

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 CMGR01,1a g='GROUPA'
```

mqsilistmqpubsub command

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 547

Purpose:

Use the **mqsilistmqpubsub** command to display the status of the WebSphere MQ Publish/Subscribe neighbor brokers to the specified WebSphere Message Broker broker.

This command indicates the status of the activity started by a previous join request (see "mqsijoinmqpubsub command" on page 475). 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:

```
mqsjlistmqpubsub—BrokerName
```

Parameters:

BrokerName

(Required) The name of the broker for which you want a list of neighbors.

Authorization:

On Linux and UNIX systems, the user ID must be a member of the **mqbrkrs** group.

On Windows platforms, no authorization is required.

Examples:

If there are no WebSphere MQ Publish/Subscribe brokers, and no **mqsjoinmqpubsub** command has been issued, this command returns the following message:

BIP8088I: There are no WebSphere MQ Publish/Subscribe neighbors

If an **mqsjoinmqpubsub** 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 **mqsjoinmqpubsub** 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

The `mqsimanagexalinks` command is used to set up a supported database on Linux and UNIX systems for XA coordination under the control of WebSphere MQ.

Purpose

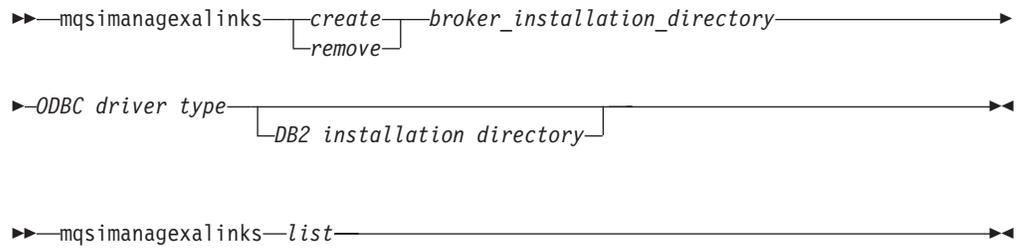
Use `mqsimanagexalinks` to set up a supported database on Linux and UNIX systems for XA coordination under the control of WebSphere MQ.

The command also sets up or removes the links required by WebSphere MQ. Run this command for each version of the ODBC drivers you want to use on your machine.

The `mqsimanagexalinks` command also provides a list of supported ODBC drivers.

Complete the `mqsimanagexalinks` procedure before you create a broker database. It is recommended that do this after you have installed your database.

Syntax



Parameters

create

(Required) Specify that the requested links are created.

remove

(Required) Specify that the requested links are removed.

Broker_installation_directory

(Required) The full path to your broker installation.

ODBC Driver TYPE

(Required) The ODBC driver type. See the list command for details on how to find the supported ODBC drivers.

DB2 installation directory

(Optional). If you are using DB2 then the full path to your db2 installation directory must also be supplied.

list

(Required) Display the list of supported ODBC drivers.

Links created for DB2 version 8

The command will create the appropriate links for your platform in the /var/mqm/exits directory to the following files for db2 version 8.

```

DB2_install_dir/lib/libdb2.so
DB2_install_dir/lib/libdb2.so.1
Broker_install_dir/sample/xatm/db2swit32

```

The command will create the appropriate links for your platform in the /var/mqm/exits directory to the following files db2 version 8.

```

DB2_install_dir/lib64/libdb2.so
DB2_install_dir/lib64/libdb2.so.1
Broker_install_dir/sample/xatm/db2swit

```

Not all links may be created on all platforms.

Links created for DB2 version 9

The command will create the appropriate links for your platform in the /var/mqm/exits directory to the following files for db2 version 9.

```

DB2_install_dir/lib32/libdb2.so
DB2_install_dir/lib32/libdb2.so.1
Broker_install_dir/sample/xatm/db2swit32

```

The command will create the appropriate links for your platform in the /var/mqm/exits directory to the following files db2 version 9.

```
DB2_install_dir/lib64/libdb2.so
DB2_install_dir/lib64/libdb2.so.1
Broker_install_dir/sample/xatm/db2swit
```

Not all links may be created on all platforms.

Links created for DataDirect V5.0 ODBC driver package

The command will create the appropriate links for the HP-UX (PA-RISC platform) in the /var/mqm/exits and /var/mqm/exits64 directory to the following files.

For Oracle in /var/mqm/exits:

```
broker_install_dir/ODBC32/V5.0/lib/libUKicu20.sl
broker_install_dir/ODBC32/V5.0/lib/UKor820.sl
broker_install_dir/ODBC32/V5.0/lib/UKor8dtc20.sl
```

For Oracle in /var/mqm/exits64:

```
broker_install_dir/ODBC64/V5.0/lib/libUKicu20.sl
broker_install_dir/ODBC64/V5.0/lib/UKor820.sl
broker_install_dir/ODBC64/V5.0/lib/IUKor8dtc20.sl
```

For Sybase in /var/mqm/exits:

```
broker_install_dir/ODBC32/V5.0/lib/libUKicu20.sl
broker_install_dir/ODBC32/V5.0/lib/UKase20.sl
broker_install_dir/ODBC32/V5.0/lib/UKasedtc20.sl
```

For Sybase in /var/mqm/exits64:

```
broker_install_dir/ODBC64/V5.0/lib/libUKicu20.sl
broker_install_dir/ODBC64/V5.0/lib/UKase20.sl
broker_install_dir/ODBC64/V5.0/lib/UKasedtc20.sl
```

Links created for DataDirect V5.2 ODBC driver package

The command will create the appropriate links for your platform in the /var/mqm/exits and /var/mqm/exits64 directory to the following files.

For Oracle in /var/mqm/exits:

```
broker_install_dir/ODBC32/V5.2/lib/libUKicu22.x
broker_install_dir/ODBC32/V5.2/lib/UKor822.x
broker_install_dir/ODBC32/V5.2/lib/UKor8dtc22.x
```

For Oracle in /var/mqm/exits64:

```
broker_install_dir/ODBC64/V5.2/lib/libUKicu22.x
broker_install_dir/ODBC64/V5.2/lib/UKora22.x
broker_install_dir/ODBC64/V5.2/lib/UKoradtc22.x
or broker_install_dir/ODBC64/V5.2/lib/UKor8dtc22.x
```

For Sybase in /var/mqm/exits:

```
broker_install_dir/ODBC32/V5.2/lib/libUKicu22.x
broker_install_dir/ODBC32/V5.2/lib/UKase22.x
broker_install_dir/ODBC32/V5.2/lib/UKasedtc22.x
```

For Sybase in /var/mqm/exits64:

```
broker_install_dir/ODBC64/V5.2/lib/libUKicu22.x
broker_install_dir/ODBC64/V5.2/lib/UKase22.x
broker_install_dir/ODBC64/V5.2/lib/UKasedtc22.x
```

Where x is the relevant library extension on your platform. Not all links may be created on all platforms.

Authorization:

The user ID used to invoke this command must be root.

mqsimigratecomponents command

The `mqsimigratecomponents` command migrates 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 BIPMGCOMP.

Purpose:

Migrate components to WebSphere Message Broker Version 6.1 from Version 6.0 or from Version 5.0.

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.

For Version 5.0 of the product, Version 5.0.0.4 (Fix Pack 4) is the earliest release of the product that is supported.

You *must* have an installation of the product at the required version, 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 on a component, without making any changes, to ensure that the component is suitable for the required migration (**-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).

The `mqsigratecomponents` command updates your registry and file system, WebSphere MQ definitions, and database definitions. If the user who is issuing the command does not have the authority to perform all of these steps, the command can be run 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 using the `-1`, `-2`, and `-3` parameters.

If you are using the `mqsigratecomponents` command to migrate a broker that uses Sybase for its broker database, you must modify the database by performing the following actions:

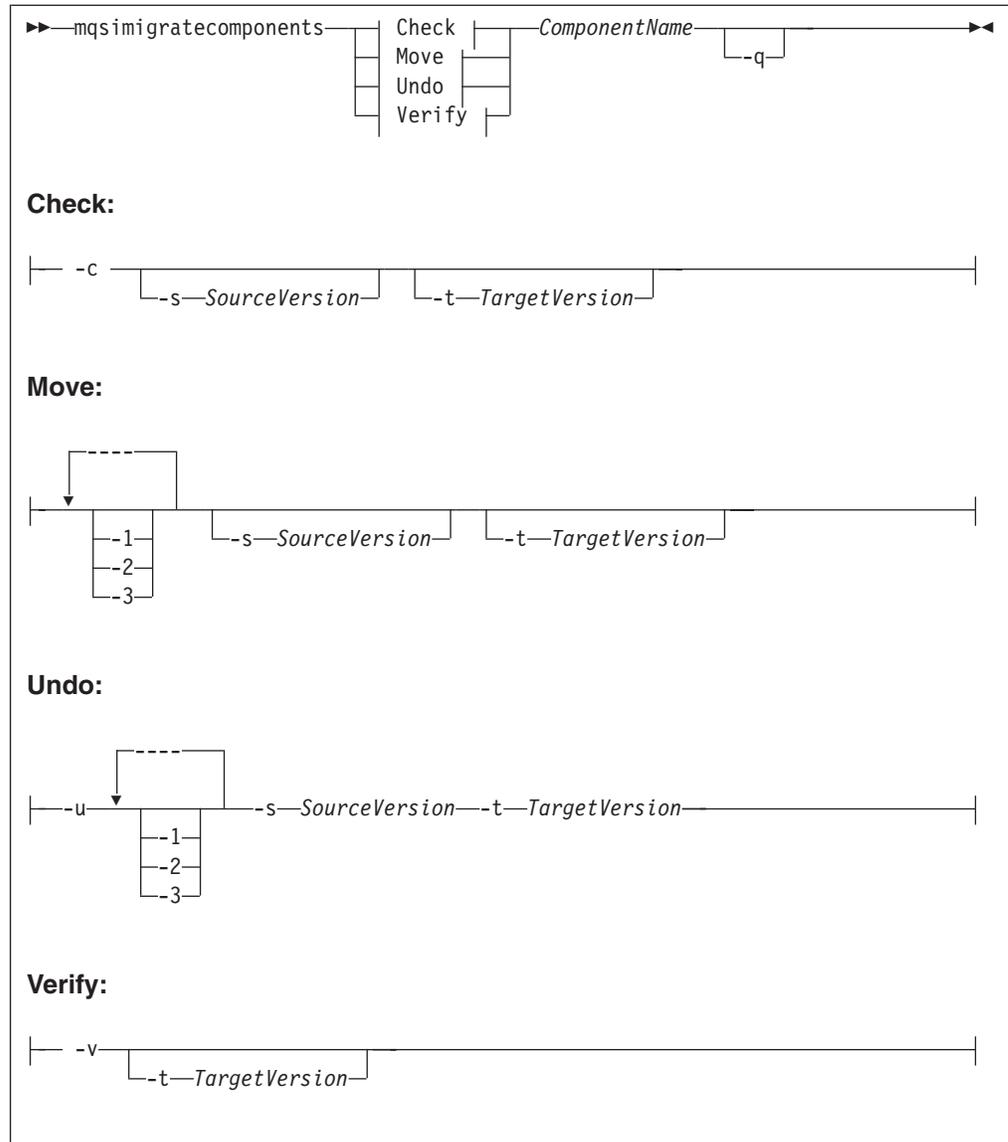
1. Log on to ISQL using a system administrator account.

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) Do a check of a specified component before you migrate it, to ensure that:
 - The auto-detected version of the broker matches any version specified on the command line.
 - 64-bit execution groups are not supported if you are migrating from Version 6.1 to Version 5.0.
 - The database tables to be copied from a previous release do not contain any rows that are incorrectly indexed.

The migration check can be run against 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) Do a post-migration check of a specified component to ensure that:
 - The correct database tables and queues exist for the specified version.
 - The registry is in the correct format for the specified version.
- q (Optional) Print fewer status messages during the operation.
- 1 (Optional) Do only registry and file system work.
 - When you migrate to Version 6.1, use the **-1** parameter before the **-2** or **-3** parameters.
 - When migrating backwards 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 are sharing 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:

```
mqsigratecomponents 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:

```
mqsigratecomponents BROKERB -1 -2
```
- In two steps:

```
mqsigratecomponents BROKERB -1  
mqsigratecomponents BROKERB -2
```

- u (Optional) Undo a failed migration step; you must also specify at least one of **-1**, **-2**, or **-3**. You should only use this option when migration has failed, and also failed to auto-recover (a failure during split migration being one example).

-s *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 `mqsigratecomponents` command with the **-1** parameter, as shown in the split migration example.
- See “Purpose” on page 491 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, as shown in the split migration example.

- See "Purpose" on page 491 for the restrictions to the version numbers of the product that are supported.

ComponentName

(Required) The name of the component to migrate.

Authorization:

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 databases associated with the component
- Modify queue definitions

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 as 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:

```
mqsigratecomponents BROKER -1
mqsigratecomponents BROKER -s 5.0.0.5 -2
mqsigratecomponents 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:

```
mqsigratecomponents -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.

mqsigratecomponents 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
```

mqsireadlog command

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 547

Purpose:

Use the mqsireadlog command to retrieve the trace log for the specified component. This command is valid for:

User trace

Specify the `-u` option.

Service trace

Specify the `-t` option. You are recommended to use this option only if directed to do so by the action described in a BIPxxxx message, or by your IBM Support Center.

You can specify the output to be directed to file, or to stdout. The trace records returned by this command are in XML format and can be browsed with an XML browser. If you direct output to file, the data is written in code page utf-8. The file is therefore platform-independent, and can be transferred to other systems for browsing or formatting using the mqsiformatlog command.

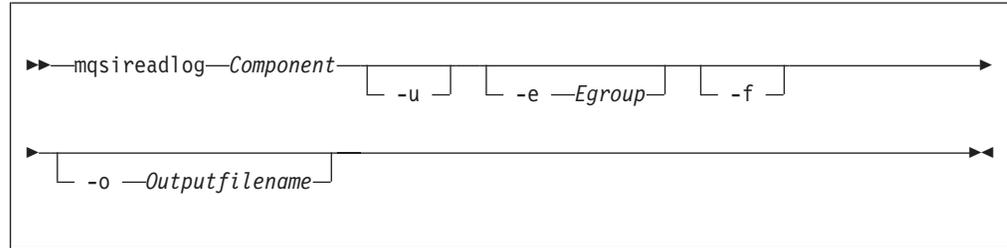
On HP-UX you are recommended to set the size parameter of the mqsichange trace command to be less than 500 MB. Note that the size of the XML generated files is often half as much again as the original trace file, and setting the value of the size parameter to be greater than 500 MB can cause problems.

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).

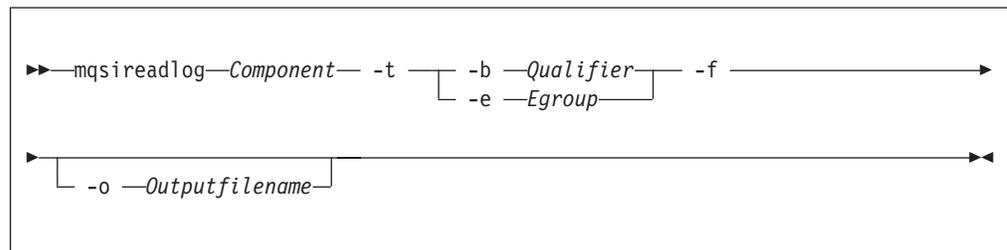
If you specify a broker, or any of its resources (execution group or message flow) you must have deployed them before you can start trace and read the log files.

Syntax:

User trace:



Service trace:



Parameters:

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).

-u (Optional) Read the log contents from the user trace log. This is valid **only** if you select the broker component.

-e *Egroup*

(Optional) The label of the execution group for which log information is to be read.

-o *Outputfilename*

(Optional) The name of the file into which to write the log data. If you specify a full pathname, the file is created in the directory specified. If you specify just the filename, the file is created in the current working directory. The contents of the file are written in code page utf-8, which is platform-independent and preserves data such as DBCS characters.

You must specify a file name if you want to format the log using the mqsiformatlog command. If you do not specify a filename, the contents of the log are written to stdout. You are recommended to use a file extension of .xml.

-f (Optional 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. Further details are given in “Additional parameters exclusive to service trace” on page 498.

If you specify this option, stop tracing (using `mqsichangetrace`) before you use the `mqsireadlog` command. If the log file is in use when you issue this command with this flag specified, partial XML records might be returned. You can reduce the risk of this happening by specifying `-m safe` on the `mqsichangetrace` command. If the component being traced has itself stopped, you do not then need to issue a `mqsichangetrace` command.

If you do not stop tracing before you issue this command, check the contents of the log file created and remove any partial records from the end using a text editor before using the `mqsiformatlog` command, as partial records cannot be read by the format command.

Additional parameters exclusive to service trace:

Use these options only when directed to do so by your IBM Support Center or by a BIPxxxx message.

-t (Required) Read the log contents from the service trace log.

-b *Qualifier*

(Required) Read the contents of the log for the broker agent, Configuration Manager agent, or User Name Server agent, or for the specified command utility program. This option is valid only if you have specified `-t` (service trace).

The following table shows the valid combinations of *qualifier* and component for service trace.

This option is generally used to trace the commands themselves. If you want to trace a particular command, run that command with environment variable `MQSI_UTILITY_TRACE` set to `debug` or `normal` before you issue this command to read the trace output generated.

Enter these values exactly as shown.

The agent trace is initiated when you specify the `-b` flag on the `mqsichangetrace` command. Do this only when directed to do so by a WebSphere 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
```

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.

