8.0

Troubleshooting and Support for IBM MQ

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 539.

This edition applies to version 8 release 0 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Troubleshooting and support

If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

For an introduction to troubleshooting and support, see "Troubleshooting overview" on page 7.

There are some initial checks that you can make for your platform to help determine the causes of some common problems. See the appropriate topic for your platform:

- Windows Linux UNIX "Making initial checks on Windows, UNIX and Linux systems" on page 9
- IBM i "Making initial checks on IBM i" on page 18
- z/OS "Making initial checks on z/OS" on page 28

For information about solving problems, see "Making initial checks" on page 8.

For information about solving problems for IBM MQ Telemetry, see "IBM MQ Telemetry troubleshooting" on page 500.

For information about solving problems when you are using channel authentication records, see "Channel authentication records troubleshooting" on page 441.

Information that is produced by IBM MQ can help you to find and resolve problems. For more information, see the following topics:

- "Using logs" on page 341
- "Using trace" on page 350
- z/OS "Problem determination on z/OS" on page 389
- " First Failure Support Technology (FFST" on page 330

For information about recovering after a problem, see "Recovering after failure" on page 512.

See also "Contacting IBM Software Support" on page 328.

If an IBM MQ component or command has returned an error, and you want further information about a message written to the screen or the log, you can browse for details of the message, see "Reason codes and exceptions" on page 43.

Related information

Troubleshooting and support reference

Troubleshooting overview

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

A basic troubleshooting strategy at a high level involves:

1. "Recording the symptoms of the problem" on page 7
2. "Re-creating the problem" on page 8
3. "Eliminating possible causes" on page 8

Recording the symptoms of the problem

Depending on the type of problem that you have, whether it be with your application, your server, or your tools, you might receive a message that indicates that something is wrong. Always record the error message that you see. As simple as this sounds, error messages sometimes contain codes that might make more sense as you investigate your problem further. You might also receive multiple error messages that look similar but have subtle differences. By recording the details of each one, you can learn more about where your problem exists.

Sources of error messages:

- Problems view
- Local error log
- Eclipse log
- User trace
- Service trace
- Error dialog boxes

Re-creating the problem

Think back to what steps you were doing that led to the problem. Try those steps again to see if you can easily re-create the problem. If you have a consistently repeatable test case, it is easier to determine what solutions are necessary.

- How did you first notice the problem?
- Did you do anything different that made you notice the problem?
- Is the process that is causing the problem a new procedure, or has it worked successfully before?
- If this process worked before, what has changed? (The change can refer to any type of change that is made to the system, ranging from adding new hardware or software, to reconfiguring existing software.)
- What was the first symptom of the problem that you witnessed? Were there other symptoms occurring around the same time?
- Does the same problem occur elsewhere? Is only one machine experiencing the problem or are multiple machines experiencing the same problem?
- What messages are being generated that might indicate what the problem is?

Windows Linux UNIX You can find more information about these types of question in "Making initial checks on Windows, UNIX and Linux systems" on page 9.

Eliminating possible causes

Narrow the scope of your problem by eliminating components that are not causing the problem. By using a process of elimination, you can simplify your problem and avoid wasting time in areas that are not responsible. Consult the information in this product and other available resources to help you with your elimination process.

Making initial checks

There are some initial checks that you can make that may provide answers to common problems that you may have.

Carry out the initial checks for your platform:

- Windows Linux UNIX "Making initial checks on Windows, UNIX and Linux systems" on page 9
- z/OS "Making initial checks on z/OS" on page 28
- IBM i "Making initial checks on IBM i" on page 18

For logging and tracing, see the following information:

- “Using logs” on page 341
- “Using trace” on page 350

Use the information and general advice given in the subtopics to help you rectify the problem.

Related concepts

“Troubleshooting and support” on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

Related information

Troubleshooting and support reference

Making initial checks on Windows, UNIX and Linux systems

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in:

- IBM MQ
- The network
- The application
- Other applications that you have configured to work with IBM MQ

This section contains a list of questions to consider. As you go through the list, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause straight away, they might be useful later if you have to carry out a systematic problem determination exercise.

- “Has IBM MQ run successfully before?” on page 11
- “Have any changes been made since the last successful run?” on page 11
- “Are there any error messages or return codes to explain the problem?” on page 11
- “Can you reproduce the problem?” on page 12
- “Are you receiving an error code when creating or starting a queue manager? (Windows only)” on page 12
- “Does the problem affect only remote queues?” on page 13
- “Have you obtained incorrect output?” on page 13
- “Are some of your queues failing?” on page 15
- “Have you failed to receive a response from a PCF command?” on page 16
- “Has the application run successfully before?” on page 16
- “Is your application or system running slowly?” on page 18
- “Does the problem affect specific parts of the network?” on page 18
- “Does the problem occur at specific times of the day?” on page 18
- “Is the problem intermittent?” on page 18

See the following sections for some additional tips for problem determination for system administrators and application developers.

Tips for system administrators

- Check the error logs for messages for your operating system:
  - Windows Linux UNIX “Error logs on Windows, UNIX and Linux systems” on page 342

  - IBM i “Error logs on IBM i” on page 346
  - z/OS “Diagnostic information produced on IBM MQ for z/OS” on page 396
- Check the contents of `qm.ini` for any configuration changes or errors. For more information on changing configuration information, see:
  - Windows Linux UNIX Changing configuration information on Windows, UNIX and Linux® systems
  - IBM i Changing configuration information on IBM i
  - z/OS Customizing your queue managers on z/OS
- If your application development teams are reporting something unexpected, you use trace to investigate the problems. For information about using trace, see “Using trace” on page 350.

Tips for application developers

- Check the return codes from the MQI calls in your applications. For a list of reason codes, see “API reason codes” on page 44. Use the information provided in the return code to determine the cause of the problem. Follow the steps in the Programmer response sections of the reason code to resolve the problem.
- If you are unsure whether your application is working as expected, for example, you are not unsure of the parameters being passed into the MQI or out of the MQI, you can use trace to collect information about all the inputs and outputs of your MQI calls. For more information about using trace, see “Using trace” on page 350.
- For more information about handling errors in MQI applications, see Handling program errors.

Related concepts

“Troubleshooting and support” on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

“Making initial checks on z/OS” on page 28
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks on IBM i” on page 18
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks” on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

“Reason codes and exceptions” on page 43
You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

Related tasks

“Contacting IBM Software Support” on page 328
You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

Related reference

“PCF reason codes” on page 244

Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

Related information

Troubleshooting and support reference

Has IBM MQ run successfully before?

If IBM MQ has not run successfully before, it is likely that you have not yet set it up correctly. See Installing IBM MQ and select the platform, or platforms, that your enterprise uses to check that you have installed the product correctly.

To run the verification procedure, see:

- Verifying a server installation
- Verifying a client installation

Also look at Configuring for information about post-installation configuration of IBM MQ.

Have any changes been made since the last successful run?

Changes that have been made to your IBM MQ configuration, maintenance updates, or changes to other programs that interact with IBM MQ could be the cause of your problem.

When you are considering changes that might recently have been made, think about the IBM MQ system, and also about the other programs it interfaces with, the hardware, and any new applications. Consider also the possibility that a new application that you are not aware of might have been run on the system.

- Have you changed, added, or deleted any queue definitions?
- Have you changed or added any channel definitions? Changes might have been made to either IBM MQ channel definitions or any underlying communications definitions required by your application.
- Do your applications deal with return codes that they might get as a result of any changes you have made?
- Have you changed any component of the operating system that could affect the operation of IBM MQ? For example, have you modified the Windows Registry.

Have you applied any maintenance updates?

If you have applied a maintenance update to IBM MQ, check that the update action completed successfully and that no error message was produced.

- Did the update have any special instructions?
- Was any test run to verify that the update was applied correctly and completely?
- Does the problem still exist if IBM MQ is restored to the previous maintenance level?
- If the installation was successful, check with the IBM Support Center for any maintenance package errors.
- If a maintenance package has been applied to any other program, consider the effect it might have on the way IBM MQ interfaces with it.

Are there any error messages or return codes to explain the problem?

You might find error messages or return codes that help you to determine the location and cause of your problem.

IBM MQ uses error logs to capture messages concerning its own operation, any queue managers that you start, and error data coming from the channels that are in use. Check the error logs to see if any messages have been recorded that are associated with your problem.

IBM MQ also logs errors in the Windows Application Event Log. On Windows, check if the Windows Application Event Log shows any IBM MQ errors. To open the log, from the Computer Management panel, expand **Event Viewer** and select **Application**.

Windows Linux UNIX For information about the locations and contents of the error logs, see "Error logs on Windows, UNIX and Linux systems" on page 342

For each IBM MQ Message Queue Interface (MQI) and IBM MQ Administration Interface (MQAI) call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call. If your application gets a return code indicating that a Message Queue Interface (MQI) call has failed, check the reason code to find out more about the problem.

For a list of reason codes, see "API completion and reason codes" on page 43.

Detailed information on return codes is contained within the description of each MQI call.

Related reference

"PCF reason codes" on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

"Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes" on page 315
IBM MQ can use Secure Sockets Layer (SSL) with the various communication protocols. Use this topic to identify the error codes that can be returned by SSL.

"WCF custom channel exceptions" on page 320
Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Related information

IBM MQ AMQ messages

z/OS IBM MQ for z/OS messages, completion, and reason codes

Troubleshooting and support reference

Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which it is reproduced:

- Is it caused by a command or an equivalent administration request?

  Does the operation work if it is entered by another method? If the command works if it is entered on the command line, but not otherwise, check that the command server has not stopped, and that the queue definition of the SYSTEM.ADMIN.COMMAND.QUEUE has not been changed.
- Is it caused by a program? Does it fail on all IBM MQ systems and all queue managers, or only on some?
- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.

Are you receiving an error code when creating or starting a queue manager? (Windows only)

If the IBM MQ Explorer, or the `amqmdain` command, fails to create or start a queue manager, indicating an authority problem, it might be because the user under which the IBM MQ Windows service is running has insufficient rights.

Ensure that the user with which the IBM MQ Windows service is configured has the rights described in User rights required for an IBM MQ Windows Service. By default this service is configured to run as the `MUSR_MQADMIN` user. For subsequent installations, the Prepare IBM MQ Wizard creates a user account named `MUSR_MQADMINx` , where x is the next available number representing a user ID that does not exist.

Does the problem affect only remote queues?

Things to check if the problem affects only remote queues.

If the problem affects only remote queues, perform the following checks:

- Check that required channels have started, can be triggered, and any required initiators are running.
- Check that the programs that should be putting messages to the remote queues have not reported problems.
- If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on. Also, check that the trigger monitor is running.
- Check the error logs for messages indicating channel errors or problems.
- If necessary, start the channel manually.

Have you obtained incorrect output?

In this section, *incorrect output* refers to your application: not receiving a message that you were expecting it to receive; receiving a message containing unexpected or corrupted information; receiving a message that you were not expecting it to receive, for example, one that was destined for a different application.

Messages that do not arrive on the queue

If messages do not arrive when you are expecting them, check for the following:

- Has the message been put on the queue successfully?
  - Has the queue been defined correctly? For example, is MAXMSGL sufficiently large?
  - Is the queue enabled for putting?
  - Is the queue already full?
  - Has another application got exclusive access to the queue?
- Are you able to get any messages from the queue?
  - Do you need to take a sync point?

    If messages are being put or retrieved within sync point, they are not available to other tasks until the unit of recovery has been committed.
  - Is your wait interval long enough?

    You can set the wait interval as an option for the MQGET call. Ensure that you are waiting long enough for a response.
  - Are you waiting for a specific message that is identified by a message or correlation identifier (*MsgId* or *CorrelId*)?

    Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful MQGET call sets both these values to that of the message retrieved, so you might need to reset these values in order to get another message successfully.

    Also, check whether you can get other messages from the queue.
  - Can other applications get messages from the queue?
  - Was the message you are expecting defined as persistent?

    If not, and IBM MQ has been restarted, the message has been lost.
  - Has another application got exclusive access to the queue?

If you cannot find anything wrong with the queue, and IBM MQ is running, check the process that you expected to put the message onto the queue for the following:

- Did the application start?

If it should have been triggered, check that the correct trigger options were specified.

- Did the application stop?
- Is a trigger monitor running?
- Was the trigger process defined correctly?
- Did the application complete correctly?

  Look for evidence of an abnormal end in the job log.
- Did the application commit its changes, or were they backed out?

If multiple transactions are serving the queue, they can conflict with one another. For example, suppose one transaction issues an MQGET call with a buffer length of zero to find out the length of the message, and then issues a specific MQGET call specifying the *MsgId* of that message. However, in the meantime, another transaction issues a successful MQGET call for that message, so the first application receives a reason code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multiple server environment must be designed to cope with this situation.

Consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If so, refer to the subsequent information in this topic.

Messages that contain unexpected or corrupted information

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following:

- Has your application, or the application that put the message onto the queue, changed?

  Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.

  For example, the format of the message data might have been changed, in which case, both applications must be recompiled to pick up the changes. If one application has not been recompiled, the data will appear corrupted to the other.
- Is an application sending messages to the wrong queue?

  Check that the messages your application is receiving are not intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.

  If your application uses an alias queue, check that the alias points to the correct queue.
- Has the trigger information been specified correctly for this queue?

  Check that your application should have started; or should a different application have started?

If these checks do not enable you to solve the problem, check your application logic, both for the program sending the message, and for the program receiving it.

Problems with incorrect output when using distributed queues

If your application uses distributed queues, consider the following points:

- Has IBM MQ been correctly installed on both the sending and receiving systems, and correctly configured for distributed queuing?
- Are the links available between the two systems?

  Check that both systems are available, and connected to IBM MQ. Check that the connection between the two systems is active.

  You can use the MQSC command PING against either the queue manager (PING QMGR) or the channel (PING CHANNEL) to verify that the link is operable.
- Is triggering set on in the sending system?

- Is the message for which you are waiting a reply message from a remote system?

  Check that triggering is activated in the remote system.
- Is the queue already full?

  If so, check if the message has been put onto the dead-letter queue.

  The dead-letter queue header contains a reason or feedback code explaining why the message could not be put onto the target queue. See Using the dead-letter (undelivered message) queue and MQDLH - Dead-letter header for information about the dead-letter queue header structure.
- Is there a mismatch between the sending and receiving queue managers?

  For example, the message length could be longer than the receiving queue manager can handle.
- Are the channel definitions of the sending and receiving channels compatible?

  For example, a mismatch in sequence number wrap can stop the distributed queuing component. See Distributed queuing and clusters for more information about distributed queuing.
- Is data conversion involved? If the data formats between the sending and receiving applications differ, data conversion is necessary. Automatic conversion occurs when the MQGET call is issued if the format is recognized as one of the built-in formats.

  If the data format is not recognized for conversion, the data conversion exit is taken to allow you to perform the translation with your own routines.

  Refer to Data conversion for further information about data conversion.

Are some of your queues failing?

If you suspect that the problem occurs with only a subset of queues, check the local queues that you think are having problems.

Perform the following checks:

1. Display the information about each queue. You can use the MQSC command DISPLAY QUEUE to display the information.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.
   - If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:
     - If triggering is being used:
       - Is the trigger monitor running?
       - Is the trigger depth too great? That is, does it generate a trigger event often enough?
       - Is the process name correct?
       - Is the process available and operational?
     - Can the queue be shared? If not, another application could already have it open for input.
     - Is the queue enabled appropriately for GET and PUT?
   - If there are no application processes getting messages from the queue, determine why this is so. It could be because the applications need to be started, a connection has been disrupted, or the MQOPEN call has failed for some reason.

     Check the queue attributes IPPROCS and OPPROCS. These attributes indicate whether the queue has been opened for input and output. If a value is zero, it indicates that no operations of that type can occur. The values might have changed; the queue might have been open but is now closed.

     You need to check the status at the time you expect to put or get a message.

If you are unable to solve the problem, contact your IBM Support Center for help.

Have you failed to receive a response from a PCF command?

Considerations if you have issued a command but have not received a response.

If you have issued a command but have not received a response, consider the following checks:

- Is the command server running?

  Work with the `dspmqcsv` command to check the status of the command server.

  - If the response to this command indicates that the command server is not running, use the `strmqcsv` command to start it.
  - If the response to the command indicates that the SYSTEM.ADMIN.COMMAND.QUEUE is not enabled for MQGET requests, enable the queue for MQGET requests.
- Has a reply been sent to the dead-letter queue?

  The dead-letter queue header structure contains a reason or feedback code describing the problem. See MQDLH - Dead-letter header and Using the dead-letter (undelivered message) queue for information about the dead-letter queue header structure (MQDLH).

  If the dead-letter queue contains messages, you can use the provided browse sample application (amqsbcg) to browse the messages using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.
- Has a message been sent to the error log?

  See "Error log directories on UNIX, Linux, and Windows" on page 344 for further information.
- Are the queues enabled for put and get operations?
- Is the *`WaitInterval`* long enough?

  If your MQGET call has timed out, a completion code of MQCC_FAILED and a reason code of MQRC_NO_MSG_AVAILABLE are returned. (See WaitInterval (MQLONG) for information about the *`WaitInterval`* field, and completion and reason codes from MQGET.)
- If you are using your own application program to put commands onto the SYSTEM.ADMIN.COMMAND.QUEUE, do you need to take a sync point?

  Unless you have excluded your request message from sync point, you need to take a sync point before receiving reply messages.
- Are the MAXDEPTH and MAXMSGL attributes of your queues set sufficiently high?
- Are you using the *`CorrelId`* and *`MsgId`* fields correctly?

  Set the values of *`MsgId`* and *`CorrelId`* in your application to ensure that you receive all messages from the queue.

Try stopping the command server and then restarting it, responding to any error messages that are produced.

If the system still does not respond, the problem could be with either a queue manager or the whole of the IBM MQ system. First, try stopping individual queue managers to isolate a failing queue manager. If this step does not reveal the problem, try stopping and restarting IBM MQ, responding to any messages that are produced in the error log.

If the problem still occurs after restart, contact your IBM Support Center for help.

Has the application run successfully before?

Use the information in this topic to help diagnose common problems with applications.

If the problem appears to involve one particular application, consider whether the application has run successfully before.

Before you answer **Yes** to this question, consider the following:

- Have any changes been made to the application since it last ran successfully?

  If so, it is likely that the error lies somewhere in the new or modified part of the application. Take a look at the changes and see if you can find an obvious reason for the problem. Is it possible to retry using a back level of the application?
- Have all the functions of the application been fully exercised before?

  Could it be that the problem occurred when part of the application that had never been invoked before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

  If a program has been run successfully on many previous occasions, check the current queue status and the files that were being processed when the error occurred. It is possible that they contain some unusual data value that invokes a rarely-used path in the program.
- Does the application check all return codes?

  Has your IBM MQ system been changed, perhaps in a minor way, such that your application does not check the return codes it receives as a result of the change. For example, does your application assume that the queues it accesses can be shared? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
- Does the application run on other IBM MQ systems?

  Could it be that there is something different about the way that this IBM MQ system is set up that is causing the problem? For example, have the queues been defined with the same message length or priority?

Before you look at the code, and depending upon which programming language the code is written in, examine the output from the translator, or the compiler and linkage editor, to see if any errors have been reported.

If your application fails to translate, compile, or link-edit into the load library, it will also fail to run if you attempt to invoke it. See Developing applications for information about building your application.

If the documentation shows that each of these steps was accomplished without error, consider the coding logic of the application. Do the symptoms of the problem indicate the function that is failing and, therefore, the piece of code in error? See the following section for some examples of common errors that cause problems with IBM MQ applications.

Common programming errors

The errors in the following list illustrate the most common causes of problems encountered while running IBM MQ programs. Consider the possibility that the problem with your IBM MQ system could be caused by one or more of these errors:

- Assuming that queues can be shared, when they are in fact exclusive.
- Passing incorrect parameters in an MQI call.
- Passing insufficient parameters in an MQI call. This might mean that IBM MQ cannot set up completion and reason codes for your application to process.
- Failing to check return codes from MQI requests.
- Passing variables with incorrect lengths specified.
- Passing parameters in the wrong order.
- Failing to initialize *MsgId* and *CorrelId* correctly.
- Failing to initialize *Encoding* and *CodedCharSetId* following MQRC_TRUNCATED_MSG_ACCEPTED.

Is your application or system running slowly?

If your application is running slowly, it might be in a loop, or waiting for a resource that is not available, or there might be a performance problem.

Perhaps your system is operating near the limits of its capacity. This type of problem is probably worst at peak system load times, typically at mid-morning and mid-afternoon. (If your network extends across more than one time zone, peak system load might seem to occur at some other time.)

A performance problem might be caused by a limitation of your hardware.

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly-designed application program is probably to blame. This could appear to be a problem that only occurs when certain queues are accessed.

If the performance issue persists, the problem might lie with IBM MQ itself. If you suspect this, contact your IBM Support Center for help.

A common cause of slow application performance, or the build up of messages on a queue (usually a transmission queue) is one or more applications that write persistent messages outside a unit of work; for more information, see Message persistence.

Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote message queue manager is not working, the messages cannot flow to a remote queue.

Check that the connection between the two systems is available, and that the intercommunication component of IBM MQ has started.

Check that messages are reaching the transmission queue, and check the local queue definition of the transmission queue and any remote queues.

Have you made any network-related changes, or changed any IBM MQ definitions, that might account for the problem?

Does the problem occur at specific times of the day?

If the problem occurs at specific times of day, it could be that it depends on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so these are the times when load-dependent problems are most likely to occur. (If your IBM MQ network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

Is the problem intermittent?

An intermittent problem could be caused by the way that processes can run independently of each other. For example, a program might issue an MQGET call without specifying a wait option before an earlier process has completed. An intermittent problem might also be seen if your application tries to get a message from a queue before the call that put the message has been committed.

IBM i Making initial checks on IBM i

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in any of the following:

- Hardware
- Operating system

- Related software, for example, a language compiler
- The network
- The IBM MQ product
- Your IBM MQ application
- Other applications
- Site operating procedures

This section contains a list of questions to consider. As you go through the list, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause straight away, they might be useful later if you have to carry out a systematic problem determination exercise.

The following steps are intended to help you isolate the problem and are taken from the viewpoint of an IBM MQ application. Check all the suggestions at each stage.

1. Has IBM MQ for IBM i run successfully before?

   Yes
   Proceed to Step "2" on page 19.

   No
   It is likely that you have not installed or set up IBM MQ correctly.

2. Has the IBM MQ application run successfully before?

   Yes
   Proceed to Step "3" on page 19.

   No
   Consider the following:

   a. The application might have failed to compile or link, and fails if you attempt to invoke it. Check the output from the compiler or linker.

      Refer to the appropriate programming language reference information, or see Developing applications, for information about how to build your application.

   b. Consider the logic of the application. For example, do the symptoms of the problem indicate that a function is failing and, therefore, that a piece of code is in error.

      Check the following common programming errors:

      - Assuming that queues can be shared, when they are in fact exclusive.
      - Trying to access queues and data without the correct security authorization.
      - Passing incorrect parameters in an MQI call; if the wrong number of parameters is passed, no attempt can be made to complete the completion code and reason code fields, and the task is ended abnormally.
      - Failing to check return codes from MQI requests.
      - Using incorrect addresses.
      - Passing variables with incorrect lengths specified.
      - Passing parameters in the wrong order.
      - Failing to initialize *MsgId* and *CorrelId* correctly.

3. Has the IBM MQ application changed since the last successful run?

   Yes
   It is likely that the error lies in the new or modified part of the application. Check all the changes and see if you can find an obvious reason for the problem.

   a. Have all the functions of the application been fully exercised before?

      Could it be that the problem occurred when part of the application that had never been invoked before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

  b. If the program has run successfully before, check the current queue status and files that were being processed when the error occurred. It is possible that they contain some unusual data value that causes a rarely used path in the program to be invoked.
  c. The application received an unexpected MQI return code. For example:
     - Does your application assume that the queues it accesses are shareable? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
     - Have any queue definition or security profiles been changed? An MQOPEN call could fail because of a security violation; can your application recover from the resulting return code?

     See MQI Applications reference for your programming language for a description of each return code.
  d. If you have applied any PTF to IBM MQ for IBM i, check that you received no error messages when you installed the PTF.

No
Ensure that you have eliminated all the preceding suggestions and proceed to Step “4” on page 20.

4. Has the server system remained unchanged since the last successful run?

   Yes
   Proceed to “Problem characteristics” on page 22.

   No
   Consider all aspects of the system and review the appropriate documentation on how the change might have affected the IBM MQ application. For example :
   - Interfaces with other applications
   - Installation of new operating system or hardware
   - Application of PTFs
   - Changes in operating procedures

See the following sections for some additional tips for problem determination for system administrators and application developers.

Tips for system administrators

- Check the error logs for messages for your operating system:
  - Windows Linux UNIX “Error logs on Windows, UNIX and Linux systems” on page 342
  - IBM i “Error logs on IBM i” on page 346
  - z/OS “Diagnostic information produced on IBM MQ for z/OS” on page 396
- Check the contents of `qm.ini` for any configuration changes or errors. For more information on changing configuration information, see:
  - Windows Linux UNIX Changing configuration information on Windows, UNIX and Linux systems
  - IBM i Changing configuration information on IBM i
  - z/OS Customizing your queue managers on z/OS
- If your application development teams are reporting something unexpected, you use trace to investigate the problems. For information about using trace, see “Using trace” on page 350.

Tips for application developers

- Check the return codes from the MQI calls in your applications. For a list of reason codes, see "API reason codes" on page 44. Use the information provided in the return code to determine the cause of the problem. Follow the steps in the Programmer response sections of the reason code to resolve the problem.
- If you are unsure whether your application is working as expected, for example, you are not unsure of the parameters being passed into the MQI or out of the MQI, you can use trace to collect information about all the inputs and outputs of your MQI calls. For more information about using trace, see "Using trace" on page 350.
- For more information about handling errors in MQI applications, see Handling program errors.

Related concepts

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Required authority" on page 24
Some IBM MQ commands rely on using IBM i system commands for creating and managing objects, files, and libraries.

"Making initial checks on Windows, UNIX and Linux systems" on page 9
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

"Making initial checks on z/OS" on page 28
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

"Making initial checks" on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

"Reason codes and exceptions" on page 43
You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

Related tasks

"Contacting IBM Software Support" on page 328
You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

Related reference

"Determining problems with IBM MQ for IBM i applications" on page 25
If you have not yet found the cause, start to look at the problem in greater detail.

"PCF reason codes" on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

Related information

Troubleshooting and support reference

Problem characteristics

Perhaps by using the preliminary checks, you have found the cause of the problem. If so, you can probably now resolve it, possibly with the help of other sections in the IBM MQ product documentation, and in the libraries of other licensed programs.

If you have not yet found the cause, start to look at the problem in greater detail. Use the following questions as pointers to the problem. Answering the appropriate question, or questions, should lead you to the cause of the problem:

- “Can you reproduce the problem?” on page 22
- “Is the problem intermittent?” on page 22
- “Problems with commands” on page 22
- “Does the problem affect all users of the IBM MQ for IBM i application?” on page 23
- “Does the problem affect specific parts of the network?” on page 23
- “Does the problem occur only on IBM MQ?” on page 23
- “Does the problem occur at specific times of the day?” on page 23
- “Have you failed to receive a response from a command?” on page 23

Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which you do so:

- Is it caused by a command?

  Does the operation work if it is entered by another method? If the command works if it is entered on the command line, but not otherwise, check that the command server has not stopped. You must also check that the queue definition of the `SYSTEM.ADMIN.COMMAND.QUEUE` has not been changed.
- Is it caused by a program? If so, does it fail in batch? Does it fail on all IBM MQ for IBM i systems, or only on some?
- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.
- Does the problem occur with any queue manager, or when connected to one specific queue manager?
- Does the problem occur with the same type of object on any queue manager, or only one particular object? What happens after this object has been cleared or redefined?
- Is the problem independent of any message persistence settings?
- Does the problem occur only when sync points are used?
- Does the problem occur only when one or more queue-manager events are enabled?

Is the problem intermittent?

An intermittent problem might be caused by failing to take into account the fact that processes can run independently of each other. For example, a program might issue an MQGET call, without specifying a wait option, before an earlier process has completed. You might also encounter this problem if your application tries to get a message from a queue while the call that put the message is in-doubt (that is, before it has been committed or backed out).

Problems with commands

Use this information to avoid potential problems with special characters. Be careful when including special characters, for example backslash (\) and quotation marks (") characters, in descriptive text for some commands. If you use either of these characters in descriptive text, precede them with a backslash (\) character, for example:

- Enter `\\` if you need a backslash (\) character in your text.

- Enter \" if you need quotation marks (") characters in your text.

Queue managers and their associated object names are case-sensitive. By default, IBM i uses uppercase characters, unless you surround the name in apostrophe (') characters.

For example, `MYQUEUE` and `myqueue` translate to `MYQUEUE`, whereas `'myqueue'` translates to `myqueue`.

Does the problem affect all users of the IBM MQ for IBM i application?

If the problem affects only some users, look for differences in how the users configure their systems and queue manager settings.

Check the library lists and user profiles. Can the problem be circumvented by having `*ALLOBJ` authority?

Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote message queue manager is not working, the messages cannot flow to a remote queue.

Check these points:

- Is the connection between the two systems available, and has the intercommunication component of IBM MQ for IBM i started?

  Check that messages are reaching the transmission queue, the local queue definition of the transmission queue, and any remote queues.
- Have you made any network-related changes that might account for the problem or changed any IBM MQ for IBM i definitions?
- Can you distinguish between a channel definition problem and a channel message problem?

  For example, redefine the channel to use an empty transmission queue. If the channel starts correctly, the definition is correctly configured.

Does the problem occur only on IBM MQ?

If the problem occurs only on this version of IBM MQ, check the appropriate database on RETAIN, or the https://www.ibm.com/support/entry/portal/Overview/Software/WebSphere®/WebSphere_MQ, to ensure that you have applied all the relevant PTFs.

Does the problem occur at specific times of the day?

If the problem occurs at specific times of day, it might be that it is dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, and so these times are when load-dependent problems are most likely to occur. (If your IBM MQ for IBM i network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

Have you failed to receive a response from a command?

If you have issued a command but you have not received a response, consider the following questions:

- Is the command server running?

  Work with the `DSPMQMCSVR` command to check the status of the command server.

  - If the response to this command indicates that the command server is not running, use the `STRMQMCSVR` command to start it.
  - If the response to the command indicates that the SYSTEM.ADMIN.COMMAND.QUEUE is not enabled for MQGET requests, enable the queue for MQGET requests.
- Has a reply been sent to the dead-letter queue?

The dead-letter queue header structure contains a reason or feedback code describing the problem. See MQDLH - Dead-letter header for information about the dead-letter queue header structure (MQDLH).

If the dead-letter queue contains messages, you can use the provided browse sample application (amqsbcg) to browse the messages using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.

- Has a message been sent to the error log?

  See "Error logs on IBM i" on page 346 for further information.
- Are the queues enabled for put and get operations?
- Is the *WaitInterval* long enough?

  If your MQGET call has timed out, a completion code of MQCC_FAILED and a reason code of MQRC_NO_MSG_AVAILABLE are returned. (See Getting messages from a queue using the MQGET call for more information about the *WaitInterval* field, and completion and reason codes from MQGET.)
- If you are using your own application program to put commands onto the SYSTEM.ADMIN.COMMAND.QUEUE, do you need to take a sync point?

  Unless you have excluded your request message from sync point, you must take a sync point before attempting to receive reply messages.
- Are the MAXDEPTH and MAXMSGL attributes of your queues set sufficiently high?
- Are you using the *CorrelId* and *MsgId* fields correctly?

  Set the values of *MsgId* and *CorrelId* in your application to ensure that you receive all messages from the queue.

Related concepts

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Required authority" on page 24
Some IBM MQ commands rely on using IBM i system commands for creating and managing objects, files, and libraries.

Related reference

"Determining problems with IBM MQ for IBM i applications" on page 25
If you have not yet found the cause, start to look at the problem in greater detail.

Required authority

Some IBM MQ commands rely on using IBM i system commands for creating and managing objects, files, and libraries.

For example, `CRTMQM` (create queue manager) and `DLTMQM` (delete queue manager). Similarly some IBM MQ program code, for example a queue manager, relies on using IBM i system programs.

To enable this reliance, the commands and programs listed must either have `*PUBLIC *USE` authority, or explicit *USE authority to the IBM MQ user profiles `QMQM` and `QMQMADM`.

Such authority is applied automatically as part of the install process, and you do not need to apply it yourself.

However, if you encounter problems, here is how to do it manually:

1. Set the authorities for commands using GRTOBJAUT with an OBJTYPE(*CMD) parameter, for example:

   ```
   GRTOBJAUT OBJ(QSYS/ADDLIBLE) OBJTYPE(*CMD) USER(QMQMADM) AUT(*USE)
   ```

   - QSYS/ADDLIBLE

- QSYS/ADDPFM
- QSYS/CALL
- QSYS/CHGCURLIB
- QSYS/CHGJOB
- QSYS/CRTJRN
- QSYS/CRTJRNRCV
- QSYS/CRTJOBQ
- QSYS/CRTJOBD
- QSYS/CRTLIB
- QSYS/CRTMSGQ
- QSYS/CRTPF
- QSYS/CRTPGM
- QSYS/CRTSRCPF
- QSYS/DLTJRN
- QSYS/DLTJRNRCV
- QSYS/DLTLIB
- QSYS/DLTMSGQ
- QSYS/OVRPRTF
- QSYS/RCLACTGRP
- QSYS/RTVJRNE
- QSYS/RCVJRNE
- QSYS/SBMJOB

2. Set the authorities for programs using GRTOBJAUT with an OBJTYPE(*PGM) parameter, for example:

```
GRTOBJAUT OBJ(QSYS/QWTSETP) OBJTYPE(*PGM) USER(QMQMADM) AUT(*USE)
```

   - QSYS/QWTSETP(*PGM)
   - QSYS/QSYRLSPH(*PGM)
   - QSYS/QSYGETPH(*PGM)

IBM i Determining problems with IBM MQ for IBM i applications

If you have not yet found the cause, start to look at the problem in greater detail.

This section contains information about problems you might encounter with IBM MQ applications, commands, and messages. Use the following questions as pointers to the problem. Answering the appropriate question, or questions, should lead you to the cause of the problem.

Are some of your queues working?

If you suspect that the problem occurs with only a subset of queues, select the name of a local queue that you think is having problems.

1. Display the information about this queue, using WRKMQMQSTS or DSPMQMQ.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.
   - If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:

    - If triggering is being used:
        - Is the trigger monitor running?
        - Is the trigger depth too large?
        - Is the process name correct?
    - Can the queue be shared? If not, another application might already have it open for input.
    - Is the queue enabled appropriately for GET and PUT?
  - If there are no application processes getting messages from the queue, determine why (for example, because the applications must be started, a connection has been disrupted, or because the MQOPEN call has failed for some reason).

If you cannot solve the problem, contact your IBM support center for help.

Does the problem affect only remote queues?

If the problem affects only remote queues, check the subsequent points:

1. Check that the programs that should be putting messages to the remote queues have run successfully.
2. If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on. Also, check that the trigger monitor is running.
3. If necessary, start the channel manually. See Distributed queuing and clusters.
4. Check the channel with a PING command.

Are messages failing to arrive on the queue?

If messages do not arrive when you are expecting them, check for the following:

- Have you selected the correct queue manager, that is, the default queue manager or a named queue manager?
- Has the message been put on the queue successfully?
  - Has the queue been defined correctly, for example, is MAXMSGLEN sufficiently large?
  - Can applications put messages on the queue (is the queue enabled for putting)?
  - If the queue is already full, it might mean that an application was unable to put the required message on the queue.
- Can you get the message from the queue?
  - Must you take a sync point?

    If messages are being put or retrieved within sync point, they are not available to other tasks until the unit of recovery has been committed.
  - Is your timeout interval long enough?
  - Are you waiting for a specific message that is identified by a message identifier or correlation identifier (*MsgId* or *CorrelId*)?

    Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful MQGET call sets both these values to that of the message retrieved, so you might need to reset these values in order to get another message successfully.

    Also check if you can get other messages from the queue.
  - Can other applications get messages from the queue?
  - Was the message you are expecting defined as persistent?

    If not, and IBM MQ for IBM i has been restarted, the message has been lost.

If you cannot find anything wrong with the queue, and the queue manager itself is running, make the following checks on the process that you expected to put the message on to the queue:

- Did the application start?

  If it should have been triggered, check that the correct trigger options were specified.
- Is a trigger monitor running?
- Was the trigger process defined correctly?
- Did it complete correctly?

  Look for evidence of an abnormal end in the job log.
- Did the application commit its changes, or were they backed out?

If multiple transactions are serving the queue, they might occasionally conflict with one another. For example, one transaction might issue an MQGET call with a buffer length of zero to find out the length of the message, and then issue a specific MQGET call specifying the *MsgId* of that message. However, in the meantime, another transaction might have issued a successful MQGET call for that message, so the first application receives a completion code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multi-server environment must be designed to cope with this situation.

Consider that the message might have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If so, see "Are unexpected messages received when using distributed queues?" on page 27.

Do messages contain unexpected or corrupted information?

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following points:

- Has your application, or the application that put the message on to the queue, changed?

  Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.

  For example, a copyfile formatting the message might have been changed, in which case, recompile both applications to pick up the changes. If one application has not been recompiled, the data appears corrupted to the other.
- Is an application sending messages to the wrong queue?

  Check that the messages your application is receiving are not intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.

  If your application has used an alias queue, check that the alias points to the correct queue.
- Has the trigger information been specified correctly for this queue?

  Check that your application should have been started, or should a different application have been started?
- Has the CCSID been set correctly, or is the message format incorrect because of data conversion.

If these checks do not enable you to solve the problem, check your application logic, both for the program sending the message, and for the program receiving it.

Are unexpected messages received when using distributed queues?

If your application uses distributed queues, consider the following points:

- Has distributed queuing been correctly installed on both the sending and receiving systems?
- Are the links available between the two systems?

  Check that both systems are available, and connected to IBM MQ for IBM i. Check that the connection between the two systems is active.
- Is triggering set on in the sending system?
- Is the message you are waiting for a reply message from a remote system?

Check that triggering is activated in the remote system.

- Is the queue already full?

  If so, it might mean that an application was unable to put the required message on to the queue. Check that the message has been put onto the undelivered-message queue.

  The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message could not be put on to the target queue. See MQDLH - Dead-letter header or IBM i Application Programming Reference (ILE/RPG), as appropriate, for information about the dead-letter header structure.
- Is there a mismatch between the sending and receiving queue managers?

  For example, the message length could be longer than the receiving queue manager can handle.
- Are the channel definitions of the sending and receiving channels compatible?

  For example, a mismatch in sequence number wrap stops the distributed queuing component. See Distributed queuing and clusters.

Making initial checks on z/OS

Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in:

- IBM MQ
- The network
- The application
- Other applications that you have configured to work with IBM MQ

This section contains a list of questions to consider. As you go through the list, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause straight away, they might be useful later if you have to carry out a systematic problem determination exercise.

- “Has IBM MQ for z/OS run successfully before?” on page 30
- “Have you applied any APARs or PTFs?” on page 30
- “Are there any error messages, return codes or other error conditions?” on page 30
- “Has your application or IBM MQ for z/OS stopped processing work?” on page 32
- “Is there a problem with the IBM MQ queues?” on page 33
- “Are some of your queues working?” on page 33
- “Are the correct queues defined?” on page 34
- “Does the problem affect only remote or cluster queues?” on page 34
- “Does the problem affect only shared queues?” on page 35
- “Does the problem affect specific parts of the network?” on page 36
- “Problems that occur at specific times of the day or affect specific users” on page 36
- “Is the problem intermittent or does the problem occur with all z/OS, CICS, or IMS systems?” on page 36
- “Has the application run successfully before?” on page 36
- “Have any changes been made since the last successful run?” on page 38
- “Do you have a program error?” on page 38
- “Has there been an abend?” on page 39
- “Have you obtained incorrect output?” on page 40
- “Can you reproduce the problem?” on page 41

- “Have you failed to receive a response from an MQSC command?” on page 41
- “Is your application or IBM MQ for z/OS running slowly?” on page 42

See the following sections for some additional tips for problem determination for system administrators and application developers.

Tips for system administrators

- Check the error logs for messages for your operating system:
  - Windows Linux UNIX “Error logs on Windows, UNIX and Linux systems” on page 342
  - IBM i “Error logs on IBM i” on page 346
  - z/OS “Diagnostic information produced on IBM MQ for z/OS” on page 396
- Check the contents of `qm.ini` for any configuration changes or errors. For more information on changing configuration information, see:
  - Windows Linux UNIX Changing configuration information on Windows, UNIX and Linux systems
  - IBM i Changing configuration information on IBM i
  - z/OS Customizing your queue managers on z/OS
- If your application development teams are reporting something unexpected, you use trace to investigate the problems. For information about using trace, see “Using trace” on page 350.

Tips for application developers

- Check the return codes from the MQI calls in your applications. For a list of reason codes, see “API reason codes” on page 44. Use the information provided in the return code to determine the cause of the problem. Follow the steps in the Programmer response sections of the reason code to resolve the problem.
- If you are unsure whether your application is working as expected, for example, you are not unsure of the parameters being passed into the MQI or out of the MQI, you can use trace to collect information about all the inputs and outputs of your MQI calls. For more information about using trace, see “Using trace” on page 350.
- For more information about handling errors in MQI applications, see Handling program errors.

Related concepts

“Troubleshooting and support” on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

“Making initial checks on Windows, UNIX and Linux systems” on page 9
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks on IBM i” on page 18
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks” on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

“Reason codes and exceptions” on page 43

You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

Related tasks

"Contacting IBM Software Support" on page 328
You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

Related reference

"PCF reason codes" on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

Related information

Troubleshooting and support reference

Has IBM MQ for z/OS run successfully before?

Knowing whether IBM MQ for z/OS has successfully run before can help with problem determination, and there are checks you can perform to help you.

If the answer to this question is **No**, consider the following:

- Check your setup.

  If IBM MQ has not run successfully on z/OS before, it is likely that you have not yet set it up correctly. See the information about installing and customizing the queue manager in Installing the IBM MQ for z/OS product for further guidance.
- Verify the installation.
- Check that message CSQ9022I was issued in response to the START QMGR command (indicating normal completion).
- Ensure that z/OS displays IBM MQ as an installed subsystem. To determine if IBM MQ is an installed subsystem use the z/OS command `D OPDATA`.
- Check that the installation verification program (IVP) ran successfully.
- Issue the command `DISPLAY DQM` to check that the channel initiator address space is running, and that the appropriate listeners are started.

Have you applied any APARs or PTFs?

APARs and PTFs can occasionally cause unexpected problems with IBM MQ. These fixes can have been applied to IBM MQ or to other z/OS systems.

If an APAR or PTF has been applied to IBM MQ for z/OS, check that no error message was produced. If the installation was successful, check with the IBM support center for any APAR or PTF error.

If an APAR or PTF has been applied to any other product, consider the effect it might have on the way IBM MQ interfaces with it.

Ensure that you have followed any instructions in the APAR that affect your system. (For example, you might have to redefine a resource.)

Are there any error messages, return codes or other error conditions?

Use this topic to investigate error messages, return codes, and conditions where the queue manager or channel initiator terminated.

The problem might produce the following types of error message or return codes:

CSQ messages and reason codes
: IBM MQ for z/OS error messages have the prefix CSQ; if you receive any messages with this prefix (for example, in the console log, or the CICS log), see IBM MQ for z/OS messages, completion, and reason codes for an explanation.

Other messages
For messages with a different prefix, look in the appropriate messages and codes topic for a suggested course of action.

Unusual messages
Be aware of unusual messages associated with the startup of IBM MQ for z/OS, or issued while the system was running before the error occurred. Any unusual messages might indicate some system problem that prevented your application from running successfully.

Application MQI return codes
If your application gets a return code indicating that an MQI call has failed, see Return codes for a description of that return code.

Have you received an unexpected error message or return code?

If your application has received an unexpected error message, consider whether the error message has originated from IBM MQ or from another program.

IBM MQ error messages

IBM MQ for z/OS error messages are prefixed with the letters CSQ.

If you get an unexpected IBM MQ error message (for example, in the console log, or the CICS log), see IBM MQ for z/OS messages, completion, and reason codes for an explanation.

IBM MQ for z/OS messages, completion, and reason codes might give you enough information to resolve the problem quickly, or it might redirect you to another manual for further guidance. If you cannot deal with the message, you might have to contact the IBM support center for help.

Non- IBM MQ error messages

If you get an error message from another IBM program, or from the operating system, look in the messages and codes manual from the appropriate library for an explanation of what it means.

In a queue-sharing environment, look for the following error messages:

- XES (prefixed with the letters IXL)
- Db2 (prefixed with the letters DSN)
- RRS (prefixed with the letters ATR)

Unexpected return codes
If your application has received an unexpected return code from IBM MQ, see Return codes for information about how your application can handle IBM MQ return codes.

Check for error messages

Issue the DISPLAY THREAD(*) command to check if the queue manager is running. For more information about the command, see DISPLAY THREAD. If the queue manager has stopped running, look for any message that might explain the situation. Messages are displayed on the z/OS console, or on your terminal if you are using the operations and control panels. Use the DISPLAY DQM command to see if the channel initiator is working, and the listeners are active. The z/OS command

```
DISPLAY R,L
```

lists messages with outstanding replies. Check to see whether any of these replies are relevant. In some circumstances, for example, when it has used all its active logs, IBM MQ for z/OS waits for operator intervention.

No error messages issued

If no error messages have been issued, perform the following procedure to determine what is causing the problem:

1. Issue the z/OS commands

   ```
   DISPLAY A,xxxxMSTR
   DISPLAY A,xxxxCHIN
   ```

   (where xxxx is the IBM MQ for z/OS subsystem name). If you receive a message telling you that the queue manager or channel initiator has not been found, this message indicates that the subsystem has terminated. This condition could be caused by an abend or by operator shutdown of the system.
2. If the subsystem is running, you receive message IEE105I. This message includes the *CT=nnnn* field, which contains information about the processor time being used by the subsystem. Note the value of this field, and reissue the command.
   - If the *CT=* value has not changed, this indicates that the subsystem is not using any processor time. This could indicate that the subsystem is in a wait state (or that it has no work to do). If you can issue a command like DISPLAY DQM and you get output back, this indicates there is no work to do rather than a hang condition.
   - If the *CT=* value has changed dramatically, and continues to do so over repeated displays, this could indicate that the subsystem is busy or possibly in a loop.
   - If the reply indicates that the subsystem is now not found, this indicates that it was in the process of terminating when the first command was issued. If a dump is being taken, the subsystem might take a while to terminate. A message is produced at the console before terminating.

     To check that the channel initiator is working, issue the DISPLAY DQM command. If the response does not show the channel initiator working this could be because it is getting insufficient resources (like the processor). In this case, use the z/OS monitoring tools, such as RMF, to determine if there is a resource problem. If it is not, restart the channel initiator.

Has the queue manager or channel initiator terminated abnormally?

Look for any messages saying that the queue manager or channel initiator address space has abnormally terminated. If you get a message for which the system action is to terminate IBM MQ, find out whether a system dump was produced, see IBM MQ dumps.

IBM MQ for z/OS might still be running

Consider also that IBM MQ for z/OS might still be running, but only slowly. If it is running slowly, you probably have a performance problem. To confirm this, see Is your application or IBM MQ for z/OS running slowly. Refer to Dealing with performance problems for advice about what to do next.

Has your application or IBM MQ for z/OS stopped processing work?

There are several reasons why your system would unexpectedly stop processing work including problems with the queue manager, the application, z/OS, and the data sets.

There are several reasons why your system would unexpectedly stop processing work. These include:

Queue manager problems
The queue manager might be shutting down.

Application problems
An application programming error might mean that the program branches away from its normal processing, or the application might get in a loop. There might also have been an application abend.

IBM MQ problems
Your queues might have become disabled for MQPUT or MQGET calls, the dead-letter queue might be full, or IBM MQ for z/OS might be in a wait state, or a loop.

z/OS and other system problems

z/OS might be in a wait state, or CICS or IMS might be in a wait state or a loop. There might be problems at the system or sysplex level that are affecting the queue manager or the channel initiator. For example, excessive paging. It might also indicate DASD problems, or higher priority tasks with high processor usage.

Db2 and RRS problems

Check that Db2 and RRS are active.

In all cases, carry out the following checks to determine the cause of the problem:

Is there a problem with the IBM MQ queues?

Use this topic for investigating potential problems with IBM MQ queues.

If you suspect that there is a problem affecting the queues on your subsystem, use the operations and control panels to display the system-command input queue.

If the system responds

If the system responds, then at least one queue is working. In this case, follow the procedure in "Are some of your queues working?" on page 33.

If the system does not respond

The problem might be with the whole subsystem. In this instance, try stopping and restarting the queue manager, responding to any error messages that are produced.