-f (Required) Read the log file directly from the file system. When used with service trace, this flag has the same characteristics as when used with user trace. It remains optional if the `-e` flag is specified. You must specify this option if you specify the `-b` flag.

| Qualifier | Component=
<broker_name> | Component=
ConfigMgr_name | Component=
UserNameServer | Component=
workbench | Component=
utility |
|---------------------------|-----------------------------|------------------------------|------------------------------|-------------------------|-----------------------|
| mqsichangebroker | x | | | | |
| mqsichangeconfigmgr | | x | | | |
| mqsichangetrace | x | x | x | | |
| mqsichangeusername server | | | x | | |
| mqsiclearmqpubsub | x | | | | |

| Qualifier | Component=
<broker_name> | Component=
ConfigMgr_name | Component=
UserNameServer | Component=
workbench | Component=
utility |
|-----------------------------|-----------------------------|------------------------------|------------------------------|-------------------------|-----------------------|
| mqscreateaclentry | | | | | x |
| mqscreatebroker | x | | | | |
| mqscreateconfigmgr | | x | | | |
| mqscreateusernameserver | | | x | | |
| mqsdeleteaclentry | | | | | x |
| mqsdeletebroker | x | | | | |
| mqsdeleteconfigmgr | | x | | | |
| mqsdeleteusernameserver | | | x | | |
| mqsifformatlog ¹ | | | | | x |
| mqsijoinmqpubsub | | x | | | |
| mqsilist ² | | x | | | x |
| mqsilist | x | | | | |
| mqsilistaclentry | | | | | x |
| mqsireadlog | x | x | x | | x |
| mqsireporttrace | | x | | x | |
| mqsistart | x | x | x | | |
| mqsistop | x | x | x | | |
| agent | x | x | x | | |
| service | x | x | x | | |
| workbench | | | | x | |
| 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 this command is invoked without a component, trace information is recorded in, and retrieved from, the *utility* trace files in addition to component specific files. For further details see the `mqsichangetrace` command.

Authorization:

On UNIX platforms, the user ID must be a member of the **mqbrkrs** group. If the `-f` flag is specified, the user ID used to invoke this command must also have access to the trace file.

On Windows platforms, if the `-f` flag is specified, the user ID used to invoke this command must have access to the trace file. If the `-f` flag is not specified, the user ID used to issue the command must have **mqbrkrs** authority.

Examples:

User trace for broker WBRK_BROKER:

```
mqsireadlog WBRK_BROKER -u -e default -o trace.xml
```

Service trace for component ConfigMgr:
mqsireadlog ConfigMgr -t -b agent -f -o trace.xml

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) using the command **mqsiformatlog**, or view it using an XML editor or viewer.

mqsireload command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS - as a console command

Purpose:

Use the **mqsireload** command to request the broker to stop and restart execution groups.

When you issue this command, a message is sent to the broker, which stops and restarts all its execution groups. You can specify a single execution group to be reloaded, but you are recommended to use the default form of this command to reload all execution groups.

Because an execution group does not stop until all message flows within it terminates, the ability of the broker to reload quickly depends on the processing time for the longest running message flow. This affects the performance of this command, and you are recommended to review any long-running message flows.

If you have included a 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, you are recommended 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 501

Authorization:

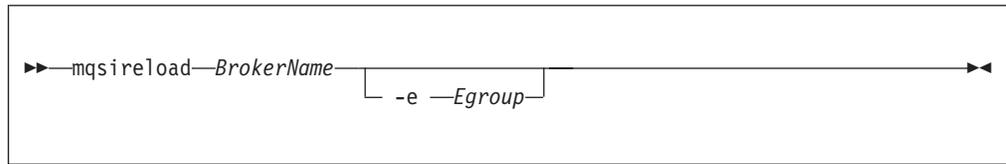
On Windows, Linux, and UNIX systems the user ID used to issue the command must be a member of the group **mqbrkrs**.

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:

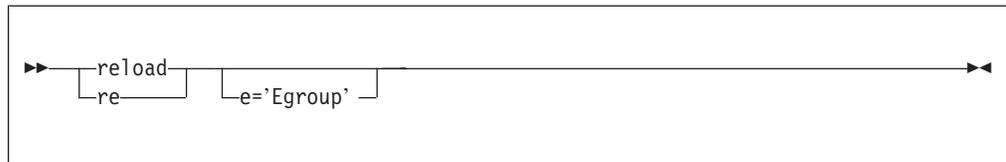
```
mqsireload broker1
```

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

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. You can also run this command by customizing and submitting BIPRLSEC; see “Contents of the Configuration Manager PDSE” on page 550.

Purpose:

Use the **mqswireloadsecurity** command to force the immediate expiry of some or all of the entries in the security cache.

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 reset. If re-authentication fails, the entry remains marked as expired. Any entries in the security cache marked as expired are removed when the next sweep of the cache is performed.

Use the **mqsicchangeproperties** 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, they must be re-authenticated.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

- “mqswireloadsecurity command - Windows, Linux, and UNIX systems”
- “mqswireloadsecurity command - z/OS” on page 503

Authorization:

On Windows the user ID used to invoke this command must have administrator authority on the local system.

On Linux and UNIX systems the user ID used to issue the command must be a member of the group **mqbrkrs**.

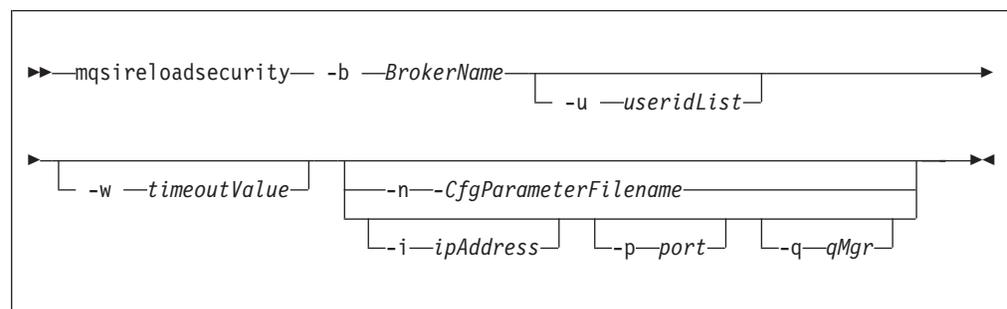
On z/OS the user ID used to invoke this command must be a member of a group that has read and write access to the component directory.

Responses:

No additional responses are returned.

mqswireloadsecurity command - Windows, Linux, and UNIX systems:

Syntax:



Parameters:

BrokerName

(Required) The name of the broker to which the request to reload the security cache entries is to be sent. If no other options are selected, 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 this parameter is not specified, all entries in the security cache are reloaded. For cached certificates this 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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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
```

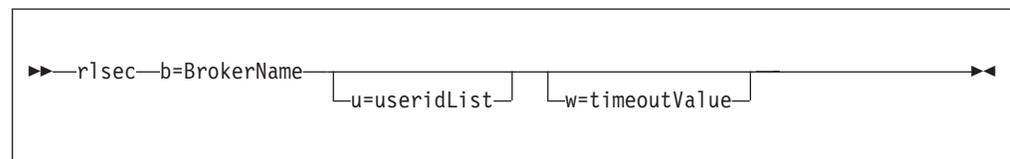
Reload the cache for a list of users on the specified broker, and wait for five seconds before returning:

```
mqsireloadsecurity -b a_broker -u user1:user2:user3 -w 5
```

mqsireloadsecurity command - z/OS:

Syntax:

z/OS console command:



Parameters:

b=BrokerName

(Required) The name of the broker to which the reloadsecurity request is sent. If no other options are selected, 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 this parameter is not specified, all entries in the security cache are reloaded. For cached certificates this 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=r1sec
```

Reload the cache for a single user on the specified broker:

```
F MQP1BRK, b=r1sec u=user1
```

Reload the cache for a list of users on the specified broker and wait for five seconds before returning:

```
F MQP1BRK, b=r1sec u=user1:user2:user3 w=5
```

mqsiereportflowstats 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 BIPRPM5; see "Contents of the broker PDSE" on page 547

Purpose:

Use the **mqsiereportflowstats** command to display the current options for accounting and statistics that have been set using the **mqsierechangefflowstats** command.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

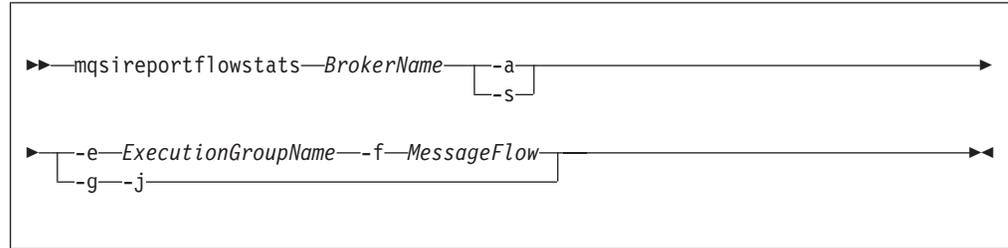
- "mqsiereportflowstats command - Windows, Linux and UNIX systems"
- "mqsiereportflowstats command - z/OS" on page 506

Authorization:

On Windows, Linux, and UNIX systems, the user ID used to issue the command must have mqbrkr authority.

mqsiereportflowstats 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:

```
mqsiereportflowstats BrokerA -s -a -e default -f MyFlow1
```

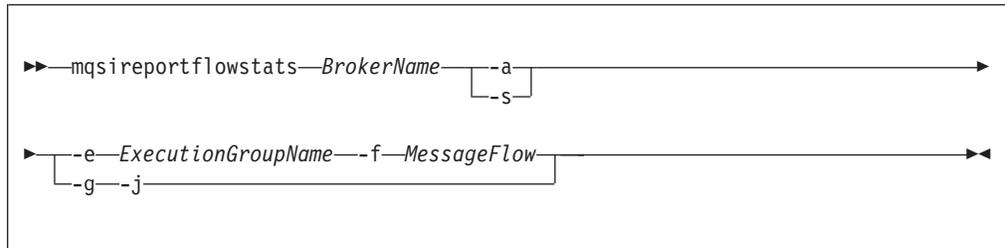
Request a report of the snapshot options that are currently stored for all message flows in all execution groups for broker "BrokerA" :

```
mqsireportflowstats BrokerA -s -g -j
```

mqsireportflowstats command - z/OS:

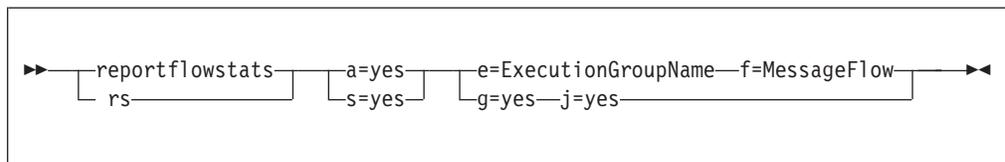
Syntax:

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:

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
```

Request a report of the snapshot options that are currently stored for all message flows in all execution groups:

```
mqsireportflowstats -s -g -j
```

mqsireportflowuserexits command

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 BIPRPUE; see "Contents of the broker PDSE" on page 547

Purpose:

Use the **mqsireportflowuserexits** command to report the list of active and inactive user exits for a given message flow.

Select the appropriate link for details of this command on the platform, or platforms, that your enterprise uses:

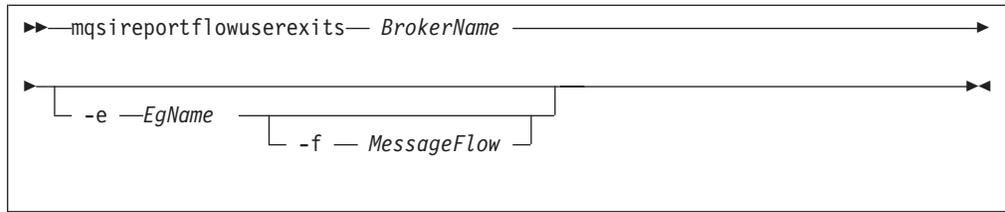
- "mqsireportflowuserexits command - Windows, Linux, and UNIX systems"
- "mqsireportflowuserexits command - z/OS" on page 508

Authorization:

On Windows, Linux, and UNIX systems, the user ID that is used to invoke this command must have **mqbrkrs** group authority.

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.

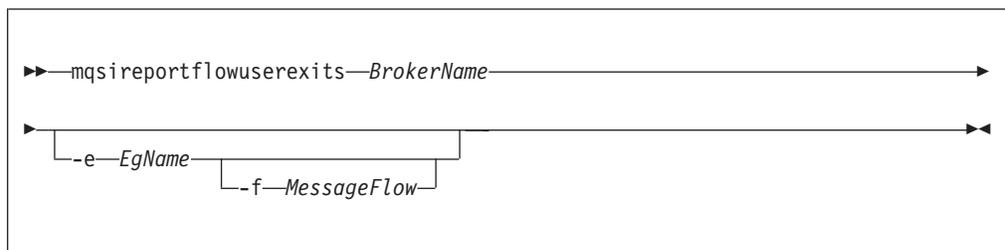
Examples:

```
mqsiereportflowuserexits 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
```

mqsiereportflowuserexits command - z/OS:

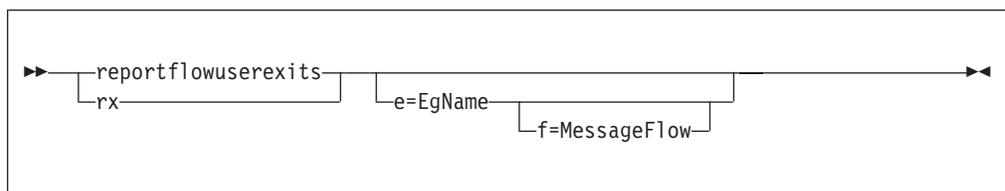
Syntax:

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:

```
mqsiportflowuserexits 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
```

mqsiportproperties command

Use the `mqsiportproperties` command to display properties that relate to a broker or its associated configurable services.

Supported platforms:

- Windows systems.
- Linux and UNIX systems.
- z/OS. Run this command by customizing and submitting BIPRPPR; see “Contents of the broker PDSE” on page 547.

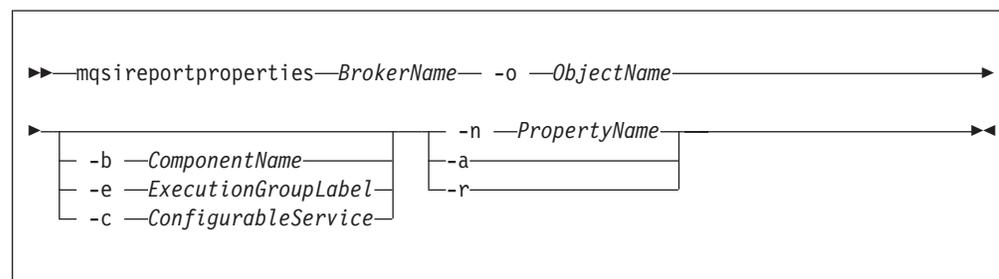
Purpose:

Use the `mqsiportproperties` command to examine the values of properties that are set using the `mqsichangeproperties` command, or created using the `mqsicreateconfigurableService` command.

Usage notes:

- Before you run this 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.

-e *ExecutionGroupLabel*

(Optional) The label of the execution group for which a report is required.

The special name `AllExecutionGroups` is allowed, and is used to view settings for the Connector object name; this allows you to view broker-wide settings that affect all execution groups.

-b *ComponentName*

(Optional) The name of the component selected. Valid values are *httplistener* and *securitycache*.

-c *ConfigurableService*

(Optional) The type of the configurable service, such as `JMSProviders`, `EISProviders`, `JDBCProviders`, `SecurityProfiles`, `PolicySets`, `PolicySetBindings`, or `FtpServer`. Specify a value of `AllTypes` to report on all configurable service types.

-o *ObjectName*

(Required) The name of the object whose properties you want to read.

The values:

- `SAP`, `Siebel`, and `PeopleSoft` are accepted for WebSphere Adapters components.
- `HTTPConnector`, and `HTTPSCConnector` are accepted for all execution groups, and allowed if you select the value *httplistener* for *ComponentName*.
- `SecurityCache`
- `SecurityProfileName` to be changed.

For example, if you set the **-c** parameter to `JMSProviders`, the expected object name is either a default JMS provider name that has been predefined by IBM, or a user-defined JMS provider name. If you set the **-c** parameter to `EISProviders`, the expected object name is a default EIS provider name that has been predefined by IBM. If you set the **-c** parameter to `JDBCProvider`, the expected object name is either a default JDBC provider name that has been predefined by IBM, or a user-defined JDBC provider name.

If you set the **-c** parameter to `SecurityProfiles`, the expected object name is the name of the security profile for the broker. Use the `mqsireportproperties` command to view the list of JMS provider names, JDBC provider names, or security profiles that are configured for this broker. 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

-a (Optional) Indicates that all property values of the object are displayed, and does not recurse into properties that have child values.

-r (Optional) Indicates that all property values of the object are displayed and, additionally, displays the child values for all properties that have child values.

You must select only one option from **-n**, **-a**, and **-r**.

For a list of the options you can select, see “`mqsichangeproperties` command” on page 381.

The *ObjectName* parameter can have a value for inter-broker communications or for the *HttpListener* component. For inter-broker communications, the following values are available:

- *DynamicSubscriptionEngine*
- *Axis2Manager*

For the *HttpListener* component, which includes the HTTPInput and HTTPReply nodes, the following *ObjectName* values are available:

- *HTTPListener* for controlling the HTTPListener process.
- *HTTPConnector* for controlling HTTP communication with the HTTPInput and HTTPReply nodes.
- *HTTPSConnector* for controlling HTTPS communication with the HTTPInput and HTTPReply nodes.

HttpListener is required when using the HTTP *ObjectName* settings.

For the *HttpListener* component, **mqsireportproperties** does not report those properties that have not been explicitly set with **mqsichangeproperties**, even if those properties have a default setting. For example, the default HTTPSConnector port that is used (unless it has been changed) is 7083. However this value is not reported by **mqsireportproperties** unless it has been changed from this default with **mqsichangeproperties**. To see the default values for all properties that **mqsireportproperties** can report on, see the documentation for **mqsichangeproperties**.

For the *securitycache* component, the following *ObjectName* value is available:

- *SecurityCache* for controlling the security cache settings.

Authorization:

No specific authority is required to invoke this command.

Responses:

These are of the form:

- *Axis2Manager*
 - *ActiveHandlerChains*
 - NodeLabel = "goop"
 - FlowLabel = "test"
 - WSAddressingConfigured = true
 - PolicySetName = "WSS default 1.0"
 -
- *HTTPConnector*
 - PortNumber = 7800
- *HTTPSConnector*
 - PortNumber = 7843

Examples:

Display all the current HTTPListener settings (that have been changed from their default value)

```
mqsireportproperties TEST -b httplistener -o HTTPListener -a
```

Display the HTTPSConnector port setting (if it has been changed from the default value)

```
mqsiportproperties TEST -b httplistener -o HTTPSConnector -n port
```

Display recursively all the current settings for the inter-broker communication.

```
mqsiportproperties TEST -e default -o DynamicSubscriptionEngine -r
```

Display the properties for all the broker's JMS provider resources (default JMS provider resources and those configurable services that are defined using the `mqsicreateconfigurable-service` command).

```
mqsiportproperties WBRK6_DEFAULT_BROKER -c JMSProviders  
-o AllReportableEntityNames -r
```

Enter the command on a single line; a line break has been added to enhance readability.

Display the properties for all the JMS provider resources of JMS_Default_IBM_MQ.

```
mqsiportproperties WBRK6_DEFAULT_BROKER -c JMSProviders -o IBM_MQ -r
```

Display properties and configuration information for Axis2.

```
mqsiportproperties TEST -e MyExecutionGroup -o Axis2Manager -r
```

Display the properties for all the broker's security profiles (default security profiles and those defined using the `mqsicreateconfigurable-service` command).

```
mqsiportproperties WBRK_6_DEFAULT_BROKER -c SecurityProfiles  
-o AllReportableEntityNames -r
```

Enter the command on a single line; a line break has been added to enhance readability.

Response:

```
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:

```
mqsiportproperties WBRK_6_DEFAULT_BROKER -c SecurityProfiles  
-o MyFirstSecurityProfile -r
```

Enter the command on a single line; a line break has been added to enhance readability.

Response:

```
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 the properties for the object Axis2Manager on the execution group MyExecutionGroup for broker TEST:

```
mqsireportproperties TEST -e MyExecutionGroup -o Axis2Manager -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 policysset.xml:

```
mqsireportproperties WBRK_BROKER -c PolicySets -o Policy_2 -n ws-security
-p policysset.xml
```

This is used to move policy sets between brokers, and for backup. Enter the command on a single line; a line break has been added to enhance readability.

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. It is used to move policy set bindings between brokers, and for backup. Enter the command on a single line; a line break has been added to enhance readability.

Report the dependencies for the WebSphere Adapter for SAP:

```
mqsireportproperties WBRK_BROKER -c EISProviders -o SAP -r
```

Report the properties of the Oracle JDBCProvider configurable service:

```
mqsireportproperties WBRK_BROKER -c JDBCProviders -o Oracle -r
```

Configurable services: The following command was used to generate this output:

```
mqsireportproperties brokerName -c AllTypes -o AllReportableEntityNames -r
```

| Configurable Service Name (-c) | Any default Configurable Services (-o) | Attributes (-n) |
|--------------------------------|--|-----------------------|
| EISProviders | PeopleSoft
SAP
Siebel
Twineball | jarsURL
nativeLibs |

| Configurable Service Name (-c) | Any default Configurable Services (-o) | Attributes (-n) |
|--------------------------------|--|--|
| FtpServer | None | accountInfo
scanDelay
transferMode
connectionType
remoteDirectory
securityIdentity
serverName |
| JDBCProviders | DB2
Informix
Informix_With_Date_Format
Microsoft_SQL_Server_2000
Oracle
Sybase_JConnect6_05 | connectionUrlFormat
connectionUrlFormatAttr1
connectionUrlFormatAttr2
connectionUrlFormatAttr3
connectionUrlFormatAttr4
connectionUrlFormatAttr5
databaseName
databaseType
databaseVersion
description
jarsURL
portNumber
securityIdentity
serverName
type4DatasourceClassName
type4DriverClassName |

| Configurable Service Name (-c) | Any default Configurable Services (-o) | Attributes (-n) |
|---------------------------------------|---|--|
| JMSProviders | ActiveMQ
BEA_Weblogic
Fiorano MQ
Client for WebSphere Application Server
JBoss
JOnAS
OpenJMS
Oracle OEMS
SeeBeyond
SonicMQ
SwiftMQ
Tibco_EMS
WebSphere_MQ
Generic_File
Generic_LDAP | jarsURL
nativeLibs |
| PolicySetBindings | WSS10Default | ws-security |
| PolicySets | WSS10Default | ws-security |
| SMTP | n/a | serverName
securityIdentity |
| SecurityProfiles | Default_Propagation | authentication
authenticationConfig
authorization
authorizationConfig
keyStore
mapping
mappingConfig
passwordValue
propagation
trustStore |

| Configurable Service Name (-c) | Any default Configurable Services (-o) | Attributes (-n) |
|--------------------------------|--|---|
| ServiceRegistries | DefaultWSRR | connectionFactoryName
endpointAddress
initialContextFactory
locationJNDIBinding
needCache
predefinedCacheQueries
refreshQueriesAfterNotification
subscriptionTopic |

mqsiporttrace command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS as a console command

Purpose:

Use the **mqsiporttrace** command to display the trace options currently in effect. This command is valid for:

- User trace. Specify the **-u** option.
On z/OS, user trace is run against a specific execution group for a broker. You can 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.
Unlike user trace, service trace can also be run against an active agent (that is, a broker, User Name Server, or Configuration Manager).
- 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 component 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:

- “mqsiporttrace command - Windows, Linux and UNIX systems”
- “mqsiporttrace command - z/OS” on page 518

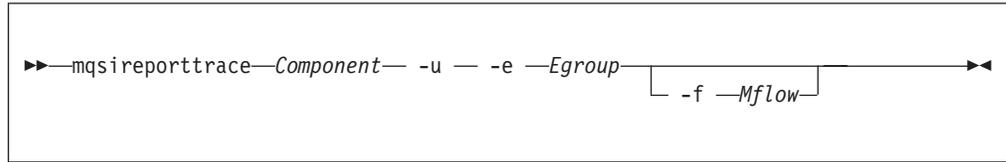
Authorization:

On Windows, Linux, and UNIX systems, the user ID used to issue the command must have **mqbrkrs** authority.

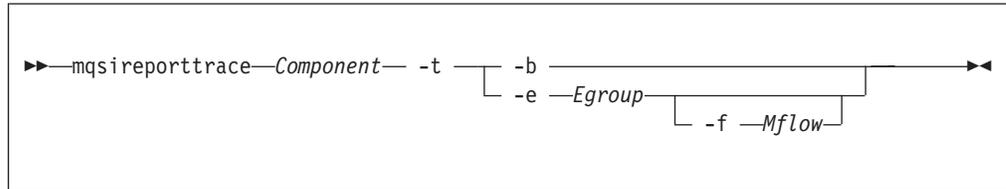
mqsiporttrace command - Windows, Linux and UNIX systems:

Syntax:

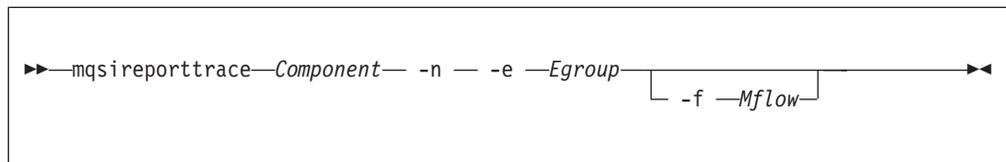
User trace:



Service trace:



Trace nodes:



Parameters:

Component

(Required) The name of a broker, or of a Configuration Manager, for which options are reported, or the fixed value `UserNameServer` (all are case sensitive on Linux and UNIX systems).

-u (Required for user trace) Derive report information from the user trace.

-e Egroup

(Required for user trace, otherwise optional) The label of the execution group for which a report is required. This is valid only if you have specified a broker as the component.

-f Mflow

(Optional) The label of the message flow for which a report is required. This is valid only if you have specified both a broker as the component and an execution group.