Check for any messages on the console needing action. Resolve any that might affect IBM MQ, such as a request to mount a tape for an archive log. See if other subsystems or CICS regions are affected.

Use the DISPLAY QMGR COMMANDQ command to identify the name of the system command input queue.

If the problem still occurs after restart

Contact your IBM support center for help (see "Contacting IBM Software Support" on page 328).

Related concepts

"Are the correct queues defined?" on page 34
IBM MQ requires certain predefined queues. Problems can occur if these queues are not defined correctly.

"Does the problem affect only remote or cluster queues?" on page 34
Use this topic for further investigation if the problem only occurs on remote or cluster queues.

"Does the problem affect only shared queues?" on page 35
Use this topic to investigate possible queue-sharing group issues which can cause problems for shared queues.

Are some of your queues working?

Use this topic to investigate when problems occur with a subset of your queues.

If you suspect that the problem occurs with only a subset of queues, select the name of a local queue that you think is having problems and perform the following procedures:

Display queue information

Use the DISPLAY QUEUE and DISPLAY QSTATUS commands to display information about the queue.

Is the queue being processed?

- If CURDEPTH is at MAXDEPTH, it might indicate that the queue is not being processed. Check that all applications that use the queue are running normally (for example, check that transactions in your CICS system are running or that applications started in response to Queue Depth High events are running).
- Issue DISPLAY QSTATUS(xx) IPPROCS to see if the queue is open for input. If not, start the application.

- If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:
  - If triggering is being used:
    - Is the trigger monitor running?
    - Is the trigger depth too big?
    - Is the process name correct?
    - Have **all** the trigger conditions been met?

      Issue DISPLAY QSTATUS(xx) IPPROCS to see if an application has the same queue open for input. In some triggering scenarios, a trigger message is not produced if the queue is open for input. Stop the application to cause the triggering processing to be invoked.
  - Can the queue be shared? If not, another application (batch, IMS, or CICS) might already have it open for input.
  - Is the queue enabled appropriately for GET and PUT?

Do you have a long-running unit of work?
If CURDEPTH is not zero, but when you attempt to MQGET a message the queue manager replies that there is no message available, issue either DIS QSTATUS(xx) TYPE(HANDLE) to show you information about applications that have the queue open, or issue DIS CONN(xx) to give you more information about an application that is connected to the queue.

How many tasks are accessing the queues?
Issue DISPLAY QSTATUS(xx) OPPROCS IPPROCS to see how many tasks are putting messages on to, and getting messages from the queue. In a queue-sharing environment, check OPPROCS and IPPROCS on each queue manager. Alternatively, use the CMDSCOPE attribute to check all the queue managers. If there are no application processes getting messages from the queue, determine the reason (for example, because the applications need to be started, a connection has been disrupted, or because the MQOPEN call has failed for some reason).

Is this queue a shared queue? Does the problem affect only shared queues?

Check that there is not a problem with the sysplex elements that support shared queues. For example, check that there is not a problem with the IBM MQ-managed Coupling Facility list structure.

Use D XCF, STRUCTURE, STRNAME=ALL to check that the Coupling Facility structures are accessible.

Use D RRS to check that RRS is active.

Is this queue part of a cluster?
Check to see if the queue is part of a cluster (from the CLUSTER or CLUSNL attribute). If it is, verify that the queue manager that hosts the queue is still active in the cluster.

If you cannot solve the problem
Contact your IBM support center for help (see “Contacting IBM Software Support” on page 328).

Are the correct queues defined?

IBM MQ requires certain predefined queues. Problems can occur if these queues are not defined correctly.

Check that the system-command input queue, the system-command reply model queue, and the reply-to queue are correctly defined, and that the MQOPEN calls were successful.

If you are using the system-command reply model queue, check that it was defined correctly.

If you are using clusters, you need to define the SYSTEM.CLUSTER.COMMAND.QUEUE to use commands relating to cluster processing.

Does the problem affect only remote or cluster queues?

Use this topic for further investigation if the problem only occurs on remote or cluster queues.

If the problem affects only remote or cluster queues, check:

Are the remote queues being accessed?

Check that the programs putting messages to the remote queues have run successfully (see "Dealing with incorrect output" on page 425).

Is the system link active?

Use APPC or TCP/IP commands as appropriate to check whether the link between the two systems is active.

Use PING or OPING for TCP/IP or D NET ID=xxxxx, E for APPC.

Is triggering working?

If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on and that the queue is get-enabled.

Is the channel or listener running?

If necessary, start the channel or the listener manually, or try stopping and restarting the channel. See Configuring distributed queuing for more information.

Look for error messages on the startup of the channel initiator and listener. See IBM MQ for z/OS messages, completion, and reason codes and Configuring distributed queuing to determine the cause.

What is the channel status?

Check the channel status using the DISPLAY CHSTATUS (channel_name) command.

Are your process and channel definitions correct?

Check your process definitions and your channel definitions.

See Configuring distributed queuing for information about how to use distributed queuing, and for information about how to define channels.

Does the problem affect only shared queues?

Use this topic to investigate possible queue-sharing group issues which can cause problems for shared queues.

If the problem affects only queue-sharing groups, use the VERIFY QSG function of the CSQ5PQSG utility. This command verifies that the Db2 setup is consistent in terms of the bitmap allocation fields, and object definition for the Db2 queue manager, structure, and shared queue objects, and reports details of any inconsistency that is discovered.

The following is an example of a VERIFY QSG report with errors:

```
CSQU501I  VERIFY QSG function requested
CSQU503I  QSG=SQ02, DB2 DSG=DSN710P5, DB2 ssid=DFP5
CSQU517I  XCF group CSQGSQ02 already defined
CSQU520I  Summary information for XCF group CSQGSQ02
CSQU522I  Member=MQ04, state=QUIESCED, system=MV4A
CSQU523I  User data=D4E5F4C15AD4D8F0F4404040C4C5....
CSQU522I  Member=MQ03, state=QUIESCED, system=MV4A
CSQU523I  User data=D4E5F4C15AD4D8F0F3404040C4C6....
CSQU526I  Connected to Db2 DF4A
CSQU572E  Usage map T01_ARRAY_QMGR and Db2 table CSQ.ADMIN_B_QMGR inconsistent
CSQU573E  QMGR MQ04 in table entry 1 not set in usage map
CSQU574E  QMGR 27 in usage map has no entry in table
CSQU572E  Usage map T01_ARRAY_STRUC and Db2 table CSQ.ADMIN_B_STRUCTURE inconsistent
CSQU575E  Structure APPL2 in table entry 4 not set in usage map
CSQU576E  Structure 55 in usage map has no entry in table
CSQU572E  Usage map T03_LH_ARRAY and Db2 table CSQ.OBJ_B_QUEUE inconsistent
CSQU577E  Queue MYSQ in table entry 13 not set in usage map for structure APPL1
CSQU576E  Queue 129 in usage map for structure APPL1 has no entry in table
CSQU528I  Disconnected from Db2 DF4A
CSQU148I  CSQ5PQSG Utility completed, return code=12
```

Does the problem affect specific parts of the network?

Network problems can cause related problems for MQ for z/OS. Use this topic to review possible sources of networks problems.

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote queue manager is not working, the messages cannot flow to a target queue on the target queue manager. Check that the connection between the two systems is available, and that the channel initiator and listener have been started. Use the MQSC `PING CHANNEL` command to check the connection.

Check that messages are reaching the transmission queue, and check the local queue definition of the transmission queue, and any remote queues. Use the MQSC BYTSSENT keyword of the `DISPLAY CHSTATUS` command to check that data is flowing along the channel. Use `DISPLAY QLOCAL (XMITQ) CURDEPTH` to check whether there are messages to be sent on the transmission queue. Check for diagnostic messages at both ends of the channel informing you that messages have been sent to the dead-letter queue.

If you are using IBM MQ clusters, check that the clustering definitions have been set up correctly.

Have you made any network-related changes that might account for the problem?

Have you changed any IBM MQ definitions, or any CICS or IMS definitions? Check the triggering attributes of the transmission queue.

Problems that occur at specific times of the day or affect specific users

Use this topic to review IBM MQ problems that occur at specific times of the day or specific groups of users.

If the problem occurs at specific times of day, it might be that it is dependent on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, and so these periods are the times when load-dependent problems are most likely to occur. (If your network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

If you think that your IBM MQ for z/OS system has a performance problem, see "Dealing with performance problems on z/OS" on page 419.

If the problem only affects some users, is it because some users do not have the correct security authorization? See User IDs for security checking for information about user IDs checked by IBM MQ for z/OS.

Is the problem intermittent or does the problem occur with all z/OS, CICS, or IMS systems?

Review this topic to consider if problems are caused by application interaction or are related to other z/OS systems.

An intermittent problem could be caused by failing to take into account the fact that processes can run independently of each other. For example, a program might issue an `MQGET` call, without specifying WAIT, before an earlier process has completed. You might also encounter this type of problem if your application tries to get a message from a queue while it is in sync point (that is, before it has been committed).

If the problem only occurs when you access a particular z/OS, IMS, or CICS system, consider what is different about this system. Also consider whether any changes have been made to the system that might affect the way it interacts with IBM MQ.

Has the application run successfully before?

Application errors can often be determined by determining if they have run successfully before or if they have produced error messages and unexpected return codes.

If the problem appears to involve one particular application, consider whether the application has run successfully before.

Before you answer Yes to this question, consider:

Have any changes been made to the application since it last ran successfully?
If so, it is likely that the error lies somewhere in the new or modified part of the application. Investigate the changes and see if you can find an obvious reason for the problem.

Have all the functions of the application been fully exercised before?

Did problem occur when part of the application that had never been started before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.

If a program has been run successfully on many previous occasions, check the current queue status and files that were being processed when the error occurred. It is possible that they contain some unusual data value that causes a rarely used path in the program to be invoked.

Does the application check all return codes?
Has your system has been changed, perhaps in a minor way. Check the return codes your application receives as a result of the change. For example:

- Does your application assume that the queues it accesses can be shared? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
- Have any security profiles been altered? An MQOPEN call might fail because of a security violation; can your application recover from the resulting return code?

Does the application expect particular message formats?
If a message with an unexpected message format has been put onto a queue (for example, a message from a queue manager on a different platform), it might require data conversion or another different form of processing.

Does the application run on other IBM MQ for z/OS systems?
Is something different about the way that this queue manager is set up that is causing the problem? For example, have the queues been defined with the same maximum message length, or default priority?

Does the application use the MQSET call to change queue attributes?
Is the application is designed to set a queue to have no trigger, then process some work, then set the queue to have a trigger? The application might have failed before the queue had been reset to have a trigger.

Does the application handle messages that cause an application to fail?
If an application fails because of a corrupted message, the message retrieved is rolled back. The next application might get the same message and fail in the same way. Ensure that applications use the backout count; when the backout count threshold has been reached, the message in question is put onto the backout queue.

If your application has never run successfully before, examine your application carefully to see if you can find any of the following errors:

Translation and compilation problems
Before you look at the code, examine the output from the translator, the compiler or assembler, and the linkage editor, to see if any errors have been reported. If your application fails to translate, compile/assemble, or link edit into the load library, it also fails to run if you attempt to invoke it. See Developing applications for information about building your application, and for examples of the job control language (JCL) statements required.

Batch and TSO programs
For batch and TSO programs, check that the correct stub has been included. There is one batch stub and two RRS stubs. If you are using RRS, check that you are not using the MQCMIT and MQBACK calls with the CSQBRSTB stub. Use the CSQBRRSI stub if you want to continue using these calls with RRS.

CICS programs
For CICS programs, check that the program, the IBM MQ CICS stub, and the CICS stub have been linked in the correct order. Also, check that your program or transaction is defined to CICS.

IMS programs
For IMS programs, check that the link includes the program, the IBM MQ stub, and the IMS language interface module. Ensure that the correct entry point has been specified. A program that is loaded dynamically from an IMS program must have the stub and language interface module linked also if it is to use IBM MQ.

Possible code problems
If the documentation shows that each step was accomplished without error, consider the coding of the application. Do the symptoms of the problem indicate the function that is failing and, therefore, the piece of code in error? See "Do you have a program error?" on page 38 for some examples of common errors that cause problems with IBM MQ applications.

Do applications report errors from IBM MQ ?
For example, a queue might not be enabled for "gets". It receives a return code specifying this condition but does not report it. Consider where your applications report any errors or problems.

Have any changes been made since the last successful run?

Recent changes made since the last successful run are often the source of unexpected errors. This topic contains information about some of the changes which can be investigated as part of your problem determination.

When you are considering changes that might recently have been made, think about IBM MQ, and also about the other programs it interfaces with, the hardware, and any new applications. Consider also the possibility that a new application that you don not yet know about might have been run on the system.

Has your initialization procedure been changed?
Consider whether that might be the cause of the problem. Have you changed any data sets, or changed a library definition? Has z/OS been initialized with different parameters? In addition, check for error messages sent to the console during initialization.

Have you changed any queue definitions or security profiles?
Consider whether some of your queues have been altered so that they are members of a cluster. This change might mean that messages arrive from different sources (for example, other queue managers or applications).

Have you changed any definitions in your sysplex that relate to the support and implementation of shared queues?
Consider the effect that changes to such definitions as your sysplex couple data set, or Coupling Facility resource management policy. These changes might have on the operation of shared queues. Also, consider the effect of changes to the Db2 data sharing environment.

Has any of the software on your z/OS system been upgraded to a later release?
Consider whether there are any necessary post-installation or migration activities that you need to perform.

Has your z/OS subsystem name table been changed?
Changes to levels of corequisite software like z/OS or LE might require additional changes to IBM MQ.

Do your applications deal with return codes that they might get as a result of any changes you have made?
Ensure that your applications deal with any new return codes that you introduce.

Do you have a program error?

Use this topic to investigate if a program error is causing an IBM MQ problem.

The examples that follow illustrate the most common causes of problems encountered while running IBM MQ programs. Consider the possibility that the problem with your system could be caused by one of these errors.

- Programs issue MQSET to change queue attributes and fail to reset attributes of a queue. For example, setting a queue to NOTRIGGER.

- Making incorrect assumptions about the attributes of a queue. This assumption could include assuming that queues can be opened with MQOPEN when they are MQOPEN-exclusive, and assuming that queues are not part of a cluster when they are.
- Trying to access queues and data without the correct security authorization.
- Linking a program with no stub, or with the wrong stub (for example, a TSO program with the CICS stub). This can cause either a long-running unit of work, or an X'0C4' or other abend.
- Passing incorrect or invalid parameters in an MQI call; if the wrong number of parameters are passed, no attempt can be made to complete the completion code and reason code fields, and the task is abended. (This is an X'0C4' abend.)

  This problem might occur if you attempt to run an application on an earlier version of MQSeries® than it was written for, where some of the MQI values are invalid.
- Failing to define the IBM MQ modules to z/OS correctly (this error causes an X'0C4' abend in CSQYASCP).
- Failing to check return codes from MQI requests.

  This problem might occur if you attempt to run an application on a later version of IBM MQ than it was written for, where new return codes have been introduced that are not checked for.
- Failing to open objects with the correct options needed for later MQI calls, for example using the MQOPEN call to open a queue but not specifying the correct options to enable the queue for subsequent MQGET calls.
- Failing to initialize *MsgId* and *CorrelId* correctly.

  This error is especially true for MQGET.
- Using incorrect addresses.
- Using storage before it has been initialized.
- Passing variables with incorrect lengths specified.
- Passing parameters in the wrong order.
- Failing to define the correct security profiles and classes to RACF®.

  This might stop the queue manager or prevent you from carrying out any productive work.
- Relying on default MQI options for a ported application.

  For example, z/OS defaults to MQGET and MQPUT in sync point. The distributed-platform default is out of sync point.
- Relying on default behavior at a normal or abnormal end of a portal application.

  On z/OS, a normal end does an implicit MQCMIT and an abnormal end does an implicit rollback.

Has there been an abend?

Use this topic to investigate common causes of abends and the different types of abend that can cause problems.

If your application has stopped running, it can be caused by an abnormal termination (abend).

You are notified of an abend in one of the following places, depending on what type of application you are using:

Batch
: Your listing shows the abend.

CICS
: You see a CICS transaction abend message. If your task is a terminal task, this message is displayed on your screen. If your task is not attached to a terminal, the message is displayed on the CICS CSMT log.

IMS
In all cases, you see a message at the IBM MQ for IMS master terminal and in the listing of the dependent region involved. If an IMS transaction that had been entered from a terminal was being processed, an error message is also sent to that terminal.

TSO
You might see a TSO message with a return code on your screen. (Whether this message is displayed depends on the way your system is set up, and the type of error.)

Common causes of abends

Abends can be caused by the user ending the task being performed before it terminates normally; for example, if you purge a CICS transaction. Abends can also be caused by an error in an application program.

Address space dumps and transaction dumps

For some abends, an address space dump is produced. For CICS transactions, a transaction dump showing the storage areas of interest to the transaction is provided.

- If an application passes some data, the address of which is no longer valid, a dump is sometimes produced in the address space of the user.

  Note: For a batch dump, the dump is formatted and written to SYSUDUMP. For information about SYSUDUMPs, see "SYSUDUMP information" on page 416. For CICS, a system dump is written to the SYS1.DUMP data sets, as well as a transaction dump being taken.
- If a problem with IBM MQ for z/OS itself causes an abend, an abend code of X'5C6' or X'6C6' is returned, along with an abend reason code. This reason code uniquely describes the cause of the problem. See "IBM MQ for z/OS abends" on page 393 for information about the abend codes, and see Return codes for an explanation of the reason code.

Abnormal program termination

If your program has terminated abnormally, see "Dealing with program abends" on page 394.

If your system has terminated abnormally, and you want to analyze the dump produced, see "IBM MQ for z/OS dumps" on page 400. This section tells you how to format the dump, and how to interpret the data contained in it.

Have you obtained incorrect output?

Use this topic to review any incorrect output you have received.

If you have obtained what you believe to be some incorrect output, consider the following:

Classifying incorrect output
"Incorrect output˒ might be regarded as any output that you were not expecting. However, use this term with care in the context of problem determination because it might be a secondary effect of some other type of error. For example, looping could be occurring if you get any repetitive output, even though that output is what you expected.

Error messages
IBM MQ also responds to many errors it detects by sending error messages. You might regard these messages as "incorrect output˒, but they are only symptoms of another type of problem. If you have received an error message from IBM MQ that you were not expecting, refer to "Are there any error messages, return codes or other error conditions?" on page 30.

Unexpected messages
If your application has not received a message that it was expecting, has received a message containing unexpected or corrupted information, or has received a message that it was not expecting (for example, one that was destined for a different application), refer to "Dealing with incorrect output" on page 425.

Can you reproduce the problem?

Reproducing the problem can be used to assist problem determination for IBM MQ for z/OS. Use this topic to further isolate the type of problem reproduction.

If you can reproduce the problem, consider the conditions under which you can reproduce it. For example:

Is it caused by a command?
If so, is the command issued from the z/OS console, from CSQUTIL, from a program written to put commands onto the SYSTEM.COMMAND.INPUT queue, or by using the operations and control panels?

Does the command work if it is entered by another method?
If the command works when it is entered at the console, but not otherwise, check that the command server has not stopped, and that the queue definition of the SYSTEM.COMMAND.INPUT queue has not been changed.

Is the command server running?
Issue the command `DIS CMDSERV` to check.

Is it caused by an application?

If so, does it fail in CICS, IMS, TSO, or batch?

Does it fail on all IBM MQ systems, or only on some?

Is an application causing the problem?
Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.

Have you failed to receive a response from an MQSC command?

Use this topic for investigating problems where you fail to receive a response from an MQSC command.

If you have issued an MQSC command from an application (and not from a z/OS console), but you have not received a response, consider the subsequent questions:

Is the command server running?

Check that the command server is running, as follows:

1. Use the DISPLAY CMDSERV command at the z/OS console to display the status of the command server.
2. If the command server is not running, start it using the START CMDSERV command.
3. If the command server is running, issue the DISPLAY QUEUE command. Use the name of the system-command input queue and the CURDEPTH and MAXDEPTH attributes to define the data displayed.

   If these values show that the queue is full, and the command server has been started, the messages are not being read from the queue.
4. Try stopping the command server and then restarting it, responding to any error messages that are produced.
5. Issue the display command again to see if it is working now.

Has a reply been sent to the dead-letter queue?

Use the DISPLAY QMGR DEADQ command to find out the name of the system dead-letter queue (if you do not know what it is).

Use this name in the DISPLAY QUEUE command with the CURDEPTH attribute to see if there are any messages on the queue.

The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code describing the problem. (See Reason (MQLONG) for information about the dead-letter header structure.)

Are the queues enabled for PUTs and GETs?
Use the DISPLAY QUEUE command from the console to check, for example, DISPLAY QUEUE(SYSTEM.COMMAND.INPUT) PUT GET.

Is the `WaitInterval` parameter set to a sufficiently long time?
If your MQGET call has timed out, your application receives completion code of 2 and a reason code of 2033 (MQRC_NO_MSG_AVAILABLE). (See WaitInterval (MQLONG) and MQGET - Get message for information about the *`WaitInterval`* parameter, and completion and reason codes from MQGET.)

Is a sync point required?

If you are using your own application program to put commands onto the system-command input queue, consider whether you must take a sync point.

You must take a sync point after putting messages to a queue, and before attempting to receive reply messages, or use MQPMO_NO_SYNCPOINT when putting them. Unless you have excluded your request message from sync point, you must take a sync point before attempting to receive reply messages.

Are the `MaxDepth` and `MaxMsgL` parameters of your queues set sufficiently high?
See CSQO016E for information about defining the system-command input queue and the reply-to queue.

Are you using the *`CorrelId`* and *`MsgId`* parameters correctly?

You must identify the queue and then display the CURDEPTH. Use the DISPLAY QUEUE command from the console (for example, DISPLAY QUEUE (MY.REPLY.QUEUE) CURDEPTH), to see if there are messages on the reply-to queue that you have not received.

Set the values of *`MsgId`* and *`CorrelId`* in your application to ensure that you receive all messages from the queue.

The following questions are applicable if you have issued an MQSC command from either a z/OS console (or its equivalent), or an application, but have not received a response:

Is the queue manager still running, or did your command cause an abend?
Look for error messages indicating an abend, and if one occurred, see "IBM MQ for z/OS dumps" on page 400.

Were any error messages issued?
Check to see if any error messages were issued that might indicate the nature of the error.

See Issuing commands for information about the different methods you can use to enter MQSC commands.

Is your application or IBM MQ for z/OS running slowly?

Slow applications can be caused by the application itself or underlying software including IBM MQ. Use this topic for initial investigations into slow applications.

If your application is running slowly, this could indicate that it is in a loop, or waiting for a resource that is not available.

Is the problem worse at peak system load times?
This could also be caused by a performance problem. Perhaps it is because your system needs tuning, or because it is operating near the limits of its capacity. This type of problem is probably worst at peak system load times, typically at mid-morning and mid-afternoon. (If your network extends across more than one time zone, peak system load might seem to you to occur at some other time.)

Does the problem occur when the system is lightly loaded?
If you find that degrading performance is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly designed application program is probably to blame. This could manifest itself as a problem that only occurs when specific queues are accessed.

Is IBM MQ for z/OS running slowly?

The following symptoms might indicate that IBM MQ for z/OS is running slowly:

- If your system is slow to respond to commands.
- If repeated displays of the queue depth indicate that the queue is being processed slowly for an application with which you would expect a large amount of queue activity.

You can find guidance on dealing with waits and loops in "Dealing with applications that are running slowly or have stopped on z/OS" on page 420, and on dealing with performance problems in "Dealing with performance problems on z/OS" on page 419.

Reason codes and exceptions

You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

- IBM MQ AMQ messages
- "API completion and reason codes" on page 43
- "PCF reason codes" on page 244
- "Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes" on page 315
- "WCF custom channel exceptions" on page 320
- z/OS IBM MQ for z/OS messages, completion, and reason codes
- WMQ JMS Exception Messages

API completion and reason codes

For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call.

For more information about the IBM MQ API, see Developing applications, and the reference information in Developing applications reference.

For a full list and explanation of the API reason codes, see "API reason codes" on page 44.

API completion codes

The following is a list of the completion codes (MQCC) returned by IBM MQ

0: Successful completion (MQCC_OK)

The call completed fully; all output parameters have been set.

The *Reason* parameter always has the value MQRC_NONE in this case.

1: Warning (partial completion) (MQCC_WARNING)

The call completed partially. Some output parameters might have been set in addition to the *CompCode* and *Reason* output parameters.

The *Reason* parameter gives additional information.

2: Call failed (MQCC_FAILED)

The processing of the call did not complete, and the state of the queue manager is normally unchanged; exceptions are specifically noted. Only the *CompCode* and *Reason* output parameters have been set; all other parameters are unchanged.

The reason might be a fault in the application program, or it might be a result of some situation external to the program, for example the application's authority might have been revoked. The *Reason* parameter gives additional information.

Related reference

"PCF reason codes" on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

"Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes" on page 315
IBM MQ can use Secure Sockets Layer (SSL) with the various communication protocols. Use this topic to identify the error codes that can be returned by SSL.

"WCF custom channel exceptions" on page 320
Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Related information

IBM MQ AMQ messages

z/OS IBM MQ for z/OS messages, completion, and reason codes

API reason codes

The reason code parameter (*Reason*) is a qualification to the completion code parameter (*CompCode*).

If there is no special reason to report, MQRC_NONE is returned. A successful call returns MQCC_OK and MQRC_NONE.

If the completion code is either MQCC_WARNING or MQCC_FAILED, the queue manager always reports a qualifying reason; details are given under each call description.

Where user exit routines set completion codes and reasons, they should adhere to these rules. In addition, any special reason values defined by user exits should be less than zero, to ensure that they do not conflict with values defined by the queue manager. Exits can set reasons already defined by the queue manager, where these are appropriate.

Reason codes also occur in:

- The *Reason* field of the MQDLH structure
- The *Feedback* field of the MQMD structure

The following is a list of reason codes, in numeric order, providing detailed information to help you understand them, including:

- An explanation of the circumstances that have caused the code to be raised
- The associated completion code
- Suggested programmer actions in response to the code

0 (0000) (RC0): MQRC_NONE

Explanation

The call completed normally. The completion code (*CompCode*) is MQCC_OK.

Completion Code

MQCC_OK

Programmer response

None.

900 (0384) (RC900): MQRC_APPL_FIRST

Explanation

This is the lowest value for an application-defined reason code returned by a data-conversion exit. Data-conversion exits can return reason codes in the range MQRC_APPL_FIRST through MQRC_APPL_LAST to indicate particular conditions that the exit has detected.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

As defined by the writer of the data-conversion exit.

999 (03E7) (RC999): MQRC_APPL_LAST

Explanation

This is the highest value for an application-defined reason code returned by a data-conversion exit. Data-conversion exits can return reason codes in the range MQRC_APPL_FIRST through MQRC_APPL_LAST to indicate particular conditions that the exit has detected.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

As defined by the writer of the data-conversion exit.

2001 (07D1) (RC2001): MQRC_ALIAS_BASE_Q_TYPE_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued specifying an alias queue as the destination, but the *BaseQName* in the alias queue definition resolves to a queue that is not a local queue, a local definition of a remote queue, or a cluster queue V 8.0.0.6, or a queue in a distribution list contains an alias queue that is pointing to a topic object.

Completion Code

MQCC_FAILED

Programmer response

Correct the queue definitions.

This reason code is also used to identify the corresponding event message Alias Base Queue Type Error.

2002 (07D2) (RC2002): MQRC_ALREADY_CONNECTED

Explanation

An MQCONN or MQCONNX call was issued, but the application is already connected to the queue manager.

- On z/OS, this reason code occurs for batch and IMS applications only; it does not occur for CICS applications.
- On UNIX, IBM i, Linux and Windows, this reason code occurs if the application attempts to create a nonshared handle when a nonshared handle exists for the thread. A thread can have no more than one nonshared handle.
- On UNIX, IBM i, Linux and Windows, this reason code occurs if an MQCONN call is issued from within an MQ channel exit, API Crossing Exit, or Async Consume Callback function, and a shared hConn is bound to this thread.
- On UNIX, IBM i, Linux and Windows, this reason code occurs if an MQCONNX call that does not specify one of the MQCNO_HANDLE_SHARE_* options is issued from within an MQ channel exit, API Crossing Exit, or Async Consume Callback function, and a shared hConn is bound to this thread
- On Windows, MTS objects do not receive this reason code, as additional connections to the queue manager are permitted.

Completion Code

MQCC_WARNING

Programmer response

None. The *Hconn* parameter returned has the same value as was returned for the previous MQCONN or MQCONNX call.

An MQCONN or MQCONNX call that returns this reason code does *not* mean that an additional MQDISC call must be issued to disconnect from the queue manager. If this reason code is returned because the application has been called in a situation where the MQCONN has already been done, do *not* issue a corresponding MQDISC, because this causes the application that issued the original MQCONN or MQCONNX call to be disconnected as well.

2003 (07D3) (RC2003): MQRC_BACKED_OUT

Explanation

The current unit of work encountered an unrecoverable error or was backed out. This reason code is issued in the following cases:

- On an MQCMIT or MQDISC call, when the commit operation fails and the unit of work is backed out. All resources that participated in the unit of work are returned to their state at the start of the unit of work. The MQCMIT or MQDISC call completes with MQCC_WARNING in this case.
  - On z/OS, this reason code occurs only for batch applications.
- On an MQGET, MQPUT, or MQPUT1 call that is operating within a unit of work, when the unit of work already encountered an error that prevents the unit of work from being committed (for example, when the log space is exhausted). The application must issue the appropriate call to back out the unit of work. (For a unit of work that is coordinated by the queue manager, this call is the MQBACK call, although the MQCMIT call has the same effect in these circumstances.) The MQGET, MQPUT, or MQPUT1 call completes with MQCC_FAILED in this case.
  - On z/OS, this case does not occur.
- On an asynchronous consumption callback (registered by an MQCB call), the unit of work is backed out and the asynchronous consumer should call MQBACK.
  - On z/OS, this case does not occur.
- For the IBM MQ client on HP Integrity NonStop Server using TMF, this return code can occur:
  - For MQGET, MQPUT, and MQPUT1 calls, if you have an active transaction that is being coordinated by TMF, but the IBM MQ part of the transaction is rolled back because of inactivity on the transaction.

  - If the TMF/Gateway detects that TMF is rolling back the current transaction before the application finishes with it.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Check the returns from previous calls to the queue manager. For example, a previous MQPUT call might have failed.

2004 (07D4) (RC2004): MQRC_BUFFER_ERROR

Explanation

The *Buffer* parameter is not valid for one of the following reasons:

- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The parameter pointer points to storage that cannot be accessed for the entire length specified by *BufferLength*.
- For calls where *Buffer* is an output parameter: the parameter pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2005 (07D5) (RC2005): MQRC_BUFFER_LENGTH_ERROR

Explanation

The *BufferLength* parameter is not valid, or the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason can also be returned to an MQ MQI client program on the MQCONN or MQCONNX call if the negotiated maximum message size for the channel is smaller than the fixed part of any call structure.

This reason should also be returned by the MQZ_ENUMERATE_AUTHORITY_DATA installable service component when the *AuthorityBuffer* parameter is too small to accommodate the data to be returned to the invoker of the service component.

This reason code can also be returned when a zero length multicast message has been supplied where a positive length is required.

Completion Code

MQCC_FAILED

Programmer response

Specify a value that is zero or greater. For the mqAddString and mqSetString calls, the special value MQBL_NULL_TERMINATED is also valid.

2006 (07D6) (RC2006): MQRC_CHAR_ATTR_LENGTH_ERROR

Explanation

CharAttrLength is negative (for MQINQ or MQSET calls), or is not large enough to hold all selected attributes (MQSET calls only). This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Specify a value large enough to hold the concatenated strings for all selected attributes.

2007 (07D7) (RC2007): MQRC_CHAR_ATTRS_ERROR

Explanation

CharAttrs is not valid. The parameter pointer is not valid, or points to read-only storage for MQINQ calls or to storage that is not as long as implied by *CharAttrLength*. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2008 (07D8) (RC2008): MQRC_CHAR_ATTRS_TOO_SHORT

Explanation

For MQINQ calls, *CharAttrLength* is not large enough to contain all of the character attributes for which MQCA_* selectors are specified in the *Selectors* parameter.

The call still completes, with the *CharAttrs* parameter string filled in with as many character attributes as there is room for. Only complete attribute strings are returned: if there is insufficient space remaining to accommodate an attribute in its entirety, that attribute and subsequent character attributes are omitted. Any space at the end of the string not used to hold an attribute is unchanged.

An attribute that represents a set of values (for example, the namelist *Names* attribute) is treated as a single entity—either all of its values are returned, or none.

Completion Code

MQCC_WARNING

Programmer response

Specify a large enough value, unless only a subset of the values is needed.

2009 (07D9) (RC2009): MQRC_CONNECTION_BROKEN

Explanation

Connection to the queue manager has been lost. This can occur because the queue manager has ended. If the call is an MQGET call with the MQGMO_WAIT option, the wait has been canceled. All connection and object handles are now invalid.

For MQ MQI client applications, it is possible that the call did complete successfully, even though this reason code is returned with a *CompCode* of MQCC_FAILED.

Completion Code

MQCC_FAILED

Programmer response

Applications can attempt to reconnect to the queue manager by issuing the MQCONN or MQCONNX call. It might be necessary to poll until a successful response is received.

- z/OS On z/OS for CICS applications, it is not necessary to issue the MQCONN or MQCONNX call, because CICS applications are connected automatically.

Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

z/OS For z/OS IMS check that the subsystem is started using the IMS DIS SUBSYS command, and if necessary start it using the IMS STA SUBSYS command.

Related information

IBM MQ and IMS

2010 (07DA) (RC2010): MQRC_DATA_LENGTH_ERROR

Explanation

The *DataLength* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason can also be returned to an MQ MQI client program on the MQGET, MQPUT, or MQPUT1 call, if the *BufferLength* parameter exceeds the maximum message size that was negotiated for the client channel.

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

If the error occurs for an MQ MQI client program, also check that the maximum message size for the channel is big enough to accommodate the message being sent; if it is not big enough, increase the maximum message size for the channel.

2011 (07DB) (RC2011): MQRC_DYNAMIC_Q_NAME_ERROR

Explanation

On the MQOPEN call, a model queue is specified in the *ObjectName* field of the *ObjDesc* parameter, but the *DynamicQName* field is not valid, for one of the following reasons:

- *DynamicQName* is completely blank (or blank up to the first null character in the field).
- Characters are present that are not valid for a queue name.
- An asterisk is present beyond the 33rd position (and before any null character).
- An asterisk is present followed by characters that are not null and not blank.

This reason code can also sometimes occur when a server application opens the reply queue specified by the *ReplyToQ* and *ReplyToQMgr* fields in the MQMD of a message that the server has just received. In this case the reason code indicates that the application that sent the original message placed incorrect values into the *ReplyToQ* and *ReplyToQMgr* fields in the MQMD of the original message.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid name.

2012 (07DC) (RC2012): MQRC_ENVIRONMENT_ERROR

Explanation

The call is not valid for the current environment.

- z/OS On z/OS, when one of the following applies:
  - An MQCONN or MQCONNX call was issued, but the application has been linked with an adapter that is not supported in the environment in which the application is running. For example, this can arise when the application is linked with the MQ RRS adapter, but the application is running in a Db2 Stored Procedure address space. RRS is not supported in this environment. Stored Procedures wishing to use the MQ RRS adapter must run in a Db2 WLM-managed Stored Procedure address space.
  - An MQCMIT or MQBACK call was issued, but the application has been linked with the RRS batch adapter CSQBRSTB. This adapter does not support the MQCMIT and MQBACK calls.
  - An MQCMIT or MQBACK call was issued in the CICS or IMS environment.
  - The RRS subsystem is not operational on the z/OS system that ran the application.
  - An MQCTL call with MQOP_START or an MQCB call registering an Event Listener, was issued but the application is not allowed to create a POSIX thread.
  - An IBM MQ classes for Java application has instantiated an MQQueueManager object using the CLIENT transport. The z/OS environment only supports the use of the BINDINGS transport.
- On IBM i, HP Integrity NonStop Server, UNIX systems, and Windows, when one of the following applies:
  - The application is linked to unsupported libraries.
  - The application is linked to the wrong libraries (threaded or nonthreaded).
  - An MQBEGIN, MQCMIT, or MQBACK call was issued, but an external unit-of-work manager is in use. For example, this reason code occurs on Windows when an MTS object is running as a DTC transaction. This reason code also occurs if the queue manager does not support units of work.
  - The MQBEGIN call was issued in an MQ MQI client environment.
  - An MQXCLWLN call was issued, but the call did not originate from a cluster workload exit.

  - An MQCONNX call was issued specifying the option MQCNO_HANDLE_SHARE_NONE from within an MQ channel exit, an API Exit, or a Callback function. The reason code occurs only if a shared hConn is bound to the application thread.
  - An IBM MQ Object is unable to connect fastpath.
  - An IBM MQ classes for Java application has created an MQQueueManager object that uses the CLIENT transport, and then called MQQueueManager.begin(). This method can only be called on MQQueueManager objects that use the BINDINGS transport.
- On Windows, when using the managed .NET client, an attempt was made to use one of the unsupported features:
  - Unmanaged channel exits
  - Secure Sockets Layer (SSL)
  - XA Transactions
  - Communications other than TCP/IP
  - Channel compression

Completion Code

MQCC_FAILED

Programmer response

Do one of the following (as appropriate):

- On z/OS:
  - Link the application with the correct adapter.
  - Modify the application to use the SRRCMIT and SRRBACK calls in place of the MQCMIT and MQBACK calls. Alternatively, link the application with the RRS batch adapter CSQBRRSI. This adapter supports MQCMIT and MQBACK in addition to SRRCMIT and SRRBACK.
  - For a CICS or IMS application, issue the appropriate CICS or IMS call to commit or backout the unit of work.
  - Start the RRS subsystem on the z/OS system that is running the application.
  - If your application uses Language Environment (LE) ensure that it uses the DLL interface and it runs with POSIX(ON).
  - Ensure that your application is allowed to use Unix System Services (USS).
  - Ensure that your Connection Factory definitions for local z/OS applications and WebSphere Application Server applications use Transport Type with bindings mode connections.
- In the other environments:
  - Link the application with the correct libraries (threaded or nonthreaded).
  - Remove from the application the call or feature that is not supported.
  - Change your application to run setuid, if you want to run fastpath.

2013 (07DD) (RC2013): MQRC_EXPIRY_ERROR

Explanation

On an MQPUT or MQPUT1 call, the value specified for the *Expiry* field in the message descriptor MQMD is not valid.

V 8.0.0.4 This reason code is also generated by JMS applications specifying a delivery delay value greater than either the:

- Message expiry time specified by the application, or

- Expiry time set by the **CUSTOM**(*CAPEXPRY*) attribute of the objects used in the resolution of the target queue or topic.

Completion Code

MQCC_FAILED

Programmer response

Specify a value that is greater than zero, or the special value MQEI_UNLIMITED.

V 8.0.0.4 Ensure that the delivery delay specified by JMS applications is less than the:

- Message expiry time specified by the application, or
- Expiry time set by the **CUSTOM**(*CAPEXPRY*) attribute of the objects used in the resolution of the target queue or topic.

2014 (07DE) (RC2014): MQRC_FEEDBACK_ERROR

Explanation

On an MQPUT or MQPUT1 call, the value specified for the *Feedback* field in the message descriptor MQMD is not valid. The value is not MQFB_NONE, and is outside both the range defined for system feedback codes and the range defined for application feedback codes.

Completion Code

MQCC_FAILED

Programmer response

Specify MQFB_NONE, or a value in the range MQFB_SYSTEM_FIRST through MQFB_SYSTEM_LAST, or MQFB_APPL_FIRST through MQFB_APPL_LAST.

2016 (07E0) (RC2016): MQRC_GET_INHIBITED

Explanation

MQGET calls are currently inhibited for the queue, or for the queue to which this queue resolves.

Completion Code

MQCC_FAILED

Programmer response

If the system design allows get requests to be inhibited for short periods, retry the operation later.

This reason code is also used to identify the corresponding event message Get Inhibited.

System programmer action

Use ALTER QLOCAL(...) GET(ENABLED) to allow messages to be got.

2017 (07E1) (RC2017): MQRC_HANDLE_NOT_AVAILABLE

Explanation

An MQOPEN, MQPUT1 or MQSUB call was issued, but the maximum number of open handles allowed for the current task has already been reached. Be aware that when a distribution list is specified on the MQOPEN or MQPUT1 call, each queue in the distribution list uses one handle.

- On z/OS, "task˵ means a CICS task, a z/OS task, or an IMS-dependent region.

In addition, the MQSUB call allocates two handles when you do not provide an object handle on input.

Completion Code

MQCC_FAILED

Programmer response

Check whether the application is issuing MQOPEN calls without corresponding MQCLOSE calls. If it is, modify the application to issue the MQCLOSE call for each open object as soon as that object is no longer needed.

Also check whether the application is specifying a distribution list containing a large number of queues that are consuming all of the available handles. If it is, increase the maximum number of handles that the task can use, or reduce the size of the distribution list. The maximum number of open handles that a task can use is given by the *MaxHandles* queue manager attribute.

2018 (07E2) (RC2018): MQRC_HCONN_ERROR

Explanation

The connection handle *Hconn* is not valid, for one of the following reasons:

- The parameter pointer is not valid, or (for the MQCONN or MQCONNX call) points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The value specified was not returned by a preceding MQCONN or MQCONNX call.
- The value specified has been made invalid by a preceding MQDISC call.
- The handle is a shared handle that has been made invalid by another thread issuing the MQDISC call.
- The handle is a shared handle that is being used on the MQBEGIN call (only nonshared handles are valid on MQBEGIN).
- The handle is a nonshared handle that is being used a thread that did not create the handle.
- The call was issued in the MTS environment in a situation where the handle is not valid (for example, passing the handle between processes or packages; note that passing the handle between library packages *is* supported).
- The conversion program is not defined as OPENAPI, when the MQXCNVC call is invoked by running a character conversion exit program with CICS TS 3.2 or higher. When the conversion process runs, the TCB is switched to the Quasi Reentrant (QR) TCB, making the connection incorrect.

Completion Code

MQCC_FAILED

Programmer response

Ensure that a successful MQCONN or MQCONNX call is performed for the queue manager, and that an MQDISC call has not already been performed for it. Ensure that the handle is being used within its valid scope (see the description of MQCONN in MQCONN for more information about MQCONN).

- On z/OS, also check that the application has been linked with the correct stub; this is CSQCSTUB for CICS applications, CSQBSTUB for batch applications, and CSQQSTUB for IMS applications. Also, the stub used must not belong to a release of the queue manager that is more recent than the release on which the application will run.

Ensure the character conversion exit program run by your CICS TS 3.2 or higher application, which invokes the MQXCNVC call, is defined as OPENAPI. This definition prevents the 2018 MQRC_HCONN_ERROR error caused by from an incorrect connection, and allows the MQGET to complete.

2019 (07E3) (RC2019): MQRC_HOBJ_ERROR

Explanation

The object handle *Hobj* is not valid, for one of the following reasons:

- The parameter pointer is not valid, or (for the MQOPEN call) points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The value specified was not returned by a preceding MQOPEN call.
- The value specified has been made invalid by a preceding MQCLOSE call.
- The handle is a shared handle that has been made invalid by another thread issuing the MQCLOSE call.
- The handle is a nonshared handle that is being used by a thread that did not create the handle.
- The call is MQGET or MQPUT, but the object represented by the handle is not a queue.

Completion Code

MQCC_FAILED

Programmer response

Ensure that a successful MQOPEN call is performed for this object, and that an MQCLOSE call has not already been performed for it. Ensure that the handle is being used within its valid scope (see the description of MQOPEN in MQOPEN for more information).

2020 (07E4) (RC2020): MQRC_INHIBIT_VALUE_ERROR

Explanation

On an MQSET call, the value specified for either the MQIA_INHIBIT_GET attribute or the MQIA_INHIBIT_PUT attribute is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value for the *InhibitGet* or *InhibitPut* queu attribute.

2021 (07E5) (RC2021): MQRC_INT_ATTR_COUNT_ERROR

Explanation

On an MQINQ or MQSET call, the *IntAttrCount* parameter is negative (MQINQ or MQSET), or smaller than the number of integer attribute selectors (MQIA_*) specified in the *Selectors* parameter (MQSET only). This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Specify a value large enough for all selected integer attributes.

2022 (07E6) (RC2022): MQRC_INT_ATTR_COUNT_TOO_SMALL

Explanation

On an MQINQ call, the *IntAttrCount* parameter is smaller than the number of integer attribute selectors (MQIA_*) specified in the *Selectors* parameter.

The call completes with MQCC_WARNING, with the *IntAttrs* array filled in with as many integer attributes as there is room for.

Completion Code

MQCC_WARNING

Programmer response

Specify a large enough value, unless only a subset of the values is needed.

2023 (07E7) (RC2023): MQRC_INT_ATTRS_ARRAY_ERROR

Explanation

On an MQINQ or MQSET call, the *IntAttrs* parameter is not valid. The parameter pointer is not valid (MQINQ and MQSET), or points to read-only storage or to storage that is not as long as indicated by the *IntAttrCount* parameter (MQINQ only). (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2024 (07E8) (RC2024): MQRC_SYNCPOINT_LIMIT_REACHED

Explanation

An MQGET, MQPUT, or MQPUT1 call failed because it would have caused the number of uncommitted messages in the current unit of work to exceed the limit defined for the queue manager (see the

MaxUncommittedMsgs queue-manager attribute). The number of uncommitted messages is the sum of the following since the start of the current unit of work:

- Messages put by the application with the MQPMO_SYNCPOINT option
- Messages retrieved by the application with the MQGMO_SYNCPOINT option
- Trigger messages and COA report messages generated by the queue manager for messages put with the MQPMO_SYNCPOINT option
- COD report messages generated by the queue manager for messages retrieved with the MQGMO_SYNCPOINT option
- On HP Integrity NonStop Server , this reason code occurs when the maximum number of I/O operations in a single TM/MP transaction has been exceeded.

When publishing messages out of syncpoint to topics it is possible to receive this reason code; see Publications under syncpoint for more information.

Completion Code

MQCC_FAILED

Programmer response

Check whether the application is looping. If it is not, consider reducing the complexity of the application. Alternatively, increase the queue-manager limit for the maximum number of uncommitted messages within a unit of work.

- On z/OS, the limit for the maximum number of uncommitted messages can be changed by using the ALTER QMGR command.
- On IBM i , the limit for the maximum number of uncommitted messages can be changed by using the CHGMQM command.
- On HP Integrity NonStop Server , the application should cancel the transaction and retry with a smaller number of operations in the unit of work. See the *MQSeries for Tandem NonStop Kernel System Management Guide* for more details.

2025 (07E9) (RC2025): MQRC_MAX_CONNS_LIMIT_REACHED

Explanation

The MQCONN or MQCONNX call was rejected because the maximum number of concurrent connections has been exceeded.

- On z/OS, the connection limits are 32767 for both TSO and Batch.
- On IBM i, HP Integrity NonStop Server, UNIX systems, and Windows, this reason code can also occur on the MQOPEN call.
- When using Java applications, the connection manager might define a limit to the number of concurrent connections.

  Note: The application using IBM MQ might have delegated the management of connections to a framework or connection pool, for example, a Java EE application server, an application framework such as Spring, an IBM Container (for IBM Cloud® (formerly Bluemix®)), or a combination of these. For more information, see IBM MQ classes for JMS object pooling.

Completion Code

MQCC_FAILED

Programmer response

Either increase the size of the appropriate parameter value, or reduce the number of concurrent connections.

Related information

Connection pooling in IBM MQ classes for Java

2026 (07EA) (RC2026): MQRC_MD_ERROR

Explanation

The MQMD structure is not valid, for one of the following reasons:

- The *StrucId* field is not MQMD_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQMD structure are set correctly.

2027 (07EB) (RC2027): MQRC_MISSING_REPLY_TO_Q

Explanation

On an MQPUT or MQPUT1 call, the *ReplyToQ* field in the message descriptor MQMD is blank, but one or both of the following is true:

- A reply was requested (that is, MQMT_REQUEST was specified in the *MsgType* field of the message descriptor).
- A report message was requested in the *Report* field of the message descriptor.

Completion Code

MQCC_FAILED

Programmer response

Specify the name of the queue to which the reply message or report message is to be sent.

2029 (07ED) (RC2029): MQRC_MSG_TYPE_ERROR

Explanation

Either:

- On an MQPUT or MQPUT1 call, the value specified for the *MsgType* field in the message descriptor (MQMD) is not valid.

- A message processing program received a message that does not have the expected message type. For example, if the IBM MQ command server receives a message which is not a request message (MQMT_REQUEST) then it rejects the request with this reason code.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value for the *MsgType* field. In the case where a request is rejected by a message processing program, refer to the documentation for that program for details of the message types that it supports.

2030 (07EE) (RC2030): MQRC_MSG_TOO_BIG_FOR_Q

Explanation

An MQPUT or MQPUT1 call was issued to put a message on a queue, but the message was too long for the queue and MQMF_SEGMENTATION_ALLOWED was not specified in the *MsgFlags* field in MQMD. If segmentation is not allowed, the length of the message cannot exceed the lesser of the queue *MaxMsgLength* attribute and queue-manager *MaxMsgLength* attribute.

- On z/OS, the queue manager does not support the segmentation of messages; if MQMF_SEGMENTATION_ALLOWED is specified, it is accepted but ignored.

This reason code can also occur when MQMF_SEGMENTATION_ALLOWED *is* specified, but the nature of the data present in the message prevents the queue manager splitting it into segments that are small enough to place on the queue:

- For a user-defined format, the smallest segment that the queue manager can create is 16 bytes.
- For a built-in format, the smallest segment that the queue manager can create depends on the particular format, but is greater than 16 bytes in all cases other than MQFMT_STRING (for MQFMT_STRING the minimum segment size is 16 bytes).

MQRC_MSG_TOO_BIG_FOR_Q can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

Completion Code

MQCC_FAILED

Programmer response

Check whether the *BufferLength* parameter is specified correctly; if it is, do one of the following:

- Increase the value of the queue's *MaxMsgLength* attribute; the queue-manager's *MaxMsgLength* attribute may also need increasing.
- Break the message into several smaller messages.
- Specify MQMF_SEGMENTATION_ALLOWED in the *MsgFlags* field in MQMD; this will allow the queue manager to break the message into segments.

2031 (07EF) (RC2031): MQRC_MSG_TOO_BIG_FOR_Q_MGR

Explanation

An MQPUT or MQPUT1 call was issued to put a message on a queue, but the message was too long for the queue manager and MQMF_SEGMENTATION_ALLOWED was not specified in the *MsgFlags* field in MQMD. If segmentation is not allowed, the length of the message cannot exceed the lesser of the queue-manager *MaxMsgLength* attribute and queue *MaxMsgLength* attribute.

This reason code can also occur when MQMF_SEGMENTATION_ALLOWED *is* specified, but the nature of the data present in the message prevents the queue manager splitting it into segments that are small enough for the queue-manager limit:

- For a user-defined format, the smallest segment that the queue manager can create is 16 bytes.
- For a built-in format, the smallest segment that the queue manager can create depends on the particular format, but is greater than 16 bytes in all cases other than MQFMT_STRING (for MQFMT_STRING the minimum segment size is 16 bytes).

MQRC_MSG_TOO_BIG_FOR_Q_MGR can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

This reason also occurs if a channel, through which the message is to pass, has restricted the maximum message length to a value that is actually less than that supported by the queue manager, and the message length is greater than this value.

- On z/OS, this return code is issued only if you are using CICS for distributed queuing. Otherwise, MQRC_MSG_TOO_BIG_FOR_CHANNEL is issued.

Completion Code

MQCC_FAILED

Programmer response

Check whether the *BufferLength* parameter is specified correctly; if it is, do one of the following:

- Increase the value of the queue-manager's *MaxMsgLength* attribute; the queue's *MaxMsgLength* attribute may also need increasing.
- Break the message into several smaller messages.
- Specify MQMF_SEGMENTATION_ALLOWED in the *MsgFlags* field in MQMD; this will allow the queue manager to break the message into segments.
- Check the channel definitions.

2033 (07F1) (RC2033): MQRC_NO_MSG_AVAILABLE

Explanation

An MQGET call was issued, but there is no message on the queue satisfying the selection criteria specified in MQMD (the *MsgId* and *CorrelId* fields), and in MQGMO (the *Options* and *MatchOptions* fields). Either the MQGMO_WAIT option was not specified, or the time interval specified by the *WaitInterval* field in MQGMO has expired. This reason is also returned for an MQGET call for browse, when the end of the queue has been reached.

This reason code can also be returned by the mqGetBag and mqExecute calls. mqGetBag is similar to MQGET. For the mqExecute call, the completion code can be either MQCC_WARNING or MQCC_FAILED:

- If the completion code is MQCC_WARNING, some response messages were received during the specified wait interval, but not all. The response bag contains system-generated nested bags for the messages that were received.
- If the completion code is MQCC_FAILED, no response messages were received during the specified wait interval.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

If this is an expected condition, no corrective action is required.

If this is an unexpected condition, check that:

- The message was put on the queue successfully.
- The unit of work (if any) used for the MQPUT or MQPUT1 call was committed successfully.
- The options controlling the selection criteria are specified correctly. All of the following can affect the eligibility of a message for return on the MQGET call:
  - MQGMO_LOGICAL_ORDER
  - MQGMO_ALL_MSGS_AVAILABLE
  - MQGMO_ALL_SEGMENTS_AVAILABLE
  - MQGMO_COMPLETE_MSG
  - MQMO_MATCH_MSG_ID
  - MQMO_MATCH_CORREL_ID
  - MQMO_MATCH_GROUP_ID
  - MQMO_MATCH_MSG_SEQ_NUMBER
  - MQMO_MATCH_OFFSET
  - Value of *MsgId* field in MQMD
  - Value of *CorrelId* field in MQMD

Consider waiting longer for the message.

2034 (07F2) (RC2034): MQRC_NO_MSG_UNDER_CURSOR

Explanation

An MQGET call was issued with either the MQGMO_MSG_UNDER_CURSOR or the MQGMO_BROWSE_MSG_UNDER_CURSOR option. However, the browse cursor is not positioned at a retrievable message. This is caused by one of the following:

- The cursor is positioned logically before the first message (as it is before the first MQGET call with a browse option has been successfully performed).
- The message the browse cursor was positioned on has been locked or removed from the queue (probably by some other application) since the browse operation was performed.
- The message the browse cursor was positioned on has expired.

Completion Code

MQCC_FAILED

Programmer response

Check the application logic. This may be an expected reason if the application design allows multiple servers to compete for messages after browsing. Consider also using the MQGMO_LOCK option with the preceding browse MQGET call.

2035 (07F3) (RC2035): MQRC_NOT_AUTHORIZED

General explanation

Explanation

The user of the application or channel that produced the error, is not authorized to perform the operation attempted:

- On an MQCONN or MQCONNX call, the user is not authorized to connect to the queue manager.
  - On z/OS, for CICS applications, MQRC_CONNECTION_NOT_AUTHORIZED is issued instead.
- On an MQCONNX call, the length of the user ID or password is greater than the maximum length permitted. The maximum length of the user ID is dependent on the platform. For more information, see User IDs.
- On an MQOPEN or MQPUT1 call, the user is not authorized to open the object for the option(s) specified.
  - On z/OS, if the object being opened is a model queue, this reason also arises if the user is not authorized to create a dynamic queue with the required name.
- On an MQCLOSE call, the user is not authorized to delete the object, which is a permanent dynamic queue, and the *Hobj* parameter specified on the MQCLOSE call is not the handle returned by the MQOPEN call that created the queue.
- On a command, the user is not authorized to issue the command, or to access the object it specifies.
- On an MQSUB call, the user is not authorized to subscribe to the topic.
- On an MQSUB call, using non-managed destination queues, the user is not authorized to use the destination queue.

This reason code can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct queue manager or object was specified, and that appropriate authority exists.

This reason code is also used to identify the corresponding event message,

- MQCONN or MQCONNX Not Authorized (type 1).
- MQOPEN or MQPUT1 Not Authorized (type 2).
- MQCLOSE Not Authorized (type 3).
- Command Not Authorized (type 4).
- MQSUB Not Authorized (type 5).
- MQSUB destination Not Authorized (type 6).

Specific problems generating RC2035

JMSWMQ2013 invalid security authentication

See Invalid security authentication for information when your IBM MQ JMS application fails with security authentication errors.

MQRC_NOT_AUTHORIZED on a queue or channel

See MQRC_NOT_AUTHORIZED on a queue for information when MQRC 2035 (MQRC_NOT_AUTHORIZED) is returned where a user is not authorized to perform the function. Determine which object the user cannot access and provide the user access to the object.

MQRC_NOT_AUTHORIZED (AMQ4036 on a client) as an administrator

See MQRC_NOT_AUTHORIZED as an administrator for information when MQRC 2035 (MQRC_NOT_AUTHORIZED) is returned where you try to use a user ID that is an IBM MQ Administrator, to remotely access the queue manager through a client connection.

MQS_REPORT_NOAUTH

See MQS_REPORT_NOAUTH for information on using this environment variable to better diagnose return code 2035 (MQRC_NOT_AUTHORIZED). The use of this environment variable generates errors in the queue manager error log, but does not generate a Failure Data Capture (FDC).

MQSAUTHERRORS

See MQSAUTHERRORS for information on using this environment variable to generate FDC files related to return code 2035 (MQRC_NOT_AUTHORIZED). The use of this environment variable generates an FDC, but does not generate errors in the queue manager error log.

2036 (07F4) (RC2036): MQRC_NOT_OPEN_FOR_BROWSE

Explanation

An MQGET call was issued with one of the following options:

- MQGMO_BROWSE_FIRST
- MQGMO_BROWSE_NEXT
- MQGMO_BROWSE_MSG_UNDER_CURSOR
- MQGMO_MSG_UNDER_CURSOR

but either the queue had not been opened for browse, or you are using IBM MQ Multicast messaging.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_BROWSE when the queue is opened.

If you are using IBM MQ Multicast messaging, you cannot specify browse options with an MQGET call.

2037 (07F5) (RC2037): MQRC_NOT_OPEN_FOR_INPUT

Explanation

An MQGET call was issued to retrieve a message from a queue, but the queue had not been opened for input.

Completion Code

MQCC_FAILED

Programmer response

Specify one of the following when the queue is opened:

- MQOO_INPUT_SHARED
- MQOO_INPUT_EXCLUSIVE

- MQOO_INPUT_AS_Q_DEF

2038 (07F6) (RC2038): MQRC_NOT_OPEN_FOR_INQUIRE

Explanation

An MQINQ call was issued to inquire object attributes, but the object had not been opened for inquire.

An MQINQ call was issued for a topic handle in IBM MQ Multicast.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_INQUIRE when the object is opened.

MQINQ is not supported for topic handles in IBM MQ Multicast.

2039 (07F7) (RC2039): MQRC_NOT_OPEN_FOR_OUTPUT

Explanation

An MQPUT call was issued to put a message on a queue, but the queue had not been opened for output.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_OUTPUT when the queue is opened.

2040 (07F8) (RC2040): MQRC_NOT_OPEN_FOR_SET

Explanation

An MQSET call was issued to set queue attributes, but the queue had not been opened for set.

An MQSET call was issued for a topic handle in IBM MQ Multicast.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_SET when the object is opened.

MQSET is not supported for topic handles in IBM MQ Multicast.

2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED

Explanation

Object definitions that affect this object have been changed since the `Hobj` handle used on this call was returned by the MQOPEN call. For more information about the MQOPEN call, see MQOPEN.

This reason does not occur if the object handle is specified in the *Context* field of the *PutMsgOpts* parameter on the MQPUT or MQPUT1 call.

Completion Code

MQCC_FAILED

Programmer response

Issue an MQCLOSE call to return the handle to the system. It is then usually sufficient to reopen the object and retry the operation. However, if the object definitions are critical to the application logic, an MQINQ call can be used after reopening the object, to obtain the new values of the object attributes.

2042 (07FA) (RC2042): MQRC_OBJECT_IN_USE

Explanation

An MQOPEN call was issued, but the object in question has already been opened by this or another application with options that conflict with those specified in the **Options** parameter. This arises if the request is for shared input, but the object is already open for exclusive input; it also arises if the request is for exclusive input, but the object is already open for input (of any sort).

MCAs for receiver channels, or the intra-group queuing agent (IGQ agent), may keep the destination queues open even when messages are not being transmitted; this results in the queues appearing to be *in use*. Use the MQSC command DISPLAY QSTATUS to find out who is keeping the queue open.

IBM MQ opens a queue for shared input if the application uses the MQOO_INPUT_SHARED open option or, if the application uses MQOO_INPUT_AS_Q_DEF and the default sharing option queue attribute is set to DEFSOPT(SHARED). However, there is an administrative override in the form of the SHARE/NOSHARE option.

If the queue definition shows NOSHARE, then IBM MQ will make the input handle exclusive regardless of the options set by the application.

- z/OS On z/OS, this reason can also occur for an MQOPEN or MQPUT1 call, if the object to be opened (which can be a queue, or for MQOPEN a namelist or process object) is in the process of being deleted.
- z/OS The default setting on z/OS is NOSHARE.

Completion Code

MQCC_FAILED

Programmer response

System design should specify whether an application is to wait and retry, or take other action.

2043 (07FB) (RC2043): MQRC_OBJECT_TYPE_ERROR

Explanation

On the MQOPEN or MQPUT1 call, the *ObjectType* field in the object descriptor MQOD specifies a value that is not valid. For the MQPUT1 call, the object type must be MQOT_Q.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid object type.

2044 (07FC) (RC2044): MQRC_OD_ERROR

Explanation

On the MQOPEN or MQPUT1 call, the object descriptor MQOD is not valid, for one of the following reasons:

- The *StrucId* field is not MQOD_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQOD structure are set correctly.

2045 (07FD) (RC2045): MQRC_OPTION_NOT_VALID_FOR_TYPE

Explanation

On an MQOPEN or MQCLOSE call, an option is specified that is not valid for the type of object or queue being opened or closed.

For the MQOPEN call, this includes the following cases:

- An option that is inappropriate for the object type (for example, MQOO_OUTPUT for an MQOT_PROCESS object).
- An option that is unsupported for the queue type (for example, MQOO_INQUIRE for a remote queue that has no local definition).
- One or more of the following options:
  - MQOO_INPUT_AS_Q_DEF
  - MQOO_INPUT_SHARED
  - MQOO_INPUT_EXCLUSIVE
  - MQOO_BROWSE
  - MQOO_INQUIRE
  - MQOO_SET

  when either:

  - the queue name is resolved through a cell directory, or
  - *ObjectQMgrName* in the object descriptor specifies the name of a local definition of a remote queue (to specify a queue-manager alias), and the queue named in the *RemoteQMgrName* attribute of the definition is the name of the local queue manager.

For the MQCLOSE call, this includes the following case:

- The MQCO_DELETE or MQCO_DELETE_PURGE option when the queue is not a dynamic queue.

This reason code can also occur on the MQOPEN call when the object being opened is of type MQOT_NAMELIST, MQOT_PROCESS, or MQOT_Q_MGR, but the *ObjectQMgrName* field in MQOD is neither blank nor the name of the local queue manager.

Completion Code

MQCC_FAILED

Programmer response

Specify the correct option. For the MQOPEN call, ensure that the *ObjectQMgrName* field is set correctly. For the MQCLOSE call, either correct the option or change the definition type of the model queue that is used to create the new queue.

2046 (07FE) (RC2046): MQRC_OPTIONS_ERROR

Explanation

The *Options* parameter or field contains options that are not valid, or a combination of options that is not valid.

- For the MQOPEN, MQCLOSE, MQXCNVC, mqBagToBuffer, mqBufferToBag, mqCreateBag, and mqExecute calls, *Options* is a separate parameter on the call.

  This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- For the MQBEGIN, MQCONNX, MQGET, MQPUT, and MQPUT1 calls, *Options* is a field in the relevant options structure (MQBO, MQCNO, MQGMO, or MQPMO).
- For more information about option errors for IBM MQ Multicast see: MQI concepts and how they relate to multicast.

Completion Code

MQCC_FAILED

Programmer response

Specify valid options. Check the description of the *Options* parameter or field to determine which options and combinations of options are valid. If multiple options are being set by adding the individual options together, ensure that the same option is not added twice. For more information, see Rules for validating MQI options.

2047 (07FF) (RC2047): MQRC_PERSISTENCE_ERROR

Explanation

On an MQPUT or MQPUT1 call, the value specified for the *Persistence* field in the message descriptor MQMD is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify one of the following values:

- MQPER_PERSISTENT

- MQPER_NOT_PERSISTENT
- MQPER_PERSISTENCE_AS_Q_DEF

2048 (0800) (RC2048): MQRC_PERSISTENT_NOT_ALLOWED

Explanation

On an MQPUT or MQPUT1 call, the value specified for the *Persistence* field in MQMD (or obtained from the *DefPersistence* queue attribute) specifies MQPER_PERSISTENT, but the queue on which the message is being placed does not support persistent messages. Persistent messages cannot be placed on temporary dynamic queues.

This reason code can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

Completion Code

MQCC_FAILED

Programmer response

Specify MQPER_NOT_PERSISTENT if the message is to be placed on a temporary dynamic queue. If persistence is required, use a permanent dynamic queue or predefined queue in place of a temporary dynamic queue.

Be aware that server applications are recommended to send reply messages (message type MQMT_REPLY) with the same persistence as the original request message (message type MQMT_REQUEST). If the request message is persistent, the reply queue specified in the *ReplyToQ* field in the message descriptor MQMD cannot be a temporary dynamic queue. Use a permanent dynamic queue or predefined queue as the reply queue in this situation.

On z/OS, you cannot put persistent messages to a shared queue if the CFSTRUCT that the queue uses is defined with RECOVER(NO). Either put only non-persistent messages to this queue or change the CFSTRUCT definition to RECOVER(YES). If you put a persistent message to a queue that uses a CFSTRUCT with RECOVER(NO) the put will fail with MQRC_PERSISTENT_NOT_ALLOWED.

2049 (0801) (RC2049): MQRC_PRIORITY_EXCEEDS_MAXIMUM

Explanation

An MQPUT or MQPUT1 call was issued, but the value of the *Priority* field in the message descriptor MQMD exceeds the maximum priority supported by the local queue manager, as shown by the *MaxPriority* queue-manager attribute. The message is accepted by the queue manager, but is placed on the queue at the queue manager's maximum priority. The *Priority* field in the message descriptor retains the value specified by the application that put the message.

Completion Code

MQCC_WARNING

Programmer response

None required, unless this reason code was not expected by the application that put the message.

2050 (0802) (RC2050): MQRC_PRIORITY_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the value of the *Priority* field in the message descriptor MQMD is not valid. The maximum priority supported by the queue manager is given by the *MaxPriority* queue-manager attribute.

Completion Code

MQCC_FAILED

Programmer response

Specify a value in the range zero through *MaxPriority*, or the special value MQPRI_PRIORITY_AS_Q_DEF.

2051 (0803) (RC2051): MQRC_PUT_INHIBITED

Explanation

MQPUT and MQPUT1 calls are currently inhibited for the queue, or for the queue to which this queue resolves.

This reason code can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

Completion Code

MQCC_FAILED

Programmer response

If the system design allows put requests to be inhibited for short periods, retry the operation later.

This reason code is also used to identify the corresponding event message Put Inhibited.

System programmer action

Use ALTER QLOCAL(...) PUT(ENABLED) to allow messages to be put.

2052 (0804) (RC2052): MQRC_Q_DELETED

Explanation

An *Hobj* queue handle specified on a call refers to a dynamic queue that has been deleted since the queue was opened. For more information about the deletion of dynamic queues, see the description of MQCLOSE in MQCLOSE.

- On z/OS, this can also occur with the MQOPEN and MQPUT1 calls if a dynamic queue is being opened, but the queue is in a logically-deleted state. See MQCLOSE for more information about this.

Completion Code

MQCC_FAILED

Programmer response

Issue an MQCLOSE call to return the handle and associated resources to the system (the MQCLOSE call will succeed in this case). Check the design of the application that caused the error.

2053 (0805) (RC2053): MQRC_Q_FULL

Explanation

An MQPUT or MQPUT1 call, or a command, failed because the queue is full, that is, it already contains the maximum number of messages possible, as specified by the *MaxQDepth* queue attribute.

This reason code can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

Completion Code

MQCC_FAILED

Programmer response

Retry the operation later. Consider increasing the maximum depth for this queue, or arranging for more instances of the application to service the queue.

This reason code is also used to identify the corresponding event message Queue Full.

2055 (0807) (RC2055): MQRC_Q_NOT_EMPTY

Explanation

An MQCLOSE call was issued for a permanent dynamic queue, but the call failed because the queue is not empty or still in use. One of the following applies:

- The MQCO_DELETE option was specified, but there are messages on the queue.
- The MQCO_DELETE or MQCO_DELETE_PURGE option was specified, but there are uncommitted get or put calls outstanding against the queue.

See the usage notes pertaining to dynamic queues for the MQCLOSE call for more information.

This reason code is also returned from a command to clear or delete or move a queue, if the queue contains uncommitted messages (or committed messages in the case of delete queue without the purge option).

Completion Code

MQCC_FAILED

Programmer response

Check why there might be messages on the queue. Be aware that the *CurrentQDepth* queue attribute might be zero even though there are one or more messages on the queue; this can happen if the messages have been retrieved as part of a unit of work that has not yet been committed. If the messages can be discarded, try using the MQCLOSE call with the MQCO_DELETE_PURGE option. Consider retrying the call later.

2056 (0808) (RC2056): MQRC_Q_SPACE_NOT_AVAILABLE

Explanation

An MQPUT or MQPUT1 call was issued, but there is no space available for the queue on disk or other storage device.

This reason code can also occur in the *Feedback* field in the message descriptor of a report message; in this case it indicates that the error was encountered by a message channel agent when it attempted to put the message on a remote queue.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Check whether an application is putting messages in an infinite loop. If not, make more disk space available for the queue.

2057 (0809) (RC2057): MQRC_Q_TYPE_ERROR

Explanation

One of the following occurred:

- On an MQOPEN call, the *ObjectQMgrName* field in the object descriptor MQOD or object record MQOR specifies the name of a local definition of a remote queue (to specify a queue-manager alias), and in that local definition the *RemoteQMgrName* attribute is the name of the local queue manager. However, the *ObjectName* field in MQOD or MQOR specifies the name of a model queue on the local queue manager; this is not allowed. For more information, see MQOPEN.
- On an MQPUT1 call, the object descriptor MQOD or object record MQOR specifies the name of a model queue.
- On a previous MQPUT or MQPUT1 call, the *ReplyToQ* field in the message descriptor specified the name of a model queue, but a model queue cannot be specified as the destination for reply or report messages. Only the name of a predefined queue, or the name of the *dynamic* queue created from the model queue, can be specified as the destination. In this situation the reason code MQRC_Q_TYPE_ERROR is returned in the *Reason* field of the MQDLH structure when the reply message or report message is placed on the dead-letter queue.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid queue.

This reason code is also used to identify the corresponding event message Queue Type Error.

2058 (080A) (RC2058): MQRC_Q_MGR_NAME_ERROR

Explanation

On an MQCONN or MQCONNX call, the value specified for the *QMgrName* parameter is not valid or not known. This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

- z/OS On z/OS for CICS applications, this reason can occur on *any* call if the original connect specified an incorrect or unrecognized name.

z/OS For CICS, this reason can be caused by the wrong resync value. For example, Groupresync is specified and the queue manager is not in a queue sharing group.

This reason code can also occur if an MQ MQI client application attempts to connect to a queue manager within an MQ-client queue-manager group (see the *QMgrName* parameter of MQCONN), and either:

- Queue-manager groups are not supported.
- There is no queue-manager group with the specified name.

z/OS For IMS adapter on z/OS, this can be caused by the library containing your CSQQDEFV module being after the SCSQAUTH data set in steplib. Move your library above the SCSQAUTH data set.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Use an all-blank name if possible, or verify that the name used is valid.

If you are using CICS Resyncmember(Groupresync), use the queue-sharing group (QSG) name in the MQNAME rather than the queue manager name.

2059 (080B) (RC2059): MQRC_Q_MGR_NOT_AVAILABLE

Explanation

This error occurs:

1. On an MQCONN or MQCONNX call, the queue manager identified by the *QMgrName* parameter is not available for connection.
   - On z/OS:
     - For batch applications, this reason can be returned to applications running in LPARs that do not have a queue manager installed.
     - For CICS applications, this reason can occur on any call if the original connect specified a queue manager with a name that was recognized, but which is not available.
   - On IBM i, this reason can also be returned by the MQOPEN and MQPUT1 calls, when MQHC_DEF_HCONN is specified for the *Hconn* parameter by an application running in compatibility mode.
2. On an MQCONN or MQCONNX call from an IBM MQ MQI client application:
   - Attempting to connect to a queue manager within an MQ-client queue-manager group when none of the queue managers in the group is available for connection (see the *QMgrName* parameter of the MQCONN call).
   - If the client channel fails to connect, perhaps because of an error with the client-connection or the corresponding server-connection channel definitions.
3. If a command uses the *CommandScope* parameter specifying a queue manager that is not active in the queue-sharing group.
4. In a multiple installation environment, where an application attempts to connect to a queue manager associated with an installation of IBM WebSphere MQ 7.1, or later, but has loaded libraries from IBM WebSphere MQ 7.0.1. IBM WebSphere MQ 7.0.1 cannot load libraries from other versions of IBM MQ.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the queue manager has been started. If the connection is from a client application, check the channel definitions, channel status, and error logs.

In a multiple installation environment, ensure that IBM WebSphere MQ 7.1, or later, libraries are loaded by the operating system. For more information, see Connecting applications in a multiple installation environment.

2061 (080D) (RC2061): MQRC_REPORT_OPTIONS_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the *Report* field in the message descriptor MQMD contains one or more options that are not recognized by the local queue manager. The options that cause this reason code to be returned depend on the destination of the message; see the description of REPORT in Report options and message flags for more details.

This reason code can also occur in the *Feedback* field in the MQMD of a report message, or in the *Reason* field in the MQDLH structure of a message on the dead-letter queue; in both cases it indicates that the destination queue manager does not support one or more of the report options specified by the sender of the message.

Completion Code

MQCC_FAILED

Programmer response

Do the following:

- Ensure that the *Report* field in the message descriptor is initialized with a value when the message descriptor is declared, or is assigned a value prior to the MQPUT or MQPUT1 call. Specify MQRO_NONE if no report options are required.
- Ensure that the report options specified are valid; see the *Report* field described in the description of MQMD in Report options and message flags for valid report options.
- If multiple report options are being set by adding the individual report options together, ensure that the same report option is not added twice.
- Check that conflicting report options are not specified. For example, do not add both MQRO_EXCEPTION and MQRO_EXCEPTION_WITH_DATA to the *Report* field; only one of these can be specified.

2062 (080E) (RC2062): MQRC_SECOND_MARK_NOT_ALLOWED

Explanation

An MQGET call was issued specifying the MQGMO_MARK_SKIP_BACKOUT option in the *Options* field of MQGMO, but a message has already been marked within the current unit of work. Only one marked message is allowed within each unit of work.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Modify the application so that no more than one message is marked within each unit of work.

2063 (080F) (RC2063): MQRC_SECURITY_ERROR

Explanation

An MQCONN, MQCONNX, MQOPEN, MQSUB, MQPUT1, or MQCLOSE call was issued, but it failed because a security error occurred.

- On z/OS, there are two possible reasons for this:
  - An MQCONN or MQCONNX call was issued to connect to the queue manager using the BINDINGS transport, passing in a username or password, or both, that were longer than 8 characters.
  - The security error was returned by the External Security Manager.
- If you are using AMS, this could be a set up issue.
- If you are using connection authentication with an LDAP server, this could be as a result of connectivity failure to the LDAP server, or an error from the LDAP server.

Completion Code

MQCC_FAILED

Programmer response

Note the error from the security manager, and contact your system programmer or security administrator.

- If you are using AMS, you should check the queue manager error logs.
- On z/OS, ensure that both the username and password specified, when connecting to the queue manager, have a maximum length of 8 characters.
- On IBM i, the FFST log will contain the error information.
- If you are using LDAP, use DISPLAY QMSTATUS to check the status of the connection to the LDAP server, and check the queue manager error logs for any error messages.

2065 (0811) (RC2065): MQRC_SELECTOR_COUNT_ERROR

Explanation

On an MQINQ or MQSET call, the *SelectorCount* parameter specifies a value that is not valid. This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Specify a value in the range 0 through 256.

2066 (0812) (RC2066): MQRC_SELECTOR_LIMIT_EXCEEDED

Explanation

On an MQINQ or MQSET call, the *SelectorCount* parameter specifies a value that is larger than the maximum supported (256).

Completion Code

MQCC_FAILED

Programmer response

Reduce the number of selectors specified on the call; the valid range is 0 through 256.

2067 (0813) (RC2067): MQRC_SELECTOR_ERROR

Explanation

An MQINQ or MQSET call was issued, but the `Selectors` array contains a selector that is not valid for one of the following reasons:

- The selector is not supported or out of range.
- The selector is not applicable to the type of object with attributes that are being inquired upon or set.
- The selector is for an attribute that cannot be set.

This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

An MQINQ call was issued for a managed handle in IBM MQ Multicast, inquiring a value other than `Current Depth`.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the value specified for the selector is valid for the object type represented by `Hobj`. For the MQSET call, also ensure that the selector represents an integer attribute that can be set.

MQINQ for managed handles in IBM MQ Multicast can only inquire on `Current Depth`.

2068 (0814) (RC2068): MQRC_SELECTOR_NOT_FOR_TYPE

Explanation

On the MQINQ call, one or more selectors in the `Selectors` array is not applicable to the type of the queue with attributes that are being inquired upon.

This reason also occurs when the queue is a cluster queue that resolved to a remote instance of the queue. In this case only a subset of the attributes that are valid for local queues can be inquired. See the usage notes in the description of MQINQ in MQINQ - Inquire object attributes for more information about MQINQ.

The call completes with MQCC_WARNING, with the attribute values for the inapplicable selectors set as follows:

- For integer attributes, the corresponding elements of `IntAttrs` are set to MQIAV_NOT_APPLICABLE.
- For character attributes, the appropriate parts of the `CharAttrs` string are set to a character string consisting entirely of asterisks (*).

Completion Code

MQCC_WARNING

Programmer response

Verify that the selector specified is the one that was intended.

If the queue is a cluster queue, specifying one of the MQOO_BROWSE, MQOO_INPUT_*, or MQOO_SET options in addition to MQOO_INQUIRE forces the queue to resolve to the local instance of the queue. However, if there is no local instance of the queue the MQOPEN call fails.

2069 (0815) (RC2069): MQRC_SIGNAL_OUTSTANDING

Explanation

An MQGET call was issued with either the MQGMO_SET_SIGNAL or MQGMO_WAIT option, but there is already a signal outstanding for the queue handle *Hobj*.

This reason code occurs only in the following environments: z/OS, Windows 95, Windows 98.

Completion Code

MQCC_FAILED

Programmer response

Check the application logic. If it is necessary to set a signal or wait when there is a signal outstanding for the same queue, a different object handle must be used.

2070 (0816) (RC2070): MQRC_SIGNAL_REQUEST_ACCEPTED

Explanation

An MQGET call was issued specifying MQGMO_SET_SIGNAL in the *GetMsgOpts* parameter, but no suitable message was available; the call returns immediately. The application can now wait for the signal to be delivered.

- On z/OS, the application should wait on the Event Control Block pointed to by the *Signal1* field.
- On Windows 95, Windows 98, the application should wait for the signal Windows message to be delivered.

This reason code occurs only in the following environments: z/OS, Windows 95, Windows 98.

Completion Code

MQCC_WARNING

Programmer response

Wait for the signal; when it is delivered, check the signal to ensure that a message is now available. If it is, reissue the MQGET call.

- On z/OS, wait on the ECB pointed to by the *Signal1* field and, when it is posted, check it to ensure that a message is now available.
- On Windows 95, Windows 98, the application (thread) should continue executing its message loop.

2071 (0817) (RC2071): MQRC_STORAGE_NOT_AVAILABLE

Explanation

The call failed because there is insufficient main storage available.

Completion Code

MQCC_FAILED

Programmer response

Ensure that active applications are behaving correctly, for example, that they are not looping unexpectedly. If no problems are found, make more main storage available.

- On z/OS, if no application problems are found, ask your system programmer to increase the size of the region in which the queue manager runs.

2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE

Explanation

Either the MQGMO_SYNCPOINT option was used with an MQGET call, or the MQPMO_SYNCPOINT option was used with an MQPUT or MQPUT1 call, but the local queue manager was unable to honor the request. If the queue manager does not support units of work, the *SyncPoint* queue-manager attribute has the value MQSP_NOT_AVAILABLE.

This reason code can also occur on the MQGET, MQPUT, and MQPUT1 calls when an external unit-of-work coordinator is used. If that coordinator requires an explicit call to start the unit of work, but the application has not issued that call before the MQGET, MQPUT, or MQPUT1 call, reason code MQRC_SYNCPOINT_NOT_AVAILABLE is returned.

- IBM i On IBM i, this reason code means that IBM i Commitment Control is not started, or is unavailable for use by the queue manager.
- On HP Integrity NonStop Server, this reason code means that the client has detected that the application has an active transaction that is being coordinated by the Transaction Management Facility (TMF), but that a z/OS queue manager is unable to be coordinated by TMF.

This reason code can also be returned if the MQGMO_SYNCPOINT or the MQPMO_SYNCPOINT option was used for IBM MQ Multicast messaging. Transactions are not supported for multicast.

Completion Code

MQCC_FAILED

Programmer response

Remove the specification of MQGMO_SYNCPOINT or MQPMO_SYNCPOINT, as appropriate.

- IBM i On IBM i, ensure that Commitment Control is started. If this reason code occurs after Commitment Control is started, contact your system programmer.
- On HP Integrity NonStop Server, ensure that your z/OS queue manager has the relevant APAR applied. Check with the IBM support center for APAR details.

2075 (081B) (RC2075): MQRC_TRIGGER_CONTROL_ERROR

Explanation

On an MQSET call, the value specified for the MQIA_TRIGGER_CONTROL attribute selector is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value.

2076 (081C) (RC2076): MQRC_TRIGGER_DEPTH_ERROR

Explanation

On an MQSET call, the value specified for the MQIA_TRIGGER_DEPTH attribute selector is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a value that is greater than zero.

2077 (081D) (RC2077): MQRC_TRIGGER_MSG_PRIORITY_ERR

Explanation

On an MQSET call, the value specified for the MQIA_TRIGGER_MSG_PRIORITY attribute selector is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a value in the range zero through the value of *MaxPriority* queue-manager attribute.

2078 (081E) (RC2078): MQRC_TRIGGER_TYPE_ERROR

Explanation

On an MQSET call, the value specified for the MQIA_TRIGGER_TYPE attribute selector is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value.

2079 (081F) (RC2079): MQRC_TRUNCATED_MSG_ACCEPTED

Explanation

On an MQGET call, the message length was too large to fit into the supplied buffer. The MQGMO_ACCEPT_TRUNCATED_MSG option was specified, so the call completes. The message is removed from the queue (subject to unit-of-work considerations), or, if this was a browse operation, the browse cursor is advanced to this message.

The *DataLength* parameter is set to the length of the message before truncation, the *Buffer* parameter contains as much of the message as fits, and the MQMD structure is filled in.

Completion Code

MQCC_WARNING

Programmer response

None, because the application expected this situation.

2080 (0820) (RC2080): MQRC_TRUNCATED_MSG_FAILED

Explanation

On an MQGET call, the message length was too large to fit into the supplied buffer. The MQGMO_ACCEPT_TRUNCATED_MSG option was *not* specified, so the message has not been removed from the queue. If this was a browse operation, the browse cursor remains where it was before this call, but if MQGMO_BROWSE_FIRST was specified, the browse cursor is positioned logically before the highest-priority message on the queue.

The *`DataLength`* field is set to the length of the message before truncation, the *`Buffer`* parameter contains as much of the message as fits, and the MQMD structure is filled in.

Completion Code

MQCC_WARNING

Programmer response

Supply a buffer that is at least as large as *`DataLength`*, or specify MQGMO_ACCEPT_TRUNCATED_MSG if not all of the message data is required.

2082 (0822) (RC2082): MQRC_UNKNOWN_ALIAS_BASE_Q

Explanation

An MQOPEN or MQPUT1 call was issued specifying an alias queue as the target, but the *`BaseQName`* in the alias queue attributes is not recognized as a queue name.

This reason code can also occur when *`BaseQName`* is the name of a cluster queue that cannot be resolved successfully.

MQRC_UNKNOWN_ALIAS_BASE_Q might indicate that the application is specifying the **`ObjectQmgrName`** of the queue manager that it is connecting to, and the queue manager that is hosting the alias queue. This means that the queue manager looks for the alias target queue on the specified queue manager and fails because the alias target queue is not on the local queue manager. Leave the **`ObjectQmgrName`** parameter blank so that the clustering decides which queue manager to route to.

Completion Code

MQCC_FAILED

Programmer response

Correct the queue definitions.

This reason code is also used to identify the corresponding event message Unknown Alias Base Queue.

If the reason code is seen by an application that is using IBM MQ classes for JMS, modify the JMS queue object definition that is used by the application so that the **`QMANAGER`** property is set to the empty string (""). This setting ensures that the clustering decides which queue manager to route to.

2085 (0825) (RC2085): MQRC_UNKNOWN_OBJECT_NAME

Explanation

An MQOPEN, MQPUT1 , or MQSUB call was issued, but the object identified by the *ObjectName* and *ObjectQMgrName* fields in the object descriptor MQOD cannot be found. One of the following applies:

- The *ObjectQMgrName* field is one of the following:
  - Blank
  - The name of the local queue manager
  - The name of a local definition of a remote queue (a queue-manager alias) in which the *RemoteQMgrName* attribute is the name of the local queue manager

  but no object with the specified *ObjectName* and *ObjectType* exists on the local queue manager.
- The object being opened is a cluster queue that is hosted on a remote queue manager, but the local queue manager does not have a defined route to the remote queue manager.
- The object being opened is a queue definition that has QSGDISP(GROUP). Such definitions cannot be used with the MQOPEN, MQPUT1 , or MQSUB calls.
- The MQOD in the failing application specifies the name of the local queue manager in *ObjectQMgrName*. The local queue manager does not host the particular cluster queue specified in *ObjectName*.

  The solution in this environment is to leave *ObjectQMgrName* of the MQOD blank.

This can also occur in response to a command that specifies the name of an object or other item that does not exist.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid object name. Ensure that the name is padded with blanks at the end, if necessary. If this is correct, check the object definitions.

This reason code is also used to identify the corresponding event message Unknown Object Name.

2086 (0826) (RC2086): MQRC_UNKNOWN_OBJECT_Q_MGR

Explanation

On an MQOPEN or MQPUT1 call, the *ObjectQMgrName* field in the object descriptor MQOD does not satisfy the naming rules for objects. For more information, see ObjectQMgrName (MQCHAR48).

This reason also occurs if the *ObjectType* field in the object descriptor has the value MQOT_Q_MGR, and the *ObjectQMgrName* field is not blank, but the name specified is not the name of the local queue manager.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid queue manager name. To refer to the local queue manager, a name consisting entirely of blanks or beginning with a null character can be used. Ensure that the name is padded with blanks at the end, or terminated with a null character if necessary.

2087 (0827) (RC2087): MQRC_UNKNOWN_REMOTE_Q_MGR

Explanation

On an MQOPEN or MQPUT1 call, an error occurred with the queue-name resolution, for one of the following reasons:

- *ObjectQMgrName* is blank or the name of the local queue manager, *ObjectName* is the name of a local definition of a remote queue (or an alias to one), and one of the following is true:
  - *RemoteQMgrName* is blank or the name of the local queue manager. Note that this error occurs even if *XmitQName* is not blank.
  - *XmitQName* is blank, but there is no transmission queue defined with the name of *RemoteQMgrName*, and the *DefXmitQName* queue-manager attribute is blank.
  - *RemoteQMgrName* and *RemoteQName* specify a cluster queue that cannot be resolved successfully, and the *DefXmitQName* queue-manager attribute is blank.
  - On z/OS only, the *RemoteQMgrName* is the name of a queue manager in the Queue Sharing group but intra-group queuing is disabled.
- *ObjectQMgrName* is the name of a local definition of a remote queue (containing a queue-manager alias definition), and one of the following is true:
  - *RemoteQName* is not blank.
  - *XmitQName* is blank, but there is no transmission queue defined with the name of *RemoteQMgrName*, and the *DefXmitQName* queue-manager attribute is blank.
- *ObjectQMgrName* is not:
  - Blank
  - The name of the local queue manager
  - The name of a transmission queue
  - The name of a queue-manager alias definition (that is, a local definition of a remote queue with a blank *RemoteQName*)

  but the *DefXmitQName* queue-manager attribute is blank and the queue manager is not part of a queue-sharing group with intra-group queuing enabled.
- *ObjectQMgrName* is the name of a model queue.
- The queue name is resolved through a cell directory. However, there is no queue defined with the same name as the remote queue manager name obtained from the cell directory, and the *DefXmitQName* queue-manager attribute is blank.

Completion Code

MQCC_FAILED

Programmer response

Check the values specified for *ObjectQMgrName* and *ObjectName*. If these are correct, check the queue definitions.

This reason code is also used to identify the corresponding event message Unknown Remote Queue Manager.

2090 (082A) (RC2090): MQRC_WAIT_INTERVAL_ERROR

Explanation

On the MQGET call, the value specified for the *WaitInterval* field in the *GetMsgOpts* parameter is not valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a value greater than or equal to zero, or the special value MQWI_UNLIMITED if an indefinite wait is required.

2091 (082B) (RC2091): MQRC_XMIT_Q_TYPE_ERROR

Explanation

On an MQOPEN or MQPUT1 call, a message is to be sent to a remote queue manager. The *ObjectName* or *ObjectQMgrName* field in the object descriptor specifies the name of a local definition of a remote queue but one of the following applies to the *XmitQName* attribute of the definition:

- *XmitQName* is not blank, but specifies a queue that is not a local queue
- *XmitQName* is blank, but *RemoteQMgrName* specifies a queue that is not a local queue

This reason also occurs if the queue name is resolved through a cell directory, and the remote queue manager name obtained from the cell directory is the name of a queue, but this is not a local queue.

Completion Code

MQCC_FAILED

Programmer response

Check the values specified for *ObjectName* and *ObjectQMgrName*. If these are correct, check the queue definitions.

This reason code is also used to identify the corresponding event message Transmission Queue Type Error.

2092 (082C) (RC2092): MQRC_XMIT_Q_USAGE_ERROR

Explanation

On an MQOPEN or MQPUT1 call, a message is to be sent to a remote queue manager, but one of the following occurred:

- *ObjectQMgrName* specifies the name of a local queue, but it does not have a *Usage* attribute of MQUS_TRANSMISSION.
- The *ObjectName* or *ObjectQMgrName* field in the object descriptor specifies the name of a local definition of a remote queue but one of the following applies to the *XmitQName* attribute of the definition:
  - *XmitQName* is not blank, but specifies a queue that does not have a *Usage* attribute of MQUS_TRANSMISSION
  - *XmitQName* is blank, but *RemoteQMgrName* specifies a queue that does not have a *Usage* attribute of MQUS_TRANSMISSION
  - *XmitQName* specifies the queue SYSTEM.QSG.TRANSMIT.QUEUE the IGQ queue manager attribute indicates that IGQ is DISABLED.
- The queue name is resolved through a cell directory, and the remote queue manager name obtained from the cell directory is the name of a local queue, but it does not have a *Usage* attribute of MQUS_TRANSMISSION.

Completion Code

MQCC_FAILED

Programmer response

Check the values specified for *ObjectName* and *ObjectQMgrName*. If these are correct, check the queue definitions.

This reason code is also used to identify the corresponding event message Transmission Queue Usage Error.

2093 (082D) (RC2093): MQRC_NOT_OPEN_FOR_PASS_ALL

Explanation

An MQPUT call was issued with the MQPMO_PASS_ALL_CONTEXT option specified in the *PutMsgOpts* parameter, but the queue had not been opened with the MQOO_PASS_ALL_CONTEXT option.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_PASS_ALL_CONTEXT (or another option that implies it) when the queue is opened.

2094 (082E) (RC2094): MQRC_NOT_OPEN_FOR_PASS_IDENT

Explanation

An MQPUT call was issued with the MQPMO_PASS_IDENTITY_CONTEXT option specified in the *PutMsgOpts* parameter, but the queue had not been opened with the MQOO_PASS_IDENTITY_CONTEXT option.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_PASS_IDENTITY_CONTEXT (or another option that implies it) when the queue is opened.

2095 (082F) (RC2095): MQRC_NOT_OPEN_FOR_SET_ALL

Explanation

An MQPUT call was issued with the MQPMO_SET_ALL_CONTEXT option specified in the *PutMsgOpts* parameter, but the queue had not been opened with the MQOO_SET_ALL_CONTEXT option.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_SET_ALL_CONTEXT when the queue is opened.

2096 (0830) (RC2096): MQRC_NOT_OPEN_FOR_SET_IDENT

Explanation

An MQPUT call was issued with the MQPMO_SET_IDENTITY_CONTEXT option specified in the *PutMsgOpts* parameter, but the queue had not been opened with the MQOO_SET_IDENTITY_CONTEXT option.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_SET_IDENTITY_CONTEXT (or another option that implies it) when the queue is opened.

2097 (0831) (RC2097): MQRC_CONTEXT_HANDLE_ERROR

Explanation

On an MQPUT or MQPUT1 call, MQPMO_PASS_IDENTITY_CONTEXT or MQPMO_PASS_ALL_CONTEXT was specified, but the handle specified in the *Context* field of the *PutMsgOpts* parameter is either not a valid queue handle, or it is a valid queue handle but the queue was not opened with MQOO_SAVE_ALL_CONTEXT.

Completion Code

MQCC_FAILED

Programmer response

Specify MQOO_SAVE_ALL_CONTEXT when the queue referred to is opened.

2098 (0832) (RC2098): MQRC_CONTEXT_NOT_AVAILABLE

Explanation

On an MQPUT or MQPUT1 call, MQPMO_PASS_IDENTITY_CONTEXT or MQPMO_PASS_ALL_CONTEXT was specified, but the queue handle specified in the *Context* field of the *PutMsgOpts* parameter has no context associated with it. This arises if no message has yet been successfully retrieved with the queue handle referred to, or if the last successful MQGET call was a browse.

This condition does not arise if the message that was last retrieved had no context associated with it.

- On z/OS, if a message is received by a message channel agent that is putting messages with the authority of the user identifier in the message, this code is returned in the *Feedback* field of an exception report if the message has no context associated with it.

Completion Code

MQCC_FAILED

Programmer response

Ensure that a successful nonbrowse get call has been issued with the queue handle referred to.

2099 (0833) (RC2099): MQRC_SIGNAL1_ERROR

Explanation

An MQGET call was issued, specifying MQGMO_SET_SIGNAL in the *GetMsgOpts* parameter, but the *Signal1* field is not valid.

- On z/OS, the address contained in the *Signal1* field is not valid, or points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- On Windows 95, Windows 98, the window handle in the *Signal1* field is not valid.

This reason code occurs only in the following environments: z/OS, Windows 95, Windows 98.

Completion Code

MQCC_FAILED

Programmer response

Correct the setting of the *Signal1* field.

2100 (0834) (RC2100): MQRC_OBJECT_ALREADY_EXISTS

Explanation

An MQOPEN call was issued to create a dynamic queue, but a queue with the same name as the dynamic queue already exists.

- On z/OS, a rare "race condition˵ can also give rise to this reason code; see the description of reason code MQRC_NAME_IN_USE for more details.

Completion Code

MQCC_FAILED

Programmer response

If supplying a dynamic queue name in full, ensure that it obeys the naming conventions for dynamic queues; if it does, either supply a different name, or delete the existing queue if it is no longer required. Alternatively, allow the queue manager to generate the name.

If the queue manager is generating the name (either in part or in full), reissue the MQOPEN call.

2101 (0835) (RC2101): MQRC_OBJECT_DAMAGED

Explanation

The object accessed by the call is damaged and cannot be used. For example, this might be because the definition of the object in main storage is not consistent, or because it differs from the definition of the object on disk, or because the definition on disk cannot be read. The object can be deleted, although it might not be possible to delete the associated user space.

- On z/OS, this reason occurs when the Db2 list header or structure number associated with a shared queue is zero. This situation arises as a result of using the MQSC command DELETE CFSTRUCT to delete the Db2 structure definition. The command resets the list header and structure number to zero for each of the shared queues that references the deleted CF structure.

Completion Code

MQCC_FAILED

Programmer response

It might be necessary to stop and restart the queue manager, or to restore the queue-manager data from backup storage.

- On IBM i, HP Integrity NonStop Server, and UNIX systems, consult the FFST record to obtain more detail about the problem.
- On z/OS, delete the shared queue and redefine it using the MQSC command DEFINE QLOCAL. This automatically defines a CF structure and allocates list headers for it.