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 report derive report information from service trace for execution group exgrp1 in broker WBRK_BROKER, enter the command:

```
mqsiporttrace 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:

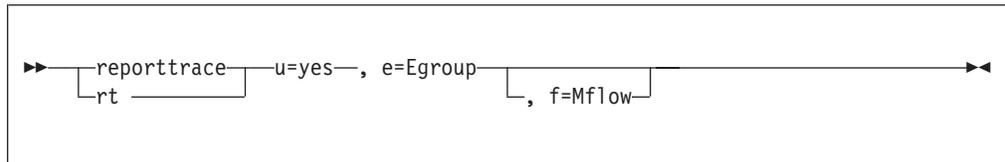
```
mqsiporttrace WBRK_BROKER -n -e "exgrp1"
```

mqsiporttrace command - z/OS:

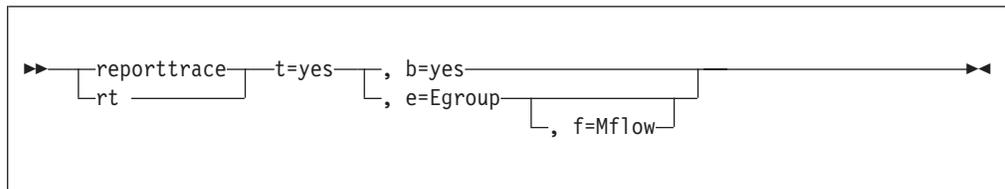
Syntax:

z/OS console command:

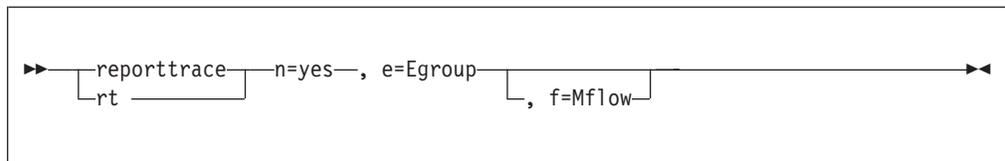
User trace:



Service trace:



Trace nodes:



Parameters:

-u (Required for user trace) Derive report information from the user trace.

-e *Egroup*

(Required for user trace, otherwise optional) The label of the execution group for which a report is required. This is valid only if you have specified a broker as the component.

This name is case sensitive; include the name in single quotes if it contains mixed-case characters.

-f *Mflow*

(Optional) The label of the message flow for which a report is required. This is valid only if you have specified both a broker as the component and an execution group.

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 derive 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

Command to restore previously backed up Configuration Manager archive files that you created using the mqsibbackupconfigmgr 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 550

Purpose:

This command restores a previously backed up Configuration Manager archive file that you created using the mqsibbackupconfigmgr command.

Usage notes:

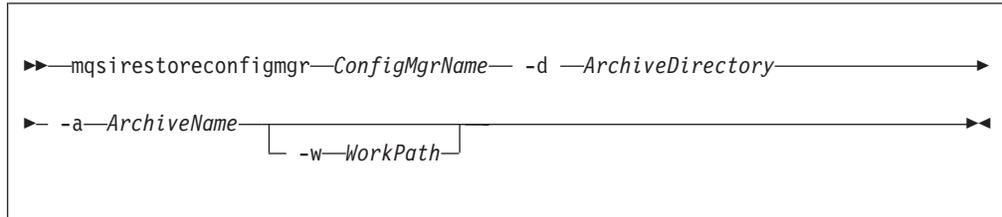
1. Before you run the command, stop the Configuration Manager.
2. Run this command on the computer on which the Configuration Manager has been created.
3. If you are recovering the Configuration Manager because the configuration repository is damaged, restore the repository from a previously successful 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) Specifies the backup archive name.

-w *WorkPath*

(Optional) Specifies the path for the Configuration Manager repository.

Authorization:

On Windows this command changes security privileges for the ServiceUserID; the user ID used to invoke this command must be a member of the Windows **Administrators** group on one local system. In addition, for Linux and UNIX systems, the user ID must be a member of mqbrkrs.

mqsisetdbparms command

Use the mqsisetdbparms command to associate a specific user ID and password with a data source name (DSN) that is accessed from a message flow.

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPSDBP.

Purpose:

Use the mqsisetdbparms command to create, alter, or delete the user ID and password pairs that are associated with the specified data source name (DSN).

Data source names are used in the following nodes:

- Compute
- Database
- DatabaseRetrieve

- DatabaseRoute
- DataDelete
- DataInsert
- DataUpdate
- FileInput
- FileOutput
- Filter
- Mapping
- Warehouse

If you use the same DSN in multiple nodes to refer to the same database instance, the same user ID and password combination is used.

Data source names are also used to associate user ID and password pairs with a JDBC type 4 connection. Specify the DSN that is used in this command as the **-n securityIdentity** parameter on the `mqsicreateconfigurableservice` or `mqsichangeproperties` command.

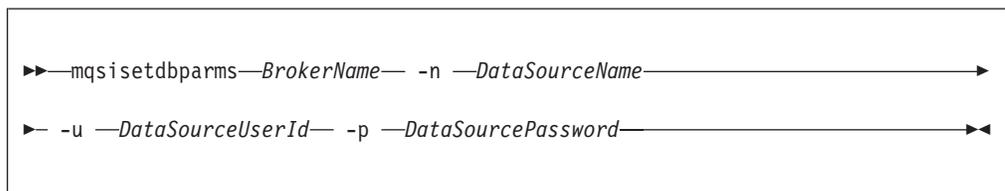
The user ID and password pair is created in the DSN folder under the broker's registry folder.

This command does not run if the broker is running. You must stop the broker before you run this command.

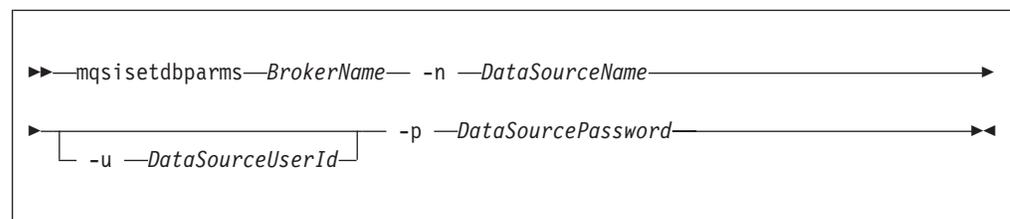
The `mqsisetdbparms` command does not apply to the broker database. Although you can issue this command successfully to specify the data source name that is used by the broker to access its database, it has no effect on the user ID that is used to access any database, including any user databases that is referenced by that data source name.

Syntax:

Create:



Alter:



Delete:

```
►►—mqsisetdbparms—BrokerName— -n —DataSourceName— -d —————►►
```

Parameters:

BrokerName

(Required) The name of the broker for which settings are to be created, altered, or deleted.

-n *DataSourceName*

(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.
- The name of the security identity that is used to authenticate a connection to an FTP server. The *DataSourceName* 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 *securityIdentity* property of the FtpServer configurable service definition.
- The name of the security identity that is used to authenticate a JDBC type 4 connection. You must run the `mqsischangeproperties` command to modify the *securityIdentity* property of the associated JDBCProvider configurable service to match the *DataSourceName* that you specify on this command.

-u *DataSourceUserId*

(Required for Create; Optional for Alter) The user ID to be associated with this data source.

-p *DataSourcePassword*

(Required for Create and Alter) The password to be associated with this data source.

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 (Required for Delete) This parameter deletes the user ID and password pair for this data source from the registry.

Authorization:

Windows On Windows systems, the user ID that is used to invoke this command must have Administrator authority on the local system.

Linux **UNIX** On Linux and UNIX systems, the user ID that is used to invoke this command must be a member of the `mqbrkrs` group.

z/OS On z/OS, the user ID that is used to invoke 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 data sources, 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:

The following examples show the usage of the command without the additional Universal Record Identifier (URI):

```
mqsisetdbparms WBRK_BROKER -n MQBroker1 -u MQUserId -p password
mqsisetdbparms WBRK_BROKER -n MQBroker1 -d
```

The following examples show the use of the command when the URI for a JMS or JNDI resource name is substituted for the DataSourceName that is associated with the `-n` parameter.

For a JMS resource, the prefix is "jms::"; for JNDI, the prefix is "jndi::"; and for a JDBC provider, it is "jdbc::".

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
```

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 itself comprises the message flow name concatenated with the node label by means of the underscore character "_", that is:

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 use the account name string in the DataSource option of the `mqsisetdbparms` command this by prefixing the account name with the resource type, and concatenating the account name with an ampersand (&) character followed by the resource name itself:

<resource type><account name>&<resource name>

Therefore, assuming a JMS resource name of `tcf1`, used by JMSInput node `MyJMSInput1` in message flow `MyJMSFlow1`, the following DataSourceName is used:

`jms::MyJMSFlow1_MyJMSInput1&tcf1`

Specifying a user ID of `myuserid`, a password of `secret`, a broker name of `MyBroker1`, and the DataSourceName name created from the account name, as described above, use the following syntax:

```
mqsisetdbparms MyBroker1 -n jms::MyJMSFlow1_MyJMSInput1@tcf1
-u myuserid -p secret
```

To configure a security identity for a secure JDBC data source, enter the following command. `myDBID` is a JDBC data source security identity, which is a property that is configured for a `JDBCProvider` that is defined in the broker configurable services. The user ID and password are configured for the secure JDBC data source.

```
mqsisetdbparms MyBroker1 -n jdbc::myDBID -u myUserID -p myPassword
```

You can 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 *****
```

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 DataSourceName` must match the value of the `securityIdentity` property of the associated `JDBCProvider` configurable service.

```
mqsisetdbparms broker name -n datasource_name
-u userID -p password
```

For example:

```
mqsisetdbparms BROKER1 -n OracleXADatasource1 -u myuserid -p secretpw
```

mqsisetsecurity command

Supported platforms:

- Windows

Purpose:

Use the **mqsisetsecurity** command to create the Windows groups that WebSphere Message Broker requires for secure access to its runtime libraries and data.

This 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 MQM group provided you have **Administrator** authority.

Syntax:

```
▶▶ mqsisetsecurity —————▶▶
```

Parameters:

None

Authorization:

The user ID used to invoke this command must belong to the **Administrators** group on the local system because creating groups is an administrative task.

mqsi_setupdatabase command

Supported platforms:

- AIX
- HP-UX (PA-RISC platform)
- Solaris (SPARC platform)

Purpose:

Use the `mqsi_setupdatabase` command to set up database links for an Oracle database on AIX, HP-UX (PA-RISC platform), or Solaris (SPARC platform) systems that support 32-bit brokers. 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 broker database. If necessary, though, you can run the command before you install the Oracle database manager as long as you specify the intended database installation directory correctly.

Syntax:

```
►—mqsi_setupdatabase— Database—Database_Home_Directory—◄◄
```

Parameters:

Database

(Required) The database that you are installing. Supported values are:

- oracle9
- oracle10

Database_Home_Directory

(Required) The name of the directory in which the database is (or will be) installed (for example, `/usr/1pp/oracle9`).

Authorization:

The user ID used to run this command must be a member of the **mqbrkrs** group.

mqsistart command

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:

- “mqsisstart command - Windows, Linux and UNIX systems”
- “mqsisstart command - z/OS” on page 527

mqsisstart command - Windows, Linux and UNIX systems:

Purpose:

Use **mqsisstart** to start a WebSphere Message Broker component. If the queue manager associated with this component (defined in the corresponding create command) is not already running, it is also started by this command. However, no listeners, channels, or channel initiators associated with the started component are started. If you have WebSphere MQ Version 6.0 installed, use the WebSphere MQ Explorer to start any required listeners, channels, or channel initiators. For earlier supported versions of WebSphere MQ, use the WebSphere MQ Services snap-in.

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. Any 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 “Creating a Configuration Manager on Windows” on page 180. (If you lock your workstation, the queue manager does not stop).

Syntax:

```
►—mqsisstart—Component—————►
```

Parameters:

Component

(Required) The component must have a broker name, a Configuration Manager name, or one of the following fixed values:

- **Windows** UserNameServer or DatabaseInstanceMgr on Windows platforms
- **Linux** **UNIX** UserNameServer on Linux and UNIX systems

Linux **UNIX** All of the names are case sensitive on Linux and UNIX systems.

Do not use the mqsisstop 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:

```
mqsistart WBRK_BROKER
mqsistart DatabaseInstanceMgr
```

mqsistart command - z/OS:

Purpose:

The following table distinguishes between the **start (/S)** and **startcomponent** commands, and lists the available options:

- Use the **start (/S)** command to bring a component into a state in which you can run the appropriate **change** command.
- Use the **startcomponent** command to start a broker, Configuration Manager, or User Name Server when its controller (control process) is already running.

| Component | Command | Description |
|-----------------------|---|---|
| Broker | /S <Broker started task name>
/F <Broker started task name>,SC | Start broker.
Starts the broker from a 'stop component' state. |
| Configuration Manager | /S <Configuration Manager started task name>
/F <Configuration Manager started task name>,SC | Start 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 | Start User Name Server.
Starts the User Name Server from a 'stop component' state. |

When the controller address space is started, this causes the component to start automatically. This behavior can be changed by an optional start parameter in the started task. If the parameter is set to MAN, the component does not start automatically; the default is AUTO.

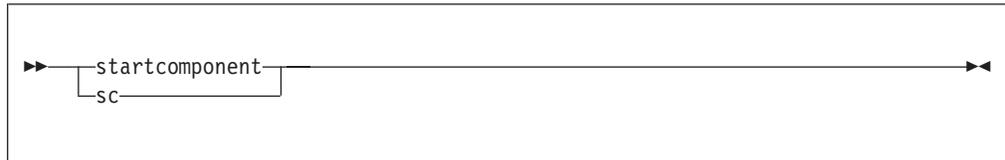
Issuing commands against the controller means issuing start, stop, or modify commands from the console to the controller address space. There are two scenarios using this command. The first is that the controller is started with the

parameter MAN instead of AUTO. The second is that, after a **stopcomponent** command, the component has to be restarted.

Syntax:

z/OS console command - startcomponent:

Synonym: sc



Examples:

F MQ00BRK,sc

mqsisstartmsgflow command

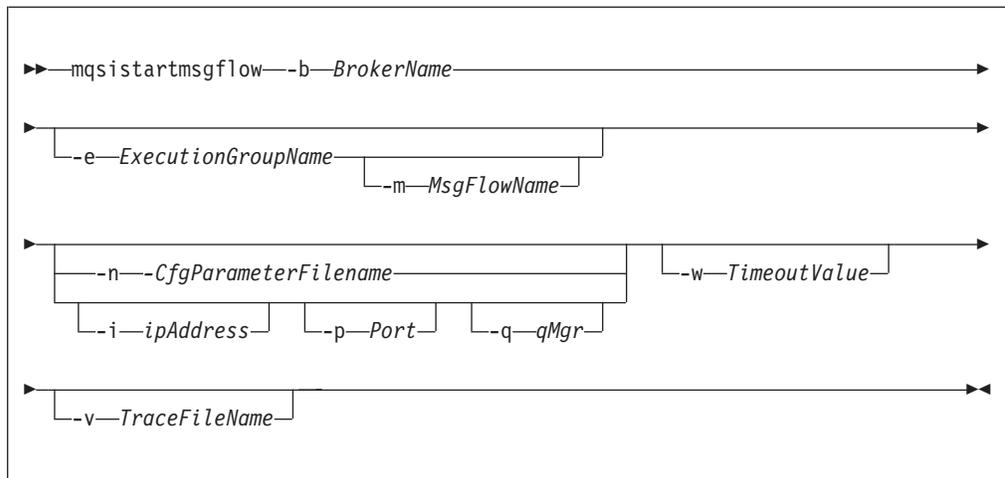
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPSTMF; see “Contents of the Configuration Manager PDSE” on page 550

Purpose:

Use the **mqsisstartmsgflow** command to start message flows.

Syntax:



Parameters:

-b *BrokerName*

(Required) The name of the broker on which to start message flows.

If you do not specify the **-e** and **-m** flags, all message flows on the broker are started.

-e *ExecutionGroupName*

(Optional) The name of the execution group on which message flows are started.

-m *MsgFlowName*

(Optional) The name of the message flow being started.

You can specify only one message flow in a single command. However, if you do not specify this parameter, all message flows on the execution group or broker are started.

If you specify this flag you must also specify the **-e** flag.

-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 as:

```
<?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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *Port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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:

To start message flows you must have sufficient authority defined in the Configuration Manager's access control list.

The permissions required are the same as the permission required to do the equivalent function in the Message Broker Toolkit; see “ACL permissions” on page 591 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

- 0 (Success) States that the request completed successfully and the state of all message flows has been updated.
- 2 (Failure) States that at least one message flow can not be put into the correct state for any reason.
- 98 States that the Configuration Manager cannot be reached.
- 99 States that the supplied arguments to the utility are not valid.

Examples:

Starts all message flows on broker B1, which is controlled by the Configuration Manager whose connection details are described in `cm1.configmgr`. Control is returned to the caller when all message flows in the broker are reported as started, or the default time of one minute elapses, whichever is sooner.

```
mqsistartmsgflow -n cm1.configmgr -b B1
```

Starts all message flows on broker B1, which is controlled by the Configuration Manager whose connection details are described in `cm1.configmgr`. Control is returned to the caller when all message flows in the broker are reported as started, or two minutes elapses, whichever is sooner.

```
mqsistartmsgflow -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 which is on local host:1414.

```
mqsistartmsgflow -q QM1 -i localhost -p 1414 -b B1
```

Enter **mqsistartmsgflow** to display usage information:

```
> mqsistartmsgflow
BIP1024I: Starts message flows.

> Syntax:
mqsistartmsgflow (-n cfgParameterFileName | ([-i ipAddress] [-p port] [-q qMgr]))
  -b brokerName [-e executionGroupName [-m flowName]] [-w timeoutValue]
  [-v traceFileName]
Command Options:
'-n cfgParameterFileName' File containing Configuration Manager connection parameters (.configmgr)
'-i ipAddress' IP address or host name of the Configuration Manager
'-p port' port number of the Configuration Manager
'-q qMgr' queue manager of the Configuration Manager
'-b brokerName' name of the broker on which to start message flows
'-e executionGroupName' name of the execution group on which to start message flows.
If this is not specified, all message flows on the broker will be started.
'-m flowName' name of the message flow to start.
If this is not specified, all message flows on the execution group will be started.
'-w timeoutValue' time to wait (in seconds) for message flows to start (Default=60)
'-v traceFileName' send verbose internal trace to the specified file.
```

mqsistop command

Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS 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 532

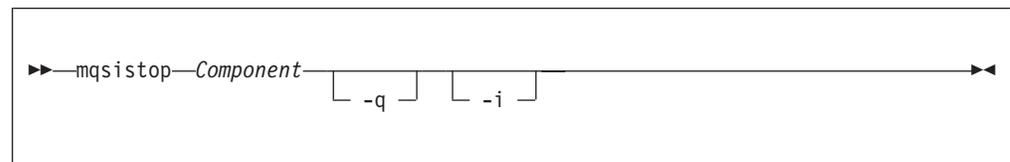
mqsistop command - Windows, Linux, and UNIX systems:

This topic shows you 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** UserNameServer or DatabaseInstanceMgr on Windows platforms
- **Linux** **UNIX** UserNameServer on Linux and UNIX systems

Linux **UNIX** All of the names are case sensitive on Linux and UNIX systems.

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 `mqsisstop` command.

-i (Optional) Immediately stops the broker.

Only specify this flag if you have already tried, and failed, to stop the broker in a controlled fashion using the `mqsisstop` command without the `-i` flag.

Authorization:

Windows On Windows platforms, the user ID used to start the `mqsisstop` command must belong to the **Administrators** group.

Linux **UNIX** On Linux and UNIX systems, the user ID used to invoke the `mqsisstop` command must conform to the following requirements:

- The user ID must be a member of the **mqbrkrs** group.
- The user ID must either be *root*, or be the same as the user ID that started the component.
- If you specify the `-q` parameter, the user ID must be a member of the **mqm** group.

The security requirements for using the `mqsisstop` command are summarized in the following topics:

- “Security requirements for Windows platforms” on page 593
- “Security requirements for Linux and UNIX platforms” on page 592

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:

Windows **Windows**

To stop the Database Instance manager:
`mqsisstop DatabaseInstanceMgr`

Windows **Linux** **UNIX** **Windows, Linux, and UNIX systems**

To stop the broker, *mybroker*, and the WebSphere MQ queue manager associated with it:

```
mqsisstop mybroker -q
```

mqsisstop command - z/OS:

Purpose:

Use the **mqsisstop** command to stop a WebSphere Message Broker component; the controller must be running.

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>/P <Broker started task name>
 /F <Broker started task name>,P

 /F <Broker started task name>,PC</p> | <p>Stop broker.</p> <p>Stop broker; this is the same as /P. You can also use /F <broker started task name>,STOP</p> <p>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 “mqsischangebroker command” on page 356 console command. Restart the broker afterwards by running the “mqsisstart command” on page 525 (SC) console command.</p> |
| Configuration Manager | <p>/P <Configuration Manager started task name>
 /F <Configuration Manager started task name>,P

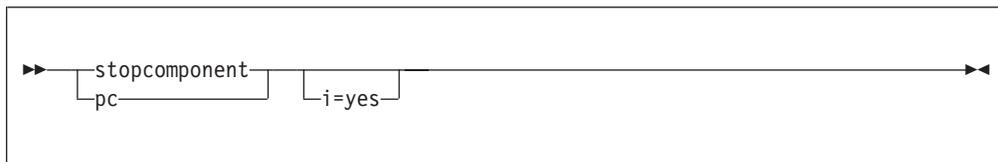
 /F <Configuration Manager started task name>,PC</p> | <p>Stop Configuration Manager.</p> <p>Stop Configuration Manager; this is the same as /P. You can also use /F <configMgr started task name>,STOP</p> <p>Stop Configuration Manager component. This stops the Configuration Manager process, but leaves the message broker console command server running inside the controller address space. This allows you to run the “mqsischangeconfigmgr command” on page 366 console command. Restart the broker afterwards by running the “mqsisstart command” on page 525 (SC) console command.</p> |
| User Name Server | <p>/P <User Name Server started task name>
 /F <User Name Server started task name>,P

 /F <User Name Server started task name>,PC</p> | <p>Stop User Name Server.</p> <p>Stop User Name Server; this is the same as /P. You can also use /F <User Name Server started task name>,STOP</p> <p>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 “mqsischangeusernameserver command” on page 404 console command. Restart the broker afterwards by running the “mqsisstart command” on page 525 (SC) console command.</p> |

Syntax:

z/OS console command - stopcomponent:

Synonym: pc



Parameters:

-i (Optional) Immediately stop the broker.

Only specify this flag if you have already tried, and failed, to stop the broker in a controlled fashion using the `mqsisstop` 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
```

mqsisstopmsgflow command

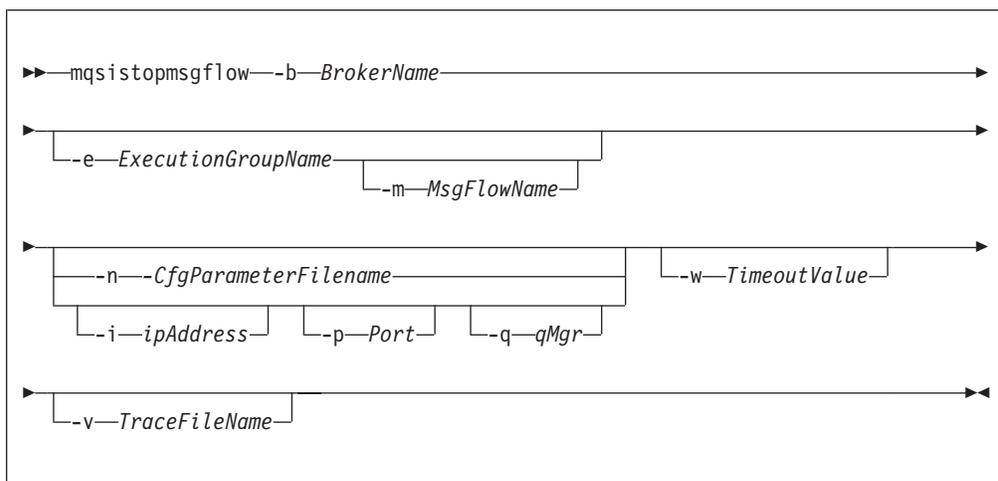
Supported platforms:

- Windows
- Linux and UNIX systems
- z/OS. Run this command by customizing and submitting BIPSPMF; see “Contents of the Configuration Manager PDSE” on page 550

Purpose:

Use the `mqsisstopmsgflow` command to stop message flows.

Syntax:



Parameters:

-b *BrokerName*

(Required) The name of the broker on which to stop message flows.

If you do not specify the **-e** and **-m** flags, all message flows on the broker are stopped.

-e *ExecutionGroupName*

(Optional) The name of the execution group on which message flows are stopped.

-m *MsgFlowName*

(Optional) The name of the message flow being stopped.

You can specify only one message flow in a single command. However, if you do not specify this parameter, all message flows on the execution group or broker are stopped.

If you specify this flag you must also specify the **-e** flag.

-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 as:

```
<?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 this parameter is not specified, the default value "" is used, which results in a local binding connection.

-p *Port*

(Optional) This parameter is the port number of the Configuration Manager. If this parameter is not specified, 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 this parameter is not specified, the default value "" 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:

To stop message flows you must have sufficient authority defined in the Configuration Manager's access control list.

The permissions required are the same as the permission required to do the equivalent function in the Message Broker Toolkit; see "ACL permissions" on page 591 for a list of permissions that can be defined in the Configuration Manager.

Responses:

This command returns the following responses:

- 0 (Success) States that the request completed successfully and the state of all message flows has been updated.
- 2 (Failure) States that at least one message flow can not be put into the correct state for any reason.
- 98 States that the Configuration Manager cannot be reached.
- 99 States that the supplied arguments to the utility are not valid.

Examples:

Stops all message flows on execution group default on broker B1, which is controlled by the Configuration Manager whose connection details are described in cml.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 cml.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 cml.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 cml.configmgr -b B1 -e default -m flow1
```

Enter **mqsistopmsgflow** to display usage information:

```
> mqsistopmsgflow
BIP1025I: Stops message flows.

> Syntax:
mqsistopmsgflow (-n cfgParameterFileName | ([-i ipAddress] [-p port] [-q qMgr]))
  -b brokerName [-e executionGroupName [-m flowName]] [-w timeoutValue]
  [-v traceFileName]
Command Options:
'-n cfgParameterFileName' Configuration Manager connection file (.configmgr)
'-i ipAddress' IP address or host name of the Configuration Manager
'-p port' port number of the Configuration Manager
'-q qMgr' queue manager of the Configuration Manager
'-b brokerName' name of the broker on which to stop message flows
'-e executionGroupName' name of the execution group on which to stop message flows.
If this is not specified, all message flows on the broker will be stopped.
'-m flowName' name of the message flow to start.
If this is not specified, all message flows on the execution group will be stopped.
'-w timeoutValue' time to wait (in seconds) for message flows to stop (Default=60)
'-v traceFileName' send verbose internal trace to the specified file.
```

z/OS specific information

Follow the links below for more information:

- “Administration in z/OS”
- “z/OS customization” on page 539
- “z/OS JCL variables” on page 551

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” on page 538
- “START and STOP commands on z/OS” on page 538

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='lowmf1''
//STEP1 EXEC PGM=IEFBR14
```

If your product for issuing console commands supports mixed case input, you might need to take special actions to use mixed case. For example in SDSF on OS/390® V2.10 and above, you can enter mixed case keyword values by typing /, pressing ENTER, and entering the command in the popup window. If you enter / followed by the command, the command is translated to uppercase. In NetView, prefix the command with NETVASIS to get lowercase support.

Guidance for issuing console commands in z/OS

These examples use a broker called MQP1BRK. Start and stop the broker or User Name Server using the MVS START (S) and STOP (P) commands. If you want to pass information to the broker or User Name Server while it is running, use the MVS MODIFY command (F) to issue commands. For example:

```
F MQP1BRK,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 “mqsicchangebroker command” on page 356. You can do this by using the MODIFY (/F) command on the started component, and using the startcomponent (SC) or stopcomponent (PC) options. See “mqsisstart command” on page 525 and “mqsisstop command” on page 531 for more information.

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 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/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 performed by other people in your organization are identified in the table below:

| Role | Task |
|-----------------------------------|---|
| z/OS systems programmer | "Customizing the z/OS environment" on page 137 |
| | "Setting up z/OS security" on page 65 |
| | "Summary of required access (z/OS)" on page 541 |
| | "Customizing UNIX System Services on z/OS" on page 148 |
| | "Creating the broker component" on page 174 |
| | "Starting and stopping a broker on z/OS" on page 288 |
| | "Checking APF attributes of bipimain on z/OS" on page 157 |
| DB2 administrator | "Setting up DB2 security on z/OS" on page 67 |
| | "Summary of required access (z/OS)" on page 541 |
| | "DB2 planning on z/OS" on page 150 |
| | "Priming DB2" on page 173 |
| WebSphere MQ administrator | "Setting up WebSphere MQ" on page 68 |
| | "Summary of required access (z/OS)" on page 541 |
| | "WebSphere MQ planning for z/OS" on page 153 |
| | "Creating the broker component" on page 174 |
| | "Defining the started tasks to z/OS Workload Manager (WLM)" on page 154 |

| | |
|---|---|
| WebSphere Message Broker administrator | "Setting up workbench access on z/OS" on page 69 |
| | "Summary of required access (z/OS)" |
| | "Creating a broker on z/OS" on page 167 |
| | "Creating the broker component" on page 174 |
| | "Checking APF attributes of bipimain on z/OS" on page 157 |
| | |
| Performance specialist | "Defining the started tasks to z/OS Workload Manager (WLM)" on page 154 |
| | |
| Security administrator | "Setting up z/OS security" on page 65 |
| | "Setting up DB2 security on z/OS" on page 67 |
| | "Summary of required access (z/OS)" |
| | "Creating Publish/Subscribe user IDs" on page 69 |
| | |
| Data administrator | "Setting up z/OS security" on page 65 |
| | "Summary of required access (z/OS)" |
| | "Disk space requirements on z/OS" on page 545 |
| | "Starting and stopping a broker on z/OS" on page 288 |
| | "Using the file system on z/OS" on page 142 |
| | "Binding a DB2 plan to use data-sharing groups on z/OS" on page 546 |

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 directory authorizations required for all WebSphere Message Broker components are:

- *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.

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.RRSF 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.RRSF)
```

and the following command gives the required access:

```
PERMIT DB2P.RRSF 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.

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 Configuration Manager Proxy 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 Configuration Manager Proxy 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 WebSphere MQ, 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 68.

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 WebSphere Message Broker for z/OS installation uses approximately 400 MB of disk space; you should plan on using 500 MB which will also allow for new service to be applied and also for the component directories.

When you apply service, if you do not replace your existing install (for example you apply the new fix pack level alongside your existing install) then you need to plan the same amount of disk space for the higher service level libraries.

If you are transferring the files using *tar* to package them, you need approximately 200 MB of space for the .tar file.

You can check how much space is used and how much is free in a file system 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 z/OS components.

| <i>Component</i> | <i>Space required</i> | <i>Purpose</i> |
|-------------------------------------|-----------------------|--|
| DB2 database (broker only) | 6 MB | Holds the broker system tables. |
| Component directory | 20 MB | Holds the runtime information and output directories for the component.

This includes trace files and other user problem determination data, which can become large. |
| Component PDSE | 1 MB | Holds the customization and administration jobs, procedures, and data for the component.

The data set must be allocated with a fixed record length of 80 (LRECL=80) and a format of FB 80. Reserve directory space for 50 members, or use a PDSE if possible. |
| Started task user ID home directory | 8 GB | Collect diagnostic materials, for example dumps. Dumps are usually more than 500MB in size.

8 GB of space needs to be available in the 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 different to ensure that, when dumps are taken in the Started task user ID home directory, they do not cause problems with the runtime broker that still needs to write to the Component directory.

Binding a DB2 plan to use data-sharing groups 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. For further information, see:

- “Installation information - broker and User Name Server” on page 168
- “Installation information - Configuration Manager” on page 181
- “DB2 information” on page 168
- “Component information - broker” on page 169
- “Component information - Configuration Manager” on page 182
- “Component information - User Name Server” on page 191

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 581 lists the shipped BIPDSNAO file. | BIPDSNAO |
| Job to run the broker DB2 database QUIESCE | BIPQSDB |
| Job to run the broker DB2 database report | BIPRPDB |

| Description | Name |
|--|----------|
| 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 mqsibrowse command. Use this command only at the request of IBM service. | BIPBRWS |
| Job to run the mqsichangebroker command | BIPCHBK |
| Job to run the mqsichangeflowstats command | BIPCHMS |
| Job to run the mqsichangeproperties command | BIPCHPR |
| Job to run the mqsichangeflowuserexits command | BIPCHUE |
| Removes an MQSeries Publish/Subscribe broker as a neighbor; runs the mqsiclearmqpubsub command | BIPCLMP |
| Job to run the mqsicreatebroker command to: <ul style="list-style-type: none"> • Create the DB2 tables • Create the WebSphere MQ resources • Create the broker registry “Sample BIPCRBK file” on page 563 lists the shipped BIPCRBK file. | BIPCRBK |
| Job to create the DB2 storage group, database and table spaces.
“Sample BIPCRDB file” on page 571 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 582 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 584 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 mqsireadlog command. | BIPRELG |
| Job to run the mqsireportflowstats command | BIPRPMS |
| Job to run the mqsireportproperties command | BIPRPPR |
| Job to run the mqsireportflowuserexits command | BIPRPUE |

| | |
|---|---------|
| Job 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 560 lists the shipped BIPBRKP file. | BIPBRKP |

Use the standard z/OS IPCS facilities to take a dump. You require access to the following:

- COMPONENT_PDSE
- ComponentDirectory
- SYS1.MIGLIB
- SYS1.SBLSCLI0
- SYS1.PARMLIB
- IPCS

Contents of the User Name Server PDSE

After successfully customizing the User Name Server, the members in your User Name Server PDSE are as follows:

User Name Server PDSE members originating in <hlq>.SBIPSAMP

| Description | Name |
|---|---------|
| User Name Server profile

"Sample BIPUPROF file" on page 588 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: <ul style="list-style-type: none"> • Create the WebSphere MQ resources • Create the User Name Server registry "Sample BIPCRUN file" on page 580 lists the shipped BIPCRUN file. | BIPCRUN |
| Job to run the mqsdeleteusernameserver 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 582 lists the shipped BIPEDIT file. | BIPEDIT |
| Job to run the mqsiformatlog command. | BIPFMLG |
| Generate ENVFILE.