2102 (0836) (RC2102): MQRC_RESOURCE_PROBLEM

Explanation

There are insufficient system resources to complete the call successfully. On z/OS this can indicate that Db2 errors occurred when using shared queues, or that the maximum number of shared queues that can be defined in a single coupling facility list structure has been reached.

Completion Code

MQCC_FAILED

Programmer response

Run the application when the machine is less heavily loaded.

- On z/OS, check the operator console for messages that might provide additional information.
- On IBM i, HP Integrity NonStop Server, and UNIX systems, consult the FFST record to obtain more detail about the problem.

2103 (0837) (RC2103): MQRC_ANOTHER_Q_MGR_CONNECTED

Explanation

An MQCONN or MQCONNX call was issued, but the thread or process is already connected to a different queue manager. The thread or process can connect to only one queue manager at a time.

- On z/OS, this reason code does not occur.
- On Windows, MTS objects do not receive this reason code, as connections to other queue managers are allowed.

Completion Code

MQCC_FAILED

Programmer response

Use the MQDISC call to disconnect from the queue manager that is already connected, and then issue the MQCONN or MQCONNX call to connect to the new queue manager.

Disconnecting from the existing queue manager closes any queues that are currently open; it is suggested that any uncommitted units of work are committed or backed out before the MQDISC call is issued.

2104 (0838) (RC2104): MQRC_UNKNOWN_REPORT_OPTION

Explanation

An MQPUT or MQPUT1 call was issued, but the *Report* field in the message descriptor MQMD contains one or more options that are not recognized by the local queue manager. The options are accepted.

The options that cause this reason code to be returned depend on the destination of the message; see the description of REPORT in Report options and message flags for more information.

Completion Code

MQCC_WARNING

Programmer response

If this reason code is expected, no corrective action is required. If this reason code is not expected, do the following:

- Ensure that the *Report* field in the message descriptor is initialized with a value when the message descriptor is declared, or is assigned a value prior to the MQPUT or MQPUT1 call.
- Ensure that the report options specified are valid; see the *Report* field described in the description of MQMD in MQMD - Message descriptor for valid report options.
- If multiple report options are being set by adding the individual report options together, ensure that the same report option is not added twice.
- Check that conflicting report options are not specified. For example, do not add both MQRO_EXCEPTION and MQRO_EXCEPTION_WITH_DATA to the *Report* field; only one of these can be specified.

2105 (0839) (RC2105): MQRC_STORAGE_CLASS_ERROR

Explanation

The MQPUT or MQPUT1 call was issued, but the storage-class object defined for the queue does not exist.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Create the storage-class object required by the queue, or modify the queue definition to use an existing storage class. The name of the storage-class object used by the queue is given by the *StorageClass* queue attribute.

2106 (083A) (RC2106): MQRC_COD_NOT_VALID_FOR_XCF_Q

Explanation

An MQPUT or MQPUT1 call was issued, but the *Report* field in the message descriptor MQMD specifies one of the MQRO_COD_* options and the target queue is an XCF queue. MQRO_COD_* options cannot be specified for XCF queues.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Remove the relevant MQRO_COD_* option.

2107 (083B) (RC2107): MQRC_XWAIT_CANCELED

Explanation

An MQXWAIT call was issued, but the call has been canceled because a STOP CHINIT command has been issued (or the queue manager has been stopped, which causes the same effect). See MQXWAIT for more information about the MQXWAIT call.

Completion Code

MQCC_FAILED

Programmer response

Tidy up and terminate.

2108 (083C) (RC2108): MQRC_XWAIT_ERROR

Explanation

An MQXWAIT call was issued, but the invocation was not valid for one of the following reasons:

- The wait descriptor MQXWD contains data that is not valid.
- The linkage stack level is not valid.
- The addressing mode is not valid.
- There are too many wait events outstanding.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Obey the rules for using the MQXWAIT call. For more information about MQWAIT, see MQXWAIT.

2109 (083D) (RC2109): MQRC_SUPPRESSED_BY_EXIT

Explanation

On any call other than MQCONN or MQDISC, the API crossing exit suppressed the call.

Completion Code

MQCC_FAILED

Programmer response

Obey the rules for MQI calls that the exit enforces. To find out the rules, see the writer of the exit.

2110 (083E) (RC2110): MQRC_FORMAT_ERROR

Explanation

An MQGET call was issued with the MQGMO_CONVERT option specified in the *GetMsgOpts* parameter, but the message cannot be converted successfully due to an error associated with the message format. Possible errors include:

- The format name in the message is MQFMT_NONE.
- A user-written exit with the name specified by the *Format* field in the message cannot be found.
- The message contains data that is not consistent with the format definition.

The message is returned unconverted to the application issuing the MQGET call, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

Completion Code

MQCC_WARNING

Programmer response

Check the format name that was specified when the message was put. If this is not one of the built-in formats, check that a suitable exit with the same name as the format is available for the queue manager to load. Verify that the data in the message corresponds to the format expected by the exit.

2111 (083F) (RC2111): MQRC_SOURCE_CCSID_ERROR

Explanation

The coded character-set identifier from which character data is to be converted is not valid or not supported.

This can occur on the MQGET call when the MQGMO_CONVERT option is included in the *GetMsgOpts* parameter; the coded character-set identifier in error is the *CodedCharSetId* field in the message being retrieved. In this case, the message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

This reason can also occur on the MQGET call when the message contains one or more MQ header structures (MQCIH, MQDLH, MQIIH, MQRMH), and the *CodedCharSetId* field in the message specifies a character set that does not have SBCS characters for the characters that are valid in queue names. MQ header structures containing such characters are not valid, and so the message is returned unconverted. The Unicode character set UCS-2 is an example of such a character set.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

This reason can also occur on the MQXCNVC call; the coded character-set identifier in error is the *SourceCCSID* parameter. Either the *SourceCCSID* parameter specifies a value that is not valid or not supported, or the *SourceCCSID* parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason can also occur on a MQSETMP/MQINQMP/MQDLTMP call when the application issuing the calls does not use Language Environment (LE) and defines CCSID values of MQCCSI_APPL (-3) for message property names and string property values.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Check the character-set identifier that was specified when the message was put, or that was specified for the `SourceCCSID` parameter on the MQXCNVC call. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the specified character set, conversion must be carried out by the application.

If this reason happens as result of a MQSETMP/MQINQMP/MQDLTMP call issued in a non-LE application program that has specified CCSID as MQCCSI_APPL (-3) then applications should be changed to specify the CCSID value used by the application to encode the property names or property string values.

Your applications should override the value of MQCCSI_APPL (-3) with the correct CCSID used as described in Redefinition of MQCCSI_APPL, or they should set the explicit CCSID value used to encode text strings in MQCHARV or similar structures.

2112 (0840) (RC2112): MQRC_SOURCE_INTEGER_ENC_ERROR

Explanation

On an MQGET call, with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the message being retrieved specifies an integer encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

This reason code can also occur on the MQXCNVC call, when the *Options* parameter contains an unsupported MQDCC_SOURCE_* value, or when MQDCC_SOURCE_ENC_UNDEFINED is specified for a UCS-2 code page.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Check the integer encoding that was specified when the message was put. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required integer encoding, conversion must be carried out by the application.

2113 (0841) (RC2113): MQRC_SOURCE_DECIMAL_ENC_ERROR

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the message being retrieved specifies a decimal encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

Completion Code

MQCC_WARNING

Programmer response

Check the decimal encoding that was specified when the message was put. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required decimal encoding, conversion must be carried out by the application.

2114 (0842) (RC2114): MQRC_SOURCE_FLOAT_ENC_ERROR

Explanation

On an MQGET call, with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the message being retrieved specifies a floating-point encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

Completion Code

MQCC_WARNING

Programmer response

Check the floating-point encoding that was specified when the message was put. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required floating-point encoding, conversion must be carried out by the application.

2115 (0843) (RC2115): MQRC_TARGET_CCSID_ERROR

Explanation

The coded character-set identifier to which character data is to be converted is not valid or not supported.

This can occur on the MQGET call when the MQGMO_CONVERT option is included in the *GetMsgOpts* parameter; the coded character-set identifier in error is the *CodedCharSetId* field in the *MsgDesc* parameter. In this case, the message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

This reason can also occur on the MQGET call when the message contains one or more MQ header structures (MQCIH, MQDLH, MQIIH, MQRMH), and the *CodedCharSetId* field in the *MsgDesc* parameter specifies a character set that does not have SBCS characters for the characters that are valid in queue names. The Unicode character set UCS-2 is an example of such a character set.

This reason can also occur on the MQXCNVC call; the coded character-set identifier in error is the *TargetCCSID* parameter. Either the *TargetCCSID* parameter specifies a value that is not valid or not supported, or the *TargetCCSID* parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Check the character-set identifier that was specified for the *CodedCharSetId* field in the *MsgDesc* parameter on the MQGET call, or that was specified for the *SourceCCSID* parameter on the MQXCNVC call. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the specified character set, conversion must be carried out by the application.

2116 (0844) (RC2116): MQRC_TARGET_INTEGER_ENC_ERROR

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the *MsgDesc* parameter specifies an integer encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message being retrieved, and the call completes with MQCC_WARNING.

This reason code can also occur on the MQXCNVC call, when the *Options* parameter contains an unsupported MQDCC_TARGET_* value, or when MQDCC_TARGET_ENC_UNDEFINED is specified for a UCS-2 code page.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Check the integer encoding that was specified. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required integer encoding, conversion must be carried out by the application.

2117 (0845) (RC2117): MQRC_TARGET_DECIMAL_ENC_ERROR

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the *MsgDesc* parameter specifies a decimal encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

Completion Code

MQCC_WARNING

Programmer response

Check the decimal encoding that was specified. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required decimal encoding, conversion must be carried out by the application.

2118 (0846) (RC2118): MQRC_TARGET_FLOAT_ENC_ERROR

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the *Encoding* value in the *MsgDesc* parameter specifies a floating-point encoding that is not recognized. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

Completion Code

MQCC_WARNING

Programmer response

Check the floating-point encoding that was specified. If this is correct, check that it is one for which queue-manager conversion is supported. If queue-manager conversion is not supported for the required floating-point encoding, conversion must be carried out by the application.

2119 (0847) (RC2119): MQRC_NOT_CONVERTED

Explanation

An MQGET call was issued with the MQGMO_CONVERT option specified in the *GetMsgOpts* parameter, but an error occurred during conversion of the data in the message. The message data is returned unconverted, the values of the *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter are set to those of the message returned, and the call completes with MQCC_WARNING.

If the message consists of several parts, each of which is described by its own *CodedCharSetId* and *Encoding* fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various *CodedCharSetId* and *Encoding* fields always correctly describe the relevant message data.

This error may also indicate that a parameter to the data-conversion service is not supported.

Completion Code

MQCC_WARNING

Programmer response

Check that the message data is correctly described by the *Format*, *CodedCharSetId* and *Encoding* parameters that were specified when the message was put. Also check that these values, and the *CodedCharSetId* and *Encoding* specified in the *MsgDesc* parameter on the MQGET call, are supported for queue-manager conversion. If the required conversion is not supported, conversion must be carried out by the application.

2120 (0848) (RC2120): MQRC_CONVERTED_MSG_TOO_BIG

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, the message data expanded during data conversion and exceeded the size of the buffer provided by the application. However, the message had already been removed from the queue because prior to conversion the message data could be accommodated in the application buffer without truncation.

The message is returned unconverted, with the *CompCode* parameter of the MQGET call set to MQCC_WARNING. If the message consists of several parts, each of which is described by its own character-set and encoding fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts may be converted and other parts not converted. However, the values returned in the various character-set and encoding fields always correctly describe the relevant message data.

This reason also occurs on the MQXCNVC call, when the *TargetBuffer* parameter is too small to accommodate the converted string, and the string has been truncated to fit in the buffer. The length of valid data returned is given by the *DataLength* parameter; in the case of a DBCS string or mixed SBCS/DBCS string, this length may be *less than* the length of *TargetBuffer*.

Completion Code

MQCC_WARNING

Programmer response

For the MQGET call, check that the exit is converting the message data correctly and setting the output length *DataLength* to the appropriate value. If it is, the application issuing the MQGET call must provide a larger buffer for the *Buffer* parameter.

For the MQXCNVC call, if the string must be converted without truncation, provide a larger output buffer.

2121 (0849) (RC2121): MQRC_NO_EXTERNAL_PARTICIPANTS

Explanation

An MQBEGIN call was issued to start a unit of work coordinated by the queue manager, but no participating resource managers have been registered with the queue manager. As a result, only changes to MQ resources can be coordinated by the queue manager in the unit of work.

This reason code occurs in the following environments: AIX®, HP-UX, IBM i, Solaris, Windows.

Completion Code

MQCC_WARNING

Programmer response

If the application does not require non-MQ resources to participate in the unit of work, this reason code can be ignored or the MQBEGIN call removed. Otherwise consult your system programmer to determine why the required resource managers have not been registered with the queue manager; the queue manager's configuration file might be in error.

2122 (084A) (RC2122): MQRC_PARTICIPANT_NOT_AVAILABLE

Explanation

An MQBEGIN call was issued to start a unit of work coordinated by the queue manager, but one or more of the participating resource managers that had been registered with the queue manager is not available. As a result, changes to those resources cannot be coordinated by the queue manager in the unit of work.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows.

Completion Code

MQCC_WARNING

Programmer response

If the application does not require non-MQ resources to participate in the unit of work, this reason code can be ignored. Otherwise consult your system programmer to determine why the required resource managers are not available. The resource manager might have been halted temporarily, or there might be an error in the queue manager's configuration file.

2123 (084B) (RC2123): MQRC_OUTCOME_MIXED

Explanation

The queue manager is acting as the unit-of-work coordinator for a unit of work that involves other resource managers, but one of the following occurred:

- An MQCMIT or MQDISC call was issued to commit the unit of work, but one or more of the participating resource managers backed-out the unit of work instead of committing it. As a result, the outcome of the unit of work is mixed.
- An MQBACK call was issued to back out a unit of work, but one or more of the participating resource managers had already committed the unit of work.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Examine the queue-manager error logs for messages relating to the mixed outcome; these messages identify the resource managers that are affected. Use procedures local to the affected resource managers to resynchronize the resources.

This reason code does not prevent the application initiating further units of work.

2124 (084C) (RC2124): MQRC_OUTCOME_PENDING

Explanation

The queue manager is acting as the unit-of-work coordinator for a unit of work that involves other resource managers, and an MQCMIT or MQDISC call was issued to commit the unit of work, but one or more of the participating resource managers has not confirmed that the unit of work was committed successfully.

The completion of the commit operation will happen at some point in the future, but there remains the possibility that the outcome will be mixed.

Windows Solaris HP-UX AIX This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

z/OS On z/OS, this situation can occur if a queue manager loses connectivity to a coupling facility structure while a unit of work that affects messages on shared queues is being committed or backed out.

Completion Code

MQCC_WARNING

Programmer response

Windows Solaris HP-UX AIX Use the normal error-reporting mechanisms to determine whether the outcome was mixed. If it was, take appropriate action to resynchronize the resources.

Windows Solaris HP-UX AIX This reason code does not prevent the application initiating further units of work.

z/OS If this reason code was returned as a result of loss of connectivity to a coupling facility structure on z/OS, the operation will be completed either when the queue manager reconnects to the affected structure, or when another queue manager in the queue-sharing group is able to perform peer recovery on the structure.

2125 (084D) (RC2125): MQRC_BRIDGE_STARTED

Explanation

The IMS bridge has been started.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Bridge Started.

2126 (084E) (RC2126): MQRC_BRIDGE_STOPPED

Explanation

The IMS bridge has been stopped.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Bridge Stopped.

2127 (084F) (RC2127): MQRC_ADAPTER_STORAGE_SHORTAGE

Explanation

On an MQCONN call, the adapter was unable to acquire storage.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Notify the system programmer. The system programmer should determine why the system is short on storage, and take appropriate action, for example, increase the region size on the step or job card.

2128 (0850) (RC2128): MQRC_UOW_IN_PROGRESS

Explanation

An MQBEGIN call was issued to start a unit of work coordinated by the queue manager, but a unit of work is already in existence for the connection handle specified. This may be a global unit of work started by a previous MQBEGIN call, or a unit of work that is local to the queue manager or one of the cooperating resource managers. No more than one unit of work can exist concurrently for a connection handle.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Review the application logic to determine why there is a unit of work already in existence. Move the MQBEGIN call to the appropriate place in the application.

2129 (0851) (RC2129): MQRC_ADAPTER_CONN_LOAD_ERROR

Explanation

On an MQCONN call, the connection handling module could not be loaded, so the adapter could not link to it. The connection handling module name is:

- CSQBCON for batch applications
- CSQQCONN or CSQQCON2 for IMS applications

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified in the batch application program execution JCL, and in the queue-manager startup JCL.

2130 (0852) (RC2130): MQRC_ADAPTER_SERV_LOAD_ERROR

Explanation

On an MQI call, the batch adapter could not load one of the following API service module, and so could not link to it:

- CSQBSRV

- CSQAPEPL
- CSQBCRMH
- CSQBAPPL

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified in the batch application program execution JCL, and in the queue-manager startup JCL.

2131 (0853) (RC2131): MQRC_ADAPTER_DEFS_ERROR

Explanation

On an MQCONN call, the subsystem definition module (CSQBDEFV for batch and CSQQDEFV for IMS) does not contain the required control block identifier.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check your library concatenation. If this is correct, check that the CSQBDEFV or CSQQDEFV module contains the required subsystem ID.

2132 (0854) (RC2132): MQRC_ADAPTER_DEFS_LOAD_ERROR

Explanation

On an MQCONN call, the subsystem definition module (CSQBDEFV for batch and CSQQDEFV for IMS) could not be loaded.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified in the application program execution JCL, and in the queue-manager startup JCL.

2133 (0855) (RC2133): MQRC_ADAPTER_CONV_LOAD_ERROR

Explanation

On an MQGET call, the adapter (batch or IMS) could not load the data conversion services modules.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified in the batch application program execution JCL, and in the queue-manager startup JCL.

2134 (0856) (RC2134): MQRC_BO_ERROR

Explanation

On an MQBEGIN call, the begin-options structure MQBO is not valid, for one of the following reasons:

- The *StrucId* field is not MQBO_STRUC_ID.
- The *Version* field is not MQBO_VERSION_1.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQBO structure are set correctly.

2135 (0857) (RC2135): MQRC_DH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQDH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQDH_STRUC_ID.
- The *Version* field is not MQDH_VERSION_1.
- The *StrucLength* field specifies a value that is too small to include the structure plus the arrays of MQOR and MQPMR records.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2136 (0858) (RC2136): MQRC_MULTIPLE_REASONS

Explanation

An MQOPEN, MQPUT or MQPUT1 call was issued to open a distribution list or put a message to a distribution list, but the result of the call was not the same for all of the destinations in the list. One of the following applies:

- The call succeeded for some of the destinations but not others. The completion code is MQCC_WARNING in this case.
- The call failed for all of the destinations, but for differing reasons. The completion code is MQCC_FAILED in this case.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Examine the MQRR response records to identify the destinations for which the call failed, and the reason for the failure. Ensure that sufficient response records are provided by the application on the call to enable the error(s) to be determined. For the MQPUT1 call, the response records must be specified using the MQOD structure, and not the MQPMO structure.

2137 (0859) (RC2137): MQRC_OPEN_FAILED

Explanation

A queue or other MQ object could not be opened successfully, for one of the following reasons:

- An MQCONN or MQCONNX call was issued, but the queue manager was unable to open an object that is used internally by the queue manager. As a result, processing cannot continue. The error log will contain the name of the object that could not be opened.
- An MQPUT call was issued to put a message to a distribution list, but the message could not be sent to the destination to which this reason code applies because that destination was not opened successfully by the MQOPEN call. This reason occurs only in the *Reason* field of the MQRR response record.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Do one of the following:

- If the error occurred on the MQCONN or MQCONNX call, ensure that the required objects exist by running the following command and then retrying the application:

```
STRMQM -c qmgr
```

where `qmgr` should be replaced by the name of the queue manager.

- If the error occurred on the MQPUT call, examine the MQRR response records specified on the MQOPEN call to determine the reason that the queue failed to open. Ensure that sufficient response records are provided by the application on the call to enable the error(s) to be determined.

2138 (085A) (RC2138): MQRC_ADAPTER_DISC_LOAD_ERROR

Explanation

On an MQDISC call, the disconnect handling module (CSQBDSC for batch and CSQQDISC for IMS) could not be loaded, so the adapter could not link to it.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified in the application program execution JCL, and in the queue-manager startup JCL. Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2139 (085B) (RC2139): MQRC_CNO_ERROR

Explanation

On an MQCONNX call, the connect-options structure MQCNO is not valid, for one of the following reasons:

- The *StrucId* field is not MQCNO_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the parameter pointer points to read-only storage.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQCNO structure are set correctly.

2140 (085C) (RC2140): MQRC_CICS_WAIT_FAILED

Explanation

On any MQI call, the CICS adapter issued an EXEC CICS WAIT request, but the request was rejected by CICS.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Examine the CICS trace data for actual response codes. The most likely cause is that the task has been canceled by the operator or by the system.

2141 (085D) (RC2141): MQRC_DLH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQDLH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQDLH_STRUC_ID.
- The *Version* field is not MQDLH_VERSION_1.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2142 (085E) (RC2142): MQRC_HEADER_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQ header structure that is not valid. Possible errors include the following:

- The *StrucId* field is not valid.
- The *Version* field is not valid.
- The *StrucLength* field specifies a value that is too small.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2143 (085F) (RC2143): MQRC_SOURCE_LENGTH_ERROR

Explanation

On the MQXCNVC call, the *SourceLength* parameter specifies a length that is less than zero or not consistent with the string's character set or content (for example, the character set is a double-byte character set, but the length is not a multiple of two). This reason also occurs if the *SourceLength* parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason code can also occur on the MQGET call when the MQGMO_CONVERT option is specified. In this case it indicates that the MQRC_SOURCE_LENGTH_ERROR reason was returned by an MQXCNVC call issued by the data conversion exit.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Specify a length that is zero or greater. If the reason code occurs on the MQGET call, check that the logic in the data-conversion exit is correct.

2144 (0860) (RC2144): MQRC_TARGET_LENGTH_ERROR

Explanation

On the MQXCNVC call, the *TargetLength* parameter is not valid for one of the following reasons:

- *TargetLength* is less than zero.
- The *TargetLength* parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The MQDCC_FILL_TARGET_BUFFER option is specified, but the value of *TargetLength* is such that the target buffer cannot be filled completely with valid characters. This can occur when *TargetCCSID* is a pure DBCS character set (such as UCS-2), but *TargetLength* specifies a length that is an odd number of bytes.

This reason code can also occur on the MQGET call when the MQGMO_CONVERT option is specified. In this case it indicates that the MQRC_TARGET_LENGTH_ERROR reason was returned by an MQXCNVC call issued by the data conversion exit.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Specify a length that is zero or greater. If the MQDCC_FILL_TARGET_BUFFER option is specified, and *TargetCCSID* is a pure DBCS character set, ensure that *TargetLength* specifies a length that is a multiple of two.

If the reason code occurs on the MQGET call, check that the logic in the data-conversion exit is correct.

2145 (0861) (RC2145): MQRC_SOURCE_BUFFER_ERROR

Explanation

On the MQXCNVC call, the *SourceBuffer* parameter pointer is not valid, or points to storage that cannot be accessed for the entire length specified by *SourceLength*. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason code can also occur on the MQGET call when the MQGMO_CONVERT option is specified. In this case it indicates that the MQRC_SOURCE_BUFFER_ERROR reason was returned by an MQXCNVC call issued by the data conversion exit.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Specify a valid buffer. If the reason code occurs on the MQGET call, check that the logic in the data-conversion exit is correct.

2146 (0862) (RC2146): MQRC_TARGET_BUFFER_ERROR

Explanation

On the MQXCNVC call, the *TargetBuffer* parameter pointer is not valid, or points to read-only storage, or to storage that cannot be accessed for the entire length specified by *TargetLength*. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

This reason code can also occur on the MQGET call when the MQGMO_CONVERT option is specified. In this case it indicates that the MQRC_TARGET_BUFFER_ERROR reason was returned by an MQXCNVC call issued by the data conversion exit.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Specify a valid buffer. If the reason code occurs on the MQGET call, check that the logic in the data-conversion exit is correct.

2148 (0864) (RC2148): MQRC_IIH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQIIH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQIIH_STRUC_ID.
- The *Version* field is not MQIIH_VERSION_1.
- The *StrucLength* field is not MQIIH_LENGTH_1.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2149 (0865) (RC2149): MQRC_PCF_ERROR

Explanation

An MQPUT or MQPUT1 call was issued to put a message containing PCF data, but the length of the message does not equal the sum of the lengths of the PCF structures present in the message. This can occur for messages with the following format names:

- MQFMT_ADMIN
- MQFMT_EVENT
- MQFMT_PCF

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the length of the message specified on the MQPUT or MQPUT1 call equals the sum of the lengths of the PCF structures contained within the message data.

2150 (0866) (RC2150): MQRC_DBCS_ERROR

Explanation

An error was encountered attempting to convert a double-byte character set (DBCS) string. This can occur in the following cases:

- On the MQXCNVC call, when the *SourceCCSID* parameter specifies the coded character-set identifier of a double-byte character set, but the *SourceBuffer* parameter does not contain a valid DBCS string. This may be because the string contains characters that are not valid DBCS characters, or because the string is a mixed SBCS/DBCS string and the shift-out/shift-in characters are not correctly paired. The completion code is MQCC_FAILED in this case.
- On the MQGET call, when the MQGMO_CONVERT option is specified. In this case it indicates that the MQRC_DBCS_ERROR reason code was returned by an MQXCNVC call issued by the data conversion exit. The completion code is MQCC_WARNING in this case.
- For the z/OS dead letter handler utility CSQUDLQH, when the rule being processed uses the default of CONVERT(YES). Modify the rule to use CONVERT(NO) if the data does not need to be converted.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Specify a valid string.

If the reason code occurs on the MQGET call, check that the data in the message is valid, and that the logic in the data-conversion exit is correct.

2152 (0868) (RC2152): MQRC_OBJECT_NAME_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued to open a distribution list (that is, the *RecsPresent* field in MQOD is greater than zero), but the *ObjectName* field is neither blank nor the null string.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

If it is intended to open a distribution list, set the *ObjectName* field to blanks or the null string. If it is not intended to open a distribution list, set the *RecsPresent* field to zero.

2153 (0869) (RC2153): MQRC_OBJECT_Q_MGR_NAME_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued to open a distribution list (that is, the *RecsPresent* field in MQOD is greater than zero), but the *ObjectQMgrName* field is neither blank nor the null string.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

If it is intended to open a distribution list, set the *ObjectQMgrName* field to blanks or the null string. If it is not intended to open a distribution list, set the *RecsPresent* field to zero.

2154 (086A) (RC2154): MQRC_RECS_PRESENT_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued, but the call failed for one of the following reasons:

- *RecsPresent* in MQOD is less than zero.
- *ObjectType* in MQOD is not MQOT_Q, and *RecsPresent* is not zero. *RecsPresent* must be zero if the object being opened is not a queue.
- IBM MQ Multicast is being used and *RecsPresent* in MQOD is not set to zero. IBM MQ Multicast does not use distribution lists.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

If it is intended to open a distribution list, set the *ObjectType* field to MQOT_Q and *RecsPresent* to the number of destinations in the list. If it is not intended to open a distribution list, set the *RecsPresent* field to zero.

2155 (086B) (RC2155): MQRC_OBJECT_RECORDS_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued to open a distribution list (that is, the *RecsPresent* field in MQOD is greater than zero), but the MQOR object records are not specified correctly. One of the following applies:

- *ObjectRecOffset* is zero and *ObjectRecPtr* is zero or the null pointer.
- *ObjectRecOffset* is not zero and *ObjectRecPtr* is not zero and not the null pointer.
- *ObjectRecPtr* is not a valid pointer.
- *ObjectRecPtr* or *ObjectRecOffset* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that one of *ObjectRecOffset* and *ObjectRecPtr* is zero and the other nonzero. Ensure that the field used points to accessible storage.

2156 (086C) (RC2156): MQRC_RESPONSE_RECORDS_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued to open a distribution list (that is, the *RecsPresent* field in MQOD is greater than zero), but the MQRR response records are not specified correctly. One of the following applies:

- *ResponseRecOffset* is not zero and *ResponseRecPtr* is not zero and not the null pointer.
- *ResponseRecPtr* is not a valid pointer.
- *ResponseRecPtr* or *ResponseRecOffset* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that at least one of *ResponseRecOffset* and *ResponseRecPtr* is zero. Ensure that the field used points to accessible storage.

2157 (086D) (RC2157): MQRC_ASID_MISMATCH

Explanation

On any MQI call, the caller's primary ASID was found to be different from the home ASID.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Correct the application (MQI calls cannot be issued in cross-memory mode). Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2158 (086E) (RC2158): MQRC_PMO_RECORD_FLAGS_ERROR

Explanation

An MQPUT or MQPUT1 call was issued to put a message, but the *PutMsgRecFields* field in the MQPMO structure is not valid, for one of the following reasons:

- The field contains flags that are not valid.
- The message is being put to a distribution list, and put message records have been provided (that is, *RecsPresent* is greater than zero, and one of *PutMsgRecOffset* or *PutMsgRecPtr* is nonzero), but *PutMsgRecFields* has the value MQPMRF_NONE.
- MQPMRF_ACCOUNTING_TOKEN is specified without either MQPMO_SET_IDENTITY_CONTEXT or MQPMO_SET_ALL_CONTEXT.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that *PutMsgRecFields* is set with the appropriate MQPMRF_* flags to indicate which fields are present in the put message records. If MQPMRF_ACCOUNTING_TOKEN is specified, ensure that either MQPMO_SET_IDENTITY_CONTEXT or MQPMO_SET_ALL_CONTEXT is also specified. Alternatively, set both *PutMsgRecOffset* and *PutMsgRecPtr* to zero.

2159 (086F) (RC2159): MQRC_PUT_MSG_RECORDS_ERROR

Explanation

An MQPUT or MQPUT1 call was issued to put a message to a distribution list, but the MQPMR put message records are not specified correctly. One of the following applies:

- *PutMsgRecOffset* is not zero and *PutMsgRecPtr* is not zero and not the null pointer.
- *PutMsgRecPtr* is not a valid pointer.
- *PutMsgRecPtr* or *PutMsgRecOffset* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that at least one of *PutMsgRecOffset* and *PutMsgRecPtr* is zero. Ensure that the field used points to accessible storage.

2160 (0870) (RC2160): MQRC_CONN_ID_IN_USE

Explanation

On an MQCONN call, the connection identifier assigned by the queue manager to the connection between a CICS or IMS allied address space and the queue manager conflicts with the connection identifier of another connected CICS or IMS system. The connection identifier assigned is as follows:

- For CICS, the applid
- For IMS, the IMSID parameter on the IMSCTRL (sysgen) macro, or the IMSID parameter on the execution parameter (EXEC card in IMS control region JCL)
- For batch, the job name
- For TSO, the user ID

A conflict arises only if there are two CICS systems, two IMS systems, or one each of CICS and IMS, having the same connection identifiers. Batch and TSO connections need not have unique identifiers.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the naming conventions used in different systems that might connect to the queue manager do not conflict.

2161 (0871) (RC2161): MQRC_Q_MGR_QUIESCING

Explanation

An MQI call was issued, but the call failed because the queue manager is quiescing (preparing to shut down).

When the queue manager is quiescing, the MQOPEN, MQPUT, MQPUT1, and MQGET calls can still complete successfully, but the application can request that they fail by specifying the appropriate option on the call:

- MQOO_FAIL_IF_QUIESCING on MQOPEN
- MQPMO_FAIL_IF_QUIESCING on MQPUT or MQPUT1
- MQGMO_FAIL_IF_QUIESCING on MQGET

Specifying these options enables the application to become aware that the queue manager is preparing to shut down.

- On z/OS:
  - For batch applications, this reason can be returned to applications running in LPARs that do not have a queue manager installed.
  - For CICS applications, this reason can be returned when no connection was established.

- On IBM i for applications running in compatibility mode, this reason can be returned when no connection was established.

Completion Code

MQCC_FAILED

Programmer response

The application should tidy up and end. If the application specified the MQOO_FAIL_IF_QUIESCING, MQPMO_FAIL_IF_QUIESCING, or MQGMO_FAIL_IF_QUIESCING option on the failing call, the relevant option can be removed and the call reissued. By omitting these options, the application can continue working to complete and commit the current unit of work, but the application does not start a new unit of work.

2162 (0872) (RC2162): MQRC_Q_MGR_STOPPING

Explanation

An MQI call was issued, but the call failed because the queue manager is shutting down. If the call was an MQGET call with the MQGMO_WAIT option, the wait has been canceled. No more MQI calls can be issued.

For MQ MQI client applications, it is possible that the call did complete successfully, even though this reason code is returned with a *CompCode* of MQCC_FAILED.

- On z/OS, the MQRC_CONNECTION_BROKEN reason may be returned instead if, as a result of system scheduling factors, the queue manager shuts down before the call completes.

Completion Code

MQCC_FAILED

Programmer response

The application should tidy up and end. If the application is in the middle of a unit of work coordinated by an external unit-of-work coordinator, the application should issue the appropriate call to back out the unit of work. Any unit of work that is coordinated by the queue manager is backed out automatically.

2163 (0873) (RC2163): MQRC_DUPLICATE_RECOV_COORD

Explanation

On an MQCONN or MQCONNX call, a recovery coordinator already exists for the connection name specified on the connection call issued by the adapter.

A conflict arises only if there are two CICS systems, two IMS systems, or one each of CICS and IMS, having the same connection identifiers. Batch and TSO connections need not have unique identifiers.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the naming conventions used in different systems that might connect to the queue manager do not conflict.

2173 (087D) (RC2173): MQRC_PMO_ERROR

Explanation

On an MQPUT or MQPUT1 call, the MQPMO structure is not valid, for one of the following reasons:

- The *StrucId* field is not MQPMO_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQPMO structure are set correctly.

2182 (0886) (RC2182): MQRC_API_EXIT_NOT_FOUND

Explanation

The API crossing exit entry point could not be found.

Completion Code

MQCC_FAILED

Programmer response

Check the entry point name is valid for the library module.

2183 (0887) (RC2183): MQRC_API_EXIT_LOAD_ERROR

Explanation

The API crossing exit module could not be linked to. If this message is returned when the API crossing exit is called *after* the process has been run, the process itself might have completed correctly.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library concatenation has been specified, and that the API crossing exit module is executable and correctly named. Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2184 (0888) (RC2184): MQRC_REMOTE_Q_NAME_ERROR

Explanation

On an MQOPEN or MQPUT1 call, one of the following occurred:

- A local definition of a remote queue (or an alias to one) was specified, but the *RemoteQName* attribute in the remote queue definition is entirely blank. Note that this error occurs even if the *XmitQName* in the definition is not blank.
- The *ObjectQMgrName* field in the object descriptor is not blank and not the name of the local queue manager, but the *ObjectName* field is blank.

Completion Code

MQCC_FAILED

Programmer response

Alter the local definition of the remote queue and supply a valid remote queue name, or supply a nonblank *ObjectName* in the object descriptor, as appropriate.

This reason code is also used to identify the corresponding event message Remote Queue Name Error.

2185 (0889) (RC2185): MQRC_INCONSISTENT_PERSISTENCE

Explanation

An MQPUT call was issued to put a message in a group or a segment of a logical message, but the value specified or defaulted for the *Persistence* field in MQMD is not consistent with the current group and segment information retained by the queue manager for the queue handle. All messages in a group and all segments in a logical message must have the same value for persistence, that is, all must be persistent, or all must be nonpersistent.

If the current call specifies MQPMO_LOGICAL_ORDER, the call fails. If the current call does not specify MQPMO_LOGICAL_ORDER, but the previous MQPUT call for the queue handle did, the call succeeds with completion code MQCC_WARNING.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Modify the application to ensure that the same value of persistence is used for all messages in the group, or all segments of the logical message.

2186 (088A) (RC2186): MQRC_GMO_ERROR

Explanation

On an MQGET call, the MQGMO structure is not valid, for one of the following reasons:

- The *StrucId* field is not MQGMO_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQGMO structure are set correctly.

2187 (088B) (RC2187): MQRC_CICS_BRIDGE_RESTRICTION

Explanation

It is not permitted to issue MQI calls from user transactions that are run in an MQ/CICS bridge environment where the bridge exit also issues MQI calls. The MQI call fails. If it occurs in the bridge exit, it results in a transaction abend. If it occurs in the user transaction, it can result in a transaction abend.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The transaction cannot be run using the MQ/CICS bridge. Refer to the appropriate CICS manual for information about restrictions in the MQ/CICS bridge environment.

2188 (088C) (RC2188): MQRC_STOPPED_BY_CLUSTER_EXIT

Explanation

An MQOPEN, MQPUT, or MQPUT1 call was issued to open or put a message on a cluster queue, but the cluster workload exit rejected the call.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check the cluster workload exit to ensure that it has been written correctly. Determine why it rejected the call and correct the problem.

2189 (088D) (RC2189): MQRC_CLUSTER_RESOLUTION_ERROR

Explanation

An MQOPEN, MQPUT, or MQPUT1 call was issued to open or put a message on a cluster queue, but the queue definition could not be resolved correctly because a response was required from the repository manager but none was available.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the repository manager is operating and that the queue and channel definitions are correct.

2190 (088E) (RC2190): MQRC_CONVERTED_STRING_TOO_BIG

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, a string in a fixed-length field in the message expanded during data conversion and exceeded the size of the field. When this happens, the queue manager tries discarding trailing blank characters and characters following the first null character to make the string fit, but in this case there were insufficient characters that could be discarded.

This reason code can also occur for messages with a format name of MQFMT_IMS_VAR_STRING. When this happens, it indicates that the IMS variable string expanded such that its length exceeded the capacity of the 2 byte binary length field contained within the structure of the IMS variable string. (The queue manager never discards trailing blanks in an IMS variable string.)

The message is returned unconverted, with the *CompCode* parameter of the MQGET call set to MQCC_WARNING. If the message consists of several parts, each of which is described by its own character-set and encoding fields (for example, a message with format name MQFMT_DEAD_LETTER_HEADER), some parts might be converted and other parts not converted. However, the values returned in the various character-set and encoding fields always correctly describe the relevant message data.

This reason code does not occur if the string can be made to fit by discarding trailing blank characters.

Completion Code

MQCC_WARNING

Programmer response

Check that the fields in the message contain the correct values, and that the character-set identifiers specified by the sender and receiver of the message are correct. If they are, the layout of the data in the message must be modified to increase the lengths of the field, or fields so that there is sufficient space to permit the string, or strings to expand when converted.

2191 (088F) (RC2191): MQRC_TMC_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQTMC2 structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQTMC_STRUC_ID.
- The *Version* field is not MQTMC_VERSION_2.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2192 (0890) (RC2192): MQRC_PAGESET_FULL

Explanation

Former name for MQRC_STORAGE_MEDIUM_FULL.

2192 (0890) (RC2192): MQRC_STORAGE_MEDIUM_FULL

Explanation

An MQI call or command was issued to operate on an object, but the call failed because the external storage medium is full. One of the following applies:

- A page-set data set is full (nonshared queues only).
- A coupling-facility structure is full (shared queues only).
- A coupling-facility is full. This situation can arise when the coupling facility structure is configured to use SCM storage (SCMMAXSIZE configured in CFRM policy) and messages are offloaded to SCM storage because the coupling facility structure has reached 90% threshold. Additional SCM use requires further augmented storage for the structure and there is insufficient storage in the coupling-facility to support this.
- The SMDS was full.

You can get this reason code when the page set or SMDS were expanding, but the space was not yet available. Check the messages in the job log to see the status of any expansion.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check which queues contain messages and look for applications that might be filling the queues unintentionally. Be aware that the queue that has caused the page set or coupling-facility structure to become full is not necessarily the queue referenced by the MQI call that returned MQRC_STORAGE_MEDIUM_FULL.

Check that all of the usual server applications are operating correctly and processing the messages on the queues.

If the applications and servers are operating correctly, increase the number of server applications to cope with the message load, or request the system programmer to increase the size of the page-set data sets.

2193 (0891) (RC2193): MQRC_PAGESET_ERROR

Explanation

An error was encountered with the page set while attempting to access it for a locally defined queue. This could be because the queue is on a page set that does not exist. A console message is issued that tells you the number of the page set in error. For example if the error occurred in the TEST job, and your user identifier is ABCDEFG, the message is:

```
CSQI041I CSQIALLC JOB TEST USER ABCDEFG HAD ERROR ACCESSING PAGE SET 27
```

If this reason code occurs while attempting to delete a dynamic queue with MQCLOSE, the dynamic queue has not been deleted.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check that the storage class for the queue maps to a valid page set using the DISPLAY Q(xx) STGCLASS, DISPLAY STGCLASS(xx), and DISPLAY USAGE PSID commands. If you are unable to resolve the problem, notify the system programmer who should:

- Collect the following diagnostic information:
  - A description of the actions that led to the error
  - A listing of the application program being run at the time of the error
  - Details of the page sets defined for use by the queue manager
- Attempt to re-create the problem, and take a system dump immediately after the error occurs
- Contact your IBM Support Center

2194 (0892) (RC2194): MQRC_NAME_NOT_VALID_FOR_TYPE

Explanation

An MQOPEN call was issued to open the queue manager definition, but the *ObjectName* field in the *ObjDesc* parameter is not blank.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the *ObjectName* field is set to blanks.

2195 (0893) (RC2195): MQRC_UNEXPECTED_ERROR

Explanation

The call was rejected because an unexpected error occurred.

Completion Code

MQCC_FAILED

Programmer response

Check the application's parameter list to ensure, for example, that the correct number of parameters was passed, and that data pointers and storage keys are valid. If the problem cannot be resolved, contact your system programmer.

- On z/OS, check the joblog and logrec, and whether any information has been displayed on the console. If this error occurs on an MQCONN or MQCONNX call, check that the subsystem named is an active MQ

subsystem. In particular, check that it is not a Db2 subsystem. If the problem cannot be resolved, rerun the application with a CSQSNAP DD card (if you have not already got a dump) and send the resulting dump to IBM.
- On IBM i, consult the FFST record to obtain more detail about the problem.
- On HP Integrity NonStop Server, and UNIX systems, consult the FDC file to obtain more detail about the problem.

2196 (0894) (RC2196): MQRC_UNKNOWN_XMIT_Q

Explanation

On an MQOPEN or MQPUT1 call, a message is to be sent to a remote queue manager. The *ObjectName* or the *ObjectQMgrName* in the object descriptor specifies the name of a local definition of a remote queue (in the latter case queue-manager aliasing is being used), but the *XmitQName* attribute of the definition is not blank and not the name of a locally-defined queue.

Completion Code

MQCC_FAILED

Programmer response

Check the values specified for *ObjectName* and *ObjectQMgrName*. If these are correct, check the queue definitions.

This reason code is also used to identify the corresponding event message Unknown Transmission Queue.

2197 (0895) (RC2197): MQRC_UNKNOWN_DEF_XMIT_Q

Explanation

An MQOPEN or MQPUT1 call was issued specifying a remote queue as the destination. If a local definition of the remote queue was specified, or if a queue-manager alias is being resolved, the *XmitQName* attribute in the local definition is blank.

Because there is no queue defined with the same name as the destination queue manager, the queue manager has attempted to use the default transmission queue. However, the name defined by the *DefXmitQName* queue-manager attribute is not the name of a locally-defined queue.

Completion Code

MQCC_FAILED

Programmer response

Correct the queue definitions, or the queue-manager attribute.

This reason code is also used to identify the corresponding event message Unknown Default Transmission Queue.

2198 (0896) (RC2198): MQRC_DEF_XMIT_Q_TYPE_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued specifying a remote queue as the destination. Either a local definition of the remote queue was specified, or a queue-manager alias was being resolved, but in either case the *XmitQName* attribute in the local definition is blank.

Because there is no transmission queue defined with the same name as the destination queue manager, the local queue manager has attempted to use the default transmission queue. However, although there is a queue defined by the *DefXmitQName* queue-manager attribute, it is not a local queue.

Completion Code

MQCC_FAILED

Programmer response

Do one of the following:

- Specify a local transmission queue as the value of the *XmitQName* attribute in the local definition of the remote queue.
- Define a local transmission queue with a name that is the same as that of the remote queue manager.
- Specify a local transmission queue as the value of the *DefXmitQName* queue-manager attribute.

See XmitQName for more information about transmission queue names.

This reason code is also used to identify the corresponding event message Default Transmission Queue Type Errorr.

2199 (0897) (RC2199): MQRC_DEF_XMIT_Q_USAGE_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued specifying a remote queue as the destination. Either a local definition of the remote queue was specified, or a queue-manager alias was being resolved, but in either case the *XmitQName* attribute in the local definition is blank.

Because there is no transmission queue defined with the same name as the destination queue manager, the local queue manager has attempted to use the default transmission queue. However, the queue defined by the *DefXmitQName* queue-manager attribute does not have a *Usage* attribute of MQUS_TRANSMISSION.

This reason code is returned from MQOPEN or MQPUT1, if the queue manager's Default Transmission Queue is about to be used, but the name of this queue is SYSTEM.CLUSTER.TRANSMIT.QUEUE. This queue is reserved for clustering, so it is not valid to set the queue manager's Default Transmission Queue to this name.

Completion Code

MQCC_FAILED

Programmer response

Do one of the following:

- Specify a local transmission queue as the value of the *XmitQName* attribute in the local definition of the remote queue.
- Define a local transmission queue with a name that is the same as that of the remote queue manager.
- Specify a different local transmission queue as the value of the *DefXmitQName* queue-manager attribute.
- Change the *Usage* attribute of the *DefXmitQName* queue to MQUS_TRANSMISSION.

See XmitQName for more information about transmission queue names.

This reason code is also used to identify the corresponding event message Default Transmission Queue Usage Error.

2201 (0899) (RC2201): MQRC_NAME_IN_USE

Explanation

An MQOPEN call was issued to create a dynamic queue, but a queue with the same name as the dynamic queue already exists. The existing queue is one that is logically deleted, but for which there are still one or more open handles. For more information, see MQOPEN.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Either ensure that all handles for the previous dynamic queue are closed, or ensure that the name of the new queue is unique; see the description for reason code MQRC_OBJECT_ALREADY_EXISTS.

2202 (089A) (RC2202): MQRC_CONNECTION_QUIESCING

Explanation

This reason code is issued when the connection to the queue manager is in quiescing state, and an application issues one of the following calls:

- MQCONN or MQCONNX
- MQOPEN, with no connection established, or with MQOO_FAIL_IF_QUIESCING included in the *Options* parameter
- MQGET, with MQGMO_FAIL_IF_QUIESCING included in the *Options* field of the *GetMsgOpts* parameter
- MQPUT or MQPUT1, with MQPMO_FAIL_IF_QUIESCING included in the *Options* field of the *PutMsgOpts* parameter

MQRC_CONNECTION_QUIESCING is also issued by the message channel agent (MCA) when the queue manager is in quiescing state.

Completion Code

MQCC_FAILED

Programmer response

The application should tidy up and terminate. Any uncommitted changes in a unit of work should be backed out.

2203 (089B) (RC2203): MQRC_CONNECTION_STOPPING

Explanation

This reason code is issued when the connection to the queue manager is shutting down, and the application issues an MQI call. No more message-queuing calls can be issued. For the MQGET call, if the MQGMO_WAIT option was specified, the wait is canceled.

Note that the MQRC_CONNECTION_BROKEN reason may be returned instead if, as a result of system scheduling factors, the queue manager shuts down before the call completes.

MQRC_CONNECTION_STOPPING is also issued by the message channel agent (MCA) when the queue manager is shutting down.

For MQ MQI client applications, it is possible that the call did complete successfully, even though this reason code is returned with a *CompCode* of MQCC_FAILED.

Completion Code

MQCC_FAILED

Programmer response

The application should tidy up and terminate. Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2204 (089C) (RC2204): MQRC_ADAPTER_NOT_AVAILABLE

Explanation

This is issued only for CICS applications, if any call is issued and the CICS adapter (a Task Related User Exit) has been disabled, or has not been enabled.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The application should tidy up and terminate. Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2206 (089E) (RC2206): MQRC_MSG_ID_ERROR

Explanation

An MQGET call was issued to retrieve a message using the message identifier as a selection criterion, but the call failed because selection by message identifier is not supported on this queue.

- On z/OS, the queue is a shared queue, but the *IndexType* queue attribute does not have an appropriate value:
  - If selection is by message identifier alone, *IndexType* must have the value MQIT_MSG_ID.
  - If selection is by message identifier and correlation identifier combined, *IndexType* must have the value MQIT_MSG_ID or MQIT_CORREL_ID. However, the match-any values of MQCI_NONE and MQMI_NONE respectively are exceptions to this rule, and result in the 2206 MQRC_MSG_ID_ERROR reason code.
- On HP Integrity NonStop Server, a key file is required but has not been defined.

Completion Code

MQCC_FAILED

Programmer response

Do one of the following:

- Modify the application so that it does not use selection by message identifier: set the *MsgId* field to MQMI_NONE and do not specify MQMO_MATCH_MSG_ID in MQGMO.
- On z/OS, change the *IndexType* queue attribute to MQIT_MSG_ID.
- On HP Integrity NonStop Server, define a key file.

2207 (089F) (RC2207): MQRC_CORREL_ID_ERROR

Explanation

An MQGET call was issued to retrieve a message using the correlation identifier as a selection criterion, but the call failed because selection by correlation identifier is not supported on this queue.

- On z/OS, the queue is a shared queue, but the *IndexType* queue attribute does not have an appropriate value:
  - If selection is by correlation identifier alone, *IndexType* must have the value MQIT_CORREL_ID.
  - If selection is by correlation identifier and message identifier combined, *IndexType* must have the value MQIT_CORREL_ID or MQIT_MSG_ID.
- On HP Integrity NonStop Server, a key file is required but has not been defined.

Completion Code

MQCC_FAILED

Programmer response

Do one of the following:

- On z/OS, change the *IndexType* queue attribute to MQIT_CORREL_ID.
- On HP Integrity NonStop Server, define a key file.
- Modify the application so that it does not use selection by correlation identifier: set the *CorrelId* field to MQCI_NONE and do not specify MQMO_MATCH_CORREL_ID in MQGMO.

2208 (08A0) (RC2208): MQRC_FILE_SYSTEM_ERROR

Explanation

An unexpected return code was received from the file system, in attempting to perform an operation on a queue.

This reason code occurs only on VSE/ESA.

Completion Code

MQCC_FAILED

Programmer response

Check the file system definition for the queue that was being accessed. For a VSAM file, check that the control interval is large enough for the maximum message length allowed for the queue.

2209 (08A1) (RC2209): MQRC_NO_MSG_LOCKED

Explanation

An MQGET call was issued with the MQGMO_UNLOCK option, but no message was currently locked.

Completion Code

MQCC_WARNING

Programmer response

Check that a message was locked by an earlier MQGET call with the MQGMO_LOCK option for the same handle, and that no intervening call has caused the message to become unlocked.

2210 (08A2) (RC2210): MQRC_SOAP_DOTNET_ERROR

Explanation

This exception has been received from an external .NET environment. For more information, see the inner exception that is contained within the received exception message.

Completion Code

MQCC_FAILED

Programmer response

Refer to the .NET documentation for information about the inner exception. Follow the corrective action recommended there.

2211 (08A3) (RC2211): MQRC_SOAP_AXIS_ERROR

Explanation

An exception from the Axis environment has been received and is included as a chained exception.

Completion Code

MQCC_FAILED

Programmer response

Refer to the Axis documentation for details about the chained exception. Follow the corrective action recommended there.

2212 (08A4) (RC2212): MQRC_SOAP_URL_ERROR

Explanation

The SOAP URL has been specified incorrectly.

Completion Code

MQCC_FAILED

Programmer response

Correct the SOAP URL and rerun.

2217 (08A9) (RC2217): MQRC_CONNECTION_NOT_AUTHORIZED

Explanation

This reason code occurs only on z/OS.

If the queue manager has been configured to use Advanced Message Security this reason code is returned if an error occurs in security processing.

This reason code might indicate a privacy security policy has been defined for the target queue that does not identify any recipients.

This reason code is also returned to CICS applications if the CICS subsystem is not authorized to connect to the queue manager.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the subsystem is authorized to connect to the queue manager.

2218 (08AA) (RC2218): MQRC_MSG_TOO_BIG_FOR_CHANNEL

Explanation

A message was put to a remote queue, but the message is larger than the maximum message length allowed by the channel. This reason code is returned in the *Feedback* field in the message descriptor of a report message.

Completion Code

MQCC_FAILED

Programmer response

Check the channel definitions. Increase the maximum message length that the channel can accept, or break the message into several smaller messages.

2219 (08AB) (RC2219): MQRC_CALL_IN_PROGRESS

Explanation

The application issued an MQI call whilst another MQI call was already being processed for that connection. Only one call per application connection can be processed at a time.

Concurrent calls can arise when an application uses multiple threads, or when an exit is invoked as part of the processing of an MQI call. For example, a data-conversion exit invoked as part of the processing of the MQGET call may try to issue an MQI call.

- On z/OS, concurrent calls can arise only with batch or IMS applications; an example is when a subtask ends while an MQI call is in progress (for example, an MQGET that is waiting), and there is an end-of-task exit routine that issues another MQI call.
- On Windows, concurrent calls can also arise if an MQI call is issued in response to a user message while another MQI call is in progress.
- If the application is using multiple threads with shared handles, MQRC_CALL_IN_PROGRESS occurs when the handle specified on the call is already in use by another thread and MQCNO_HANDLE_SHARE_NO_BLOCK was specified on the MQCONNX call.

Completion Code

MQCC_FAILED

Programmer response

Ensure that an MQI call cannot be issued while another one is active. Do not issue MQI calls from within a data-conversion exit.

- On z/OS, if you want to provide a subtask to allow an application that is waiting for a message to arrive to be canceled, wait for the message by using MQGET with MQGMO_SET_SIGNAL, rather than MQGMO_WAIT.

2220 (08AC) (RC2220): MQRC_RMH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQRMH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQRMH_STRUC_ID.
- The *Version* field is not MQRMH_VERSION_1.
- The *StrucLength* field specifies a value that is too small to include the structure plus the variable-length data at the end of the structure.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2222 (08AE) (RC2222): MQRC_Q_MGR_ACTIVE

Explanation

This condition is detected when a queue manager becomes active.

- On z/OS, this event is not generated for the first start of a queue manager, only on subsequent restarts.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Manager Active.

2223 (08AF) (RC2223): MQRC_Q_MGR_NOT_ACTIVE

Explanation

This condition is detected when a queue manager is requested to stop or quiesce.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Manager Not Active.

2224 (08B0) (RC2224): MQRC_Q_DEPTH_HIGH

Explanation

An MQPUT or MQPUT1 call has caused the queue depth to be incremented to, or greater than, the limit specified in the *QDepthHighLimit* attribute.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Depth High.

2225 (08B1) (RC2225): MQRC_Q_DEPTH_LOW

Explanation

An MQGET call has caused the queue depth to be decremented to, or less than, the limit specified in the *QDepthLowLimit* attribute.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Depth Low.

2226 (08B2) (RC2226): MQRC_Q_SERVICE_INTERVAL_HIGH

Explanation

No successful gets or puts have been detected within an interval that is greater than the limit specified in the *QServiceInterval* attribute.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Service Interval High.

2227 (08B3) (RC2227): MQRC_Q_SERVICE_INTERVAL_OK

Explanation

A successful get has been detected within an interval that is less than or equal to the limit specified in the *QServiceInterval* attribute.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Queue Service Interval OK.

2228 (08B4) (RC2228): MQRC_RFH_HEADER_FIELD_ERROR

Explanation

An expected RFH header field was not found or had an invalid value. If this error occurs in an IBM MQ SOAP listener, the missing or erroneous field is either the *contentType* field or the *transportVersion* field or both.

Completion Code

MQCC_FAILED

Programmer response

If this error occurs in an IBM MQ SOAP listener, and you are using the IBM-supplied sender, contact your IBM Support Center. If you are using a bespoke sender, check the associated error message, and that the RFH2 section of the SOAP/MQ request message contains all the mandatory fields, and that these fields have valid values.

2229 (08B5) (RC2229): MQRC_RAS_PROPERTY_ERROR

Explanation

There is an error related to the RAS property file. The file might be missing, it might be not accessible, or the commands in the file might be incorrect.

Completion Code

MQCC_FAILED

Programmer response

Look at the associated error message, which explains the error in detail. Correct the error and try again.

2232 (08B8) (RC2232): MQRC_UNIT_OF_WORK_NOT_STARTED

Explanation

An MQGET, MQPUT or MQPUT1 call was issued to get or put a message within a unit of work, but no TM/MP transaction had been started. If MQGMO_NO_SYNCPOINT is not specified on MQGET, or MQPMO_NO_SYNCPOINT is not specified on MQPUT or MQPUT1 (the default), the call requires a unit of work.

Completion Code

MQCC_FAILED

Programmer response

Ensure a TM/MP transaction is available, or issue the MQGET call with the MQGMO_NO_SYNCPOINT option, or the MQPUT or MQPUT1 call with the MQPMO_NO_SYNCPOINT option, which will cause a transaction to be started automatically.

2233 (08B9) (RC2233): MQRC_CHANNEL_AUTO_DEF_OK

Explanation

This condition is detected when the automatic definition of a channel is successful. The channel is defined by the MCA.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Auto-definition OK.

2234 (08BA) (RC2234): MQRC_CHANNEL_AUTO_DEF_ERROR

Explanation

This condition is detected when the automatic definition of a channel fails; this might be because an error occurred during the definition process, or because the channel automatic-definition exit inhibited the definition. Additional information is returned in the event message indicating the reason for the failure.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING

Programmer response

This reason code is only used to identify the corresponding event message Channel Auto-definition Error.

Examine the additional information returned in the event message to determine the reason for the failure.

2235 (08BB) (RC2235): MQRC_CFH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFH structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2236 (08BC) (RC2236): MQRC_CFIL_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFIL or MQRCFIL64 structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2237 (08BD) (RC2237): MQRC_CFIN_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFIN or MQCFIN64 structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2238 (08BE) (RC2238): MQRC_CFSL_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFSL structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2239 (08BF) (RC2239): MQRC_CFST_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFST structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2241 (08C1) (RC2241): MQRC_INCOMPLETE_GROUP

Explanation

An operation was attempted on a queue using a queue handle that had an incomplete message group. This reason code can arise in the following situations:

- On the MQPUT call, when the application specifies MQPMO_LOGICAL_ORDER and attempts to put a message that is not in a group. The completion code is MQCC_FAILED in this case.
- On the MQPUT call, when the application does *not* specify MQPMO_LOGICAL_ORDER, but the previous MQPUT call for the queue handle did specify MQPMO_LOGICAL_ORDER. The completion code is MQCC_WARNING in this case.
- On the MQGET call, when the application does *not* specify MQGMO_LOGICAL_ORDER, but the previous MQGET call for the queue handle did specify MQGMO_LOGICAL_ORDER. The completion code is MQCC_WARNING in this case.
- On the MQCLOSE call, when the application attempts to close the queue that has the incomplete message group. The completion code is MQCC_WARNING in this case.

If there is an incomplete logical message as well as an incomplete message group, reason code MQRC_INCOMPLETE_MSG is returned in preference to MQRC_INCOMPLETE_GROUP.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

If this reason code is expected, no corrective action is required. Otherwise, ensure that the MQPUT call for the last message in the group specifies MQMF_LAST_MSG_IN_GROUP.

2242 (08C2) (RC2242): MQRC_INCOMPLETE_MSG

Explanation

An operation was attempted on a queue using a queue handle that had an incomplete logical message. This reason code can arise in the following situations:

- On the MQPUT call, when the application specifies MQPMO_LOGICAL_ORDER and attempts to put a message that is not a segment, or that has a setting for the MQMF_LAST_MSG_IN_GROUP flag that is different from the previous message. The completion code is MQCC_FAILED in this case.
- On the MQPUT call, when the application does *not* specify MQPMO_LOGICAL_ORDER, but the previous MQPUT call for the queue handle did specify MQPMO_LOGICAL_ORDER. The completion code is MQCC_WARNING in this case.
- On the MQGET call, when the application does *not* specify MQGMO_LOGICAL_ORDER, but the previous MQGET call for the queue handle did specify MQGMO_LOGICAL_ORDER. The completion code is MQCC_WARNING in this case.
- On the MQCLOSE call, when the application attempts to close the queue that has the incomplete logical message. The completion code is MQCC_WARNING in this case.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

If this reason code is expected, no corrective action is required. Otherwise, ensure that the MQPUT call for the last segment specifies MQMF_LAST_SEGMENT.

2243 (08C3) (RC2243): MQRC_INCONSISTENT_CCSIDS

Explanation

An MQGET call was issued specifying the MQGMO_COMPLETE_MSG option, but the message to be retrieved consists of two or more segments that have differing values for the *CodedCharSetId* field in MQMD. This can arise when the segments take different paths through the network, and some of those paths have MCA sender conversion enabled. The call succeeds with a completion code of MQCC_WARNING, but only the first few segments that have identical character-set identifiers are returned.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING

Programmer response

Remove the MQGMO_COMPLETE_MSG option from the MQGET call and retrieve the remaining message segments one by one.

2244 (08C4) (RC2244): MQRC_INCONSISTENT_ENCODINGS

Explanation

An MQGET call was issued specifying the MQGMO_COMPLETE_MSG option, but the message to be retrieved consists of two or more segments that have differing values for the *Encoding* field in MQMD. This can arise when the segments take different paths through the network, and some of those paths have MCA sender conversion enabled. The call succeeds with a completion code of MQCC_WARNING, but only the first few segments that have identical encodings are returned.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING

Programmer response

Remove the MQGMO_COMPLETE_MSG option from the MQGET call and retrieve the remaining message segments one by one.

2245 (08C5) (RC2245): MQRC_INCONSISTENT_UOW

Explanation

One of the following applies:

- An MQPUT call was issued to put a message in a group or a segment of a logical message, but the value specified or defaulted for the MQPMO_SYNCPOINT option is not consistent with the current group and segment information retained by the queue manager for the queue handle.

  If the current call specifies MQPMO_LOGICAL_ORDER, the call fails. If the current call does not specify MQPMO_LOGICAL_ORDER, but the previous MQPUT call for the queue handle did, the call succeeds with completion code MQCC_WARNING.
- An MQGET call was issued to remove from the queue a message in a group or a segment of a logical message, but the value specified or defaulted for the MQGMO_SYNCPOINT option is not consistent with the current group and segment information retained by the queue manager for the queue handle.

  If the current call specifies MQGMO_LOGICAL_ORDER, the call fails. If the current call does not specify MQGMO_LOGICAL_ORDER, but the previous MQGET call for the queue handle did, the call succeeds with completion code MQCC_WARNING.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

Modify the application to ensure that the same unit-of-work specification is used for all messages in the group, or all segments of the logical message.

2246 (08C6) (RC2246): MQRC_INVALID_MSG_UNDER_CURSOR

Explanation

An MQGET call was issued specifying the MQGMO_COMPLETE_MSG option with either MQGMO_MSG_UNDER_CURSOR or MQGMO_BROWSE_MSG_UNDER_CURSOR, but the message that is under the cursor has an MQMD with an *Offset* field that is greater than zero. Because MQGMO_COMPLETE_MSG was specified, the message is not valid for retrieval.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Reposition the browse cursor so that it is located on a message with an *Offset* field in MQMD that is zero. Alternatively, remove the MQGMO_COMPLETE_MSG option.

2247 (08C7) (RC2247): MQRC_MATCH_OPTIONS_ERROR

Explanation

An MQGET call was issued, but the value of the *MatchOptions* field in the *GetMsgOpts* parameter is not valid, for one of the following reasons:

- An undefined option is specified.
- All of the following are true:
  - MQGMO_LOGICAL_ORDER is specified.
  - There is a current message group or logical message for the queue handle.
  - Neither MQGMO_BROWSE_MSG_UNDER_CURSOR nor MQGMO_MSG_UNDER_CURSOR is specified.
  - One or more of the MQMO_* options is specified.
  - The values of the fields in the *MsgDesc* parameter corresponding to the MQMO_* options specified, differ from the values of those fields in the MQMD for the message to be returned next.
- On z/OS, one or more of the options specified is not valid for the index type of the queue.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that only valid options are specified for the field.

2248 (08C8) (RC2248): MQRC_MDE_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQMDE structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQMDE_STRUC_ID.
- The *Version* field is not MQMDE_VERSION_2.
- The *StrucLength* field is not MQMDE_LENGTH_2.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2249 (08C9) (RC2249): MQRC_MSG_FLAGS_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the *MsgFlags* field in the message descriptor MQMD contains one or more message flags that are not recognized by the local queue manager. The message flags that cause this reason code to be returned depend on the destination of the message; see the description of REPORT in Report options and message flags for more information.

This reason code can also occur in the *Feedback* field in the MQMD of a report message, or in the *Reason* field in the MQDLH structure of a message on the dead-letter queue; in both cases it indicates that the destination queue manager does not support one or more of the message flags specified by the sender of the message.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Do the following:

- Ensure that the *MsgFlags* field in the message descriptor is initialized with a value when the message descriptor is declared, or is assigned a value prior to the MQPUT or MQPUT1 call. Specify MQMF_NONE if no message flags are needed.
- Ensure that the message flags specified are valid; see the *MsgFlags* field described in the description of MQMD in MsgFlags (MQLONG) for valid message flags.
- If multiple message flags are being set by adding the individual message flags together, ensure that the same message flag is not added twice.
- On z/OS, ensure that the message flags specified are valid for the index type of the queue; see the description of the *MsgFlags* field in MQMD for further details.

2250 (08CA) (RC2250): MQRC_MSG_SEQ_NUMBER_ERROR

Explanation

An MQGET, MQPUT, or MQPUT1 call was issued, but the value of the *MsgSeqNumber* field in the MQMD or MQMDE structure is less than one or greater than 999 999 999.

This error can also occur on the MQPUT call if the *MsgSeqNumber* field would have become greater than 999 999 999 as a result of the call.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify a value in the range 1 through 999 999 999. Do not attempt to create a message group containing more than 999 999 999 messages.

2251 (08CB) (RC2251): MQRC_OFFSET_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the value of the *Offset* field in the MQMD or MQMDE structure is less than zero or greater than 999 999 999.

This error can also occur on the MQPUT call if the *Offset* field would have become greater than 999 999 999 as a result of the call.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify a value in the range 0 through 999 999 999. Do not attempt to create a message segment that would extend beyond an offset of 999 999 999.

2252 (08CC) (RC2252): MQRC_ORIGINAL_LENGTH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued to put a report message that is a segment, but the *OriginalLength* field in the MQMD or MQMDE structure is either:

- Less than the length of data in the message, or
- Less than one (for a segment that is not the last segment), or
- Less than zero (for a segment that is the last segment)

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify a value that is greater than zero. Zero is valid only for the last segment.

2253 (08CD) (RC2253): MQRC_SEGMENT_LENGTH_ZERO

Explanation

An MQPUT or MQPUT1 call was issued to put the first or an intermediate segment of a logical message, but the length of the application message data in the segment (excluding any MQ headers that may be present) is zero. The length must be at least one for the first or intermediate segment.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check the application logic to ensure that segments are put with a length of one or greater. Only the last segment of a logical message is permitted to have a length of zero.

2255 (08CF) (RC2255): MQRC_UOW_NOT_AVAILABLE

Explanation

An MQGET, MQPUT, or MQPUT1 call was issued to get or put a message outside a unit of work, but the options specified on the call required the queue manager to process the call within a unit of work. Because there is already a user-defined unit of work in existence, the queue manager was unable to create a temporary unit of work for the duration of the call.

This reason occurs in the following circumstances:

- On an MQGET call, when the MQGMO_COMPLETE_MSG option is specified in MQGMO and the logical message to be retrieved is persistent and consists of two or more segments.
- On an MQPUT or MQPUT1 call, when the MQMF_SEGMENTATION_ALLOWED flag is specified in MQMD and the message requires segmentation.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Issue the MQGET, MQPUT, or MQPUT1 call inside the user-defined unit of work. Alternatively, for the MQPUT or MQPUT1 call, reduce the size of the message so that it does not require segmentation by the queue manager.

2256 (08D0) (RC2256): MQRC_WRONG_GMO_VERSION

Explanation

An MQGET call was issued specifying options that required an MQGMO with a version number not less than MQGMO_VERSION_2, but the MQGMO supplied did not satisfy this condition.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Modify the application to pass a version-2 MQGMO. Check the application logic to ensure that the *Version* field in MQGMO has been set to MQGMO_VERSION_2. Alternatively, remove the option that requires the version-2 MQGMO.

2257 (08D1) (RC2257): MQRC_WRONG_MD_VERSION

Explanation

An MQGET, MQPUT, or MQPUT1 call was issued specifying options that required an MQMD with a version number not less than MQMD_VERSION_2, but the MQMD supplied did not satisfy this condition.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Modify the application to pass a version-2 MQMD. Check the application logic to ensure that the *Version* field in MQMD has been set to MQMD_VERSION_2. Alternatively, remove the option that requires the version-2 MQMD.

2258 (08D2) (RC2258): MQRC_GROUP_ID_ERROR

Explanation

An MQPUT or MQPUT1 call was issued to put a distribution-list message that is also a message in a group, a message segment, or has segmentation allowed, but an invalid combination of options and values was specified. All of the following are true:

- MQPMO_LOGICAL_ORDER is not specified in the *Options* field in MQPMO.
- Either there are too few MQPMR records provided by MQPMO, or the *GroupId* field is not present in the MQPMR records.
- One or more of the following flags is specified in the *MsgFlags* field in MQMD or MQMDE:
  - MQMF_SEGMENTATION_ALLOWED
  - MQMF_*_MSG_IN_GROUP
  - MQMF_*_SEGMENT
- The *GroupId* field in MQMD or MQMDE is not MQGI_NONE.

This combination of options and values would result in the same group identifier being used for all of the destinations in the distribution list; this is not permitted by the queue manager.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify MQGI_NONE for the *GroupId* field in MQMD or MQMDE. Alternatively, if the call is MQPUT specify MQPMO_LOGICAL_ORDER in the *Options* field in MQPMO.

2259 (08D3) (RC2259): MQRC_INCONSISTENT_BROWSE

Explanation

An MQGET call was issued with the MQGMO_BROWSE_NEXT option specified, but the specification of the MQGMO_LOGICAL_ORDER option for the call is different from the specification of that option for the previous call for the queue handle. Either both calls must specify MQGMO_LOGICAL_ORDER, or neither call must specify MQGMO_LOGICAL_ORDER.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Add or remove the MQGMO_LOGICAL_ORDER option as appropriate. Alternatively, to switch between logical order and physical order, specify the MQGMO_BROWSE_FIRST option to restart the scan from the beginning of the queue, omitting or specifying MQGMO_LOGICAL_ORDER as required.

2260 (08D4) (RC2260): MQRC_XQH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQXQH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQXQH_STRUC_ID.
- The *Version* field is not MQXQH_VERSION_1.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2261 (08D5) (RC2261): MQRC_SRC_ENV_ERROR

Explanation

This reason occurs when a channel exit that processes reference messages detects an error in the source environment data of a reference message header (MQRMH). One of the following is true:

- *SrcEnvLength* is less than zero.
- *SrcEnvLength* is greater than zero, but there is no source environment data.
- *SrcEnvLength* is greater than zero, but *SrcEnvOffset* is negative, zero, or less than the length of the fixed part of MQRMH.
- *SrcEnvLength* is greater than zero, but *SrcEnvOffset* plus *SrcEnvLength* is greater than *StrucLength*.

The exit returns this reason in the *Feedback* field of the MQCXP structure. If an exception report is requested, it is copied to the *Feedback* field of the MQMD associated with the report.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify the source environment data correctly.

2262 (08D6) (RC2262): MQRC_SRC_NAME_ERROR

Explanation

This reason occurs when a channel exit that processes reference messages detects an error in the source name data of a reference message header (MQRMH). One of the following is true:

- *SrcNameLength* is less than zero.
- *SrcNameLength* is greater than zero, but there is no source name data.
- *SrcNameLength* is greater than zero, but *SrcNameOffset* is negative, zero, or less than the length of the fixed part of MQRMH.
- *SrcNameLength* is greater than zero, but *SrcNameOffset* plus *SrcNameLength* is greater than *StrucLength*.

The exit returns this reason in the *Feedback* field of the MQCXP structure. If an exception report is requested, it is copied to the *Feedback* field of the MQMD associated with the report.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify the source name data correctly.

2263 (08D7) (RC2263): MQRC_DEST_ENV_ERROR

Explanation

This reason occurs when a channel exit that processes reference messages detects an error in the destination environment data of a reference message header (MQRMH). One of the following is true:

- *DestEnvLength* is less than zero.
- *DestEnvLength* is greater than zero, but there is no destination environment data.
- *DestEnvLength* is greater than zero, but *DestEnvOffset* is negative, zero, or less than the length of the fixed part of MQRMH.
- *DestEnvLength* is greater than zero, but *DestEnvOffset* plus *DestEnvLength* is greater than *StrucLength*.

The exit returns this reason in the *Feedback* field of the MQCXP structure. If an exception report is requested, it is copied to the *Feedback* field of the MQMD associated with the report.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify the destination environment data correctly.

2264 (08D8) (RC2264): MQRC_DEST_NAME_ERROR

Explanation

This reason occurs when a channel exit that processes reference messages detects an error in the destination name data of a reference message header (MQRMH). One of the following is true:

- *DestNameLength* is less than zero.
- *DestNameLength* is greater than zero, but there is no destination name data.
- *DestNameLength* is greater than zero, but *DestNameOffset* is negative, zero, or less than the length of the fixed part of MQRMH.
- *DestNameLength* is greater than zero, but *DestNameOffset* plus *DestNameLength* is greater than *StrucLength*.

The exit returns this reason in the *Feedback* field of the MQCXP structure. If an exception report is requested, it is copied to the *Feedback* field of the MQMD associated with the report.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Specify the destination name data correctly.

2265 (08D9) (RC2265): MQRC_TM_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQTM structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQTM_STRUC_ID.
- The *Version* field is not MQTM_VERSION_1.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2266 (08DA) (RC2266): MQRC_CLUSTER_EXIT_ERROR

Explanation

An MQOPEN, MQPUT, or MQPUT1 call was issued to open or put a message on a cluster queue, but the cluster workload exit defined by the queue-manager's *ClusterWorkloadExit* attribute failed unexpectedly or did not respond in time. Subsequent MQOPEN, MQPUT, and MQPUT1 calls for this queue handle are processed as though the *ClusterWorkloadExit* attribute were blank.

- On z/OS, a message giving more information about the error is written to the system log, for example message CSQV455E or CSQV456E.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check the cluster workload exit to ensure that it has been written correctly.

2267 (08DB) (RC2267): MQRC_CLUSTER_EXIT_LOAD_ERROR

Explanation

An MQCONN or MQCONNX call was issued to connect to a queue manager, but the queue manager was unable to load the cluster workload exit. Execution continues without the cluster workload exit.

- On z/OS, if the cluster workload exit cannot be loaded, a message is written to the system log, for example message CSQV453I. Processing continues as though the *ClusterWorkloadExit* attribute had been blank.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_WARNING

Programmer response

Ensure that the queue-manager's *ClusterWorkloadExit* attribute has the correct value, and that the exit has been installed into the correct location.

2268 (08DC) (RC2268): MQRC_CLUSTER_PUT_INHIBITED

Explanation

An MQOPEN call with the MQOO_OUTPUT and MQOO_BIND_ON_OPEN options in effect was issued for a cluster queue, but the call failed because all of the following are true:

- All instances of the cluster queue are currently put-inhibited (that is, all of the queue instances have the *InhibitPut* attribute set to MQQA_PUT_INHIBITED).

- There is no local instance of the queue. (If there is a local instance, the MQOPEN call succeeds, even if the local instance is put-inhibited.)
- There is no cluster workload exit for the queue, or there is a cluster workload exit but it did not choose a queue instance. (If the cluster workload exit does choose a queue instance, the MQOPEN call succeeds, even if that instance is put-inhibited.)

If the MQOO_BIND_NOT_FIXED option is specified on the MQOPEN call, the call can succeed even if all of the queues in the cluster are put-inhibited. However, a subsequent MQPUT call may fail if all of the queues are still put-inhibited at the time of the MQPUT call.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

If the system design allows put requests to be inhibited for short periods, retry the operation later. If the problem persists, determine why all of the queues in the cluster are put-inhibited.

2269 (08DD) (RC2269): MQRC_CLUSTER_RESOURCE_ERROR

Explanation

An MQOPEN, MQPUT, or MQPUT1 call was issued for a cluster queue, but an error occurred whilst trying to use a resource required for clustering.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Do the following:

- Check that the SYSTEM.CLUSTER.* queues are not put inhibited or full.
- Check the event queues for any events relating to the SYSTEM.CLUSTER.* queues, as these may give guidance as to the nature of the failure.
- Check that the repository queue manager is available.
- On z/OS, check the console for signs of the failure, such as full page sets.

2270 (08DE) (RC2270): MQRC_NO_DESTINATIONS_AVAILABLE

Explanation

An MQPUT or MQPUT1 call was issued to put a message on a cluster queue, but at the time of the call there were no longer any instances of the queue in the cluster. The message therefore could not be sent.

This situation can occur when MQOO_BIND_NOT_FIXED is specified on the MQOPEN call that opens the queue, or MQPUT1 is used to put the message.

This reason code can also occur when running the REFRESH CLUSTER command. See "Application issues seen when running REFRESH CLUSTER" on page 475

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check the queue definition and queue status to determine why all instances of the queue were removed from the cluster. Correct the problem and rerun the application.

2271 (08DF) (RC2271): MQRC_CONN_TAG_IN_USE

Explanation

An MQCONNX call was issued specifying one of the MQCNO_*_CONN_TAG_* options, but the call failed because the connection tag specified by *ConnTag* in MQCNO is in use by an active process or thread, or there is an unresolved unit of work that references this connection tag.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The problem is likely to be transitory. The application should wait a short while and then retry the operation.

2272 (08E0) (RC2272): MQRC_PARTIALLY_CONVERTED

Explanation

On an MQGET call with the MQGMO_CONVERT option included in the *GetMsgOpts* parameter, one or more MQ header structures in the message data could not be converted to the specified target character set or encoding. In this situation, the MQ header structures are converted to the queue-manager's character set and encoding, and the application data in the message is converted to the target character set and encoding. On return from the call, the values returned in the various *CodedCharSetId* and *Encoding* fields in the *MsgDesc* parameter and MQ header structures indicate the character set and encoding that apply to each part of the message. The call completes with MQCC_WARNING.

This reason code usually occurs when the specified target character set is one that causes the character strings in the MQ header structures to expand beyond the lengths of their fields. Unicode character set UCS-2 is an example of a character set that causes this to happen.

Completion Code

MQCC_FAILED

Programmer response

If this is an expected situation, no corrective action is required.

If this is an unexpected situation, check that the MQ header structures contain valid data. If they do, specify as the target character set a character set that does not cause the strings to expand.

2273 (08E1) (RC2273): MQRC_CONNECTION_ERROR

Explanation

An MQCONN or MQCONNX call failed for one of the following reasons:

- The installation and customization options chosen for IBM MQ do not allow connection by the type of application being used.
- The system parameter module is not at the same release level as the queue manager.
- The channel initiator is not at the same release level as the queue manager.
- An internal error was detected by the queue manager.

Completion Code

MQCC_FAILED

Programmer response

None, if the installation and customization options chosen for IBM MQ do not allow all functions to be used.

Otherwise, if this occurs while starting the channel initiator, ensure that the queue manager and the channel initiator are both at the same release level and that their started task JCL procedures both specify the same level of IBM MQ program libraries; if this occurs while starting the queue manager, relinkedit the system parameter module (CSQZPARM) to ensure that it is at the correct level. If the problem persists, contact your IBM support center.

2274 (08E2) (RC2274): MQRC_OPTION_ENVIRONMENT_ERROR

Explanation

An MQGET call with the MQGMO_MARK_SKIP_BACKOUT option specified was issued from a Db2 Stored Procedure. The call failed because the MQGMO_MARK_SKIP_BACKOUT option cannot be used from a Db2 Stored Procedure.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Remove the MQGMO_MARK_SKIP_BACKOUT option from the MQGET call.

2277 (08E5) (RC2277): MQRC_CD_ERROR

Explanation

An MQCONNX call was issued to connect to a queue manager, but the MQCD channel definition structure addressed by the *ClientConnOffset* or *ClientConnPtr* field in MQCNO contains data that is not valid. Consult the error log for more information about the nature of the error.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Ensure that input fields in the MQCD structure are set correctly.

2278 (08E6) (RC2278): MQRC_CLIENT_CONN_ERROR

Explanation

An MQCONNX call was issued to connect to a queue manager, but the MQCD channel definition structure is not specified correctly. One of the following applies:

- *ClientConnOffset* is not zero and *ClientConnPtr* is not zero and not the null pointer.
- *ClientConnPtr* is not a valid pointer.
- *ClientConnPtr* or *ClientConnOffset* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems. It also occurs in Java applications when a client channel definition table (CCDT) is specified to determine the name of the channel, but the table itself cannot be found.

Completion Code

MQCC_FAILED

Programmer response

Ensure that at least one of *ClientConnOffset* and *ClientConnPtr* is zero. Ensure that the field used points to accessible storage. Ensure that the URL of the client channel definition table is correct.

2279 (08E7) (RC2279): MQRC_CHANNEL_STOPPED_BY_USER

Explanation

This condition is detected when the channel has been stopped by an operator. The reason qualifier identifies the reasons for stopping.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Stopped By User.

2280 (08E8) (RC2280): MQRC_HCONFIG_ERROR

Explanation

The configuration handle *Hconfig* specified on the MQXEP call or MQZEP call is not valid. The MQXEP call is issued by an API exit function; the MQZEP call is issued by an installable service.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Specify the configuration handle that was provided by the queue manager:

- On the MQXEP call, use the handle passed in the *Hconfig* field of the MQAXP structure.
- On the MQZEP call, use the handle passed to the installable service's configuration function on the component initialization call. For more information about installable services, see Installable services and components for UNIX, Linux and Windows .

2281 (08E9) (RC2281): MQRC_FUNCTION_ERROR

Explanation

An MQXEP or MQZEP call was issued, but the function identifier *Function* specified on the call is not valid, or not supported by the installable service being configured.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Do the following:

- For the MQXEP call, specify one of the MQXF_* values.
- For the MQZEP call, specify an MQZID_* value that is valid for the installable service being configured. See MQZEP to determine which values are valid.

2282 (08EA) (RC2282): MQRC_CHANNEL_STARTED

Explanation

One of the following has occurred:

- An operator has issued a Start Channel command.
- An instance of a channel has been successfully established. This condition is detected when Initial Data negotiation is complete and resynchronization has been performed where necessary such that message transfer can proceed.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Started.

2283 (08EB) (RC2283): MQRC_CHANNEL_STOPPED

Explanation

This condition is detected when the channel has been stopped. The reason qualifier identifies the reasons for stopping.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Stopped.

2284 (08EC) (RC2284): MQRC_CHANNEL_CONV_ERROR

Explanation

This condition is detected when a channel is unable to do data conversion and the MQGET call to get a message from the transmission queue resulted in a data conversion error. The conversion reason code identifies the reason for the failure.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Conversion Error.

2285 (08ED) (RC2285): MQRC_SERVICE_NOT_AVAILABLE

Explanation

This reason should be returned by an installable service component when the requested action cannot be performed because the required underlying service is not available.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Make the underlying service available.

2286 (08EE) (RC2286): MQRC_INITIALIZATION_FAILED

Explanation

This reason should be returned by an installable service component when the component is unable to complete initialization successfully.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Correct the error and retry the operation.

2287 (08EF) (RC2287): MQRC_TERMINATION_FAILED

Explanation

This reason should be returned by an installable service component when the component is unable to complete termination successfully.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Correct the error and retry the operation.

2288 (08F0) (RC2288): MQRC_UNKNOWN_Q_NAME

Explanation

This reason should be returned by the MQZ_LOOKUP_NAME installable service component when the name specified for the *QName* parameter is not recognized.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

None. See Installable services and components for UNIX, Linux and Windows for more information about installable services.

2289 (08F1) (RC2289): MQRC_SERVICE_ERROR

Explanation

This reason should be returned by an installable service component when the component encounters an unexpected error.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Correct the error and retry the operation.

2290 (08F2) (RC2290): MQRC_Q_ALREADY_EXISTS

Explanation

This reason should be returned by the MQZ_INSERT_NAME installable service component when the queue specified by the *QName* parameter is already defined to the name service.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

None. See Installable services and components for UNIX, Linux and Windows for more information about installable services.

2291 (08F3) (RC2291): MQRC_USER_ID_NOT_AVAILABLE

Explanation

This reason should be returned by the MQZ_FIND_USERID installable service component when the user ID cannot be determined.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

None. See Installable services and components for UNIX, Linux and Windows for more information about installable services.

2292 (08F4) (RC2292): MQRC_UNKNOWN_ENTITY

Explanation

This reason should be returned by the authority installable service component when the name specified by the *EntityName* parameter is not recognized.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the entity is defined.

2294 (08F6) (RC2294): MQRC_UNKNOWN_REF_OBJECT

Explanation

This reason should be returned by the MQZ_COPY_ALL_AUTHORITY installable service component when the name specified by the *RefObjectName* parameter is not recognized.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the reference object is defined. See Installable services and components for UNIX, Linux and Windows for more information about installable services.

2295 (08F7) (RC2295): MQRC_CHANNEL_ACTIVATED

Explanation

This condition is detected when a channel that has been waiting to become active, and for which a Channel Not Activated event has been generated, is now able to become active because an active slot has been released by another channel.

This event is not generated for a channel that is able to become active without waiting for an active slot to be released.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Activated.

2296 (08F8) (RC2296): MQRC_CHANNEL_NOT_ACTIVATED

Explanation

This condition is detected when a channel is required to become active, either because it is starting or because it is about to make another attempt to establish connection with its partner. However, it is unable to do so because the limit on the number of active channels has been reached.

- On z/OS, the maximum number of active channels is given by the ACTCHL queue manager attribute.
- In other environments, the maximum number of active channels is given by the MaxActiveChannels parameter in the qm.ini file.

The channel waits until it is able to take over an active slot released when another channel ceases to be active. At that time a Channel Activated event is generated.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Not Activated.

2297 (08F9) (RC2297): MQRC_UOW_CANCELED

Explanation

An MQI call was issued, but the unit of work (TM/MP transaction) being used for the MQ operation had been canceled. This may have been done by TM/MP itself (for example, due to the transaction running for too long, or exceeding audit trail sizes), or by the application program issuing an ABORT_TRANSACTION. All updates performed to resources owned by the queue manager are backed out.

Completion Code

MQCC_FAILED

Programmer response

Refer to the operating system's *Transaction Management Operations Guide* to determine how the Transaction Manager can be tuned to avoid the problem of system limits being exceeded.

2298 (08FA) (RC2298): MQRC_FUNCTION_NOT_SUPPORTED

Explanation

The function requested is not available in the current environment.

Completion Code

MQCC_FAILED

Programmer response

Remove the call from the application.

This reason code can be used when the call requires resources or functionality that is restricted by the queue manager OPMODE setting.

If you get this reason code with CICS group connect, check that the queue manager attribute GROUPUR is enabled.

2299 (08FB) (RC2299): MQRC_SELECTOR_TYPE_ERROR

Explanation

The *Selector* parameter has the wrong data type; it must be of type Long.

Completion Code

MQCC_FAILED

Programmer response

Declare the *Selector* parameter as Long.

2300 (08FC) (RC2300): MQRC_COMMAND_TYPE_ERROR

Explanation

The mqExecute call was issued, but the value of the MQIASY_TYPE data item in the administration bag is not MQCFT_COMMAND.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the MQIASY_TYPE data item in the administration bag has the value MQCFT_COMMAND.

2301 (08FD) (RC2301): MQRC_MULTIPLE_INSTANCE_ERROR

Explanation

The *Selector* parameter specifies a system selector (one of the MQIASY_* values), but the value of the *ItemIndex* parameter is not MQIND_NONE. Only one instance of each system selector can exist in the bag.

Completion Code

MQCC_FAILED

Programmer response

Specify MQIND_NONE for the *ItemIndex* parameter.

2302 (08FE) (RC2302): MQRC_SYSTEM_ITEM_NOT_ALTERABLE

Explanation

A call was issued to modify the value of a system data item in a bag (a data item with one of the MQIASY_* selectors), but the call failed because the data item is one that cannot be altered by the application.

Completion Code

MQCC_FAILED

Programmer response

Specify the selector of a user-defined data item, or remove the call.

2303 (08FF) (RC2303): MQRC_BAG_CONVERSION_ERROR

Explanation

The mqBufferToBag or mqGetBag call was issued, but the data in the buffer or message could not be converted into a bag. This occurs when the data to be converted is not valid PCF.

Completion Code

MQCC_FAILED

Programmer response

Check the logic of the application that created the buffer or message to ensure that the buffer or message contains valid PCF.

If the message contains PCF that is not valid, the message cannot be retrieved using the mqGetBag call:

- If one of the MQGMO_BROWSE_* options was specified, the message remains on the queue and can be retrieved using the MQGET call.
- In other cases, the message has already been removed from the queue and discarded. If the message was retrieved within a unit of work, the unit of work can be backed out and the message retrieved using the MQGET call.

2304 (0900) (RC2304): MQRC_SELECTOR_OUT_OF_RANGE

Explanation

The `Selector` parameter has a value that is outside the valid range for the call. If the bag was created with the MQCBO_CHECK_SELECTORS option:

- For the mqAddInteger call, the value must be within the range MQIA_FIRST through MQIA_LAST.
- For the mqAddString call, the value must be within the range MQCA_FIRST through MQCA_LAST.

If the bag was not created with the MQCBO_CHECK_SELECTORS option:

- The value must be zero or greater.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value.

2305 (0901) (RC2305): MQRC_SELECTOR_NOT_UNIQUE

Explanation

The `ItemIndex` parameter has the value MQIND_NONE, but the bag contains more than one data item with the selector value specified by the `Selector` parameter. MQIND_NONE requires that the bag contain only one occurrence of the specified selector.

This reason code also occurs on the mqExecute call when the administration bag contains two or more occurrences of a selector for a required parameter that permits only one occurrence.

Completion Code

MQCC_FAILED

Programmer response

Check the logic of the application that created the bag. If correct, specify for `ItemIndex` a value that is zero or greater, and add application logic to process all of the occurrences of the selector in the bag.

Review the description of the administration command being issued, and ensure that all required parameters are defined correctly in the bag.

2306 (0902) (RC2306): MQRC_INDEX_NOT_PRESENT

Explanation

The specified index is not present:

- For a bag, this means that the bag contains one or more data items that have the selector value specified by the `Selector` parameter, but none of them has the index value specified by the `ItemIndex` parameter. The data item identified by the `Selector` and `ItemIndex` parameters must exist in the bag.
- For a namelist, this means that the index parameter value is too large, and outside the range of valid values.

Completion Code

MQCC_FAILED

Programmer response

Specify the index of a data item that does exist in the bag or namelist. Use the mqCountItems call to determine the number of data items with the specified selector that exist in the bag, or the nameCount method to determine the number of names in the namelist.

2307 (0903) (RC2307): MQRC_STRING_ERROR

Explanation

The *String* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2308 (0904) (RC2308): MQRC_ENCODING_NOT_SUPPORTED

Explanation

The *Encoding* field in the message descriptor MQMD contains a value that is not supported:

- For the mqPutBag call, the field in error resides in the *MsgDesc* parameter of the call.
- For the mqGetBag call, the field in error resides in:
  - The *MsgDesc* parameter of the call if the MQGMO_CONVERT option was specified.
  - The message descriptor of the message about to be retrieved if MQGMO_CONVERT was *not* specified.

Completion Code

MQCC_FAILED

Programmer response

The value must be MQENC_NATIVE.

If the value of the *Encoding* field in the message is not valid, the message cannot be retrieved using the mqGetBag call:

- If one of the MQGMO_BROWSE_* options was specified, the message remains on the queue and can be retrieved using the MQGET call.
- In other cases, the message has already been removed from the queue and discarded. If the message was retrieved within a unit of work, the unit of work can be backed out and the message retrieved using the MQGET call.

2309 (0905) (RC2309): MQRC_SELECTOR_NOT_PRESENT

Explanation

The *Selector* parameter specifies a selector that does not exist in the bag.

Completion Code

MQCC_FAILED

Programmer response

Specify a selector that does exist in the bag.

2310 (0906) (RC2310): MQRC_OUT_SELECTOR_ERROR

Explanation

The *OutSelector* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2311 (0907) (RC2311): MQRC_STRING_TRUNCATED

Explanation

The string returned by the call is too long to fit in the buffer provided. The string has been truncated to fit in the buffer.

Completion Code

MQCC_FAILED

Programmer response

If the entire string is required, provide a larger buffer. On the mqInquireString call, the *StringLength* parameter is set by the call to indicate the size of the buffer required to accommodate the string without truncation.

2312 (0908) (RC2312): MQRC_SELECTOR_WRONG_TYPE

Explanation

A data item with the specified selector exists in the bag, but has a data type that conflicts with the data type implied by the call being used. For example, the data item might have an integer data type, but the call being used might be mqSetString, which implies a character data type.

This reason code also occurs on the mqBagToBuffer, mqExecute, and mqPutBag calls when mqAddString or mqSetString was used to add the MQIACF_INQUIRY data item to the bag.

Completion Code

MQCC_FAILED

Programmer response

For the mqSetInteger and mqSetString calls, specify MQIND_ALL for the `ItemIndex` parameter to delete from the bag all existing occurrences of the specified selector before creating the new occurrence with the required data type.

For the mqInquireBag, mqInquireInteger, and mqInquireString calls, use the mqInquireItemInfo call to determine the data type of the item with the specified selector, and then use the appropriate call to determine the value of the data item.

For the mqBagToBuffer, mqExecute, and mqPutBag calls, ensure that the MQIACF_INQUIRY data item is added to the bag using the mqAddInteger or mqSetInteger calls.

2313 (0909) (RC2313): MQRC_INCONSISTENT_ITEM_TYPE

Explanation

The mqAddInteger or mqAddString call was issued to add another occurrence of the specified selector to the bag, but the data type of this occurrence differed from the data type of the first occurrence.

This reason can also occur on the mqBufferToBag and mqGetBag calls, where it indicates that the PCF in the buffer or message contains a selector that occurs more than once but with inconsistent data types.

Completion Code

MQCC_FAILED

Programmer response

For the mqAddInteger and mqAddString calls, use the call appropriate to the data type of the first occurrence of that selector in the bag.

For the mqBufferToBag and mqGetBag calls, check the logic of the application that created the buffer or sent the message to ensure that multiple-occurrence selectors occur with only one data type. A message that contains a mixture of data types for a selector cannot be retrieved using the mqGetBag call:

- If one of the MQGMO_BROWSE_* options was specified, the message remains on the queue and can be retrieved using the MQGET call.
- In other cases, the message has already been removed from the queue and discarded. If the message was retrieved within a unit of work, the unit of work can be backed out and the message retrieved using the MQGET call.

2314 (090A) (RC2314): MQRC_INDEX_ERROR

Explanation

An index parameter to a call or method has a value that is not valid. The value must be zero or greater. For bag calls, certain MQIND_* values can also be specified:

- For the mqDeleteItem, mqSetInteger and mqSetString calls, MQIND_ALL and MQIND_NONE are valid.
- For the mqInquireBag, mqInquireInteger, mqInquireString, and mqInquireItemInfo calls, MQIND_NONE is valid.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value.

2315 (090B) (RC2315): MQRC_SYSTEM_BAG_NOT_ALTERABLE

Explanation

A call was issued to add a data item to a bag, modify the value of an existing data item in a bag, or retrieve a message into a bag, but the call failed because the bag is one that had been created by the system as a result of a previous mqExecute call. System bags cannot be modified by the application.

Completion Code

MQCC_FAILED

Programmer response

Specify the handle of a bag created by the application, or remove the call.

2316 (090C) (RC2316): MQRC_ITEM_COUNT_ERROR

Explanation

The mqTruncateBag call was issued, but the *ItemCount* parameter specifies a value that is not valid. The value is either less than zero, or greater than the number of user-defined data items in the bag.

This reason also occurs on the mqCountItems call if the parameter pointer is not valid, or points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value. Use the mqCountItems call to determine the number of user-defined data items in the bag.

2317 (090D) (RC2317): MQRC_FORMAT_NOT_SUPPORTED

Explanation

The *Format* field in the message descriptor MQMD contains a value that is not supported:

- In an administration message, the format value must be one of the following: MQFMT_ADMIN, MQFMT_EVENT, MQFMT_PCF. For the mqPutBag call, the field in error resides in the *MsgDesc* parameter of the call. For the mqGetBag call, the field in error resides in the message descriptor of the message about to be retrieved.
- On z/OS, the message was put to the command input queue with a format value of MQFMT_ADMIN, but the version of MQ being used does not support that format for commands.

Completion Code

MQCC_FAILED

Programmer response

If the error occurred when putting a message, correct the format value.

If the error occurred when getting a message, the message cannot be retrieved using the mqGetBag call:

- If one of the MQGMO_BROWSE_* options was specified, the message remains on the queue and can be retrieved using the MQGET call.
- In other cases, the message has already been removed from the queue and discarded. If the message was retrieved within a unit of work, the unit of work can be backed out and the message retrieved using the MQGET call.