"Sample BIPGEN file" on page 584 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 585 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

After successfully customizing the Configuration Manager, the members in your Configuration Manager PDSE are as follows:

Configuration Manager PDSE members originating in <hlq>.SBIPSAMP

| Description | Name |
|--|----------|
| Configuration Manager profile

"Sample BIPCPROF file" on page 567 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 | BIPBUCM |
| Job to run the mqsichangeconfigmgr command | BIPCHCM |
| Job to run the mqsicreateaclentry command | BIPCRACL |
| Job to run the mqsicreateconfigmgr command to: <ul style="list-style-type: none"> • Create the WebSphere MQ resources • Create the Configuration Manager registry
"Sample BIPCRCM file" on page 570 lists the shipped BIPCRCM file. | BIPCRCM |
| Job to run the mqsicreateexecutiongroup command | BIPCREG |
| Job to run the mqsdeleteaclentry command | BIPDLACL |
| Job to run the mqsdeleteconfigmgr command | BIPDLCM |
| Job to run the mqsdeleteexecutiongroup command | BIPDLEG |
| Job to run the mqsdeploy command | BIPDPLY |
| 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 582 lists the shipped BIPEDIT file. | BIPEDIT |

| | |
|--|----------|
| Job to run the mqsiformatlog command. | BIPFMLG |
| Job to convert BIPCPROF profile to a valid ENVFILE.
"Sample BIPGEN file" on page 584 lists the shipped BIPGEN file. | BIPGEN |
| Job to run the mqsilistaclentry command | BIPLIACL |
| Job to run the mqsilist command | BIPLIST |
| Job to run the mqsireadlog command. | BIPRELG |
| Job to run the mqsireloadsecurity command. | BIPRLSEC |
| Job to run the mqsirestoreconfigmgr command | BIPRSCM |
| Job to run the mqsistopmsgflow command | BIPSPMF |
| Job to run the mqsistartmsgflow command | BIPSTMF |
| Started task. Rename BIPCMGRP to the same as the
++STARTEDTASKNAME++ .

"Sample BIPCMGRP file" on page 565 lists the shipped BIPCMGRP file. | BIPCMGRP |

Use the standard z/OS IPCS facilities to take a dump. You require access to the following:

- COMPONENT_PDSE
- ComponentDirectory
- SYS1.MIGLIB
- SYS1.SBLSCLI0
- SYS1.PARMLIB
- IPCS

z/OS JCL variables

The following table lists all the JCL variables that you can customize in alphabetical order, together with a description and an example value.

| JCL variable | Description | Example value |
|------------------------|---|------------------------|
| ++ACTIVEUSEREXITLIST++ | Active user exit list | 'MyExit1: MyExit2' |
| ++ARM++ | Specifies if a component should use Automatic Restart Management (ARM). | YES |
| ++ARMNAME++ | ARM name. Required if ++ARM++ is set to YES. | Q482BRK |
| ++ARMTYPE++ | ARM type. Required if ++ARM++ is set to YES. | SYSWMQI |
| ++COMPONENTDATASET++ | The dataset where all JCL relevant to a particular component is saved. | TESTDEV.MQP1BRK.BROKER |

| JCL variable | Description | Example value |
|------------------------|---|---------------------------------|
| ++COMPONENTDIRECTORY++ | The file system directory where the component exists. This directory includes subdirectories, for example, /log and /registry | mqs/brokers/MQP1BRK |
| ++COMPONENTNAME++ | The name you give the component when you create it. | MQP1BRK |
| ++COMPONENTPROFILE++ | Profile name. | BIPBPROF, BIPCPROF, or BIPUPROF |
| ++COMPONENTRESOURCE++ | Required specifically by BIPBRWS. It represents the resource to be displayed. | BNBRCONNECTIONS |
| ++DB2BUFFERPOOL++ | The name of the DB2 buffer pool associated with the component for which the JCL is submitted. | BP0 |
| ++DB2CONVERSION++ | Specifies the DB2 Converter. | SINGLE |
| ++DB2CURRENTSQLID++ | The DB2 user ID for the component and commands. | MQP1BRK |
| ++DB2DATABASE++ | The name of the DB2 database associated with the component for which the JCL is submitted. | DMQP1BRK |
| ++DB2HLQ++ | DB2 high-level-qualifier | SYS2.DB2.V810 |
| ++DB2INDEXBP++ | DB2 index bufferpool | BP0 |
| ++DB2LOBBP++ | DB2 LOB table bufferpool | BP0 |
| ++DB2LOCATION++ | The DB2 location value of the DB2 subsystem to which the component connects. | DSN810PK |
| ++DB2PLAN++ | The DB2 plan value. | DSNTEP81 |
| ++DB2PLANNAME++ | The DB2 plan name. | DSNACLI |
| ++DB2PROGRAM++ | The DB2 program value. | DSNTEP2 |
| ++DB2RUNLIB++ | The DB2 run library value. | DSN810PK.RUNLIB.LOAD |
| ++DB2STORAGEGROUP++ | The name of the DB2 storage group associated with the component for which the JCL is submitted. | MQP1STOR |

| JCL variable | Description | Example value |
|---------------------------|---|--------------------------|
| ++DB2SUBSYSTEM++ | The DB2 subsystem ID which the component connects. | DFK4 |
| ++DB2TABLEOWNER++ | Broker tables schema name. | MQP1BRK |
| ++EXECUTIONGROUPNAME++ | The name of an execution group. | Default |
| ++HOME++ | The file system home directory for the component's user ID. This is required to dynamically generate the ENVFILE from BIPBPROF, and is also used for STDOUT and STDERR output.

You must have the appropriate RACF authorities to write to this file system directory when submitting JCL to run a command. | /u/mqp1brk |
| ++INACTIVEUSEREXITLIST++ | Inactive user exit list | 'OtherExitA: OtherExitB' |
| ++INSTALL++ | The directory where you install the product. | /usr/lpp/mqsi |
| ++JAVA++ | Location of Java installation. | /usr/lpp/java/IBM/J1.5 |
| ++LANGLETTER++ | The letter for the language in which you want messages shown. | E (English) |
| ++LOCALE++ | Locale of environment where commands are run by submitting JCL. | C |
| ++MESSAGECASE++ | Determines if messages should appear in mixed case. | YES |
| ++MESSAGEFLOWNAME++ | Name of the message flow | MyTestMsgFlow |
| ++MQPATH++ | WebSphere MQ location | /usr/lpp/mqm |
| ++NEIGHBOURQUEUEMANAGER++ | Required specifically by BIPCLMP for the mqsiclearmqpubsub command. It represents the name of the neighboring QueueManager | MQP1 |

| JCL variable | Description | Example value |
|----------------------------|--|----------------------------|
| ++OBJECTNAME++ | Required specifically by BIPRPPR for the mqsireportproperties command. It represents the name of the object. | DynamicSubscriptionEngine |
| ++OPTIONS++ | Many commands submitted by JCL require additional options. See each command's reference material for additional information on options specific to that command. | N/A |
| ++PARENTQUEUEMANAGERNAME++ | Required specifically by BIPJNMP for the mqsijoinmqpubsub command. It represents the name of the parent QueueManager. | MQP1 |
| ++PROPERTYNAME++ | Required specifically by BIPCHPR for the mqsichangeproperties command. It represents the name of the property to be changed. | multicastEnabled |
| ++PROPERTYVALUE++ | Required specifically by BIPCHPR for the mqsichangeproperties command. It represents the new value of the property to be changed. | false |
| ++QUEUEMANAGER++ | The name of the QueueManager associated with the component for which you submit the JCL. | MQP1 |
| ++STARTEDTASKNAME++ | Name of the Started Task JCL. This can be a maximum of 8 characters. | MQP1BRK |
| ++TIMEZONE++ | Time zone of environment where commands are run by submitting JCL. | GMT0BST |
| ++WMQHLQ++ | WebSphere MQ high-level-qualifier | MQM.V600 |
| ++XMLTOOLKIT++ | IBM XML Toolkit location | /usr/lpp/ixm/IBM/xml4c-5_6 |

z/OS sample files supplied

The following files are provided as sample source that you can customize:

- “Sample BIPBPROF file”
- “Sample BIPBRKP file” on page 560
- “Sample BIPCBRK file” on page 563
- “Sample BIPCMGRP file” on page 565
- “Sample BIPCPROF file” on page 567
- “Sample BIPCRCM file” on page 570
- “Sample BIPCRDB file” on page 571
- “Sample BIPCRUN file” on page 580
- “Sample BIPDSNAO file” on page 581
- “Sample BIPEDIT file” on page 582
- “Sample BIPGEN file” on page 584
- “Sample BIPUNSP file” on page 585
- “Sample BIPUPROF file” on page 588

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/bipcpv 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)
//*****
//*
//* RMLLOCKS 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..SDSNLOAD2
// 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/bipcvc 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=*
/**SYSDUMP  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
/*******
/**
/**BIPCRMB 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 bufferpool. *
/** e.g. 'BP0' *
/** *
/** Replace ++DB2INDEXBP++ *
/** DB2 index bufferpool. *
/** e.g. 'BP0' *
/** *
/** Replace ++DB2LOBBP++ *
/** DB2 LOB table bufferpool. *
/** e.g. 'BP0' *
/**

```

```

/* Replace ++DB2SAMPLEPROGRAM++
/*          DB2 sample program name.
/*          e.g. 'DSNTEP2'
/*
/* Replace ++DB2SAMPLEPROGRAMPLAN++
/*          DB2 sample program plan name.
/*          e.g. 'DSNTEP81'
/*
/******
/*
/******
/* Create broker storagegroup
/******
/*
//STORGRP EXEC PGM=IKJEFT01,REGION=0M
/*          DB2 Runtime Libraries
//STEPLIB DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//          DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(++DB2SUBSYSTEM++)
  RUN -
    PROGRAM(++DB2SAMPLEPROGRAM++) -
    PLAN(++DB2SAMPLEPROGRAMPLAN++)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
  SET CURRENT SQLID='++DB2CURRENTSQLID++';
  CREATE STOGROUP ++DB2STORAGEGROUP++
    VOLUMES('*')
    VCAT ++DB2LOCATION++;
/*
/*
/******
/* Create broker database
/******
/*
//DATABASE EXEC PGM=IKJEFT01,REGION=0M
/*          DB2 Runtime Libraries
//STEPLIB DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT
//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//          DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM(++DB2SUBSYSTEM++)
  RUN -
    PROGRAM(++DB2SAMPLEPROGRAM++) -
    PLAN(++DB2SAMPLEPROGRAMPLAN++)
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
  SET CURRENT SQLID='++DB2CURRENTSQLID++';
  CREATE DATABASE ++DB2DATABASE++
    STOGROUP ++DB2STORAGEGROUP++
    INDEXBP ++DB2INDEXBP++;
/*
/*
/******
/* Create broker tablespaces
/******
/*
//TABLESPC EXEC PGM=IKJEFT01,REGION=0M
/*          DB2 Runtime Libraries
//STEPLIB DD DISP=SHR,DSN=++DB2HLQ++.SDSNEXIT

```

```

//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOAD
//          DD DISP=SHR,DSN=++DB2HLQ++.SDSNLOD2
//          DD DISP=SHR,DSN=++DB2RUNLIB++
//SYSTSPRT DD SYSOUT=*
//SYSTEM   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 BACLENTN
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 BROKERRA
    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 BROKERRAE
    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 BROKERRER
    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 BAGGRL0B
  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 documentation 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++ dbds subsystem_value    all"
"change ++DB2LOCATION++ db2location_value        all"
"change ++DB2TABLEOWNER++ MQ01GRP              all"
"change ++DB2CURRENTSQLID++ MQ01GRP            all"
"change ++DB2DSNACLIPLAN++ dsnaccli_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 ++DB2STORAGEEGROUP++ 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='bipmain',
//    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)
/*****
/**
/**RMLLOCKS 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

This section contains the following topics:

- “ACL permissions”
- “Security requirements for Linux and UNIX platforms” on page 592
- “Security requirements for Windows platforms” on page 593
- “Security requirements for z/OS” on page 596

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 | <ul style="list-style-type: none">• Create and delete brokers.• Create and delete collectives.• Add and remove brokers from collectives.• Create and delete connections.• Deploy topology.• All topology View permission rights |
| | View | <ul style="list-style-type: none">• View topology configuration and managed subcomponents. |
| Broker | Full control | <ul style="list-style-type: none">• 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 | <ul style="list-style-type: none">• Deploy broker configuration.• All broker View permission rights. |
| | View | <ul style="list-style-type: none">• View broker configuration and managed subcomponents.• Implicit view access to Topology. |
| Execution group | Full control | <ul style="list-style-type: none">• Edit all execution group properties.• Start and stop execution groups.• All execution group Deploy permission rights.• All execution group View permission rights. |
| | Deploy | <ul style="list-style-type: none">• Deploy execution group configuration.• Start and stop assigned message flows.• Start and stop trace.• All execution group View permission rights. |
| | View | <ul style="list-style-type: none">• View execution group configuration and managed subcomponents.• Implicit View access to parent broker and topology. |

| Object | Permission | Rights |
|--------------|--------------|---|
| Root topic | Full control | <ul style="list-style-type: none"> • Edit "Topic Access Control List". • All root topic Deploy permissions. • All root topic Edit permissions. • All root topic View permissions. |
| | Deploy | <ul style="list-style-type: none"> • Deploy entire topic configuration. • All root topic View permissions. |
| | Edit | <ul style="list-style-type: none"> • Create and delete child topics. • All root topic View permissions. |
| | View | <ul style="list-style-type: none"> • View all topics (including child topics), and any managed subcomponents. |
| Subscription | Full control | <ul style="list-style-type: none"> • Delete any subscription. • All subscription "View" permissions. |
| | View | <ul style="list-style-type: none"> • View or query all subscriptions and any managed subcomponents. |

Security requirements for Linux and UNIX platforms

A summary of the authorizations in a Linux or UNIX environment.