2318 (090E) (RC2318): MQRC_SELECTOR_NOT_SUPPORTED

Explanation

The *Selector* parameter specifies a value that is a system selector (a value that is negative), but the system selector is not one that is supported by the call.

Completion Code

MQCC_FAILED

Programmer response

Specify a selector value that is supported.

2319 (090F) (RC2319): MQRC_ITEM_VALUE_ERROR

Explanation

The mqInquireBag or mqInquireInteger call was issued, but the *ItemValue* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2320 (0910) (RC2320): MQRC_HBAG_ERROR

Explanation

A call was issued that has a parameter that is a bag handle, but the handle is not valid. For output parameters, this reason also occurs if the parameter pointer is not valid, or points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2321 (0911) (RC2321): MQRC_PARAMETER_MISSING

Explanation

An administration message requires a parameter that is not present in the administration bag. This reason code occurs only for bags created with the MQCBO_ADMIN_BAG or MQCBO_REORDER_AS_REQUIRED options.

Completion Code

MQCC_FAILED

Programmer response

Review the description of the administration command being issued, and ensure that all required parameters are present in the bag.

2322 (0912) (RC2322): MQRC_CMD_SERVER_NOT_AVAILABLE

Explanation

The command server that processes administration commands is not available.

Completion Code

MQCC_FAILED

Programmer response

Start the command server.

2323 (0913) (RC2323): MQRC_STRING_LENGTH_ERROR

Explanation

The *StringLength* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2324 (0914) (RC2324): MQRC_INQUIRY_COMMAND_ERROR

Explanation

The mqAddInquiry call was used previously to add attribute selectors to the bag, but the command code to be used for the mqBagToBuffer, mqExecute, or mqPutBag call is not recognized. As a result, the correct PCF message cannot be generated.

Completion Code

MQCC_FAILED

Programmer response

Remove the mqAddInquiry calls and use instead the mqAddInteger call with the appropriate MQIACF_*_ATTRS or MQIACH_*_ATTRS selectors.

2325 (0915) (RC2325): MQRC_NESTED_BAG_NOT_SUPPORTED

Explanation

A bag that is input to the call contains nested bags. Nested bags are supported only for bags that are output from the call.

Completion Code

MQCC_FAILED

Programmer response

Use a different bag as input to the call.

2326 (0916) (RC2326): MQRC_BAG_WRONG_TYPE

Explanation

The *Bag* parameter specifies the handle of a bag that has the wrong type for the call. The bag must be an administration bag, that is, it must be created with the MQCBO_ADMIN_BAG option specified on the mqCreateBag call.

Completion Code

MQCC_FAILED

Programmer response

Specify the MQCBO_ADMIN_BAG option when the bag is created.

2327 (0917) (RC2327): MQRC_ITEM_TYPE_ERROR

Explanation

The mqInquireItemInfo call was issued, but the *ItemType* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2328 (0918) (RC2328): MQRC_SYSTEM_BAG_NOT_DELETABLE

Explanation

An mqDeleteBag call was issued to delete a bag, but the call failed because the bag is one that had been created by the system as a result of a previous mqExecute call. System bags cannot be deleted by the application.

Completion Code

MQCC_FAILED

Programmer response

Specify the handle of a bag created by the application, or remove the call.

2329 (0919) (RC2329): MQRC_SYSTEM_ITEM_NOT_DELETABLE

Explanation

A call was issued to delete a system data item from a bag (a data item with one of the MQIASY_* selectors), but the call failed because the data item is one that cannot be deleted by the application.

Completion Code

MQCC_FAILED

Programmer response

Specify the selector of a user-defined data item, or remove the call.

2330 (091A) (RC2330): MQRC_CODED_CHAR_SET_ID_ERROR

Explanation

The *CodedCharSetId* parameter is not valid. Either the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2331 (091B) (RC2331): MQRC_MSG_TOKEN_ERROR

Explanation

An MQGET call was issued to retrieve a message using the message token as a selection criterion, but the options specified are not valid, because MQMO_MATCH_MSG_TOKEN was specified with either MQGMO_WAIT or MQGMO_SET_SIGNAL.

An Async Consumer was registered to retrieve a message using the message token as a selection criterion, but when the delivery of messages for this consumer was started no message matching the message token was available for delivery to the consumer. As a result the consumer is suspended.

Completion Code

MQCC_FAILED

Programmer response

If this reason code is returned from an MQGET call, either remove the MQMO_MATCH_MSG_TOKEN match option, or remove the MQGMO_WAIT, or MQGMO_SET_SIGNAL option which was specified.

If this reason code is returned to an Async Consume Event Handler, then the consumer has been suspended and no further messages will be delivered to the consumer. The consumer should be de-registered or modified to select a different message using the MQCB call.

2332 (091C) (RC2332): MQRC_MISSING_WIH

Explanation

An MQPUT or MQPUT1 call was issued to put a message on a queue with an *IndexType* attribute that had the value MQIT_MSG_TOKEN, but the *Format* field in the MQMD was not MQFMT_WORK_INFO_HEADER. This error occurs only when the message arrives at the destination queue manager.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Modify the application to ensure that it places an MQWIH structure at the start of the message data, and sets the *Format* field in the MQMD to MQFMT_WORK_INFO_HEADER. Alternatively, change the *ApplType* attribute of the process definition used by the destination queue to be MQAT_WLM, and specify the required service name and service step name in its *EnvData* attribute.

2333 (091D) (RC2333): MQRC_WIH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQWIH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQWIH_STRUC_ID.
- The *Version* field is not MQWIH_VERSION_1.
- The *StrucLength* field is not MQWIH_LENGTH_1.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).
- On z/OS, this error also occurs when the *IndexType* attribute of the queue is MQIT_MSG_TOKEN, but the message data does not begin with an MQWIH structure.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

- On z/OS, if the queue has an *IndexType* of MQIT_MSG_TOKEN, ensure that the message data begins with an MQWIH structure.

2334 (091E) (RC2334): MQRC_RFH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQRFH or MQRFH2 structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQRFH_STRUC_ID.
- The *Version* field is not MQRFH_VERSION_1 (MQRFH), or MQRFH_VERSION_2 (MQRFH2).
- The *StrucLength* field specifies a value that is too small to include the structure plus the variable-length data at the end of the structure.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure (the structure extends beyond the end of the message).

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value (note: MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are *not* valid in this field).

2335 (091F) (RC2335): MQRC_RFH_STRING_ERROR

Explanation

The contents of the *NameValueString* field in the MQRFH structure are not valid. *NameValueString* must adhere to the following rules:

- The string must consist of zero or more name/value pairs separated from each other by one or more blanks; the blanks are not significant.
- If a name or value contains blanks that are significant, the name or value must be enclosed in double quotation marks.
- If a name or value itself contains one or more double quotation marks, the name or value must be enclosed in double quotation marks, and each embedded double quotation mark must be doubled.
- A name or value can contain any characters other than the null, which acts as a delimiter. The null and characters following it, up to the defined length of *NameValueString*, are ignored.

The following is a valid *NameValueString*:

```
Famous_Words "The program displayed ""Hello World"""
```

Completion Code

MQCC_FAILED

Programmer response

Modify the application that generated the message to ensure that it places in the *NameValueString* field data that adheres to the rules. Check that the *StrucLength* field is set to the correct value.

2336 (0920) (RC2336): MQRC_RFH_COMMAND_ERROR

Explanation

The message contains an MQRFH structure, but the command name contained in the *NameValueString* field is not valid.

Completion Code

MQCC_FAILED

Programmer response

Modify the application that generated the message to ensure that it places in the *NameValueString* field a command name that is valid.

2337 (0921) (RC2337): MQRC_RFH_PARM_ERROR

Explanation

The message contains an MQRFH structure, but a parameter name contained in the *NameValueString* field is not valid for the command specified.

Completion Code

MQCC_FAILED

Programmer response

Modify the application that generated the message to ensure that it places in the *NameValueString* field only parameters that are valid for the specified command.

2338 (0922) (RC2338): MQRC_RFH_DUPLICATE_PARM

Explanation

The message contains an MQRFH structure, but a parameter occurs more than once in the *NameValueString* field when only one occurrence is valid for the specified command.

Completion Code

MQCC_FAILED

Programmer response

Modify the application that generated the message to ensure that it places in the *NameValueString* field only one occurrence of the parameter.

2339 (0923) (RC2339): MQRC_RFH_PARM_MISSING

Explanation

The message contains an MQRFH structure, but the command specified in the `NameValueString` field requires a parameter that is not present.

Completion Code

MQCC_FAILED

Programmer response

Modify the application that generated the message to ensure that it places in the `NameValueString` field all parameters that are required for the specified command.

2340 (0924) (RC2340): MQRC_CHAR_CONVERSION_ERROR

Explanation

This reason code is returned by the Java MQQueueManager constructor when a required character-set conversion is not available. The conversion required is between two nonUnicode character sets.

This reason code occurs in the following environment: MQ Classes for Java on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the National Language Resources component of the z/OS Language Environment is installed, and that conversion between the IBM-1047 and ISO8859-1 character sets is available.

2341 (0925) (RC2341): MQRC_UCS2_CONVERSION_ERROR

Explanation

This reason code is returned by the Java MQQueueManager constructor when a required character set conversion is not available. The conversion required is between the UCS-2 Unicode character set and the character set of the queue manager which defaults to IBM-500 if no specific value is available.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the relevant Unicode conversion tables are available for the JVM. For z/OS ensure that the Unicode conversion tables are available to the z/OS Language Environment. The conversion tables should be installed as part of the z/OS C/C++ optional feature. Refer to the *z/OS C/C++ Programming Guide* for more information about enabling UCS-2 conversions.

2342 (0926) (RC2342): MQRC_DB2_NOT_AVAILABLE

Explanation

An MQOPEN, MQPUT1, or MQSET call, or a command, was issued to access a shared queue, but it failed because the queue manager is not connected to a Db2 subsystem. As a result, the queue manager is unable to access the object definition relating to the shared queue.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Configure the Db2 subsystem so that the queue manager can connect to it.

2343 (0927) (RC2343): MQRC_OBJECT_NOT_UNIQUE

Explanation

An MQOPEN or MQPUT1 call, or a command, was issued to access a queue, but the call failed because the queue specified cannot be resolved unambiguously. There exists a shared queue with the specified name, and a nonshared queue with the same name.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

One of the queues must be deleted. If the queue to be deleted contains messages, use the MQSC command MOVE QLOCAL to move the messages to a different queue, and then use the command DELETE QLOCAL to delete the queue.

2344 (0928) (RC2344): MQRC_CONN_TAG_NOT_RELEASED

Explanation

An MQDISC call was issued when there was a unit of work outstanding for the connection handle. For CICS, IMS, and RRS connections, the MQDISC call does not commit or back out the unit of work. As a result, the connection tag associated with the unit of work is not yet available for reuse. The tag becomes available for reuse only when processing of the unit of work has been completed.

This reason code occurs only on z/OS.

Completion Code

MQCC_WARNING

Programmer response

Do not try to reuse the connection tag immediately. If the MQCONNX call is issued with the same connection tag, and that tag is still in use, the call fails with reason code MQRC_CONN_TAG_IN_USE.

2345 (0929) (RC2345): MQRC_CF_NOT_AVAILABLE

Explanation

An MQI call was issued to access a shared queue, but the call failed either because connectivity was lost to the coupling facility (CF) where the CF structure specified in the queue definition was allocated, or because allocation of the CF structure failed because there is no suitable CF to hold the structure, based on the preference list in the active CFRM policy.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

If connectivity was lost to the CF where the structure was allocated, and the queue manager has been configured to tolerate the failure and rebuild the structure, no action should be necessary. Otherwise, make available a coupling facility with one of the names specified in the CFRM policy, or modify the CFRM policy to specify the names of coupling facilities that are available.

2346 (092A) (RC2346): MQRC_CF_STRUC_IN_USE

Explanation

An MQI call or command was issued to operate on a shared queue, but the call failed because the coupling-facility structure specified in the queue definition is unavailable. The coupling-facility structure can be unavailable because a structure dump is in progress, or new connectors to the structure are currently inhibited, or an existing connector to the structure failed or disconnected abnormally and clean-up is not yet complete.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Typically, this is a temporary problem: wait for a while then retry the operation.

If the problem does not resolve itself, then connectivity problems experienced during the recovery of structures in the coupling facility could have occurred. In this case, restart the queue manager which reported the error. Resolve all the connectivity problems concerning the coupling facility before restarting the queue manager.

2347 (092B) (RC2347): MQRC_CF_STRUC_LIST_HDR_IN_USE

Explanation

An MQGET, MQOPEN, MQPUT1, or MQSET call was issued to access a shared queue, but the call failed because the list header associated with the coupling-facility structure specified in the queue definition is temporarily unavailable. The list header is unavailable because it is undergoing recovery processing.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The problem is temporary; wait a short while and then retry the operation.

2348 (092C) (RC2348): MQRC_CF_STRUC_AUTH_FAILED

Explanation

An MQOPEN or MQPUT1 call was issued to access a shared queue, but the call failed because the user is not authorized to access the coupling-facility structure specified in the queue definition.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Modify the security profile for the user identifier used by the application so that the application can access the coupling-facility structure specified in the queue definition.

2349 (092D) (RC2349): MQRC_CF_STRUC_ERROR

Explanation

An MQOPEN or MQPUT1 call was issued to access a shared queue, but the call failed because the coupling-facility structure name specified in the queue definition is not defined in the CFRM data set, or is not the name of a list structure.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Modify the queue definition to specify the name of a coupling-facility list structure that is defined in the CFRM data set.

2350 (092E) (RC2350): MQRC_CONN_TAG_NOT_USABLE

Explanation

An MQCONNX call was issued specifying one of the MQCNO_*_CONN_TAG_* options, but the call failed because the connection tag specified by *ConnTag* in MQCNO is being used by the queue manager for recovery processing, and this processing is delayed pending recovery of the coupling facility.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The problem is likely to persist. Consult the system programmer to ascertain the cause of the problem.

2351 (092F) (RC2351): MQRC_GLOBAL_UOW_CONFLICT

Explanation

An attempt was made to use inside a global unit of work a connection handle that is participating in another global unit of work. This can occur when an application passes connection handles between objects where the objects are involved in different DTC transactions. Because transaction completion is asynchronous, it is possible for this error to occur *after* the application has finalized the first object and committed its transaction.

This error does not occur for nontransactional MQI calls.

This reason code occurs only on Windows and z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check that the **`MTS Transaction Support`** attribute defined for the object's class is set correctly. If necessary, modify the application so that the connection handle is not used by objects participating in different units of work.

2352 (0930) (RC2352): MQRC_LOCAL_UOW_CONFLICT

Explanation

An attempt was made to use inside a global unit of work a connection handle that is participating in a queue-manager coordinated local unit of work. This can occur when an application passes connection handles between objects where one object is involved in a DTC transaction and the other is not.

This error does not occur for nontransactional MQI calls.

This reason code occurs only on Windows and z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check that the "MTS Transaction Support˘ attribute defined for the object's class is set correctly. If necessary, modify the application so that the connection handle is not used by objects participating in different units of work.

2353 (0931) (RC2353): MQRC_HANDLE_IN_USE_FOR_UOW

Explanation

An attempt was made to use outside a unit of work a connection handle that is participating in a global unit of work.

This error can occur when an application passes connection handles between objects where one object is involved in a DTC transaction and the other is not. Because transaction completion is asynchronous, it is possible for this error to occur *after* the application has finalized the first object and committed its transaction.

This error can also occur when a single object that was created and associated with the transaction loses that association whilst the object is running. The association is lost when DTC terminates the transaction independently of MTS. This might be because the transaction timed out, or because DTC shut down.

This error does not occur for nontransactional MQI calls.

This reason code occurs only on Windows.

Completion Code

MQCC_FAILED

Programmer response

Check that the "MTS Transaction Support" attribute defined for the object's class is set correctly. If necessary, modify the application so that objects executing within different units of work do not try to use the same connection handle.

2354 (0932) (RC2354): MQRC_UOW_ENLISTMENT_ERROR

Explanation

This reason code can occur for various reasons and occurs only on Windows , and HP Integrity NonStop Server .

On Windows, the most likely reason is that an object created by a DTC transaction does not issue a transactional MQI call until after the DTC transaction timed out. (If the DTC transaction times out after a transactional MQI call has been issued, reason code MQRC_HANDLE_IN_USE_FOR_UOW is returned by the failing MQI call.)

On HP Integrity NonStop Server, this reason occurs:

- On a transactional MQI call when the client encounters a configuration error preventing it from enlisting with the TMF/Gateway, therefore preventing participation within a global unit of work that is coordinated by the Transaction Management Facility (TMF).
- If a client application makes an enlistment request before the TMF/Gateway completes recovery of in-doubt transactions, the request is held for up to 1 second. If recovery does not complete within that time, the enlistment is rejected.

Another cause of MQRC_UOW_ENLISTMENT_ERROR is incorrect installation; On Windows, for example, the Windows NT Service pack must be installed after the Windows NT Option pack.

Completion Code

MQCC_FAILED

Programmer response

On Windows, check the DTC "Transaction timeout" value. If necessary, verify the Windows NT installation order.

On HP Integrity NonStop Server this might be a configuration error. The client issues a message to the client error log providing extra information about the configuration error. Contact your system administrator to resolve the indicated error.

2355 (0933) (RC2355): MQRC_UOW_MIX_NOT_SUPPORTED

Explanation

This reason code occurs only on Windows when you are running a version of the queue manager before version 5.2. , and on HP Integrity NonStop Server .

On Windows, the following explanations might apply:

- The mixture of calls that is used by the application to perform operations within a unit of work is not supported. In particular, it is not possible to mix within the same process a local unit of work that is coordinated by the queue manager with a global unit of work that is coordinated by DTC (Distributed Transaction Coordinator).
- An application might cause this mixture to arise if some objects in a package are coordinated by DTC and others are not. It can also occur if transactional MQI calls from an MTS client are mixed with transactional MQI calls from a library package transactional MTS object.
- No problem arises if all transactional MQI calls originate from transactional MTS objects, or all transactional MQI calls originate from non-transactional MTS objects. But when a mixture of styles is used, the first style that is used fixes the style for the unit of work, and subsequent attempts to use the other style within the process fail with reason code MQRC_UOW_MIX_NOT_SUPPORTED.
- When an application is run twice, scheduling factors in the operating system mean that it is possible for the queue-manager-coordinated transactional calls to fail in one run, and for the DTC-coordinated transactional calls to fail in the other run.

On HP Integrity NonStop Server it is not possible, within a single IBM MQ connection, to issue transactional MQI calls under the coordination of the Transaction Management Facility (TMF) if transactional MQI calls have already been made within a local unit of work that is coordinated by the queue manager until the local unit of work is completed by issuing either MQCMIT or MQBACK.

Completion Code

MQCC_FAILED

Programmer response

On Windows, check that the "MTS Transaction Support" attribute defined for the object's class is set correctly. If necessary, modify the application so that objects that run within different units of work do not try to use the same connection handle.

On HP Integrity NonStop Server, if a local unit of work that is coordinated by the queue manager is in progress, it must either be completed by issuing MQCMIT, or rolled back by issuing MQBACK before issuing any transactional MQI calls under the coordination of TMF.

2356 (0934) (RC2356): MQRC_WXP_ERROR

Explanation

An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain, but the workload exit parameter structure *ExitParms* is not valid, for one of the following reasons:

- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The *StrucId* field is not MQWXP_STRUC_ID.
- The *Version* field is not MQWXP_VERSION_2.
- The *CacheContext* field does not contain the value passed to the exit by the queue manager.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the parameter specified for *ExitParms* is the MQWXP structure that was passed to the exit when the exit was invoked.

2357 (0935) (RC2357): MQRC_CURRENT_RECORD_ERROR

Explanation

An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain, but the address specified by the *CurrentRecord* parameter is not the address of a valid record. *CurrentRecord* must be the address of a destination record (MQWDR), queue record (MQWQR), or cluster record (MQWCR) residing within the cluster cache.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the cluster workload exit passes the address of a valid record residing in the cluster cache.

2358 (0936) (RC2358): MQRC_NEXT_OFFSET_ERROR

Explanation

An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain, but the offset specified by the *NextOffset* parameter is not valid. *NextOffset* must be the value of one of the following fields:

- *ChannelDefOffset* field in MQWDR
- *ClusterRecOffset* field in MQWDR
- *ClusterRecOffset* field in MQWQR
- *ClusterRecOffset* field in MQWCR

Completion Code

MQCC_FAILED

Programmer response

Ensure that the value specified for the *NextOffset* parameter is the value of one of the fields listed.

2359 (0937) (RC2359): MQRC_NO_RECORD_AVAILABLE

Explanation

An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain, but the current record is the last record in the chain.

Completion Code

MQCC_FAILED

Programmer response

None.

2360 (0938) (RC2360): MQRC_OBJECT_LEVEL_INCOMPATIBLE

Explanation

An MQOPEN or MQPUT1 call, or a command, was issued, but the definition of the object to be accessed is not compatible with the queue manager to which the application has connected. The object definition was created or modified by a different version of the queue manager.

If the object to be accessed is a queue, the incompatible object definition could be the object specified, or one of the object definitions used to resolve the specified object (for example, the base queue to which an alias queue resolves, or the transmission queue to which a remote queue or queue-manager alias resolves).

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The application must be run on a queue manager that is compatible with the object definition. .

2361 (0939) (RC2361): MQRC_NEXT_RECORD_ERROR

Explanation

An MQXCLWLN call was issued from a cluster workload exit to obtain the address of the next record in the chain, but the address specified for the *NextRecord* parameter is either null, not valid, or the address of read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Specify a valid address for the *NextRecord* parameter.

2362 (093A) (RC2362): MQRC_BACKOUT_THRESHOLD_REACHED

Explanation

This reason code occurs only in the *Reason* field in an MQDLH structure, or in the *Feedback* field in the MQMD of a report message.

A JMS ConnectionConsumer found a message that exceeds the queue's backout threshold. The queue does not have a backout requeue queue defined, so the message was processed as specified by the disposition options in the *Report* field in the MQMD of the message.

On queue managers that do not support the *BackoutThreshold* and *BackoutRequeueQName* queue attributes, JMS ConnectionConsumer uses a value of 20 for the backout threshold. When the *BackoutCount* of a message reaches this threshold, the message is processed as specified by the disposition options.

If the *Report* field specifies one of the MQRO_EXCEPTION_* options, this reason code appears in the *Feedback* field of the report message. If the *Report* field specifies MQRO_DEAD_LETTER_Q, or the disposition report options remain at the default, this reason code appears in the *Reason* field of the MQDLH.

Completion Code

None

Programmer response

Investigate the cause of the backout count being greater than the threshold. To correct this, define the backout queue for the queue concerned.

2363 (093B) (RC2363): MQRC_MSG_NOT_MATCHED

Explanation

This reason code occurs only in the *Reason* field in an MQDLH structure, or in the *Feedback* field in the MQMD of a report message.

While performing Point-to-Point messaging, JMS encountered a message matching none of the selectors of ConnectionConsumers monitoring the queue. To maintain performance, the message was processed as specified by the disposition options in the *Report* field in the MQMD of the message.

If the *Report* field specifies one of the MQRO_EXCEPTION_* options, this reason code appears in the *Feedback* field of the report message. If the *Report* field specifies MQRO_DEAD_LETTER_Q, or the disposition report options remain at the default, this reason code appears in the *Reason* field of the MQDLH.

Completion Code

None

Programmer response

To correct this, ensure that the ConnectionConsumers monitoring the queue provide a complete set of selectors. Alternatively, set the QueueConnectionFactory to retain messages.

2364 (093C) (RC2364): MQRC_JMS_FORMAT_ERROR

Explanation

This reason code is generated by JMS applications that use either:

- ConnectionConsumers
- Activation Specifications
- WebSphere Application Server Listener Ports

and connect to an IBM MQ queue manager using IBM MQ messaging provider migration mode. When the IBM MQ classes for JMS encounter a message that cannot be parsed (for example, the message contains an invalid RFH2 header) the message is processed as specified by the disposition options in the *Report* field in the MQMD of the message.

If the *Report* field specifies one of the MQRO_EXCEPTION_* options, this reason code appears in the *Feedback* field of the report message. If the *Report* field specifies MQRO_DEAD_LETTER_Q, or the disposition report options remain at the default, this reason code appears in the *Reason* field of the MQDLH.

Completion Code

None

Programmer response

Investigate the origin of the message.

2365 (093D) (RC2365): MQRC_SEGMENTS_NOT_SUPPORTED

Explanation

An MQPUT call was issued to put a segment of a logical message, but the queue on which the message is to be placed has an *IndexType* of MQIT_GROUP_ID. Message segments cannot be placed on queues with this index type.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Modify the application to put messages that are not segments; ensure that the MQMF_SEGMENT and MQMF_LAST_SEGMENT flags in the *MsgFlags* field in MQMD are not set, and that the *Offset* is zero. Alternatively, change the index type of the queue.

2366 (093E) (RC2366): MQRC_WRONG_CF_LEVEL

Explanation

An MQOPEN or MQPUT1 call was issued specifying a shared queue, but the queue requires a coupling-facility structure with a different level of capability.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the coupling-facility structure used for the queue is at the level required to support the capabilities that the queue provides.

You can use the DISPLAY CFSTRUCT command to display the level, and ALTER CFSTRUCT() CFLEVEL() command to modify the level; see The MQSC commands.

2367 (093F) (RC2367): MQRC_CONFIG_CREATE_OBJECT

Explanation

This condition is detected when an object is created.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Create object.

2368 (0940) (RC2368): MQRC_CONFIG_CHANGE_OBJECT

Explanation

This condition is detected when an object is changed.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Change object.

2369 (0941) (RC2369): MQRC_CONFIG_DELETE_OBJECT

Explanation

This condition is detected when an object is deleted.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Delete object.

2370 (0942) (RC2370): MQRC_CONFIG_REFRESH_OBJECT

Explanation

This condition is detected when an object is refreshed.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Refresh object.

2371 (0943) (RC2371): MQRC_CHANNEL_SSL_ERROR

Explanation

This condition is detected when a connection cannot be established due to an SSL key-exchange or authentication failure.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel SSL Error.

2373 (0945) (RC2373): MQRC_CF_STRUC_FAILED

Explanation

An MQI call or command was issued to access a shared queue, but the call failed because the coupling-facility structure used for the shared queue had failed.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Report the problem to the operator or administrator, who should use the MQSC command RECOVER CFSTRUCT to initiate recovery of the coupling-facility structure, unless automatic recovery has been enabled for the structure.

2374 (0946) (RC2374): MQRC_API_EXIT_ERROR

Explanation

An API exit function returned an invalid response code, or failed in some other way.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Check the exit logic to ensure that the exit is returning valid values in the *ExitResponse* and *ExitResponse2* fields of the MQAXP structure. Consult the FFST record to see if it contains more detail about the problem.

2375 (0947) (RC2375): MQRC_API_EXIT_INIT_ERROR

Explanation

The queue manager encountered an error while attempting to initialize the execution environment for an API exit function.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Consult the FFST record to obtain more detail about the problem.

2376 (0948) (RC2376): MQRC_API_EXIT_TERM_ERROR

Explanation

The queue manager encountered an error while attempting to terminate the execution environment for an API exit function.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Consult the FFST record to obtain more detail about the problem.

2377 (0949) (RC2377): MQRC_EXIT_REASON_ERROR

Explanation

An MQXEP call was issued by an API exit function, but the value specified for the *ExitReason* parameter is either not valid, or not supported for the specified function identifier *Function*.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Modify the exit function to specify a value for *ExitReason* that is valid for the specified value of *Function*.

2378 (094A) (RC2378): MQRC_RESERVED_VALUE_ERROR

Explanation

An MQXEP call was issued by an API exit function, but the value specified for the *Reserved* parameter is not valid. The value must be the null pointer.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Modify the exit to specify the null pointer as the value of the *Reserved* parameter.

2379 (094B) (RC2379): MQRC_NO_DATA_AVAILABLE

Explanation

This reason should be returned by the MQZ_ENUMERATE_AUTHORITY_DATA installable service component when there is no more authority data to return to the invoker of the service component.

- On z/OS, this reason code does not occur.

Completion Code

MQCC_FAILED

Programmer response

None.

2380 (094C) (RC2380): MQRC_SCO_ERROR

Explanation

On an MQCONNX call, the MQSCO structure is not valid for one of the following reasons:

- The *StrucId* field is not MQSCO_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Correct the definition of the MQSCO structure.

2381 (094D) (RC2381): MQRC_KEY_REPOSITORY_ERROR

Explanation

On an MQCONN or MQCONNX call, the location of the key repository is either not specified, not valid, or results in an error when used to access the key repository. A common problem is to specify the `.kdb` suffix in the name of the keystore.

The location of the key repository is specified by one of the following:

- The value of the MQSSLKEYR environment variable (MQCONN or MQCONNX call), or
- The value of the *KeyRepository* field in the MQSCO structure (MQCONNX call only).

For the MQCONNX call, if both MQSSLKEYR and *KeyRepository* are specified, the latter is used.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid location for the key repository.

2382 (094E) (RC2382): MQRC_CRYPTO_HARDWARE_ERROR

Explanation

On an MQCONN or MQCONNX call, the configuration string for the cryptographic hardware is not valid, or results in an error when used to configure the cryptographic hardware. The configuration string is specified by one of the following:

- The value of the MQSSLCRYP environment variable (MQCONN or MQCONNX call), or

- The value of the *CryptoHardware* field in the MQSCO structure (MQCONNX call only).

For the MQCONNX call, if both MQSSLCRYP and *CryptoHardware* are specified, the latter is used.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid configuration string for the cryptographic hardware.

2383 (094F) (RC2383): MQRC_AUTH_INFO_REC_COUNT_ERROR

Explanation

On an MQCONNX call, the *AuthInfoRecCount* field in the MQSCO structure specifies a value that is less than zero.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a value for *AuthInfoRecCount* that is zero or greater.

2384 (0950) (RC2384): MQRC_AUTH_INFO_REC_ERROR

Explanation

On an MQCONNX call, the MQSCO structure does not specify the address of the MQAIR records correctly. One of the following applies:

- *AuthInfoRecCount* is greater than zero, but *AuthInfoRecOffset* is zero and *AuthInfoRecPtr* is the null pointer.
- *AuthInfoRecOffset* is not zero and *AuthInfoRecPtr* is not the null pointer.
- *AuthInfoRecPtr* is not a valid pointer.
- *AuthInfoRecOffset* or *AuthInfoRecPtr* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Ensure that one of *AuthInfoRecOffset* or *AuthInfoRecPtr* is zero and the other nonzero. Ensure that the field used points to accessible storage.

2385 (0951) (RC2385): MQRC_AIR_ERROR

Explanation

On an MQCONNX call, an MQAIR record is not valid for one of the following reasons:

- The *StrucId* field is not MQAIR_STRUC_ID.
- The *Version* field specifies a value that is not valid or not supported.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Correct the definition of the MQAIR record.

2386 (0952) (RC2386): MQRC_AUTH_INFO_TYPE_ERROR

Explanation

On an MQCONNX call, the *AuthInfoType* field in an MQAIR record specifies a value that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify MQAIT_CRL_LDAP for *AuthInfoType*.

2387 (0953) (RC2387): MQRC_AUTH_INFO_CONN_NAME_ERROR

Explanation

On an MQCONNX call, the *AuthInfoConnName* field in an MQAIR record specifies a value that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid connection name.

2388 (0954) (RC2388): MQRC_LDAP_USER_NAME_ERROR

Explanation

On an MQCONNX call, an LDAP user name in an MQAIR record is not specified correctly. One of the following applies:

- *LDAPUserNameLength* is greater than zero, but *LDAPUserNameOffset* is zero and *LDAPUserNamePtr* is the null pointer.
- *LDAPUserNameOffset* is nonzero and *LDAPUserNamePtr* is not the null pointer.
- *LDAPUserNamePtr* is not a valid pointer.
- *LDAPUserNameOffset* or *LDAPUserNamePtr* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Ensure that one of *LDAPUserNameOffset* or *LDAPUserNamePtr* is zero and the other nonzero. Ensure that the field used points to accessible storage.

2389 (0955) (RC2389): MQRC_LDAP_USER_NAME_LENGTH_ERR

Explanation

On an MQCONNX call, the *LDAPUserNameLength* field in an MQAIR record specifies a value that is less than zero.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a value for *LDAPUserNameLength* that is zero or greater.

2390 (0956) (RC2390): MQRC_LDAP_PASSWORD_ERROR

Explanation

On an MQCONNX call, the *LDAPPassword* field in an MQAIR record specifies a value when no value is allowed.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Specify a value that is blank or null.

2391 (0957) (RC2391): MQRC_SSL_ALREADY_INITIALIZED

Explanation

An MQCONN or MQCONNX call was issued when a connection is already open to the same queue manager. There is a conflict between the SSL options of the connections for one of three reasons:

- The SSL configuration options are different between the first and second connections.

- The existing connection was specified without SSL configuration options, but the second connection has SSL configuration options specified.
- The existing connection was specified with SSL configuration options, but the second connection does not have any SSL configuration options specified.

The connection to the queue manager completed successfully, but the SSL configuration options specified on the call were ignored; the existing SSL environment was used instead.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_WARNING

Programmer response

If the application must be run with the SSL configuration options defined on the MQCONN or MQCONNX call, use the MQDISC call to sever the connection to the queue manager and then stop the application. Alternatively run the application later when the SSL environment has not been initialized.

2392 (0958) (RC2392): MQRC_SSL_CONFIG_ERROR

Explanation

On an MQCONNX call, the MQCNO structure does not specify the MQSCO structure correctly. One of the following applies:

- *SSLConfigOffset* is nonzero and *SSLConfigPtr* is not the null pointer.
- *SSLConfigPtr* is not a valid pointer.
- *SSLConfigOffset* or *SSLConfigPtr* points to storage that is not accessible.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Ensure that one of *SSLConfigOffset* or *SSLConfigPtr* is zero and the other nonzero. Ensure that the field used points to accessible storage.

2393 (0959) (RC2393): MQRC_SSL_INITIALIZATION_ERROR

Explanation

An MQCONN or MQCONNX call was issued with SSL configuration options specified, but an error occurred during the initialization of the SSL environment.

This reason code occurs in the following environments: AIX, HP-UX, Solaris, Windows.

Completion Code

MQCC_FAILED

Programmer response

Check that the SSL installation is correct.

2394 (095A) (RC2394): MQRC_Q_INDEX_TYPE_ERROR

Explanation

An MQGET call was issued specifying one or more of the following options:

- MQGMO_ALL_MSGS_AVAILABLE
- MQGMO_ALL_SEGMENTS_AVAILABLE
- MQGMO_COMPLETE_MSG
- MQGMO_LOGICAL_ORDER

but the call failed because the queue is not indexed by group identifier. These options require the queue to have an *IndexType* of MQIT_GROUP_ID.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

Redefine the queue to have an *IndexType* of MQIT_GROUP_ID. Alternatively, modify the application to avoid using the options listed.

2395 (095B) (RC2395): MQRC_CFBS_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFBS structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2396 (095C) (RC2396): MQRC_SSL_NOT_ALLOWED

Explanation

A connection to a queue manager was requested, specifying SSL encryption. However, the connection mode requested is one that does not support SSL (for example, bindings connect).

Completion Code

MQCC_FAILED

Programmer response

Modify the application to request client connection mode, or to disable SSL encryption.

Note: Using a non null setting, including blanks, for the connection's cipher suite property can also cause this error.

2397 (095D) (RC2397): MQRC_JSSE_ERROR

Explanation

JSSE reported an error (for example, while connecting to a queue manager using SSL encryption). The MQException object containing this reason code references the Exception thrown by JSSE; this can be obtained by using the MQException.getCause() method. From JMS, the MQException is linked to the thrown JMSException.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Inspect the causal exception to determine the JSSE error.

2398 (095E) (RC2398): MQRC_SSL_PEER_NAME_MISMATCH

Explanation

The application attempted to connect to the queue manager using SSL encryption, but the distinguished name presented by the queue manager does not match the specified pattern.

Completion Code

MQCC_FAILED

Programmer response

Check the certificates used to identify the queue manager. Also check the value of the sslPeerName property specified by the application.

2399 (095F) (RC2399): MQRC_SSL_PEER_NAME_ERROR

Explanation

The application specified a peer name of incorrect format.

Completion Code

MQCC_FAILED

Programmer response

Check the value of the sslPeerName property specified by the application.

2400 (0960) (RC2400): MQRC_UNSUPPORTED_CIPHER_SUITE

Explanation

A connection to a queue manager was requested, specifying SSL encryption. However, JSSE reported that it does not support the CipherSuite specified by the application.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Check the CipherSuite specified by the application. Note that the names of JSSE CipherSuites differ from their equivalent CipherSpecs used by the queue manager.

Also, check that JSSE is correctly installed.

2401 (0961) (RC2401): MQRC_SSL_CERTIFICATE_REVOKED

Explanation

A connection to a queue manager was requested, specifying SSL encryption. However, the certificate presented by the queue manager was found to be revoked by one of the specified CertStores.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Check the certificates used to identify the queue manager.

2402 (0962) (RC2402): MQRC_SSL_CERT_STORE_ERROR

Explanation

A connection to a queue manager was requested, specifying SSL encryption. However, none of the CertStore objects provided by the application could be searched for the certificate presented by the queue manager. The MQException object containing this reason code references the Exception encountered when searching the first CertStore; this can be obtained using the MQException.getCause() method. From JMS, the MQException is linked to the thrown JMSException.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Inspect the causal exception to determine the underlying error. Check the CertStore objects provided by your application. If the causal exception is a java.lang.NoSuchElementException, ensure that your application is not specifying an empty collection of CertStore objects.

2406 (0966) (RC2406): MQRC_CLIENT_EXIT_LOAD_ERROR

Explanation

The external user exit required for a client connection could not be loaded because the shared library specified for it cannot be found, or the entry point specified for it cannot be found.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct library has been specified, and that the path variable for the machine environment includes the relevant directory. Ensure also that the entry point has been named properly and that the named library does export it.

2407 (0967) (RC2407): MQRC_CLIENT_EXIT_ERROR

Explanation

A failure occurred while executing a non-Java user exit for a client connection.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Check that the non-Java user exit can accept the parameters and message being passed to it and that it can handle error conditions, and that any information that the exit requires, such as user data, is correct and available.

2409 (0969) (RC2409): MQRC_SSL_KEY_RESET_ERROR

Explanation

On an MQCONN or MQCONNX call, the value of the SSL key reset count is not in the valid range of 0 through 999 999 999.

The value of the SSL key reset count is specified by either the value of the MQSSLRESET environment variable (MQCONN or MQCONNX call), or the value of the *KeyResetCount* field in the MQSCO structure (MQCONNX call only). For the MQCONNX call, if both MQSSLRESET and *KeyResetCount* are specified, the latter is used. MQCONN or MQCONNX

If you specify an SSL/TLS secret key reset count in the range 1 byte through 32Kb, SSL/TLS channels will use a secret key reset count of 32Kb. This is to avoid the overhead of excessive key resets which would occur for small SSL/TLS secret key reset values.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure and the MQSSLRESET environment variable are set correctly.

2411 (096B) (RC2411): MQRC_LOGGER_STATUS

Explanation

This condition is detected when a logger event occurs.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Logger.

2412 (096C) (RC2412): MQRC_COMMAND_MQSC

Explanation

This condition is detected when an MQSC command is executed.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Command.

2413 (096D) (RC2413): MQRC_COMMAND_PCF

Explanation

This condition is detected when a PCF command is executed.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Command.

2414 (096E) (RC2414): MQRC_CFIF_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFIF structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2415 (096F) (RC2415): MQRC_CFSF_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFSF structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2416 (0970) (RC2416): MQRC_CFGR_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFGR structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, z/OS, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2417 (0971) (RC2417): MQRC_MSG_NOT_ALLOWED_IN_GROUP

An explanation of the error, completion code, and programmer response.

Explanation

An MQPUT or MQPUT1 call was issued to put a message in a group but it is not valid to put such a message in a group. An example of an invalid message is a PCF message where the Type is MQCFT_TRACE_ROUTE.

You cannot use grouped or segmented messages with Publish/Subscribe.

Completion Code

MQCC_FAILED

Programmer response

Remove the invalid message from the group.

2418 (0972) (RC2418): MQRC_FILTER_OPERATOR_ERROR

Explanation

The **Operator** parameter supplied is not valid.

If it is an input variable then the value is not one of the MQCFOP_* constant values. If it is an output variable then the parameter pointer is not valid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredicatable results occur.)

Completion Code

MQCC_FAILED

Programmer response

Correct the parameter.

2419 (0973) (RC2419): MQRC_NESTED_SELECTOR_ERROR

Explanation

An mqAddBag call was issued, but the bag to be nested contained a data item with an inconsistent selector. This reason only occurs if the bag into which the nested bag was to be added was created with the MQCBO_CHECK_SELECTORS option.

Completion Code

MQCC_FAILED

Programmer response

Ensure that all data items within the bag to be nested have selectors that are consistent with the data type implied by the item.

2420 (0974) (RC2420): MQRC_EPH_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQEPH structure that is not valid. Possible errors include the following:

- The *StrucId* field is not MQEPH_STRUC_ID.
- The *Version* field is not MQEPH_VERSION_1.
- The *StrucLength* field specifies a value that is too small to include the structure plus the variable-length data at the end of the structure.
- The *CodedCharSetId* field is zero, or a negative value that is not valid.
- The *Flags* field contains an invalid combination of MQEPH_* values.
- The *BufferLength* parameter of the call has a value that is too small to accommodate the structure, so the structure extends beyond the end of the message.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly. Ensure that the application sets the *CodedCharSetId* field to a valid value; note that MQCCSI_DEFAULT, MQCCSI_EMBEDDED, MQCCSI_Q_MGR, and MQCCSI_UNDEFINED are not valid in this field.

2421 (0975) (RC2421): MQRC_RFH_FORMAT_ERROR

Explanation

The message contains an MQRFH structure, but its format is incorrect. If you are using IBM MQ SOAP, the error is in an incoming SOAP/MQ request message.

Completion Code

MQCC_FAILED

Programmer response

If you are using IBM MQ SOAP with the IBM-supplied sender, contact your IBM support center. If you are using IBM MQ SOAP with a bespoke sender, check that the RFH2 section of the SOAP/MQ request message is in valid RFH2 format.

2422 (0976) (RC2422): MQRC_CFBF_ERROR

Explanation

An MQPUT or MQPUT1 call was issued, but the message data contains an MQCFBF structure that is not valid.

This reason code occurs in the following environments: AIX, HP-UX, IBM i, Solaris, Windows, plus IBM MQ clients connected to these systems.

Completion Code

MQCC_FAILED

Programmer response

Check that the fields in the structure are set correctly.

2423 (0977) (RC2423): MQRC_CLIENT_CHANNEL_CONFLICT

Explanation

A client channel definition table (CCDT) was specified for determining the name of the channel, but the name has already been defined.

This reason code occurs only with Java applications.

Completion Code

MQCC_FAILED

Programmer response

Change the channel name to blank and try again.

2424 (0978) (RC2424): MQRC_SD_ERROR

Explanation

On the MQSUB call, the Subscription Descriptor MQSD is not valid, for one of the following reasons:

- The StrucId field is not MQSD_SCTRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid (it is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results can occur).
- The queue manager cannot copy the changes structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQSD structure are set correctly.

2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR

Explanation

On the MQOPEN or MQPUT1 call in the Object Descriptor MQOD, or on the MQSUB call in the Subscription Descriptor MQSD the resultant full topic string is not valid.

One of the following applies:

- ObjectName contains the name of a TOPIC object with a TOPICSTR attribute that contains an empty topic string.
- The fully resolved topic string contains the escape character `'%'` and it is not followed by one of the characters, `'*', '?' or '%'`, and the MQSO_WILDCARD_CHAR option has been used on an MQSUB call.
- On an MQOPEN, conversion cannot be performed using the CCSID specified in the MQOD structure.
- The topic string is greater than 255 characters when using IBM MQ Multicast messaging.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that there are no invalid topic string characters in either ObjectString or ObjectName.

If using IBM MQ Multicast messaging, ensure that the topic string is less than 255 characters.

2426 (097A) (RC2426): MQRC_STS_ERROR

Explanation

On an MQSTAT call, the MQSTS structure is not valid, for one of the following reasons:

- The StrucId field is not MQSTS_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.

- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQSTS structure are set correctly.

2428 (097C) (RC2428): MQRC_NO_SUBSCRIPTION

Explanation

An MQSUB call using option MQSO_RESUME was made specifying a full subscription name that does not match any existing subscription.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the subscription exists and that the full subscription name is correctly specified in your application. The full subscription name is built from the ConnTag field specified at connection time in the MQCNO structure and the SubName field specified at MQSUB time in the MQSD structure.

2429 (097D) (RC2429): MQRC_SUBSCRIPTION_IN_USE

Explanation

An MQSUB call using option MQSO_RESUME was made specifying a full subscription name that is in use.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the subscription name is correctly specified in your application. The subscription name is specified in the SubName field in the MQSD structure.

2430 (097E) (RC2430): MQRC_STAT_TYPE_ERROR

Explanation

The STS parameter contains options that are not valid for the MQSTAT call. This reason also occurs if the parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Programmer response

Specify a valid MQSTS structure as a parameter on the call to MQSTAT.

2431 (097F) (RC2431): MQRC_SUB_USER_DATA_ERROR

Explanation

On the MQSUB call in the Subscription Descriptor MQSD the SubUserData field is not valid. One of the following applies:

- SubUserData.VSLength is greater than zero, but SubUserData.VSOffset is zero and SubUserData.VSPtr is the null pointer.
- SubUserData.VSOffset is nonzero and SubUserData.VSPtr is not the null pointer (that is, it appears both fields are being used where only one is allowed).
- SubUserData.VSPtr is not a valid pointer.
- SubUserData.VSOffset or SubUserData.VSPtr points to storage that is not accessible.
- SubUserData.VSLength exceeds the maximum length allowed for this field.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that one of SubUserData.VSOffset or SubUserData.VSPtr is zero and the other nonzero. Ensure that the field used points to accessible storage. Specify a length that does not exceed the maximum length allowed for this field.

2432 (0980) (RC2432): MQRC_SUB_ALREADY_EXISTS

Explanation

An MQSUB call was issued to create a subscription, using the MQSO_CREATE option, but a subscription using the same SubName and ObjectString already exists.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the SubName and ObjectString input fields in the MQSD structure are set correctly, or use the MQSO_RESUME option to get a handle for the subscription that already exists.

2434 (0982) (RC2434): MQRC_IDENTITY_MISMATCH

Explanation

An MQSUB call using either MQSO_RESUME or MQSO_ALTER was made against a subscription that has the MQSO_FIXED_USERID option set, by a userid other than the one recorded as owning the subscription.

Completion Code

MQCC_FAILED

Programmer Response

Correct the full subscription name to one that is unique, or update the existing subscription to allow different userids to use it by using the MQSO_ANY_USERID option from an application running under the owning userid.

2435 (0983) (RC2435): MQRC_ALTER_SUB_ERROR

Explanation

An MQSUB call using option MQSO_ALTER was made changing a subscription that was created with the MQSO_IMMUTABLE option.

Completion Code

MQCC_FAILED

Programmer Response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly.

2436 (0984) (RC2436): MQRC_DURABILITY_NOT_ALLOWED

Explanation

An MQSUB call using the MQSO_DURABLE option failed. This can be for one of the following reasons:

- The topic subscribed to is defined as DURSUB(NO).
- The queue named SYSTEM.DURABLE.SUBSCRIBER.QUEUE is not available.
- The topic subscribed to is defined as both MCAST(ONLY) and DURSUB(YES) (or DURSUB(ASPARENT) and the parent is DURSUB(YES)).

Completion Code

MQCC_FAILED

Programmer Response

Durable subscriptions are stored on the SYSTEM.DURABLE.SUBSCRIBER.QUEUE. Ensure that this queue is available for use. Possible reasons for failure include the queue being full, the queue being put inhibited, the queue not existing, or (on z/OS) the pageset the queue is defined to use doesn't exist.

If the topic subscribed to is defined as DURSUB(NO) either alter the administrative topic node to use DURSUB(YES) or use the MQSO_NON_DURABLE option instead.

If the topic subscribed to is defined as MCAST(ONLY) when using IBM MQ Multicast messaging, alter the topic to use DURSUB(NO).

2437 (0985) (RC2437): MQRC_NO_RETAINED_MSG

Explanation

An MQSUBRQ call was made to a topic to request that any retained publications for this topic are sent to the subscriber. However, there are no retained publications currently stored for this topic.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that publishers to the topic are marking their publication to be retained and that publications are being made to this topic.

2438 (0986) (RC2438): MQRC_SRO_ERROR

Explanation

On the MQSUBRQ call, the Subscription Request Options MQSRO is not valid, for one of the following reasons:

- The StrucId field is not MQSRO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQSRO structure are set correctly.

2440 (0988) (RC2440): MQRC_SUB_NAME_ERROR

Explanation

On the MQSUB call in the Subscription Descriptor MQSD the SubName field is not valid or has been omitted. This is required if the MQSD option MQSO_DURABLE is specified, but may also be used if MQSO_DURABLE is not specified.

One of the following applies:

- SubName.VSLength is greater than zero, but SubName.VSOffset is zero and SubName.VSPtr is the null pointer.
- SubName.VSOffset is nonzero and SubName.VSPtr is not the null pointer (that is, it appears both fields are being used where only one is allowed).
- SubName.VSPtr is not a valid pointer.
- SubName.VSOffset or SubName.VSPtr points to storage that is not accessible.
- SubName.VSLength is zero but this field is required.
- SubName.VSLength exceeds the maximum length allowed for this field.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that SubName is specified and SubName.VSLength is nonzero. Ensure that one of SubName.VSOffset or SubName.VSPtr is zero and the other nonzero. Ensure that the field used points to accessible storage. Specify a length that does not exceed the maximum length allowed for this field.

This code can be returned if the sd.Options flags MQSO_CREATE and MQSO_RESUME are set together and sd.SubName is not initialized. You must also initialize the MQCHARV structure for sd.SubName, even if there is no subscription to resume; see Example 2: Managed MQ subscriber for more details.

2441 (0989) (RC2441): MQRC_OBJECT_STRING_ERROR

Explanation

On the MQOPEN or MQPUT1 call in the Object Descriptor MQOD, or on the MQSUB call in the Subscription Descriptor MQSD the ObjectString field is not valid.

One of the following applies:

- ObjectString.VSLength is greater than zero, but ObjectString.VSOffset is zero and ObjectString.VSPtr is the null pointer.
- ObjectString.VSOffset is nonzero and ObjectString.VSPtr is not the null pointer (that is, it appears both fields are being used where only one is allowed).
- ObjectString.VSPtr is not a valid pointer.
- ObjectString.VSOffset or ObjectString.VSPtr points to storage that is not accessible.
- ObjectString.VSLength exceeds the maximum length allowed for this field.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that one of ObjectString.VSOffset or ObjectString.VSPtr is zero and the other nonzero. Ensure that the field used points to accessible storage. Specify a length that does not exceed the maximum length allowed for this field.

2442 (098A) (RC2442): MQRC_PROPERTY_NAME_ERROR

Explanation

An attempt was made to set a property with an invalid name. Using any of the following settings results in this error:

- The name contains an invalid character.
- The name begins "JMS" or "usr.JMS" and the JMS property is not recognized.
- The name begins "mq" in any mixture of lowercase or uppercase and is not "mq_usr" and contains more than one "." character (U+002E). Multiple "." characters are not allowed in properties with those prefixes.
- The name is "NULL", "TRUE", "FALSE", "NOT", "AND", "OR", "BETWEEN", "LIKE", "IN", "IS" and "ESCAPE" or is one of these keywords prefixed by "usr.".
- The name begins with "Body" or "Root" (except for names beginning "Root.MQMD.").
- A "." character must not be followed immediately by another "." character.
- The "." character cannot be the last character in a property name.

Completion Code

MQCC_FAILED

Programmer Response

Valid property names are described in the IBM MQ documentation. Ensure that all properties in the message have valid names before reissuing the call.

2443 (098B) (RC2443): MQRC_SEGMENTATION_NOT_ALLOWED

Explanation

An MQPUT or MQPUT1 call was issued to put a segmented message or a message that may be broken up into smaller segments (MQMF_SEGMENTATION_ALLOWED). The message was found to contain one or more MQ-defined properties in the message data; MQ-defined properties are not valid in the message data of a segmented message.

IBM MQ Multicast cannot use segmented messages.

Completion Code

MQCC_FAILED

Programmer Response

Remove the invalid properties from the message data or prevent the message from being segmented.

2444 (098C) (RC2444): MQRC_CBD_ERROR

Explanation

a MQCB call the MQCBD structure is not valid for one of the following reasons:

- The StrucId field is not MQCBD_STRUC_ID
- The Version field is specifies a value that is not valid or is not supported
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQCBD structure are set correctly.

2445 (098D) (RC2445): MQRC_CTLO_ERROR

Explanation

On a MQCTL call the MQCTLO structure is not valid for one of the following reasons:

- The StrucId field is not MQCTLO_STRUC_ID
- The Version field is specifies a value that is not valid or is not supported
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQCTLO structure are set correctly.

2446 (098E) (RC2446): MQRC_NO_CALLBACKS_ACTIVE

Explanation

An MQCTL call was made with an Operation of MQOP_START_WAIT and has returned because there are no currently defined callbacks which are not suspended.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that there is at least one registered, resumed consumer function.

2448 (0990) (RC2448): MQRC_CALLBACK_NOT_REGISTERED

Explanation

An attempt to issue an MQCB call has been made against an object handle which does not currently have a registered callback.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that a callback has been registered against the object handle.

2449 (0991) (RC2449): MQRC_OPERATION_NOT_ALLOWED

Explanation

An MQCTL call was made with an Operation that is not allowed because of the state of asynchronous consumption on the hConn is currently in.

If Operation was MQOP_RESUME, the operation is not allowed because the state of asynchronous consumption on the hConn is STOPPED. Re-issue MQCTL with the MQOP_START Operation.

If Operation was MQOP_SUSPEND, the operation is not allowed because the state of asynchronous consumption on the hConn is STOPPED. If you need to get your hConn into a SUSPENDED state, issue MQCTL with the MQOP_START Operation followed by MQCTL with MQOP_SUSPEND.

If Operation was MQOP_START, the operation is not allowed because the state of asynchronous consumption on the hConn is SUSPENDED. Re-issue MQCTL with the MQOP_RESUME Operation.

If Operation was MQOP_START_WAIT, the operation is not allowed because either

- The state of asynchronous consumption on the hConn is SUSPENDED. Re-issue MQCTL with the MQOP_RESUME Operation.
- The state of asynchronous consumption on the hConn is already STARTED. Do not mix the use of MQOP_START and MQOP_START_WAIT within one application.

Completion Code

MQCC_FAILED

Programmer Response

Re-issue the MQCTL call with the correct Operation.

2457 (0999) (RC2457): MQRC_OPTIONS_CHANGED

Explanation

An MQGET call on a queue handle opened using MQOO_READ_AHEAD (or resolved to that value through the queue's default value) has altered an option that is required to be consistent between MQGET calls.

Completion Code

MQCC_FAILED

Programmer Response

Keep all required MQGET options the same between invocations of MQGET, or use MQOO_NO_READ_AHEAD when opening the queue.

2458 (099A) (RC2458): MQRC_READ_AHEAD_MSGS

Explanation

On an MQCLOSE call, the option MQCO_QUIESCE was used and there are still messages stored in client read ahead buffer that were sent to the client ahead of an application requesting them and have not yet been consumed by the application.

Completion Code

MQCC_WARNING

Programmer Response

Continue to consume messages using the queue handle until there are no more available and then issue the MQCLOSE again, or choose to discard these messages by issuing the MQCLOSE call with the MQCO_IMMEDIATE option instead.

2459 (099B) (RC2459): MQRC_SELECTOR_SYNTAX_ERROR

Explanation

An MQOPEN, MQPUT1 or MQSUB call was issued but a selection string was specified which contained a syntax error.

Completion Code

MQCC_FAILED

Programmer Response

See Message selector syntax and ensure that you have correctly followed the rules for specifying selection strings. Correct any syntax errors and resubmit the MQ API call for which the error occurred.

2460 (099C) (RC2460): MQRC_HMSG_ERROR

Explanation

On an MQCRTMH, MQDLTMH, MQSETMP, MQINQMP or MQDLT call, a message handle supplied is not valid, for one of the following reasons:

- The parameter pointer is not valid, or (for the MQCRTMH call) points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The value specified was not returned by a preceding MQCRTMH call.
- The value specified has been made invalid by a preceding MQDLTMH call.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that a successful MQCRTMH call is performed for the connection, and that an MQDLTMH call has not already been performed for it. Ensure that the handle is being used within its valid scope, for more information, see MQCRTMH - Create message handle.

2461 (099D) (RC2461): MQRC_CMHO_ERROR

Explanation

On an MQCRTMH call, the create message handle options structure MQCMHO is not valid, for one of the following reasons:

- The StrucId field is not MQCMHO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQCMHO structure are set correctly.

2462 (099E) (RC2462): MQRC_DMHO_ERROR

Explanation

On an MQDLTMH call, the delete message handle options structure MQDMHO is not valid, for one of the following reasons:

- The StrucId field is not MQCMHO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQDMHO structure are set correctly.

2463 (099F) (RC2463): MQRC_SMPO_ERROR

Explanation

On an MQSETMP call, the set message property options structure MQSMPO is not valid, for one of the following reasons:

- The StrucId field is not MQSMPO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQSMPO structure are set correctly.

2464 (09A0) (RC2464): MQRC_IMPO_ERROR

Explanation

On an MQINQMP call, the inquire message property options structure MQIMPO is not valid, for one of the following reasons:

- The StrucId field is not MQIMPO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The queue manager cannot copy the changed structure to application storage, even though the call is successful. This can occur, for example, if the pointer points to read-only storage.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQIMPO structure are set correctly.

2465 (09A1) (RC2465): MQRC_PROPERTY_NAME_TOO_BIG

Explanation

On an MQINQMP call, IBM MQ attempted to copy the name of the inquired property into the location indicated by the ReturnedName field of the InqPropOpts parameter but the buffer was too small to contain the full property name. The call failed but the VSLength field of the ReturnedName of the InqPropOpts parameter indicates how large the ReturnedName buffer needs to be.

Completion Code

MQCC_FAILED

Programmer response

The full property name can be retrieved by calling MQINQMP again with a larger buffer for the returned name, also specifying the MQIMPO_INQ_PROP_UNDER_CURSOR option. This will inquire on the same property.

2466 (09A2) (RC2466): MQRC_PROP_VALUE_NOT_CONVERTED

Explanation

An MQINQMP call was issued with the MQIMPO_CONVERT_VALUE option specified in the InqPropOpts parameter, but an error occurred during conversion of the value of the property. The property value is returned unconverted, the values of the ReturnedCCSID and ReturnedEncoding fields in the InqPropOpts parameter are set to those of the value returned.

Completion Code

MQCC_FAILED

Programmer Response

Check that the property value is correctly described by the ValueCCSID and ValueEncoding parameters that were specified when the property was set. Also check that these values, and the RequestedCCSID and RequestedEncoding specified in the InqPropOpts parameter of the MQINQMP call, are supported for MQ conversion. If the required conversion is not supported, conversion must be carried out by the application.

2467 (09A3) (RC2467): MQRC_PROP_TYPE_NOT_SUPPORTED

Explanation

An MQINQMP call was issued and the property inquired has an unsupported data type. A string representation of the value is returned and the TypeString field of the InqPropOpts parameter can be used to determine the data type of the property.

Completion Code

MQCC_WARNING

Programmer Response

Check whether the property value was intended to have a data type indicated by the TypeString field. If so the application must decide how to interpret the value. If not modify the application that set the property to give it a supported data type.

2469 (09A5) (RC2469): MQRC_PROPERTY_VALUE_TOO_BIG

Explanation

On an MQINQMP call, the property value was too large to fit into the supplied buffer. The DataLength field is set to the length of the property value before truncation and the Value parameter contains as much of the value as fits.

On an MQMHBUF call, the BufferLength was less than the size of the properties to be put in the buffer. In this case the call fails. The DataLength field is set to the length of the properties before truncation.

Completion Code

MQCC_WARNING

MQCC_FAILED

Programmer Response

Supply a buffer that is at least as large as DataLength if all of the property value data is required and call MQINQMP again with the MQIMPO_INQ_PROP_UNDER_CURSOR option specified.

2470 (09A6) (RC2470): MQRC_PROP_CONV_NOT_SUPPORTED

Explanation

On an MQINQMP call, the MQIMPO_CONVERT_TYPE option was specified to request that the property value be converted to the supplied data type before the call returned. Conversion between the actual and requested property data types is not supported. The Type parameter indicates the data type of the property value.

Completion Code

MQCC_FAILED

Programmer Response

Either call MQINQMP again without MQIMPO_CONVERT_TYPE specified, or request a data type for which conversion is supported.

2471 (09A7) (RC2471): MQRC_PROPERTY_NOT_AVAILABLE

Explanation

On an MQINQMP call, no property could be found that matched the specified name. When iterating through multiple properties, possibly using a name containing a wildcard character, this indicates that all properties matching the name have now been returned.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the correct property name was specified. If the name contains a wildcard character specify option MQIMPO_INQ_FIRST to begin iterating over the properties again.

2472 (09A8) (RC2472): MQRC_PROP_NUMBER_FORMAT_ERROR

Explanation

On an MQINQMP call, conversion of the property value was requested. The format of the property is invalid for conversion to the requested data type.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the correct property name and data type were specified. Ensure that the application setting the property gave it the correct format. See the documentation for the MQINQMP call for details on the formats required for data conversion of property values.

2473 (09A9) (RC2473): MQRC_PROPERTY_TYPE_ERROR

Explanation

On an MQSETMP call, the Type parameter does not specify a valid MQTYPE_* value. For properties beginning "Root.MQMD." or "JMS" the specified Type must correspond to the data type of the matching MQMD or JMS header field:

- For MQCHARn or Java String fields use MQTYPE_STRING.
- For MQLONG or Java int fields use MQTYPE_INT32.
- For MQBYTEn fields use MQTYPE_BYTE_STRING.
- For Java long fields use MQTYPE_INT64.

On an MQINQMP call, the Type parameter is not valid. Either the parameter pointer is not valid, the value is invalid, or it points to read-only storage. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Correct the parameter.

2478 (09AE) (RC2478): MQRC_PROPERTIES_TOO_BIG

Explanation

An MQPUT or MQPUT1 call was issued to put a message on a queue, but the properties of the message were too large. The length of the properties cannot exceed the value of the **`MaxPropertiesLength`** queue manager attribute. This return code will also be issued if a message with headers greater than 511 KB is put to a shared queue.

Completion Code

MQCC_FAILED

Programmer Response

Consider one of the following actions:

- Reduce the number or the size of the properties associated with the message. This could include moving some of the properties into the application data.
- Increase the value of the MaxPropertiesLength queue manager attribute.

2479 (09AF) (RC2479): MQRC_PUT_NOT_RETAINED

Explanation

An MQPUT or MQPUT1 call was issued to publish a message on a topic, using the MQPMO_RETAIN option, but the publication was unable to be retained. The publication is not published to any matching subscribers.

Completion Code

MQCC_FAILED

Programmer Response

Retained publications are stored on the SYSTEM.RETAINED.PUB.QUEUE. Ensure that this queue is available for use by the application. Possible reasons for failure include the queue being full, the queue being put inhibited, or the queue not existing.

2480 (09B0) (RC2480): MQRC_ALIAS_TARGTYPE_CHANGED

Explanation

An MQPUT or MQPUT1 call was issed to publish a message on a topic. One of the subscriptions matching this topic was made with a destination queue that was an alias queue which originally referenced a queue, but now references a topic object, which is not allowed. In this situation the reason code MQRC_ALIAS_TARGTYPE_CHANGED is returned in the Feedback field in the MQMD of a report message, or in the Reason field in the MQDLH structure of a message on the dead-letter queue.

Completion Code

MQCC_FAILED

Programmer Response

Find the subscriber that is using an alias queue which references a topic object and change it to reference a queue again, or change the subscription to reference a different queue.

2481 (09B1) (RC2481): MQRC_DMPO_ERROR

Explanation

On an MQDLTMP call, the delete message property options structure MQDMPO is not valid, for one of the following reasons:

- The StrucId field is not MQDMPO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQDMPO structure are set correctly.

2482 (09B2) (RC2482): MQRC_PD_ERROR

Explanation

On an MQSETMP or MQINQMP call, the property descriptor structure MQPD is not valid, for one of the following reasons:

- The StrucId field is not MQPD_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)
- The Context field contains an unrecognized value.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQPD structure are set correctly.

2483 (09B3) (RC2483): MQRC_CALLBACK_TYPE_ERROR

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER with an incorrect value for CallbackType

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the CallbackType field of the MQCBDO is specified correctly.

2484 (09B4) (RC2484): MQRC_CBD_OPTIONS_ERROR

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER with an incorrect value for the Options field of the MQCBD.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the Options are specified correctly.

2485 (09B5) (RC2485): MQRC_MAX_MSG_LENGTH_ERROR

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER with an incorrect value for the MaxMsgLength field of the MQCBD.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the MaxMsgLength are specified correctly.

2486 (09B6) (RC2486): MQRC_CALLBACK_ROUTINE_ERROR

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER failed for one of the following reasons:

- Both CallbackName and CallbackFunction are specified. Only one must be specified on the call.
- The call was made from an environment not supporting function pointers.
- A programming language that does not support Function pointer references.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the CallbackName value is specified correctly.

2487 (09B7) (RC2487): MQRC_CALLBACK_LINK_ERROR

Explanation

On an MQCTL call, the callback handling module (CSQBMCSM or CSQBMCSX for batch and DFHMQMCM for CICS) could not be loaded, so the adapter could not link to it.

This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the correct library concatenation has been specified in the application program execution JCL, and in the queue-manager startup JCL. Any uncommitted changes in a unit of work should be backed out. A unit of work that is coordinated by the queue manager is backed out automatically.

2488 (09B8) (RC2488): MQRC_OPERATION_ERROR

Explanation

An MQCTL or MQCB call was made with an invalid parameter.

There is a conflict with the value specified for **`Operation`** parameter.

This error can be caused by an invalid value in the **`Operation`** parameter, no registered consumers when using MQOP_START or MQOP_START_WAIT parameter, and trying to use non-threaded libraries with asynchronous API calls.

Completion Code

MQCC_FAILED

Programmer Response

Investigate the application program and verify the **`Operation`** parameter options are correct. Ensure you have link edited the application with the correct version of the threading libraries for asynchronous functions.

2489 (09B9) (RC2489): MQRC_BMHO_ERROR

Explanation

On an MQBUFMH call, the buffer to message handle options structure MQBMHO is not valid, for one of the following reasons:

- • The StrucId field is not MQBMHO_STRUC_ID.
- • The Version field specifies a value that is not valid or not supported.
- • The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQBMHO structure are set correctly.

2490 (09BA) (RC2490): MQRC_UNSUPPORTED_PROPERTY

Explanation

A message was found to contain a property that the queue manager does not support. The operation that failed required all the properties to be supported by the queue manager. This can occur on the MQPUT/MQPUT1 call or when a message is about to be sent down a channel to a queue manager than does not support message properties.

Completion Code

MQCC_FAILED

Programmer Response

Determine which property of the message is not supported by the queue manager and decide whether to remove the property from the message or connect to a queue manager which does support the property.

2492 (09BC) (RC2492): MQRC_PROP_NAME_NOT_CONVERTED

Explanation

An MQINQMP call was issued with the MQIMPO_CONVERT_VALUE option specified in the InqPropOpts parameter, but an error occurred during conversion of the returned name of the property. The returned name is unconverted

Completion Code

MQCC_WARNING

Programmer Response

Check that the character set of the returned name was correctly described when the property was set. Also check that these values, and the RequestedCCSID and RequestedEncoding specified in the InqPropOpts parameter of the MQINQMP call, are supported for MQ conversion. If the required conversion is not supported, conversion must be carried out by the application.

2494 (09BE) (RC2494): MQRC_GET_ENABLED

Explanation

This reason code is returned to an asynchronous consumer at the time a queue that was previously inhibited for get has been re-enabled for get.

Completion Code

MQCC_WARNING

Programmer Response

None. This reason code is used to inform the application of the change in state of the queue.

2495 (09BF) (RC2495): MQRC_MODULE_NOT_FOUND

Explanation

A native shared library could not be loaded.

Completion Code

MQCC_FAILED

Programmer Response

This problem could be caused by either of the two following reasons:

- A MQCB call was made with an Operation of MQOP_REGISTER specifying a *CallbackName* which could not be found. Ensure that the *CallbackName* value is specified correctly.
- The Java MQ code could not load a Java native shared library. This error can occur if a Java application is running in a 32-bit JRE but has been configured to load the 64-bit Java Native Libraries. Check the associated Exception stack and FFST. Ensure that the JNI shared library is specified correctly. Check also that you have specified `-Djava.library.path=/opt/mqm/java/lib`, or equivalent, when invoking the Java program.

Related information

The Java Native Interface (JNI) libraries required by IBM MQ classes for JMS applications

2496 (09C0) (RC2496): MQRC_MODULE_INVALID

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER, specifying a CallbackName which is not a valid load module.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the CallbackName value is specified correctly.

2497 (09C1) (RC2497): MQRC_MODULE_ENTRY_NOT_FOUND

Explanation

An MQCB call was made with an Operation of MQOP_REGISTER and the CallbackName identifies a function name which can't be found in the specified library.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the CallbackName value is specified correctly.

2498 (09C2) (RC2498): MQRC_MIXED_CONTENT_NOT_ALLOWED

Explanation

An attempt was made to set a property with mixed content. For example, if an application set the property "x.y" and then attempted to set the property "x.y.z" it is unclear whether in the property name hierarchy "y" contains a value or another logical grouping. Such a hierarchy would be "mixed content" and this is not supported. Setting a property which would cause mixed content is not allowed. A hierarchy within a property name is created using the "." character (U+002E).

Completion Code

MQCC_FAILED

Programmer Response

Valid property names are described in the IBM MQ documentation. Change the property name hierarchy so that it no longer contains mixed content before re-issuing the call.

2499 (09C3) (RC2499): MQRC_MSG_HANDLE_IN_USE

Explanation

A message property call was called (MQCRTMH, MQDLTMH, MQSETMP, MQINQMP, MQDLTMP or MQMHBUF) specifying a message handle that is already in use on another API call. A message handle may only be used on one call at a time.

Concurrent use of a message handle can arise, for example, when an application uses multiple threads.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the message handle cannot be used while another call is in progress.

2500 (09C4) (RC2500): MQRC_HCONN_ASYNC_ACTIVE

Explanation

An attempt to issue an MQI call has been made while the connection is started.

Completion Code

MQCC_FAILED

Programmer Response

Stop or suspend the connection using the MQCTL call and retry the operation.

2501 (09C5) (RC2501): MQRC_MHBO_ERROR

Explanation

On an MQMHBUF call, the message handle to buffer options structure MQMHBO is not valid, for one of the following reasons:

- The StrucId field is not MQMHBO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQMHBO structure are set correctly.

2502 (09C6) (RC2502): MQRC_PUBLICATION_FAILURE

Explanation

An MQPUT or MQPUT1 call was issued to publish a message on a topic. Delivery of the publication to one of the subscribers failed and due to the combination of the syncpoint option used and either:

- • The PMSGDLV attribute on the administrative TOPIC object if it was a persistent message.
- • The NPMSGDLV attribute on the administrative TOPIC object if it was a non-persistent message.

The publication has not been delivered to any of the subscribers.

Completion Code

MQCC_FAILED

Programmer response

Find the subscriber or subscribers who are having problems with their subscription queue and resolve the problem, or change the setting of the PMSGDLV or NPMSGDLV attributes on the TOPIC so that problems with one subscriber do not have an effect on other subscribers. Retry the MQPUT.

2503 (09C7) (RC2503): MQRC_SUB_INHIBITED

Explanation

MQSUB calls are currently inhibited for the topic subscribed to.

Completion Code

MQCC_FAILED

Programmer Response

If the system design allows subscription requests to be inhibited for short periods, retry the operation later.

2504 (09C8) (RC2504): MQRC_SELECTOR_ALWAYS_FALSE

Explanation

An MQOPEN, MQPUT1 or MQSUB call was issued but a selection string was specified which will never select a message

Completion Code

MQCC_FAILED

Programmer Response

Verify that the logic of the selection string which was passed in on the API is as expected. Make any necessary corrections to the logic of the string and resubmit the MQ API call for which the message occurred.

2507 (09CB) (RC2507): MQRC_XEPO_ERROR

Explanation

On an MQXEP call, the exit options structure MQXEPO is not valid, for one of the following reasons:

- The StrucId field is not MQXEPO_STRUC_ID.
- The Version field specifies a value that is not valid or not supported.
- The parameter pointer is not valid. (It is not always possible to detect parameter pointers that are not valid; if not detected, unpredictable results occur.)

Completion Code

MQCC_FAILED

Programmer Response

Ensure that input fields in the MQXEPO structure are set correctly.

2509 (09CD) (RC2509): MQRC_DURABILITY_NOT_ALTERABLE

Explanation

An MQSUB call using option MQSO_ALTER was made changing the durability of the subscription. The durability of a subscription cannot be changed.

Completion Code

MQCC_FAILED

Programmer Response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly, or change the durability option used on the MQSUB call so that it matches the existing subscription.

2510 (09CE) (RC2510): MQRC_TOPIC_NOT_ALTERABLE

Explanation

An MQSUB call using option MQSO_ALTER was made changing the one or more of the fields in the MQSD that provide the topic being subscribed to. These fields are the ObjectName, ObjectString, or wildcard options. The topic subscribed to cannot be changed.

Completion Code

MQCC_FAILED

Programmer Response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly, or change the attributes and options used on the MQSUB call so that it matches the existing subscription.

2512 (09D0) (RC2512): MQRC_SUBLEVEL_NOT_ALTERABLE

Explanation

An MQSUB call using option MQSO_ALTER was made changing the SubLevel of the subscription. The SubLevel of a subscription cannot be changed.

Completion Code

MQCC_FAILED

Programmer Response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly, or change the SubLevel field used on the MQSUB call so that it matches the existing subscription.

2513 (09D1) (RC2513): MQRC_PROPERTY_NAME_LENGTH_ERR

Explanation

An attempt was made to set, inquire or delete a property with an invalid name. This is for one of the following reasons:

- The VSLength field of the property name was set to less than or equal to zero.

- The VSLength field of the property name was set to greater than the maximum allowed value (see constant MQ_MAX_PROPERTY_NAME_LENGTH).
- The VSLength field of the property name was set to MQVS_NULL_TERMINATED and the property name was greater than the maximum allowed value.

Completion Code

MQCC_FAILED

Programmer Response

Valid property names are described in the IBM MQ documentation. Ensure that the property has a valid name length before issuing the call again.

2514 (09D2) (RC2514): MQRC_DUPLICATE_GROUP_SUB

Explanation

An MQSUB call using option MQSO_GROUP_SUB was made creating a new grouped subscription but, although it has a unique SubName, it matches the Full topic name of an existing subscription in the group.

Completion Code

MQCC_FAILED

Programmer Response

Correct the Full topic name used so that it does not match any existing subscription in the group, or correct the grouping attributes if, either a different group was intended or the subscription was not intended to be grouped at all.

2515 (09D3) (RC2515): MQRC_GROUPING_NOT_ALTERABLE

Explanation

An MQSUB call was made using option MQSO_ALTER on a grouped subscription, that is one made with the option MQSO_GROUP_SUB. Grouping of subscriptions is not alterable.

Completion Code

MQCC_FAILED

Programmer response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly, or change the various grouping fields used on the MQSUB call so that it matches the existing subscription.

2516 (09D4) (RC2516): MQRC_SELECTOR_INVALID_FOR_TYPE

Explanation

A SelectionString may only be specified in the MQOD for an MQOPEN/MQPUT1 if the following is true:

- ObjectType is MQOT_Q
- The queue is being opened using one of the MQOO_INPUT_* open options.

Completion Code

MQCC_FAILED

Programmer Response

Modify the value of ObjectType to be MQOT_Q and ensure that the queue is being opened using one of the MQOO_INPUT_* options.

2517 (09D5) (RC2517): MQRC_HOBJ_QUIESCED

Explanation

The HOBJ has been quiesced but there are no messages in the read ahead buffer which match the current selection criteria. This reason code indicates that the read ahead buffer is not empty.

Completion Code

MQCC_FAILED

Programmer Response

This reason code indicates that all messages with the current selection criteria have been processed. Do one of the following:

- If no further messages need to be processed issue an MQCLOSE without the MQCO_QUIESCE option. Any messages in the read ahead buffer will be discarded.
- Relax the current selection criteria by modifying the values in the MQGMO and reissue the call. Once all messages have been consumed the call will return MQRC_HOBJ_QUIESCED_NO_MSGS.

2518 (09D6) (RC2518): MQRC_HOBJ_QUIESCED_NO_MSGS

Explanation

The HOBJ has been quiesced and the read ahead buffer is now empty. No further messages will be delivered to this HOBJ

Completion Code

MQCC_FAILED

Programmer Response

Issue MQCLOSE against the HOBJ.

2519 (09D7) (RC2519): MQRC_SELECTION_STRING_ERROR

Explanation

The SelectionString must be specified according to the description of how to use an MQCHARV structure. Examples of why this error was returned:

- SelectionString.VSLength is greater than zero, but SelectionString.VSOffset is zero and SelectionString.VSPtr is a null pointer.
- SelectionString.VSOffset is nonzero and SelectionString.VSPtr is not the null pointer (that is, it appears both fields are being used where only one is allowed).
- SelectionString.VSPtr is not a valid pointer.
- SelectionString.VSOffset or SelectionString.VSPtr points to storage that is not accessible.

- SelectionString.VSLength exceeds the maximum length allowed for this field. The maximum length is determined by MQ_SELECTOR_LENGTH.

Completion Code

MQCC_FAILED

Programmer Response

Modify the fields of the MQCHARV so that it follows the rules for a valid MQCHARV structure.

2520 (09D8) (RC2520): MQRC_RES_OBJECT_STRING_ERROR

Explanation

On the MQOPEN or MQPUT1 call in the Object Descriptor MQOD, or on the MQSUB call in the Subscription Descriptor MQSD the ResObjectString field is not valid.

One of the following applies:

- ResObjectString.VSLength is greater than zero, but ResObjectString.VSOffset is zero and ResObjectString.VSPtr is the null pointer.
- ResObjectString.VSOffset is nonzero and ResObjectString.VSPtr is not the null pointer (that is, it appears both fields are being used where only one is allowed).
- ResObjectString.VSPtr is not a valid pointer.
- ResObjectString.VSOffset or ResObjectString.VSPtr points to storage that is not accessible.
- ResObjectString.VSBufSize is MQVS_USE_VSLENGTH and one of ResObjectString.VSOffset or ResObjectString.VSPtr have been provided.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that one of ResObjectString.VSOffset or ResObjectString.VSPtr is zero and the other nonzero and that the buffer length is provided in ResObjectString.VSBufSize. Ensure that the field used points to accessible storage.

2521 (09D9) (RC2521): MQRC_CONNECTION_SUSPENDED

Explanation

An MQCTL call with Operation MQOP_START_WAIT has returned because the asynchronous consumption of messages has been suspended. This can be for the following reasons:

- The connection was explicitly suspended using MQCTL with Operation MQOP_SUSPEND
- All consumers have been either unregistered or suspended.

Completion Code

MQCC_WARNING

Programmer Response

If this is an expected condition, no corrective action required. If this is an unexpected condition check that:

- At least one consumer is registered and not suspended
- The connection has not been suspended

2522 (09DA) (RC2522): MQRC_INVALID_DESTINATION

Explanation

An MQSUB call failed because of a problem with the destination where publications messages are to be sent, so an object handle cannot be returned to the application and the subscription is not made. This can be for one of the following reasons:

- The MQSUB call used MQSO_CREATE, MQSO_MANAGED and MQSO_NON_DURABLE and the model queue referred to by MNDURMDL on the administrative topic node does not exist
- The MQSUB call used MQSO_CREATE, MQSO_MANAGED and MQSO_DURABLE and the model queue referred to by MDURMDL on the administrative topic node does not exist, or has been defined with a DEFTYPE of TEMPDYN.
- The MQSUB call used MQSO_CREATE or MQSO_ALTER on a durable subscription and the object handle provided referred to a temporary dynamic queue. This is not an appropriate destination for a durable subscription.
- The MQSUB call used MQSO_RESUME and a Hobj of MQHO_NONE, to resume an administratively created subscription, but the queue name provided in the DEST parameter of the subscription does not exist.
- The MQSUB call used MQSO_RESUME and a Hobj of MQHO_NONE, to resume a previously created API subscription, but the queue previously used no longer exists.

Completion Code

MQCC_FAILED

Programmer Response

Ensure that the model queues referred to by MNDURMDL and MDURMDL exist and have an appropriate DEFTYPE. Create the queue referred to by the DEST parameter in an administrative subscription if one is being used. Alter the subscription to use an existing queue if the previously used one does not exist.

2523 (09DB) (RC2523): MQRC_INVALID_SUBSCRIPTION

Explanation

An MQSUB call using MQSO_RESUME or MQSO_ALTER failed because the subscription named is not valid for use by applications. This can be for one of the following reasons:

- The subscription is the SYSTEM.DEFAULT.SUB subscription which is not a valid subscription and should only be used to fill in the default values on DEFINE SUB commands.
- The subscription is a proxy type subscription which is not a valid subscription for an application to resume and is only used to enable publications to be forwarded between queue managers.
- The subscription has expired and is no longer valid for use.

Completion Code

MQCC_FAILED

Programmer Response

Ensure the subscription named in SubName field is not one of the invalid ones listed. If you have a handle open to the subscription already it must have expired. Use MQCLOSE to close the handle and then if necessary create a new subscription.

2524 (09DC) (RC2524): MQRC_SELECTOR_NOT_ALTERABLE

Explanation

An MQSUB call was issued with the MQSO_ALTER option and the MQSD contained a SelectionString. It is not valid to alter the SelectionString of a subscription.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the SelectionString field of the MQSD does not contain a valid VSPtr and that the VSLength is set to zero when making a call to MQSUB.

2525 (09DD) (RC2525): MQRC_RETAINED_MSG_Q_ERROR

Explanation

An MQSUB call which did not use the MQSO_NEW_PUBLICATIONS_ONLY option, or an MQSUBRQ call, failed because the retained publications which exist for the topic string subscribed to cannot be retrieved from the SYSTEM.RETAINED.PUB.QUEUE. This can be for one of the following reasons:

- The queue has become damaged or has been deleted.
- The queue has been set to GET(DISABLED).
- Messages have been removed from this queue directly.

An error message will be written to the log giving more details about the problem with the SYSTEM.RETAINED.PUB.QUEUE.

When this return code occurs on an MQSUB call, it can only occur using the MQSO_CREATE option, and in this case the subscription is not created.

Completion Code

MQCC_FAILED

Programmer Response

If this occurs on an MQSUB call, re-issue the MQSUB call using the option MQSO_NEW_PUBLICATIONS_ONLY, which will mean no previously retained publications are sent to this subscription, or fix the SYSTEM.RETAINED.PUB.QUEUE so that messages can be retrieved from it and re-issue the MQSUB call.

If this occurs on an MQSUBRQ call, fix the SYSTEM.RETAINED.PUB.QUEUE so that messages can be retrieved from it and re-issue the MQSUBRQ call.

2526 (09DE) (RC2526): MQRC_RETAINED_NOT_DELIVERED

Explanation

An MQSUB call which did not use the MQSO_NEW_PUBLICATIONS_ONLY option or an MQSUBRQ call, failed because the retained publications which exist for the topic string subscribed to cannot be delivered to the subscription destination queue and have subsequently failed to be delivered to the dead-letter queue.

When this return code occurs on an MQSUB call, it can only occur using the MQSO_CREATE option, and in this case the subscription is not created.

Completion Code

MQCC_FAILED

Programmer response

Fix the problems with the destination queue and the dead-letter queue and re-issue the MQSUB or MQSUBRQ call.

2527 (09DF) (RC2527): MQRC_RFH_RESTRICTED_FORMAT_ERR

Explanation

A message was put to a queue containing an MQRFH2 header which included a folder with a restricted format. However, the folder was not in the required format. These restrictions are:

- If NameValueCCSID of the folder is 1208 then only single byte UTF-8 characters are allowed in the folder, group or element names.
- Groups are not allowed in the folder.
- The values of properties may not contain any characters that require escaping.
- Only Unicode character U+0020 will be treated as white space within the folder.
- The folder tag does not contain the content attribute.
- The folder must not contain a property with a null value.

The <mq> folder requires formatting of this restricted form.

Completion Code

MQCC_FAILED

Programmer Response

Change the message to include valid MQRFH2 folders.

2528 (09E0) (RC2528): MQRC_CONNECTION_STOPPED

Explanation

An MQCTL call was issued to start the asynchronous consumption of messages, but before the connection was ready to consume messages it was stopped by one of the message consumers.

Completion Code

MQCC_FAILED

Programmer Response

If this is an expected condition, no corrective action required. If this is an unexpected condition check whether an MQCTL with Operation MQOP_STOP was issued during the MQCBCT_START callback function.

2529 (09E1) (RC2529): MQRC_ASYNC_UOW_CONFLICT

Explanation

An MQCTL call with Operation MQOP_START was issued to start the asynchronous consumption of messages, but the connection handle used already has a global unit of work outstanding. MQCTL cannot be used to start asynchronous consumption of messages while a unit of work is in existence unless the MQOP_START_WAIT Operation is used

Completion Code

MQCC_FAILED

Programmer Response

Issue an MQCMIT on the connection handle to commit the unit of work and then reissue the MQCTL call, or issue an MQCTL call using Operation MQOP_START_WAIT to use the unit of work from within the asynchronous consumption callback functions.

2530 (09E2) (RC2530): MQRC_ASYNC_XA_CONFLICT

Explanation

An MQCTL call with Operation MQOP_START was issued to start the asynchronous consumption of messages, but an external XA syncpoint coordinator has already issued an xa_open call for this connection handle. XA transactions must be done using the MQOP_START_WAIT Operation.

Completion Code

MQCC_FAILED

Programmer Response

Reissue the MQCTL call using Operation MQOP_START_WAIT.

2531 (09E3) (RC2531): MQRC_PUBSUB_INHIBITED

Explanation

MQSUB, MQOPEN, MQPUT, and MQPUT1 calls are currently inhibited for all publish/subscribe topics, either with the queue manager attribute PSMODE or because processing of publish/subscribe state at queue manager start-up has failed, or has not yet completed.

Completion Code

MQCC_FAILED

Programmer Response

If this queue manager does not intentionally inhibit publish/subscribe, investigate any error messages that describe the failure at queue manager start-up, or wait until start-up processing completes. If the queue manager is a member of cluster, then start-up is not complete until the channel initiator has also started. On z/OS, if you get this return code from the Chinit for the SYSTEM.BROKER.DEFAULT.STREAM queue or topic, then the Chinit is busy processing work, and the pubsub task starts later. Use the DISPLAY PUBSUB command to check the status of the publish/subscribe engine to ensure that it is ready for use. Additionally, on z/OS you might receive an information message CSQM076I.

2532 (09E4) (RC2532): MQRC_MSG_HANDLE_COPY_FAILURE

Explanation

An MQGET call was issued specifying a valid `MsgHandle` in which to retrieve any properties of the message. After the message had been removed from the queue the application could not allocate enough storage for the properties of the message. The message data is available to the application but the properties are not. Check the queue manager error logs for more information about how much storage was required.

Completion Code

MQCC_WARNING

Programmer response

Raise the memory limit of the application to allow it store the properties.

2533 (09E5) (RC2533): MQRC_DEST_CLASS_NOT_ALTERABLE

Explanation

An MQSUB call using option MQSO_ALTER was made changing the use of the MQSO_MANAGED option on the subscription. The destination class of a subscription cannot be changed. When the MQSO_MANAGED option is not used, the queue provided can be changed, but the class of destination (managed or not) cannot be changed.

Completion Code

MQCC_FAILED

Programmer Response

Remove the subscription using MQCLOSE and re-create it with MQSUB with the attributes set correctly, or change the use of the MQSO_MANAGED option used on the MQSUB call so that it matches the existing subscription.

2534 (09E6) (RC2534): MQRC_OPERATION_NOT_ALLOWED

Explanation

An MQCTL call was made with an Operation that is not allowed because of the state of asynchronous consumption on the hConn is currently in.

If Operation was MQOP_RESUME, the operation is not allowed because the state of asynchronous consumption on the hConn is STOPPED. Re-issue MQCTL with the MQOP_START Operation.

If Operation was MQOP_SUSPEND, the operation is not allowed because the state of asynchronous consumption on the hConn is STOPPED. If you need to get your hConn into a SUSPENDED state, issue MQCTL with the MQOP_START Operation followed by MQCTL with MQOP_SUSPEND.

If Operation was MQOP_START, the operation is not allowed because the state of asynchronous consumption on the hConn is SUSPENDED. Re-issue MQCTL with the MQOP_RESUME Operation.

If Operation was MQOP_START_WAIT, the operation is not allowed because either:

- The state of asynchronous consumption on the hConn is SUSPENDED. Re-issue MQCTL with the MQOP_RESUME Operation.
- The state of asynchronous consumption on the hConn is already STARTED. Do not mix the use of MQOP_START and MQOP_START_WAIT within one application.

Completion Code

MQCC_FAILED

Programmer Response

Re-issue the MQCTL call with the correct Operation.

2535 (09E7): MQRC_ACTION_ERROR

Explanation

An MQPUT call was issued, but the value of the Action field in the PutMsgOpts parameter is not a valid MQACTP_* value.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid value for the field.

2537 (09E9) (RC2537): MQRC_CHANNEL_NOT_AVAILABLE

Explanation

An MQCONN call was issued from a client to connect to a queue manager but the channel is not currently available. Common causes of this reason code are:

- The channel is currently in stopped state.
- The channel has been stopped by a channel exit.
- The queue manager has reached its maximum allowable limit for this channel from this client.
- The queue manager has reached its maximum allowable limit for this channel.
- The queue manager has reached its maximum allowable limit for all channels.

Completion Code

MQCC_FAILED

Programmer Response

Examine the queue manager and client error logs for messages explaining the cause of the problem.

This reason code is also used to identify the corresponding event message Channel Not Available.

2538 (09EA) (RC2538): MQRC_HOST_NOT_AVAILABLE

Explanation

An MQCONN call was issued from a client to connect to a queue manager but the attempt to allocate a conversation to the remote system failed. Common causes of this reason code are:

- The listener has not been started on the remote system.
- The connection name in the client channel definition is incorrect.
- The network is currently unavailable.
- A firewall blocking the port, or protocol-specific traffic.
- The security call initializing the IBM MQ client is blocked by a security exit on the SVRCONN channel at the server.

Completion Code

MQCC_FAILED

Programmer Response

Examine the client error log for messages explaining the cause of the problem.

If you are using a Linux server, and receiving a 2538 return code when trying to connect to a queue manager, ensure that you check your internal firewall configuration.

To diagnose the problem, issue the following commands to temporarily turn off the internal Linux firewall :

```
/etc/init.d/iptables save
/etc/init.d/iptables stop
```

To turn the internal Linux firewall back on, issue the command:

```
/etc/init.d/iptables start
```

To permanently turn off the internal Linux firewall, issue the command:

```
chkconfig iptables off
```

2539 (09EB) (RC2539): MQRC_CHANNEL_CONFIG_ERROR

Explanation

An MQCONN call was issued from a client to connect to a queue manager but the attempt to establish communication failed. Common causes of this reason code are:

- The server and client cannot agree on the channel attributes to use.
- There are errors in one or both of the QM.INI or MQCLIENT.INI configuration files.
- The server machine does not support the code page used by the client.

Completion Code

MQCC_FAILED

Programmer Response

Examine the queue manager and client error logs for messages explaining the cause of the problem.

2540 (09EC) (RC2540): MQRC_UNKNOWN_CHANNEL_NAME

Explanation

An MQCONN call was issued from a client to connect to a queue manager but the attempt to establish communication failed because the queue manager did not recognize the channel name.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the client is configured to use the correct channel name.

2541 (09ED) (RC2541): MQRC_LOOPING_PUBLICATION

Explanation

A Distributed Pub/Sub topology has been configured with a combination of Pub/Sub clusters and Pub/Sub Hierarchies such that some, or all, of the queue managers have been connected in a loop. A looping publication has been detected and put onto the dead-letter queue.

Completion Code

MQCC_FAILED

Programmer response

Examine the hierarchy and correct the loop.

2543 (09EF) (RC2543): MQRC_STANDBY_Q_MGR

Explanation

The application attempted to connect to a standby queue manager instance.

Standby queue manager instances do not accept connections. To connect to the queue manager, you must connect to its active instance.

Completion Code

MQCC_FAILED

Programmer response

Connect the application to an active queue manager instance.

2544 (09F0) (RC2544): MQRC_RECONNECTING

Explanation

The connection has started reconnecting.

If an event handler has been registered with a reconnecting connection, it is called with this reason code when reconnection attempts begin.

Completion Code

MQCC_WARNING

Programmer response

Let IBM MQ continue with its next reconnection attempt, change the interval before the reconnection, or stop the reconnection. Change any application state that depends on the reconnection.

Note: Reconnection might start while the application is in the middle of an MQI call.

2545 (09F1) (RC2545): MQRC_RECONNECTED

Explanation

The connection reconnected successfully and all handles are reinstated.

If reconnection succeeds, an event handler registered with the connection is called with this reason code.

Completion Code

MQCC_OK

Programmer response

Set any application state that depends on the reconnection.

Note: Reconnection might finish while the application is in the middle of an MQI call.

2546 (09F2) (RC2546): MQRC_RECONNECT_QMID_MISMATCH

Explanation

A reconnectable connection specified MQCNO_RECONNECT_Q_MGR and the connection attempted to reconnect to a different queue manager.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the configuration for a reconnectable client resolves to a single queue manager.

If the application does not require reconnection to exactly the same queue manager, use the `MQCONNX` option MQCNO_RECONNECT.

2547 (09F3) (RC2547): MQRC_RECONNECT_INCOMPATIBLE

Explanation

An MQI option is incompatible with reconnectable connections.

This error indicates that the option relies on information in a queue manager that is lost during reconnection. For example, the option MQPMO_LOGICAL_ORDER, requires the queue manager to remember information about logical message ordering that is lost during reconnection.

Completion Code

MQCC_FAILED

Programmer response

Modify your application to remove the incompatible option, or do not allow the application to be reconnectable.

2548 (09F4) (RC2548): MQRC_RECONNECT_FAILED

Explanation

After reconnecting, an error occurred while reinstating the handles for a reconnectable connection.

For example, an attempt to reopen a queue that had been open when the connection broke, failed.

Completion Code

MQCC_FAILED

Programmer response

Investigate the cause of the error in the error logs. Consider using the MQSTAT API to find further details of the failure.

2549 (09F5) (RC2549): MQRC_CALL_INTERRUPTED

Explanation

`MQPUT`, `MQPUT1`, or `MQCMIT` was interrupted and reconnection processing cannot reestablish a definite outcome.

This reason code is returned to a client that is using a re-connectable connection if the connection is broken between sending the request to the queue manager and receiving the response, and if the outcome is not certain. For example, an interrupted MQPUT of a message outside sync point might or might not have stored the message. Alternatively an interrupted MQPUT1 of a message outside sync point might or might not have stored the message. The timing of the failure affects whether the message remains on the queue or not. If MQCMIT was interrupted the transaction might or might not have been committed.

Completion Code

MQCC_FAILED

Programmer response

Repeat the call following reconnection, but be aware that in some cases, repeating the call might be misleading.

The application design determines the appropriate recovery action. In many cases, getting and putting messages inside sync point resolves indeterminate outcomes. Where messages need to be processed outside sync point, it might be necessary to establish whether the interrupted operation succeeded before the interruption and repeating it if it did not.

2550 (09F6) (RC2550): MQRC_NO_SUBS_MATCHED

Explanation

An MQPUT or MQPUT1 call was successful but no subscriptions matched the topic.

Completion Code

MQCC_WARNING

Programmer response

No response is required, unless this reason code was not expected by the application that put the message.

2551 (09F7) (RC2551): MQRC_SELECTION_NOT_AVAILABLE

Explanation

An MQSUB call subscribed to publications using a SelectionString. IBM MQ is unable to accept the call because it does not follow the rules for specifying selection strings, which are documented in Message selector syntax. It is possible that the selection string is acceptable to an extended message selection provider, however no extended message selection provider was available to validate the selection string. If a subscription is being created, the MQSUB fails; otherwise MQSUB completes with a warning.

An MQPUT or MQPUT1 call published a message and at least one subscriber had a content filter but IBM MQ could not determine whether the publication should be delivered to the subscriber (for example, because no extended message selection provider was available to validate the selection string). The MQPUT or MQPUT1 call will fail with MQRC_SELECTION_NOT_AVAILABLE and no subscribers will receive the publication.