| User is... | Linux or UNIX domain |
|--|---|
| Creating a component | <ul style="list-style-type: none"> • Member of mqbrkrs and mqm. • When root is used to issue the create command, it can nominate any user to run the component. |
| Installing | <ul style="list-style-type: none"> • Superuser. |
| Uninstalling | <ul style="list-style-type: none"> • Superuser. |
| Changing a component | <ul style="list-style-type: none"> • Member of mqbrkrs. |
| Deleting a component | <ul style="list-style-type: none"> • Member of mqbrkrs and mqm. |
| Starting a component | <ul style="list-style-type: none"> • Member of mqbrkrs. • Member of mqm. |
| Stopping a component | <ul style="list-style-type: none"> • Member of mqbrkrs. • The user ID must either be root, or the same as the user ID that started the component. • Member of mqm if -q is specified. |
| Listing a component | <ul style="list-style-type: none"> • Member of mqbrkrs. |
| Changing, displaying, retrieving trace information. | <ul style="list-style-type: none"> • Member of mqbrkrs. |
| Running User Name Server (login ID). | <ul style="list-style-type: none"> • Member of mqbrkrs. The User Name Server runs under the login ID specified in the create command. |
| Running broker (WebSphere MQ non-trusted application) (login ID). | <ul style="list-style-type: none"> • Member of mqbrkrs. • The broker runs under the login ID that started it. • If root started the component then the broker runs under the login ID specified as the service ID in the create command. |
| Running broker (WebSphere MQ trusted application) (login ID). | <ul style="list-style-type: none"> • Login ID must be mqm. • mqm must be a member of mqbrkrs. |
| Clearing, joining, listing WebSphere MQ publish/subscribe brokers. | <ul style="list-style-type: none"> • Member of mqbrkrs. |

| | |
|---|---|
| User is... | Linux or UNIX domain |
| Running publish/subscribe applications. | <ul style="list-style-type: none"> Any user, subject to topic and WebSphere MQ queue access control. |

Note: When the service user ID is root, all of the libraries loaded by the broker, including all of the user-written plug-in libraries 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

The following table summarizes the security requirements for the WebSphere Message Broker administrative tasks. It illustrates what group membership is required if you are using a local security domain defined on your local system SALONE, or a primary domain named PRIMARY, or a trusted domain named TRUSTED. The contents of this table assume that you have created both the Configuration Manager and the User Name Server with the same security domain.

| User is... | Local domain (SALONE) | Primary domain (PRIMARY) / Windows Single domain (PRIMARY) | Trusted domain (TRUSTED) / Windows Parent/Child domain in domain tree (TRUSTED) |
|---|--|--|--|
| Creating a broker, Configuration Manager, User Name Server, or database (with mqsicreatedb) | <ul style="list-style-type: none"> Must be a user ID defined in SALONE Member of Administrators | <ul style="list-style-type: none"> Must be a user ID defined in PRIMARY Member of SALONE\Administrators | <ul style="list-style-type: none"> Must be a user ID defined in TRUSTED Member of SALONE\Administrators |
| Changing a broker, Configuration Manager, User Name Server, DatabaseInstanceMgr | <ul style="list-style-type: none"> Must be a user ID defined in SALONE Member of Administrators | <ul style="list-style-type: none"> Must be a user ID defined in PRIMARY Member of SALONE\Administrators | <ul style="list-style-type: none"> Must be a user ID defined in TRUSTED Member of SALONE\Administrators |
| Deleting a broker, Configuration Manager, User Name Server, or database (with mqsideletedb) | <ul style="list-style-type: none"> Member of Administrators | <ul style="list-style-type: none"> Member of SALONE\Administrators | <ul style="list-style-type: none"> Member of SALONE\Administrators |
| Starting a broker, Configuration Manager, User Name Server, or DatabaseInstanceMgr | <ul style="list-style-type: none"> Member of Administrators | <ul style="list-style-type: none"> Member of SALONE\Administrators | <ul style="list-style-type: none"> Member of SALONE\Administrators |
| Listing a broker, Configuration Manager, User Name Server, or DatabaseInstanceMgr | <ul style="list-style-type: none"> Must be a user ID defined in SALONE User ID must have the authority to query the registry values under <i>WebSphereMQIntegrator</i> entry in the registry. Member of mqrbrks if issuing the command: mqsilist <broker name><execution group name>. | <ul style="list-style-type: none"> Must be a user ID defined in PRIMARY User ID must have the authority to query the registry values under <i>WebSphereMQIntegrator</i> entry in the registry. Member of PRIMARY\Domain mqrbrks if issuing the command: mqsilist <broker name><execution group name>. | <ul style="list-style-type: none"> Must be a user ID defined in TRUSTED User ID must have the authority to query the registry values under <i>WebSphereMQIntegrator</i> entry in the registry. Member of TRUSTED\Domain mqrbrks if issuing the command: mqsilist <broker name><execution group name>. |

| User is... | Local domain (SALONE) | Primary domain (PRIMARY) / Windows Single domain (PRIMARY) | Trusted domain (TRUSTED) / Windows Parent/Child domain in domain tree (TRUSTED) |
|--|--|---|---|
| Changing, displaying, retrieving trace information | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs |
| Running a User Name Server (service user ID) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs |
| Running a DatabaseInstanceMgr (service user ID) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs |
| Running a Configuration Manager (service user ID) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs • Member of mqm • Member of Administrators | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Member of SALONE\mqm (see note 1) • Member of SALONE/Administrators | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Member of SALONE\mqm (see note 2) • Member of SALONE/Administrators |
| Running a broker (WebSphere MQ fastpath off) (service user ID) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs |
| Running a broker (WebSphere MQ fastpath on) (service user ID) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs • Member of mqm | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs • Member of SALONE\mqm | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs • Member of SALONE\mqm |
| Clearing, joining, or listing WebSphere MQ Publish/Subscribe brokers | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE • Member of mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY • Member of PRIMARY\Domain mqbrkrs | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED • Member of TRUSTED\Domain mqbrkrs |
| Running a Message Brokers Toolkit (see note 3) | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE (see note 4). For example, SALONE\User1 is valid, PRIMARY\User2 and TRUSTED\User3 are not. | <ul style="list-style-type: none"> • For both domain awareness enabled and domain awareness disabled, when using Message Brokers Toolkit ACLs, user IDs must be members of any local ACL groups created on SALONE. | <ul style="list-style-type: none"> • For both domain awareness enabled and domain awareness disabled, when using Message Brokers Toolkit ACLs, user IDs must be members of any local ACL groups created on SALONE. |
| Running publish/subscribe applications | <ul style="list-style-type: none"> • Must be a user ID defined in SALONE. For example, SALONE\User1 is valid, PRIMARY\User2 and TRUSTED\User3 are not. | <ul style="list-style-type: none"> • Must be a user ID defined in PRIMARY. For example, PRIMARY\User2 is valid, SALONE\User1 and TRUSTED\User3 are not. | <ul style="list-style-type: none"> • Must be a user ID defined in TRUSTED. For example, TRUSTED\User3 is valid, SALONE\User1 and PRIMARY\User2 are not. |

Notes:

1. If you are running in a primary domain, you can also:
 - Define the user ID in the domain PRIMARY.

- Add this ID to the group **PRIMARY\Domain mqm**.
 - Add the **PRIMARY\Domain mqm** group to the group **SALONE\mqm** .
2. If you are running in a trusted domain, you can also:
 - Define the user ID in the domain TRUSTED.
 - Add this ID to the group **TRUSTED\Domain mqm**.
 - Add the **TRUSTED\Domain mqm** group to the group **SALONE\mqm** .
 3. All Message Brokers Toolkit users need read access to the WebSphere MQ java \lib subdirectory of the WebSphere MQ home directory (the default is X:\Program Files \WebSphere MQ , where X: is the operating system disk). This access is restricted to users in the local group **mqm** by WebSphere MQ. WebSphere Message Broker installation overrides this restriction and gives read access for this subdirectory to all users.
 4. If a valid user ID is defined in the domain used by the Configuration Manager (for example, PRIMARY\User4), an identical user defined in a different domain (for example, DOMAIN2\User4) can access the Message Brokers Toolkit with the authorities of PRIMARY\User4.
 5. Ensure that the service user ID has the required access to relevant directories of the product directory tree; for example, write access to the logs directory. If a workpath other than the default has been set for any component, ensure that the services user ID has appropriate access to this location.
 6. If you are running a Configuration Manager with one user ID and a broker with a different user ID on another computer, you might see an error message when trying to deploy message flows and message sets to the broker. To avoid this, do the following:
 - Ensure that the broker's user ID is a member of the **mqm** and **mqbrkrs** groups.
 - Define the broker's user ID on the computer where the Configuration Manager is running.
 - Define the Configuration Manager's user ID on the computer where the broker is running.
 - Ensure that all IDs are in lowercase so that they are compatible between computers.

Broker security changes with Windows 2000 and Windows XP

On Windows 2000 and Windows XP, 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.

To edit the registry on Windows 2000 and Windows XP:

- **Windows 2000**
 1. Run **regedt32**
 2. Select **HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebsphereMQIntegrator**
 3. Select **Permissions** from the Security menu in the title bar

4. Click **Add** on the Permissions for WebsphereMQIntegrator window. From the popup list, select the **mqbrkrs** group and click **Add**. Click **OK**.
 5. Click **Advanced** on the Permissions for WebsphereMQIntegrator window
 6. On the Access Control Settings for WebSphereMQIntegrator window, highlight **mqbrkrs** and click **View/Edit**
 7. On the Permission Entry for WebSphereMQIntegrator window, select **Set Value, Create Subkey, and Delete**. Ensure that the **Allow inheritable permissions** box is ticked and then click **OK**
 8. Click **OK** on the remaining screens to exit the editing of the registry
 9. Exit regedt32.
- **Windows XP**
 1. Run regedt32
 2. Select HKEY_LOCAL_MACHINE\SOFTWARE\IBM\WebsphereMQIntegrator
 3. Select **Permissions** by right-clicking on **WebsphereMQIntegrator**
 4. Click **Add** on the Permissions for WebsphereMQIntegrator window. From the Select Users or Groups window, select **Advanced** and then **Find Now** to list users/groups. From the list, select the **mqbrkrs** group and click **OK** twice
 5. Click **Advanced** on the Permissions for WebsphereMQIntegrator window
 6. On the Advanced Security Settings for WebSphereMQIntegrator window, highlight **mqbrkrs** and click **Edit**
 7. On the Permission Entry for WebSphereMQIntegrator window, select **Set Value, Create Subkey, and Delete**. Ensure the **Inherit from parent** box is ticked and then click **OK**
 8. Click **OK** on the remaining screens to exit the editing of the registry
 9. Exit regedt32.

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 | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> 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 | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the installation directory by the z/OS user ID installing the product |
| Uninstalling | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the installation directory by the z/OS user ID uninstalling the product |
| Changing broker, User Name Server, Configuration Manager | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS user ID issuing the command |
| Deleting broker, User Name Server, Configuration Manager | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS user ID issuing the command |
| Starting broker, User Name Server, Configuration Manager | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS assigned started task user ID |
| Stopping broker, User Name Server, Configuration Manager | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS assigned started task user ID |

| User is... | File access |
|--|--|
| Listing broker, User Name Server, Configuration Manager | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS user ID issuing the command |
| Changing, displaying, retrieving trace information. | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS user ID issuing the command |
| Clearing, joining, listing WebSphere MQ publish/subscribe brokers. | <ul style="list-style-type: none"> • <i>READ</i> and <i>WRITE</i> access to the component directory by the z/OS assigned started task user ID • Member of mqbrkrs group |
| Running publish/subscribe applications. | <ul style="list-style-type: none"> • 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

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

| | | |
|-------------|-------------------------------------|----------------|
| AIX | CICS | DB2 |
| DB2 Connect | DB2 Universal Database | developerWorks |
| Domino | | Everyplace |
| FFST | First Failure Support
Technology | IBM |
| IBMLink | IMS | IMS/ESA |
| iSeries | Language Environment | Lotus |
| MQSeries | MVS | NetView |
| OS/400 | OS/390 | POWER |
| pSeries | RACF | Rational |
| Redbooks | RETAIN | RS/6000 |
| SupportPac | S/390 | Tivoli |
| VisualAge | WebSphere | xSeries |
| z/OS | zSeries | |

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

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

Intel and Pentium are trademarks or registered trademark of Intel Corporation or its subsidiaries in the United States and other countries.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Index

A

- access control lists 70
 - permissions 591
- administration
 - broker domain 283
 - security requirements 591
 - z/OS 537
- authentication 10
 - configuring 36
 - LDAP 36
 - TFIM 37
 - implementing 52
 - services 75
 - SSL 25
- authorization 11
 - configuring 39
 - LDAP 39
 - TFIM 41
 - for configuration tasks 16

B

- backup
 - broker domain
 - distributed systems 299
 - z/OS 300
 - resources 298
 - Message Brokers toolkit workspace 301
- Broker Administration perspective
 - changing preferences 271
- broker database
 - changing access 215
 - connections 102
 - failure 215
- broker domain
 - adding a broker 227
 - administration 283
 - configuring 89
 - components 162
 - in the workbench 222
 - configuring locales 273
 - code page converters 276
 - generating code page converters 276
 - UNIX 273
 - using converters from a previous product level 278
 - Windows 275
 - z/OS 275
- connecting and disconnecting 283
- connecting and disconnecting on z/OS 284
- default configuration 200
- designing 89
- log information
 - clearing 296
 - filtering 294
 - refreshing 293
 - saving 295

- broker domain (*continued*)
 - log information (*continued*)
 - viewing 293
 - removing a broker 230
- broker domains
 - security 45
- broker networks 236
 - heterogeneous 239
 - migrated 239
- broker statistics, collecting on z/OS 157
- Broker Topology editor
 - changing properties 240
- brokers
 - adding 227
 - changing 207
 - Linux and UNIX systems 208
 - Windows 208
 - z/OS 209
 - cloned 238
 - connecting a user name server 196
 - connecting in a collective 240
 - copying 228
 - creating 163
 - Linux 165
 - UNIX 165
 - Windows 166
 - customizing a new broker on z/OS 167
 - component dataset operations 171
 - Component information 169
 - copying the started task 174
 - creating the broker
 - component 174
 - creating the broker directory 170
 - creating the broker PDSE 170
 - creating the environment file 172
 - customizing the JCL 171
 - DB2 information 168
 - installation information 168, 190
 - JCL variables 551
 - priming DB2 173
 - required information 168
 - sample files 555
- deleting 217
 - Linux and UNIX systems 218
 - Windows 218
 - z/OS 218
- execution groups, adding 232
- execution groups, copying 233
- modifying 207
 - Linux and UNIX systems 208
 - Windows 208
 - z/OS 209
- properties 323
 - changing 229
- removing 230
- removing deployed children 231
- renaming 230
- sample files, z/OS
 - BIPBPROF 555

brokers (*continued*)

- sample files, z/OS (*continued*)
 - BIPBRKP 560
 - BIPCBRK 563
 - BIPCRDB 571
 - BIPDSNAO 581
 - BIPEDIT 582
 - BIPGEN 584
- security 48
- service ID 48
- starting and stopping 287
 - UNIX 287
 - Windows 287
 - z/OS 288
- user name server, connecting on z/OS 197
- WebSphere MQ resources 93

C

- c security on z/OS 199
- channels, starting WebSphere MQ 198
- characters allowed in commands 333
- cloned brokers 238
 - adding 250
 - defining 249
 - deleting 251
- code page converters 276
 - new 276
 - using converters from a previous product level 278
- collectives 236
 - adding a broker 241
 - creating 240
 - deleting 241
 - removing a broker 242
- Command Assistant wizard 202
- commands 324
 - characters allowed in 333
 - mqsicreatedb 438
 - responses to 335
 - rules for using 334
 - runtime 354
 - mqsi_setupdatabase 525
 - mqsibackupconfigmgr 354
 - mqsicbrreport 356
 - mqsichangebroker 356
 - mqsichangeconfigmgr 366
 - mqsichangedbimgr 371
 - mqsichangeflowstats 372
 - mqsichangeflowuserexits 377
 - mqsichangeproperties 381
 - mqsichangetrace 397
 - mqsichangeusernameserver 404
 - mqsiclearmqpubsub 408
 - mqsicreateaclentry 409
 - mqsicreatebroker 416
 - mqsicreateconfigmgr 428
 - mqsicreateconfigurableservice 435
 - mqsicreatedb 438
 - mqsicreateexecutiongroup 440

- commands (*continued*)
 - runtime (*continued*)
 - mqscreateusernameserver 442
 - mqsdeleteaclentry 448
 - mqsdeletebroker 454
 - mqsdeleteconfigmgr 456
 - mqsdeleteconfigurableservice 460
 - mqsdeletedb 461
 - mqsdeleteexecutiongroup 462
 - mqsdeleteusernameserver 464
 - mqsdeploy 465
 - mqsifformatlog 474
 - mqsjoinmqpubsub 475
 - mqsilist 477
 - mqsilistaclentry 479
 - mqsilistmqpubsub 486
 - mqsimanagexalinks 488
 - mqsigratecomponents 491
 - mqsireadlog 496
 - mqsireload 500
 - mqsireloadsecurity 501
 - mqsireportflowstats 504
 - mqsireportflowuserexits 507
 - mqsireportproperties 509
 - mqsireporttrace 516
 - mqsirestoreconfigmgr 519
 - mqsisetdbparms 520
 - mqsisetsecurity 524
 - mqsistart 525
 - mqsistartmsgflow 528
 - mqsistop 531
 - mqsistopmsgflow 534
 - runtime and toolkit 335
 - mqsapplybaroverride 335
 - mqsireadbar 338
 - syntax preference 329
 - dotted decimal diagrams 331
 - railroad diagrams 329
 - toolkit 339
 - mqscreatebar 339
 - mqscreatmsgdefs 342
 - mqscreatmsgdefs C options
 - files 344
 - mqscreatmsgdefs COBOL options
 - file 345
 - mqscreatmsgdefs default options
 - file 348
 - mqscreatmsgdefs XSD options
 - file 347
 - mqscreatmsgdefsfromwsdl 350
 - mqsigratemfmaps 352
 - z/OS console
 - guidance on using 538
 - issuing to 537
- components
 - connecting 204
 - definition on z/OS 139
 - directory 140
 - PDSE 140
 - verifying 204
- configuration
 - authorization 16
 - broker database 102
 - broker domain 89
 - components 162
 - in the workbench 222
- configuration (*continued*)
 - database
 - authorizing access 111
 - connecting to 113
 - creating 105
 - creating a DB2 database on UNIX
 - systems 107
 - creating a DB2 database on
 - Windows 106
 - customizing 108
 - global coordination of
 - transactions 135
 - issuing database commands 110
 - JDBC connections 128
 - retained publications with a Sybase
 - database 134
 - using Derby databases on
 - Windows 108
 - default 200
 - publish/subscribe topology 235
 - Configuration Manager
 - changing 210
 - Linux and UNIX systems 211
 - Windows 211
 - z/OS 212
 - creating 176
 - AIX 176
 - HP-UX 177
 - Linux 178
 - Solaris 179
 - Windows 180
 - creating on z/OS 181
 - component information 182
 - copying the started task 185
 - creating the component 185
 - creating the directory 183
 - creating the PDSE 182
 - Customizing the component data
 - set 183
 - customizing the JCL 183
 - information required 181
 - installation information 181
 - deleting 219
 - Linux and UNIX systems 219
 - Windows 220
 - z/OS 220
 - modifying 210
 - moving to new queue manager 213
 - sample files, z/OS
 - BIPCMGR 565
 - BIPCPRF 567
 - BIPRCRM 570
 - security 50
 - service ID 50
 - starting and stopping 288
 - Linux and UNIX systems 288
 - Windows 289
 - z/OS 289
 - WebSphere MQ resources 93
 - configuring authentication 36
 - LDAP 36
 - TFIM 37
 - configuring authorization 39
 - LDAP 39
 - TFIM 41
 - configuring identity 34
 - configuring identity mapping 38
- creating a security profile 29
- creating user IDs 45
- customization
 - z/OS 539
 - broker PDSE, contents of 547
 - Configuration Manager PDSE,
 - contents of 550
 - DB2 using data-sharing
 - groups 546
 - disk space requirements 545
 - naming conventions 539
 - overview 138
 - planning checklist 547
 - summary of required access 541
 - tasks and roles 540
 - User Name Server PDSE, contents
 - of 549
 - z/OS environment 137
 - APF attributes, checking 157
 - Automatic Restart Manager
 - planning 155
 - DB2 planning 150
 - event log messages 142
 - file system, mounting 155
 - file system, using 142
 - installation directory, checking
 - permission of 156
 - level of Java, checking 157
 - resource recovery service
 - planning 154
 - shared libraries 148
 - temporary directories 148
 - UNIX system services 148
 - z/OS workload manager, defining
 - started tasks to 154
- CVS 271

D

- data-sharing groups, binding a DB2 plan
 - to 141
- database access, changing
 - broker database 215
- database connections
 - broker database 102
 - ODBC drivers 113
 - user database 103
- Database Instance Manager 108
- databases
 - accessing
 - setting your environment 127
 - authorizing access 111
 - broker 215
 - configuring a JDBC provider 128
 - connecting to 113
 - on Windows 115
 - creating 105
 - customizing 108
 - database commands, issuing 110
 - DB2 on UNIX systems, creating 107
 - DB2 on Windows, creating 106
 - Derby databases on Windows,
 - using 108
 - JDBC connections 128
 - authorizing access to
 - resources 133

- databases (*continued*)
 - JDBC connections (*continued*)
 - global coordination of transactions 132
 - securing 131
 - naming conventions 92
 - ODBC connections
 - on Linux and UNIX (32-bit) 118
 - on Linux and UNIX (64-bit) 122
 - security 64
 - Sybase, retained publications 134
- DbInstMgr 108
- Default Configuration wizard 200
- deployment
 - deployed children
 - removing from broker 231
 - removing from execution group 235
- development resources, security for 16
- directories, components 140
- domain awareness 27
 - enabling and disabling 46
- domain connections
 - changing properties 225
 - creating 223
 - deleting 226
 - properties 225
- dotted decimal diagrams, reading 331

E

- error logs
 - broker domain logs
 - clearing 296
 - filtering 294
 - refreshing 293
 - saving 295
 - viewing 293
- Event Log editor
 - changing preferences 298
- exception processing
 - security 16
- execution groups
 - adding 232
 - command line, using 175
 - z/OS 175
 - configuring as non-swappable on z/OS 158
 - copying 233
 - deleting 234
 - deleting through command line 216
 - deployed children, removing 235
 - properties, changing 233
 - renaming 234

F

- fastpath applications 164

G

- global coordination
 - configuration
 - 32-bit queue manager (DB2) 257
 - 32-bit queue manager (Oracle) 263

- global coordination (*continued*)
 - configuration (*continued*)
 - 64-bit queue manager (DB2) 259
 - 64-bit queue manager (Oracle) 265
 - databases 135
 - Sybase 268

H

- high-volume publish/subscribe on z/OS 249
- HTTP tunneling 26
 - implementing 64

I

- identity 8
 - configuring 34
 - configuring a message flow for 44
 - mapping 12
 - configuring 38
 - propagation 15, 44
- installation
 - directory 139
 - INSTPATH 139
 - security 28

J

- JDBC provider
 - authorizing access to resources 133
 - creating and configuring 128
 - global coordination of transactions 132
 - securing connections 131

L

- LDAP
 - security profiles 29
- leaf nodes 239
- listeners, starting WebSphere MQ 198
- locales
 - changing 273
- logs
 - broker domain logs
 - clearing 296
 - filtering 294
 - refreshing 293
 - saving 295
 - viewing 293

M

- message flow security 3
 - configuring 28
- message flows
 - database
 - connections 103
 - global coordination of transactions
 - databases 135
 - starting 286
 - stopping 286
 - throughput, optimizing 96

- message protection 77
 - using for Real-time connections 84
- message serialization
 - input between separate brokers 143
 - input between separate execution groups 145
 - input within an execution group 146
 - overview of 143
 - user tasks 147
- mqbrkrs group 70
- multicast brokers
 - choosing a protocol 248
 - mqsisetproperties command 242
 - setup parameters 242
- multicast protocols 248
- multicast publish/subscribe 238
- multicast topics 248

N

- National Language Support
 - UNIX syslog 273

O

- ODBC
 - 32-bit connection, defining on Linux and UNIX 118
 - 64-bit connection, defining on Linux and UNIX 122
 - connection
 - defining on Windows 115
 - odbc32.ini sample file 305
 - odbc64.ini sample file 311
- Oracle
 - granting user access 65
 - revoking user access 65
- order
 - choosing messaging processing order 96

P

- parent nodes 239
- password authentication 75
 - mutual challenge-response 75
 - telnet-like 75
- performance
 - considerations for design 95
 - message flow throughput 96
- planning
 - database naming conventions 92
 - product component naming
 - conventions 90
 - resource naming conventions 90
 - WebSphere MQ naming
 - conventions 91
- product component naming
 - conventions 90
- profiles, security 7
- projects
 - working sets 272
- public key cryptography 21
- publish/subscribe
 - adding a topic 252
 - configuring a topology 235

- publish/subscribe (*continued*)
 - deleting a topic 252
 - enabling applications 199
 - multicast 238
 - querying a subscription 253
 - securing the domain 78
 - security 70
 - enabling applications on z/OS 199
 - topic-based 70
 - topologies 236

Q

- quality of protection 27
 - implementing 64
- queue managers
 - starting as a Windows service 291
 - stopping 292

R

- railroad diagrams, reading 329
- Real-time connections, message protection 84
- repositories, CVS 271
- resources
 - resource naming conventions 90
- responses to commands 335
- rules
 - using commands 334
- runtime resources
 - controlling access 18
 - security for 16

S

- sample file
 - odbc32.ini 305
 - odbc64.ini 311
- security 3
 - access control list (ACL)
 - permissions 591
 - Access Control Lists 18
 - administrative tasks 591
 - authentication 10
 - configuring 36
 - authorization 11
 - configuring 39
 - TFIM and TAM 13
 - broker 48
 - broker domain 45
 - Configuration Manager 50
 - configuring authentication 36
 - LDAP 36
 - TFIM 37
 - configuring authorization 39
 - LDAP 39
 - TFIM 41
 - configuring identity 34
 - configuring identity mapping 38
 - databases 64
 - development resources 16
 - digital certificates 22
 - digital signatures 24

- security (*continued*)
 - domain, changing for the User Name Server 52
 - exception processing 16
 - exits 20, 63
 - identity 8
 - configuring 34
 - identity mapping 12
 - configuring 38
 - identity propagation 15
 - configuring a message flow for 44
 - invoking 63
 - message flow 3
 - configuring 28
 - message flow for identity propagation, configuring 44
 - planning for installation 28
 - profiles 7
 - creating 29
 - LDAP 29
 - TFIM 33
 - public key cryptography 21
 - quality of protection 27
 - runtime resources 16
 - topic-based 61, 78
 - UNIX 592
 - User Name Server 61, 78
 - Windows 593
 - workbench 46
 - z/OS 596
- SQL server
 - granting user access 65
 - revoking user access 65
- SSL authentication 25
 - implementing 52
 - MQ Java Client 52
 - Real Time node 54, 81
- SupportPac IP13 161
- Sybase
 - granting user access 65
 - revoking user access 65
- system topics 70

T

- TAM, configuring 43
- TFIM
 - security profiles 33
- TFIM module chain 42
- throughput
 - optimizing message flows 96
- topic-based security 70
 - enabling 61, 78
- topics
 - making multicast 248
- trademarks 603
- trusted applications 164
- tunneling 26
 - implementing 64

U

- Unicode 276

UNIX

- National Language Support for syslog 273
- user databases
 - connections 103
- User Name Server
 - broker, connecting on z/OS 197
 - broker, connecting to 196
 - changing 213
 - Linux and UNIX systems 214
 - Windows 214
 - z/OS 215
 - configuring publish/subscribe security 196
 - connecting to the network 195
 - creating 186
 - AIX 187
 - HP-UX 187
 - Linux 188
 - Solaris 189
 - Windows 189
 - creating on z/OS 190
 - component information 191
 - copying the started task 195
 - creating the component 194
 - creating the component dataset 193
 - creating the directory 192
 - creating the PDSE 191
 - creating the runtime environment 192
 - customizing the JCL 193
 - information required 190
 - installation information 168, 190
 - deleting 221
 - Linux and UNIX systems 221
 - Windows 222
 - z/OS 222
 - disabling 220
 - enabling 186
 - sample files, z/OS
 - BIPCRUN 580
 - BIPUNSP 585
 - BIPUPROF 588
 - security 61, 78
 - security domain, changing 52
 - starting and stopping 290
 - UNIX 290
 - Windows 290
 - z/OS 291
 - WebSphere MQ resources 94

W

- WebSphere MQ
 - infrastructure
 - designing 92
 - resources for brokers 93
 - resources for the Configuration Manager 93
 - resources for the User Name Server 94
 - naming conventions 91
 - planning for z/OS 153
 - securing resources 84
 - starting
 - channels 198

- WebSphere MQ (*continued*)
 - starting (*continued*)
 - listeners 198
 - trusted applications 164
- wildcard topics 70
- Windows service, starting a queue manager 291
- work path
 - changing 296
 - mounting 296
- workbench
 - changing preferences 270
 - configuring a broker domain 222
 - security 46
- working sets
 - projects 272

X

- XA coordination
 - configuration
 - 32-bit queue manager (DB2) 257
 - 32-bit queue manager (Oracle) 263
 - 64-bit queue manager (DB2) 259
 - 64-bit queue manager (Oracle) 265
 - databases 135
 - Sybase 268
- XPLink 141

Z

- z/OS
 - administration 537
 - broker statistics, collecting 157
 - Configuration Manager
 - connecting directly to 159
 - connecting through intermediate queue manager 160
 - customization 539
 - broker PDSE, contents of 547
 - Configuration Manager PDSE, contents of 550
 - DB2 using data-sharing groups 546
 - disk space requirements 545
 - naming conventions 539
 - planning checklist 547
 - summary of required access 541
 - tasks and roles 540
 - User Name Server PDSE, contents of 549
 - customization variables, JCL 551
 - customizing the environment 137
 - APF attributes, checking 157
 - Automatic Restart Manager
 - planning 155
 - DB2 planning 150
 - event log messages 142
 - file system 142
 - installation directory, checking permission of 156
 - level of Java, checking 157
 - mounting file systems 155

- z/OS (*continued*)
 - customizing the environment (*continued*)
 - resource recovery service
 - planning 154
 - shared libraries 148
 - temporary directory space 148
 - TMPDIR 148
 - UNIX system services 148
 - z/OS workload manager, defining
 - started tasks to 154
 - execution groups, configuring as non-swappable 158
 - guidance for issuing console commands 538
 - issuing commands to the console 537
 - Message Brokers components,
 - creating 158
 - moving from a distributed environment 216
 - security considerations 142
 - creating Publish/Subscribe user IDs 69
 - enabling the Configuration Manager to obtain user ID information 69
 - setting up DB2 67
 - setting up WebSphere MQ 68
 - setting up workbench access 69
 - setting up z/OS security 65
 - setup verification
 - SupportPac IP13 161
 - WebSphere Message Brokers and WebSphere MQ 161
 - specific information 537
 - START and STOP commands 538
 - WebSphere MQ planning 153



Printed in USA