Completion Code

MQCC_WARNING or MQCC_FAILED

Programmer response

If it was intended that the selection string should be handled by the extended message selection provider, ensure that the extended message selection provider is correctly configured and running. If extended message selection was not intended, see Message selector syntax and ensure that you have correctly followed the rules for specifying selection strings.

If a subscription is being resumed, the subscription will not be delivered any messages until a extended message selection provider is available and a message matches the SelectionString of the resumed subscription.

2552 (09F8) (RC2552): MQRC_CHANNEL_SSL_WARNING

Explanation

An SSL security event has occurred. This is not fatal to an SSL connection but is likely to be of interest to an administrator.

Completion code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel SSL Warning.

2553 (09F9) (RC2553): MQRC_OCSP_URL_ERROR

Explanation

The OCSPResponderURL field does not contain a correctly formatted HTTP URL.

Completion code

MQCC_FAILED

Programmer response

Check and correct the OCSPResponderURL. If you do not intend to access an OCSP responder, set the AuthInfoType of the authentication information object to MQAIT_CRL_LDAP.

2554 (09FA) (RC2554): MQRC_CONTENT_ERROR

Explanation

There are 2 explanations for reason code 2554:

1. An MQPUT call was issued with a message where the content could not be parsed to determine whether the message should be delivered to a subscriber with an extended message selector. No subscribers will receive the publication.
2. MQRC_CONTENT_ERROR can be returned from MQSUB and MQSUBRQ if a selection string selecting on the content of the message was specified.

Completion Code

MQCC_FAILED

Programmer response

There are 2 programmer responses for reason code 2554 because there are two causes:

1. If reason code 2554 was issued because of reason "1" on page 227 then check for error messages from the extended message selection provider and ensure that the message content is well formed before retrying the operation.
2. If reason code 2554 was issued because of reason "2" on page 227 then because the error occurred at the time that the retained message was published, either a system administrator must clear the retained queue, or you cannot specify a selection string selecting on the content.

2555 (09FB) (RC2555): MQRC_RECONNECT_Q_MGR_REQD

Explanation

The `MQCNO_RECONNECT_Q_MGR` option is required.

An option, such `MQMO_MATCH_MSG_TOKEN` in an `MQGET` call or opening a durable subscription, was specified in the client program that requires re-connection to the same queue manager.

Completion Code

`MQCC_FAILED`

Programmer response

Change the `MQCONNX` call to use `MQCNO_RECONNECT_Q_MGR`, or modify the client program not to use the conflicting option.

2556 (09FC) (RC2556): MQRC_RECONNECT_TIMED_OUT

Explanation

A reconnection attempt timed out.

The failure might occur in any MQI verb if a connection is configured to reconnect. You can customize the timeout in the MQClient.ini file

Completion Code

MQCC_FAILED

Programmer response

Look at the error logs to find out why reconnection did not complete within the time limit.

2557 (09FD) (RC2557): MQRC_PUBLISH_EXIT_ERROR

Explanation

A publish exit function returned an invalid response code, or failed in some other way. This can be returned from the MQPUT, MQPUT1, MQSUB and MQSUBRQ function calls. This reason code does not occur on IBM MQ for z/OS.

Completion Code

MQCC_FAILED

Programmer response

Check the publish exit logic to ensure that the exit is returning valid values in the ExitResponse field of the MQPSXP structure. Consult the IBM MQ error log files and FFST records for more details about the problem.

2558 (09FE) (RC2558): MQRC_COMMINFO_ERROR

Explanation

The configuration of either the name of the COMMINFO object or the object itself is incorrect.

Completion Code

MQCC_FAILED

Programmer response

Check the configuration of the TOPIC and COMMINFO objects and retry the operation.

2560 (0A00) (RC2560): MQRC_MULTICAST_ONLY

Explanation

An attempt was made to use a topic which is defined as multicast only in a non-multicast way. Possible causes for this error are:

1. An MQPUT1 call was issued to the topic
2. An MQOPEN call was issued using the MQOO_NO_MULTICAST option
3. An MQSUB call was issued using the MQSO_NO_MULTICAST option
4. The application is connected directly through bindings, that is, there is no client connection
5. The application is being run from a release prior to Version 7.1

Completion Code

MQCC_FAILED

Programmer response

Either change the topic definition to enable non-multicast, or change the application.

2561 (0A01) (RC2561): MQRC_DATA_SET_NOT_AVAILABLE

Explanation

An IBM MQI call or command was issued to operate on a shared queue, but the call failed because the data for the shared message has been offloaded to a shared message data set that is temporarily unavailable to the current queue manager. This can occur either because of a problem in accessing the data set or because the data set was previously found to be damaged, and is awaiting completion of recovery processing.

This return code can also occur if the shared message data set has not been defined for the queue manager being used. You might be using the wrong queue manager in the queue-sharing group.

- This reason code occurs only on z/OS.

Completion Code

MQCC_FAILED

Programmer response

The problem is temporary; wait a short while, and then retry the operation.

Use `DIS CFSTRUCT(...) SMDSCONN(*)` to display the status of the SMDS connection.

To start the connection if the STATUS is not OPEN, use `STA SMDSCONN(*) CFSTRUCT(...)`.

Use `DISPLAY CFSTATUS(...) TYPE(SMDS)` and check the status is active on the queue manager that you are using.

2562 (0A02) (RC2562): MQRC_GROUPING_NOT_ALLOWED

Explanation

An MQPUT call was issued to put a grouped message to a handle which is publishing over multicast.

Completion Code

MQCC_FAILED

Programmer response

Either change the topic definition to disable multicast or change the application to not use grouped messages.

2563 (0A03) (RC2563): MQRC_GROUP_ADDRESS_ERROR

Explanation

An MQOPEN or MQSUB call was issued to a multicast topic which has been defined with an incorrect group address field.

Completion Code

MQCC_FAILED

Programmer response

Correct the group address field in the COMMINFO definition linked to the TOPIC object.

2564 (0A04) (RC2564): MQRC_MULTICAST_CONFIG_ERROR

Explanation

An MQOPEN, MQSUB or MQPUT call was issued which invoked the multicast component. The call failed because the multicast configuration is incorrect.

Completion Code

MQCC_FAILED

Programmer response

Check the multicast configuration and error logs and retry the operation.

2565 (0A05) (RC2565): MQRC_MULTICAST_INTERFACE_ERROR

Explanation

An MQOPEN, MQSUB or MQPUT call was made which attempted to a network interface for multicast. The interface returned an error. Possible causes for the error are:

1. The required network interface does not exist.
2. The interface is not active.
3. The interface does not support the required IP version.

Completion Code

MQCC_FAILED

Programmer response

Verify that the IP address and the system network configuration are valid. Check the multicast configuration and error logs and retry the operation.

2566 (0A06) (RC2566): MQRC_MULTICAST_SEND_ERROR

Explanation

An MQPUT call was made which attempted to send multicast traffic over the network. The system failed to send one or more network packets.

Completion Code

MQCC_FAILED

Programmer response

Verify that the IP address and the system network configuration are valid. Check the multicast configuration and error logs and retry the operation.

2567 (0A07) (RC2567): MQRC_MULTICAST_INTERNAL_ERROR

Explanation

An MQOPEN, MQSUB or MQPUT call was issued which invoked the multicast component. An internal error occurred which prevented the operation completing successfully.

Completion Code

MQCC_FAILED

Programmer response

Tell the systems administrator.

2568 (0A08) (RC2568): MQRC_CONNECTION_NOT_AVAILABLE

Explanation

An MQCONN or MQCONNX call was made when the queue manager was unable to provide a connection of the requested connection type on the current installation. A client connection cannot be made on a server only installation. A local connection cannot be made on a client only installation.

This error can also occur when IBM MQ fails an attempt to load a library from the installation that the requested queue manager is associated with.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the connection type requested is applicable to the type of installation. If the connection type is applicable to the installation then consult the error log for more information about the nature of the error.

2569 (0A09) (RC2569): MQRC_SYNCPOINT_NOT_ALLOWED

Explanation

An MQPUT or MQPUT1 call using MQPMO_SYNCPOINT was made to a topic that is defined as MCAST(ENABLED). This is not allowed.

Completion Code

MQCC_FAILED

Programmer response

Change the application to use MQPMO_NO_SYNCPOINT, or alter the topic to disable the use of Multicast and retry the operation.

2577 (0A11) (RC2577): MQRC_CHANNEL_BLOCKED

Explanation

An inbound channel attempted to connect to the queue manager but was blocked due to matching a Channel Authentication rule.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Blocked.

2578 (0A12) (RC2578): MQRC_CHANNEL_BLOCKED_WARNING

Explanation

An inbound channel attempted to connect to the queue manager and would have been blocked due to matching a Channel Authentication rule, however the rule was defined with WARN(YES) so the rule did not block the connection.

Completion Code

MQCC_WARNING

Programmer response

None. This reason code is only used to identify the corresponding event message Channel Blocked.

2583 (0A17) (RC2583): MQRC_INSTALLATION_MISMATCH

Explanation

The application attempted to connect to a queue manager that is not associated with the same IBM MQ installation as the loaded libraries.

Completion Code

MQCC_FAILED

Programmer response

An application must use the libraries from the installation the queue manager is associated with. If the *AMQ_SINGLE_INSTALLATION* environment variable is set, you must ensure that the application connects only to queue managers associated with a single installation. Otherwise, if IBM MQ is unable to automatically locate the correct libraries, you must modify the application, or the library search path, to ensure that the correct libraries are used.

2587 (0A1B) (RC2587): MQRC_HMSG_NOT_AVAILABLE

Explanation

On an MQGET, MQPUT, or MQPUT1 call, a message handle supplied is not valid with the installation the queue manager is associated with. The message handle was created by MQCRTMH specifying the MQHC_UNASSOCIATED_HCONN option. It can be used only with queue managers associated with the first installation used in the process.

Completion Code

MQCC_FAILED

Programmer response

To pass properties between two queue managers associated with different installations, convert the message handle retrieved using MQGET into a buffer using the MQMHBUF call. Then pass that buffer into the MQPUT or MQPUT1 call of the other queue manager. Alternatively, use the **`setmqm`** command to associate one of the queue managers with the installation that the other queue manager is using. Using the **`setmqm`** command might change the version of IBM MQ that the queue manager uses.

2589 (0A1D) (RC2589) MQRC_INSTALLATION_MISSING

Explanation

On an MQCONN or MQCONNX call, an attempt was made to connect to a queue manager where the associated installation is no longer installed.

Completion Code

MQCC_FAILED

Programmer response

Associate the queue manager with a different installation using the **`setmqm`** command before attempting to connect to the queue manager again.

2590 (0A1E) (RC2590): MQRC_FASTPATH_NOT_AVAILABLE

Explanation

On an MQCONNX call, the MQCNO_FASTPATH_BINDING option was specified. However, a fastpath connection to the queue manager cannot be made. This issue can occur when a non-fastpath connection to a queue manager was made in the process before this MQCONNX call.

Completion Code

MQCC_FAILED

Programmer response

Either change all MQCONNX calls within the process to be fastpath, or use the *AMQ_SINGLE_INSTALLATION* environment variable to restrict connections to a single installation, allowing the queue manager to accept fastpath and non-fastpath connections from the same process, in any order.

2591 (0A1F) (RC2591): MQRC_CIPHER_SPEC_NOT_SUITE_B

Explanation

A client application is configured for NSA Suite B compliant operation but the CipherSpec for the client connection channel is not permitted at the configured Suite B security level. This can occur for Suite B CipherSpecs which fall outside the currently configured security level, for example if ECDHE_ECDSA_AES_128_GCM_SHA256 (which is 128-bit Suite B) is used when only the 192-bit Suite B security level is configured

For more information about which CipherSpecs are Suite B compliant, refer to Specifying CipherSpecs.

Completion Code

MQCC_FAILED

Programmer response

Select an appropriate CipherSpec which is permitted at the configured Suite B security level.

2592 (0A20) (RC2592): MQRC_SUITE_B_ERROR

Explanation

The configuration of Suite B is invalid. For example, an unrecognized value was specified in the **`MQSUITEB`** environment variable, the **`EncryptionPolicySuiteB`** SSL stanza setting or the MQSCO **`EncryptionPolicySuiteB`** field.

Completion Code

MQCC_FAILED

Programmer response

Determine the fault in the Suite B configuration and amend.

2593 (0A21)(RC2593): MQRC_CERT_VAL_POLICY_ERROR

Explanation

The certificate validation policy configuration is invalid. An unrecognized or unsupported value was specified in the **`MQCERTVPOL`** environment variable, the **`CertificateValPolicy`** SSL stanza setting or the MQSCO **`CertificateValPolicy`** field.

Completion Code

MQCC_FAILED

Programmer response

Specify a valid certificate validation policy which is supported on the current platform.

2594 (0A22)(RC2594): MQRC_PASSWORD_PROTECTION_ERROR

Explanation

An MQCONN or MQCONNX call was issued from a client connected application, but it failed to agree a password protection algorithm with the queue manager. For unencrypted channels, IBM MQ 8.0 clients try to agree a password protection mechanism to avoid sending passwords in plain text across a network.

The usual cause of this error is that the user has set the PasswordProtection attribute in the Channels stanza of mqclient.ini (or qm.ini) to ALWAYS, but the version of IBM MQ that is installed on the remote system does not support password protection.

Java and JMS clients must enable MQCSP authentication mode in order to use the PasswordProtection feature. See Connection authentication with the Java client.

Completion Code

MQCC_FAILED

Programmer response

Consider changing the PasswordProtection attribute or use SSL/TLS to protect passwords instead. If you are using SSL/TLS, you must not use a null cipher because it would send passwords in plain text which provides no protection.

More information can be found in the error log in message AMQ9296.

2595 (0A23)(RC2595): MQRC_CSP_ERROR

Explanation

The connect call failed because the MQCSP structure was not valid for one of the following reasons:

- The **`StrucId`** field is not MQCSP_STRUC_ID
- The **`Version`** field specifies a value that is not valid or not supported.
- The **`AuthenticationType`** field specifies a value that is not valid or not supported.
- The user identifier is incorrectly specified.
- The password is incorrectly specified.

Completion Code

MQCC_FAILED

Programmer response

Ensure that the MQCSP structure is correct.

z/OS On z/OS:

- Check that the IBM MQ libraries in STEPLIB are at the same or a higher level than the queue manager.
- If you are using USS, check that the LIBPATH has matching libraries, for example `LIBPATH=$LIBPATH:"/mqm/V8R0M0/java/lib/"`.

2596 (0A24)(RC2596): MQRC_CERT_LABEL_NOT_ALLOWED

Explanation

The channel definition specifies a certificate label but the environment does not support certificate label configuration.

Completion Code

MQCC_FAILED

Programmer response

Either, remove the certificate label from the channel definition, or change the configuration to ignore the label.

V 8.0.0.2 *2598 (0A26)(RC2598): MQRC_ADMIN_TOPIC_STRING_ERROR*

Explanation

This error can occur when calling MQSUB or MQOPEN. Publishing to an IBM MQ administrative topic string, starting `$SYS/MQ/` is not permitted.

When subscribing to an IBM MQ administrative topic string, the use of wildcard characters is restricted. See System topics for monitoring and activity trace for details.

Completion Code

MQCC_FAILED

Programmer response

Change the configuration to publish to an administrative topic string that does not start `$SYS/MQ/`.

6100 (17D4) (RC6100): MQRC_REOPEN_EXCL_INPUT_ERROR

Explanation

An open object does not have the correct ImqObject **open options** and requires one or more additional options. An implicit reopen is required but closure has been prevented.

Closure has been prevented because the queue is open for exclusive input and closure might result in the queue being accessed by another process or thread, before the queue is reopened by the process or thread that presently has access.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Set the **open options** explicitly to cover all eventualities so that implicit reopening is not required.

6101 (17D5) (RC6101): MQRC_REOPEN_INQUIRE_ERROR

Explanation

An open object does not have the correct ImqObject **open options** and requires one or more additional options. An implicit reopen is required but closure has been prevented.

Closure has been prevented because one or more characteristics of the object need to be checked dynamically prior to closure, and the **open options** do not already include MQOO_INQUIRE.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Set the **open options** explicitly to include MQOO_INQUIRE.

6102 (17D6) (RC6102): MQRC_REOPEN_SAVED_CONTEXT_ERR

Explanation

An open object does not have the correct ImqObject **open options** and requires one or more additional options. An implicit reopen is required but closure has been prevented.

Closure has been prevented because the queue is open with MQOO_SAVE_ALL_CONTEXT, and a destructive get has been performed previously. This has caused retained state information to be associated with the open queue and this information would be destroyed by closure.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Set the **open options** explicitly to cover all eventualities so that implicit reopening is not required.

6103 (17D7) (RC6103): MQRC_REOPEN_TEMPORARY_Q_ERROR

Explanation

An open object does not have the correct ImqObject **open options** and requires one or more additional options. An implicit reopen is required but closure has been prevented.

Closure has been prevented because the queue is a local queue of the definition type MQQDT_TEMPORARY_DYNAMIC, that would be destroyed by closure.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Set the **open options** explicitly to cover all eventualities so that implicit reopening is not required.

6104 (17D8) (RC6104): MQRC_ATTRIBUTE_LOCKED

Explanation

An attempt has been made to change the value of an attribute of an object while that object is open, or, for an ImqQueueManager object, while that object is connected. Certain attributes cannot be changed in these circumstances. Close or disconnect the object (as appropriate) before changing the attribute value.

An object might have been connected, opened, or both unexpectedly and implicitly to perform an MQINQ call. Check the attribute cross-reference table in C++ and MQI cross-reference to determine whether any of your method invocations result in an MQINQ call.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Include MQOO_INQUIRE in the ImqObject **open options** and set them earlier.

6105 (17D9) (RC6105): MQRC_CURSOR_NOT_VALID

Explanation

The browse cursor for an open queue has been invalidated since it was last used by an implicit reopen.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Set the ImqObject **open options** explicitly to cover all eventualities so that implicit reopening is not required.

6106 (17DA) (RC6106): MQRC_ENCODING_ERROR

Explanation

The encoding of the (next) message item needs to be MQENC_NATIVE for pasting.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6107 (17DB) (RC6107): MQRC_STRUC_ID_ERROR

Explanation

The structure id for the (next) message item, which is derived from the 4 characters beginning at the data pointer, is either missing or is inconsistent with the class of object into which the item is being pasted.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6108 (17DC) (RC6108): MQRC_NULL_POINTER

Explanation

A null pointer has been supplied where a nonnull pointer is either required or implied.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6109 (17DD) (RC6109): MQRC_NO_CONNECTION_REFERENCE

Explanation

The **connection reference** is null. A connection to an ImqQueueManager object is required.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6110 (17DE) (RC6110): MQRC_NO_BUFFER

Explanation

No buffer is available. For an ImqCache object, one cannot be allocated, denoting an internal inconsistency in the object state that should not occur.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6111 (17DF) (RC6111): MQRC_BINARY_DATA_LENGTH_ERROR

Explanation

The length of the binary data is inconsistent with the length of the target attribute. Zero is a correct length for all attributes.

- The correct length for an **accounting token** is MQ_ACCOUNTING_TOKEN_LENGTH.
- The correct length for an **alternate security id** is MQ_SECURITY_ID_LENGTH.
- The correct length for a **correlation id** is MQ_CORREL_ID_LENGTH.
- The correct length for a **facility token** is MQ_FACILITY_LENGTH.
- The correct length for a **group id** is MQ_GROUP_ID_LENGTH.
- The correct length for a **message id** is MQ_MSG_ID_LENGTH.
- The correct length for an **instance id** is MQ_OBJECT_INSTANCE_ID_LENGTH.
- The correct length for a **transaction instance id** is MQ_TRAN_INSTANCE_ID_LENGTH.
- The correct length for a **message token** is MQ_MSG_TOKEN_LENGTH.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6112 (17E0) (RC6112): MQRC_BUFFER_NOT_AUTOMATIC

Explanation

A user-defined (and managed) buffer cannot be resized. A user-defined buffer can only be replaced or withdrawn. A buffer must be automatic (system-managed) before it can be resized.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

6113 (17E1) (RC6113): MQRC_INSUFFICIENT_BUFFER

Explanation

There is insufficient buffer space available after the data pointer to accommodate the request. This might be because the buffer cannot be resized.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6114 (17E2) (RC6114): MQRC_INSUFFICIENT_DATA

Explanation

There is insufficient data after the data pointer to accommodate the request.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6115 (17E3) (RC6115): MQRC_DATA_TRUNCATED

Explanation

Data has been truncated when copying from one buffer to another. This might be because the target buffer cannot be resized, or because there is a problem addressing one or other buffer, or because a buffer is being downsized with a smaller replacement.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6116 (17E4) (RC6116): MQRC_ZERO_LENGTH

Explanation

A zero length has been supplied where a positive length is either required or implied.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6117 (17E5) (RC6117): MQRC_NEGATIVE_LENGTH

Explanation

A negative length has been supplied where a zero or positive length is required.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6118 (17E6) (RC6118): MQRC_NEGATIVE_OFFSET

Explanation

A negative offset has been supplied where a zero or positive offset is required.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6119 (17E7) (RC6119): MQRC_INCONSISTENT_FORMAT

Explanation

The format of the (next) message item is inconsistent with the class of object into which the item is being pasted.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6120 (17E8) (RC6120): MQRC_INCONSISTENT_OBJECT_STATE

Explanation

There is an inconsistency between this object, which is open, and the referenced ImqQueueManager object, which is not connected.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6121 (17E9) (RC6121): MQRC_CONTEXT_OBJECT_NOT_VALID

Explanation

The ImqPutMessageOptions **context reference** does not reference a valid ImqQueue object. The object has been previously destroyed.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6122 (17EA) (RC6122): MQRC_CONTEXT_OPEN_ERROR

Explanation

The ImqPutMessageOptions **context reference** references an ImqQueue object that could not be opened to establish a context. This might be because the ImqQueue object has inappropriate **open options**. Inspect the referenced object **reason code** to establish the cause.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6123 (17EB) (RC6123): MQRC_STRUC_LENGTH_ERROR

Explanation

The length of a data structure is inconsistent with its content. For an MQRMH, the length is insufficient to contain the fixed fields and all offset data.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

6124 (17EC) (RC6124): MQRC_NOT_CONNECTED

Explanation

A method failed because a required connection to a queue manager was not available, and a connection cannot be established implicitly because the IMQ_IMPL_CONN flag of the ImqQueueManager **behavior** class attribute is FALSE.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Establish a connection to a queue manager and retry.

6125 (17ED) (RC6125): MQRC_NOT_OPEN

Explanation

A method failed because an object was not open, and opening cannot be accomplished implicitly because the IMQ_IMPL_OPEN flag of the ImqObject **behavior** class attribute is FALSE.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Open the object and retry.

6126 (17EE) (RC6126): MQRC_DISTRIBUTION_LIST_EMPTY

Explanation

An ImqDistributionList failed to open because there are no ImqQueue objects referenced.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Establish at least one ImqQueue object in which the **distribution list reference** addresses the ImqDistributionList object, and retry.

6127 (17EF) (RC6127): MQRC_INCONSISTENT_OPEN_OPTIONS

Explanation

A method failed because the object is open, and the ImqObject **open options** are inconsistent with the required operation. The object cannot be reopened implicitly because the IMQ_IMPL_OPEN flag of the ImqObject **behavior** class attribute is false.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Open the object with appropriate ImqObject **open options** and retry.

6128 (17F0) (RC6128): MQRC_WRONG_VERSION

Explanation

A method failed because a version number specified or encountered is either incorrect or not supported.

For the ImqCICSBridgeHeader class, the problem is with the **version** attribute.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

If you are specifying a version number, use one that is supported by the class. If you are receiving message data from another program, ensure that both programs are using consistent and supported version numbers.

6129 (17F1) (RC6129): MQRC_REFERENCE_ERROR

Explanation

An object reference is invalid.

There is a problem with the address of a referenced object. At the time of use, the address of the object is nonnull, but is invalid and cannot be used for its intended purpose.

This reason code occurs in the IBM MQ C++ environment.

Completion Code

MQCC_FAILED

Programmer response

Check that the referenced object is neither deleted nor out of scope, or remove the reference by supplying a null address value.

PCF reason codes

Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

For more information about PCFs, see Introduction to Programmable Command Formats, Automating administration tasks, and Using Programmable Command Formats.

The following is a list of PCF reason codes, in numeric order, providing detailed information to help you understand them, including:

- An explanation of the circumstances that have caused the code to be raised
- The associated completion code
- Suggested programmer actions in response to the code

Related reference

"API completion and reason codes" on page 43
For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call.

"Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes" on page 315
IBM MQ can use Secure Sockets Layer (SSL) with the various communication protocols. Use this topic to identify the error codes that can be returned by SSL.

"WCF custom channel exceptions" on page 320
Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Related information

IBM MQ AMQ messages

z/OS IBM MQ for z/OS messages, completion, and reason codes

3001 (0BB9) (RC3001): MQRCCF_CFH_TYPE_ERROR

Explanation

Type not valid.

The MQCFH *Type* field value was not valid.

Programmer response

Specify a valid type.

3002 (0BBA) (RC3002): MQRCCF_CFH_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFH *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3003 (0BBB) (RC3003): MQRCCF_CFH_VERSION_ERROR

Explanation

Structure version number is not valid.

The MQCFH *Version* field value was not valid.

Note that z/OS requires MQCFH_VERSION_3.

Programmer response

Specify a valid structure version number.

3004 (0BBC) (RC3004): MQRCCF_CFH_MSG_SEQ_NUMBER_ERR

Explanation

Message sequence number not valid.

The MQCFH *MsgSeqNumber* field value was not valid.

Programmer response

Specify a valid message sequence number.

3005 (0BBD) (RC3005): MQRCCF_CFH_CONTROL_ERROR

Explanation

Control option not valid.

The MQCFH *Control* field value was not valid.

Programmer response

Specify a valid control option.

3006 (0BBE) (RC3006): MQRCCF_CFH_PARM_COUNT_ERROR

Explanation

Parameter count not valid.

The MQCFH *ParameterCount* field value was not valid.

Programmer response

Specify a valid parameter count.

3007 (0BBF) (RC3007): MQRCCF_CFH_COMMAND_ERROR

Explanation

Command identifier not valid.

The MQCFH *Command* field value was not valid.

Programmer response

Specify a valid command identifier.

3008 (0BC0) (RC3008): MQRCCF_COMMAND_FAILED

Explanation

Command failed.

The command has failed.

Programmer response

Refer to the previous error messages for this command.

3009 (0BC1) (RC3009): MQRCCF_CFIN_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFIN or MQCFIN64 *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3010 (0BC2) (RC3010): MQRCCF_CFST_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFST *StrucLength* field value was not valid. The value was not a multiple of four or was inconsistent with the MQCFST *StringLength* field value.

Programmer response

Specify a valid structure length.

3011 (0BC3) (RC3011): MQRCCF_CFST_STRING_LENGTH_ERR

Explanation

String length not valid.

The MQCFST *StringLength* field value was not valid. The value was negative or greater than the maximum permitted length of the parameter specified in the *Parameter* field.

Programmer response

Specify a valid string length for the parameter.

3012 (0BC4) (RC3012): MQRCCF_FORCE_VALUE_ERROR

Explanation

Force value not valid.

The force value specified was not valid.

Programmer response

Specify a valid force value.

3013 (0BC5) (RC3013): MQRCCF_STRUCTURE_TYPE_ERROR

Explanation

Structure type not valid.

The structure *Type* value was not valid.

Programmer response

Specify a valid structure type.

3014 (0BC6) (RC3014): MQRCCF_CFIN_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFIN or MQCFIN64 *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3015 (0BC7) (RC3015): MQRCCF_CFST_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFST *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3016 (0BC8) (RC3016): MQRCCF_MSG_LENGTH_ERROR

Explanation

Message length not valid.

The message data length was inconsistent with the length implied by the parameters in the message, or a positional parameter was out of sequence.

Programmer response

Specify a valid message length, and check that positional parameters are in the correct sequence.

3017 (0BC9) (RC3017): MQRCCF_CFIN_DUPLICATE_PARM

Explanation

Duplicate parameter.

Two MQCFIN or MQCFIN64 or MQCFIL or MQCFIL64 structures, or any two of those types of structure, with the same parameter identifier were present.

Programmer response

Check for and remove duplicate parameters.

3018 (0BCA) (RC3018): MQRCCF_CFST_DUPLICATE_PARM

Explanation

Duplicate parameter.

Two MQCFST structures, or an MQCFSL followed by an MQCFST structure, with the same parameter identifier were present.

Programmer response

Check for and remove duplicate parameters.

3019 (0BCB) (RC3019): MQRCCF_PARM_COUNT_TOO_SMALL

Explanation

Parameter count too small.

The MQCFH *ParameterCount* field value was less than the minimum required for the command.

Programmer response

Specify a parameter count that is valid for the command.

3020 (0BCC) (RC3020): MQRCCF_PARM_COUNT_TOO_BIG

Explanation

Parameter count too big.

The MQCFH *ParameterCount* field value was more than the maximum for the command.

Programmer response

Specify a parameter count that is valid for the command.

3021 (0BCD) (RC3021): MQRCCF_Q_ALREADY_IN_CELL

Explanation

Queue already exists in cell.

An attempt was made to define a queue with cell scope, or to change the scope of an existing queue from queue-manager scope to cell scope, but a queue with that name already existed in the cell.

Programmer response

Do one of the following:

- Delete the existing queue and retry the operation.
- Change the scope of the existing queue from cell to queue-manager and retry the operation.
- Create the new queue with a different name.

3022 (0BCE) (RC3022): MQRCCF_Q_TYPE_ERROR

Explanation

Queue type not valid.

The *QType* value was not valid.

Programmer response

Specify a valid queue type.

3023 (0BCF) (RC3023): MQRCCF_MD_FORMAT_ERROR

Explanation

Format not valid.

The MQMD *Format* field value was not MQFMT_ADMIN.

Programmer response

Specify the valid format.

3024 (0BD0) (RC3024): MQRCCF_CFSL_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFSL *StrucLength* field value was not valid. The value was not a multiple of four or was inconsistent with the MQCFSL *StringLength* field value.

Programmer response

Specify a valid structure length.

3025 (0BD1) (RC3025): MQRCCF_REPLACE_VALUE_ERROR

Explanation

Replace value not valid.

The *Replace* value was not valid.

Programmer response

Specify a valid replace value.

3026 (0BD2) (RC3026): MQRCCF_CFIL_DUPLICATE_VALUE

Explanation

Duplicate parameter value.

In the MQCFIL or MQCFIL64 structure, there was a duplicate parameter value in the list.

Programmer response

Check for and remove duplicate parameter values.

3027 (0BD3) (RC3027): MQRCCF_CFIL_COUNT_ERROR

Explanation

Count of parameter values not valid.

The MQCFIL or MQCFIL64 *Count* field value was not valid. The value was negative or greater than the maximum permitted for the parameter specified in the *Parameter* field.

Programmer response

Specify a valid count for the parameter.

3028 (0BD4) (RC3028): MQRCCF_CFIL_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFIL or MQCFIL64 *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3029 (0BD5) (RC3029): MQRCCF_MODE_VALUE_ERROR

Explanation

Mode value not valid.

The *Mode* value was not valid.

Programmer response

Specify a valid mode value.

3029 (0BD5) (RC3029): MQRCCF_QUIESCE_VALUE_ERROR

Explanation

Former name for MQRCCF_MODE_VALUE_ERROR.

3030 (0BD6) (RC3030): MQRCCF_MSG_SEQ_NUMBER_ERROR

Explanation

Message sequence number not valid.

The message sequence number parameter value was not valid.

Programmer response

Specify a valid message sequence number.

3031 (0BD7) (RC3031): MQRCCF_PING_DATA_COUNT_ERROR

Explanation

Data count not valid.

The Ping Channel *DataCount* value was not valid.

Programmer response

Specify a valid data count value.

3032 (0BD8) (RC3032): MQRCCF_PING_DATA_COMPARE_ERROR

Explanation

Ping Channel command failed.

The Ping Channel command failed with a data compare error. The data offset that failed is returned in the message (with parameter identifier MQIACF_ERROR_OFFSET).

Programmer response

Consult your systems administrator.

3033 (0BD9) (RC3033): MQRCCF_CFSL_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFSL *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3034 (0BDA) (RC3034): MQRCCF_CHANNEL_TYPE_ERROR

Explanation

Channel type not valid.

The *ChannelType* specified was not valid, or did not match the type of an existing channel being copied, changed or replaced, or the command and the specified disposition cannot be used with that type of channel.

Programmer response

Specify a valid channel name, type, or disposition.

3035 (0BDB) (RC3035): MQRCCF_PARM_SEQUENCE_ERROR

Explanation

Parameter sequence not valid.

The sequence of parameters is not valid for this command.

Programmer response

Specify the positional parameters in a valid sequence for the command.

3036 (0BDC) (RC3036): MQRCCF_XMIT_PROTOCOL_TYPE_ERR

Explanation

Transmission protocol type not valid.

The *TransportType* value was not valid.

Programmer response

Specify a valid transmission protocol type.

3037 (0BDD) (RC3037): MQRCCF_BATCH_SIZE_ERROR

Explanation

Batch size not valid.

The batch size specified was not valid.

Programmer response

Specify a valid batch size value.

3038 (0BDE) (RC3038): MQRCCF_DISC_INT_ERROR

Explanation

Disconnection interval not valid.

The disconnection interval specified was not valid.

Programmer response

Specify a valid disconnection interval.

3039 (0BDF) (RC3039): MQRCCF_SHORT_RETRY_ERROR

Explanation

Short retry count not valid.

The *ShortRetryCount* value was not valid.

Programmer response

Specify a valid short retry count value.

3040 (0BE0) (RC3040): MQRCCF_SHORT_TIMER_ERROR

Explanation

Short timer value not valid.

The *ShortRetryInterval* value was not valid.

Programmer response

Specify a valid short timer value.

3041 (0BE1) (RC3041): MQRCCF_LONG_RETRY_ERROR

Explanation

Long retry count not valid.

The long retry count value specified was not valid.

Programmer response

Specify a valid long retry count value.

3042 (0BE2) (RC3042): MQRCCF_LONG_TIMER_ERROR

Explanation

Long timer not valid.

The long timer (long retry wait interval) value specified was not valid.

Programmer response

Specify a valid long timer value.

3043 (0BE3) (RC3043): MQRCCF_SEQ_NUMBER_WRAP_ERROR

Explanation

Sequence wrap number not valid.

The *SeqNumberWrap* value was not valid.

Programmer response

Specify a valid sequence wrap number.

3044 (0BE4) (RC3044): MQRCCF_MAX_MSG_LENGTH_ERROR

Explanation

Maximum message length not valid.

The maximum message length value specified was not valid.

Programmer response

Specify a valid maximum message length.

3045 (0BE5) (RC3045): MQRCCF_PUT_AUTH_ERROR

Explanation

Put authority value not valid.

The *PutAuthority* value was not valid.

Programmer response

Specify a valid authority value.

3046 (0BE6) (RC3046): MQRCCF_PURGE_VALUE_ERROR

Explanation

Purge value not valid.

The *Purge* value was not valid.

Programmer response

Specify a valid purge value.

3047 (0BE7) (RC3047): MQRCCF_CFIL_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFIL or MQCFIL64 *Parameter* field value was not valid, or specifies a parameter that cannot be filtered, or that is also specified as a parameter to select a subset of objects.

Programmer response

Specify a valid parameter identifier.

3048 (0BE8) (RC3048): MQRCCF_MSG_TRUNCATED

Explanation

Message truncated.

The command server received a message that is larger than its maximum valid message size.

Programmer response

Check the message contents are correct.

3049 (0BE9) (RC3049): MQRCCF_CCSID_ERROR

Explanation

Coded character-set identifier error.

In a command message, one of the following occurred:

- The *CodedCharSetId* field in the message descriptor of the command does not match the coded character-set identifier of the queue manager at which the command is being processed, or
- The *CodedCharSetId* field in a string parameter structure within the message text of the command is not
  - MQCCSI_DEFAULT, or
  - the coded character-set identifier of the queue manager at which the command is being processed, as in the *CodedCharSetId* field in the message descriptor.

The error response message contains the correct value.

This reason can also occur if a ping cannot be performed because the coded character-set identifiers are not compatible. In this case the correct value is not returned.

Programmer response

Construct the command with the correct coded character-set identifier, and specify this in the message descriptor when sending the command. For ping, use a suitable coded character-set identifier.

3050 (0BEA) (RC3050): MQRCCF_ENCODING_ERROR

Explanation

Encoding error.

The *Encoding* field in the message descriptor of the command does not match that required for the platform at which the command is being processed.

Programmer response

Construct the command with the correct encoding, and specify this in the message descriptor when sending the command.

3052 (0BEC) (RC3052): MQRCCF_DATA_CONV_VALUE_ERROR

Explanation

Data conversion value not valid.

The value specified for *DataConversion* is not valid.

Programmer response

Specify a valid value.

3053 (0BED) (RC3053): MQRCCF_INDOUBT_VALUE_ERROR

Explanation

In-doubt value not valid.

The value specified for *InDoubt* is not valid.

Programmer response

Specify a valid value.

3054 (0BEE) (RC3054): MQRCCF_ESCAPE_TYPE_ERROR

Explanation

Escape type not valid.

The value specified for *EscapeType* is not valid.

Programmer response

Specify a valid value.

3062 (0BF6) (RC3062): MQRCCF_CHANNEL_TABLE_ERROR

Explanation

Channel table value not valid.

The *ChannelTable* specified was not valid, or was not appropriate for the channel type specified on an Inquire Channel or Inquire Channel Names command.

Programmer response

Specify a valid channel table value.

3063 (0BF7) (RC3063): MQRCCF_MCA_TYPE_ERROR

Explanation

Message channel agent type not valid.

The *MCAType* value specified was not valid.

Programmer response

Specify a valid value.

3064 (0BF8) (RC3064): MQRCCF_CHL_INST_TYPE_ERROR

Explanation

Channel instance type not valid.

The *ChannelInstanceType* specified was not valid.

Programmer response

Specify a valid channel instance type.

3065 (0BF9) (RC3065): MQRCCF_CHL_STATUS_NOT_FOUND

Explanation

Channel status not found.

For Inquire Channel Status, no channel status is available for the specified channel. This might indicate that the channel has not been used.

Programmer response

None, unless this is unexpected, in which case consult your systems administrator.

3066 (0BFA) (RC3066): MQRCCF_CFSL_DUPLICATE_PARM

Explanation

Duplicate parameter.

Two MQCFSL structures, or an MQCFST followed by an MQCFSL structure, with the same parameter identifier were present.

Programmer response

Check for and remove duplicate parameters.

3067 (0BFB) (RC3067): MQRCCF_CFSL_TOTAL_LENGTH_ERROR

Explanation

Total string length error.

The total length of the strings (not including trailing blanks) in a MQCFSL structure exceeds the maximum allowable for the parameter.

Programmer response

Check that the structure has been specified correctly, and if so reduce the number of strings.

3068 (0BFC) (RC3068): MQRCCF_CFSL_COUNT_ERROR

Explanation

Count of parameter values not valid.

The MQCFSL *Count* field value was not valid. The value was negative or greater than the maximum permitted for the parameter specified in the *Parameter* field.

Programmer response

Specify a valid count for the parameter.

3069 (0BFD) (RC3069): MQRCCF_CFSL_STRING_LENGTH_ERR

Explanation

String length not valid.

The MQCFSL *StringLength* field value was not valid. The value was negative or greater than the maximum permitted length of the parameter specified in the *Parameter* field.

Programmer response

Specify a valid string length for the parameter.

3070 (0BFE) (RC3070): MQRCCF_BROKER_DELETED

Explanation

Broker has been deleted.

When a broker is deleted using the *dltmqbrk* command, all broker queues created by the broker are deleted. Before this can be done the queues are emptied of all command messages; any that are found are placed on the dead-letter queue with this reason code.

Programmer response

Process the command messages that were placed on the dead-letter queue.

3071 (0BFF) (RC3071): MQRCCF_STREAM_ERROR

Explanation

Stream name is not valid.

The stream name parameter is not valid. Stream names must obey the same naming rules as for IBM MQ queues.

Programmer response

Retry the command with a valid stream name parameter.

3072 (0C00) (RC3072): MQRCCF_TOPIC_ERROR

Explanation

Topic name is invalid.

A command has been sent to the broker containing a topic name that is not valid. Note that wildcard topic names are not allowed for *Register Publisher* and *Publish* commands.

Programmer response

Retry the command with a valid topic name parameter. Up to 256 characters of the topic name in question are returned with the error response message. If the topic name contains a null character, this is assumed to terminate the string and is not considered to be part of it. A zero length topic name is not valid, as is one that contains an escape sequence that is not valid.

3073 (0C01) (RC3073): MQRCCF_NOT_REGISTERED

Explanation

Subscriber or publisher is not registered.

A *Deregister* command has been issued to remove registrations for a topic, or topics, for which the publisher or subscriber is not registered. If multiple topics were specified on the command, it fails with a completion code of MQCC_WARNING if the publisher or subscriber was registered for some, but not all, of the topics specified. This error code is also returned to a subscriber issuing a *Request Update* command for a topic for which he does not have a subscription.

Programmer response

Investigate why the publisher or subscriber is not registered. In the case of a subscriber, the subscriptions might have expired, or been removed automatically by the broker if the subscriber is no longer authorized.

3074 (0C02) (RC3074): MQRCCF_Q_MGR_NAME_ERROR

Explanation

An invalid or unknown queue manager name has been supplied.

A queue manager name has been supplied as part of a publisher or subscriber identity. This might have been supplied as an explicit parameter or in the *ReplyToQMgr* field in the message descriptor of the command. Either the queue manager name is not valid, or in the case of a subscriber identity, the subscriber's queue could not be resolved because the remote queue manager is not known to the broker queue manager.

Programmer response

Retry the command with a valid queue manager name. If appropriate, the broker includes a further error reason code within the error response message. If one is supplied, follow the guidance for that reason code in "Reason codes and exceptions" on page 43 to resolve the problem.

3075 (0C03) (RC3075): MQRCCF_INCORRECT_STREAM

Explanation

Stream name does not match the stream queue it was sent to.

A command has been sent to a stream queue that specified a different stream name parameter.

Programmer response

Retry the command either by sending it to the correct stream queue or by modifying the command so that the stream name parameter matches.

3076 (0C04) (RC3076): MQRCCF_Q_NAME_ERROR

Explanation

An invalid or unknown queue name has been supplied.

A queue name has been supplied as part of a publisher or subscriber identity. This might have been supplied as an explicit parameter or in the *ReplyToQ* field in the message descriptor of the command. Either the queue name is not valid, or in the case of a subscriber identity, the broker has failed to open the queue.

Programmer response

Retry the command with a valid queue name. If appropriate, the broker includes a further error reason code within the error response message. If one is supplied, follow the guidance for that reason code in "Reason codes and exceptions" on page 43 to resolve the problem.

3077 (0C05) (RC3077): MQRCCF_NO_RETAINED_MSG

Explanation

No retained message exists for the topic specified.

A *Request Update* command has been issued to request the retained message associated with the specified topic. No retained message exists for that topic.

Programmer response

If the topic or topics in question should have retained messages, the publishers of these topics might not be publishing with the correct publication options to cause their publications to be retained.

3078 (0C06) (RC3078): MQRCCF_DUPLICATE_IDENTITY

Explanation

Publisher or subscriber identity already assigned to another user ID.

Each publisher and subscriber has a unique identity consisting of a queue manager name, a queue name, and optionally a correlation identifier. Associated with each identity is the user ID under which that publisher or subscriber first registered. A specific identity can be assigned only to one user ID at a time. While the identity is registered with the broker all commands wanting to use it must specify the correct user ID. When a publisher or a subscriber no longer has any registrations with the broker the identity can be used by another user ID.

Programmer response

Either retry the command using a different identity or remove all registrations associated with the identity so that it can be used by a different user ID. The user ID to which the identity is currently assigned is returned within the error response message. A *Deregister* command could be issued to remove these registrations. If the user ID in question cannot be used to execute such a command, you need to have the necessary authority to open the SYSTEM.BROKER.CONTROL.QUEUE using the MQOO_ALTERNATE_USER_AUTHORITY option.

3079 (0C07) (RC3079): MQRCCF_INCORRECT_Q

Explanation

Command sent to wrong broker queue.

The command is a valid broker command but the queue it has been sent to is incorrect. *Publish* and *Delete Publication* commands need to be sent to the stream queue, all other commands need to be sent to the SYSTEM.BROKER.CONTROL.QUEUE.

Programmer response

Retry the command by sending it to the correct queue.

3080 (0C08) (RC3080): MQRCCF_CORREL_ID_ERROR

Explanation

Correlation identifier used as part of an identity is all binary zeros.

Each publisher and subscriber is identified by a queue manager name, a queue name, and optionally a correlation identifier. The correlation identifier is typically used to allow multiple subscribers to share the same subscriber queue. In this instance a publisher or subscriber has indicated within the Registration or Publication options supplied on the command that their identity does include a correlation identifier, but a valid identifier has not been supplied. The <RegOpt>CorrelAsId</RegOpt> has been specified, but the correlation identifier of the message is nulls.

Programmer response

Change the program to retry the command ensuring that the correlation identifier supplied in the message descriptor of the command message is not all binary zeros.

3081 (0C09) (RC3081): MQRCCF_NOT_AUTHORIZED

Explanation

Subscriber has insufficient authority.

To receive publications a subscriber application needs both browse authority for the stream queue that it is subscribing to, and put authority for the queue that publications are to be sent to. Subscriptions are rejected if the subscriber does not have both authorities. In addition to having browse authority for the stream queue, a subscriber would also require *altusr* authority for the stream queue to subscribe to certain topics that the broker itself publishes information on. These topics start with the MQ/SA/ prefix.

Programmer response

Ensure that the subscriber has the necessary authorities and reissue the request. The problem might occur because the subscriber's user ID is not known to the broker. This can be identified if a further error reason code of MQRC_UNKNOWN_ENTITY is returned within the error response message.

3082 (0C0A) (RC3082): MQRCCF_UNKNOWN_STREAM

Explanation

Stream is not known by the broker or could not be created.

A command message has been put to the SYSTEM.BROKER.CONTROL.QUEUE for an unknown stream. This error code is also returned if dynamic stream creation is enabled and the broker failed to create a stream queue for the new stream using the SYSTEM.BROKER.MODEL.STREAM queue.

Programmer response

Retry the command for a stream that the broker supports. If the broker should support the stream, either define the stream queue manually, or correct the problem that prevented the broker from creating the stream queue itself.

3083 (0C0B) (RC3083): MQRCCF_REG_OPTIONS_ERROR

Explanation

Invalid registration options have been supplied.

The registration options (between <RegOpt> and </RegOpt>) provided on a command are not valid.

Programmer response

Retry the command with a valid combination of options.

3084 (0C0C) (RC3084): MQRCCF_PUB_OPTIONS_ERROR

Explanation

Invalid publication options have been supplied.

The publication options provided on a Publish command are not valid.

Programmer response

Retry the command with a valid combination of options.

3085 (0C0D) (RC3085): MQRCCF_UNKNOWN_BROKER

Explanation

Command received from an unknown broker.

Within a multi-broker network, related brokers pass subscriptions and publications between each other as a series of command messages. One such command message has been received from a broker that is not, or is no longer, related to the detecting broker.

Programmer response

This situation can occur if the broker network is not quiesced while topology changes are made to the network.

If you are removing a broker from the topology when the queue manager is inactive, your changes are propagated at queue manager restart.

If you are removing a broker from the topology when the queue manager is active, make sure the channels are also active, so that your changes are immediately propagated.

3086 (0C0E) (RC3086): MQRCCF_Q_MGR_CCSID_ERROR

Explanation

Queue manager coded character set identifier error.

The coded character set value for the queue manager was not valid.

Programmer response

Specify a valid value.

3087 (0C0F) (RC3087): MQRCCF_DEL_OPTIONS_ERROR

Explanation

Invalid delete options have been supplied.

The options provided with a *Delete Publication* command are not valid.

Programmer response

Retry the command with a valid combination of options.

3088 (0C10) (RC3088): MQRCCF_CLUSTER_NAME_CONFLICT

Explanation

ClusterName and *ClusterNamelist* attributes conflict.

The command was rejected because it would have resulted in the *ClusterName* attribute and the *ClusterNamelist* attribute both being nonblank. At least one of these attributes must be blank.

Programmer response

If the command specified one of these attributes only, you must also specify the other one, but with a value of blanks. If the command specified both attributes, ensure that one of them has a value of blanks.

3089 (0C11) (RC3089): MQRCCF_REPOS_NAME_CONFLICT

Explanation

RepositoryName and *RepositoryNamelist* attributes conflict.

Either:

- The command was rejected because it would have resulted in the *RepositoryName* and *RepositoryNamelist* attributes both being nonblank. At least one of these attributes must be blank.
- For a Reset Queue Manager Cluster command, the queue manager does not provide a full repository management service for the specified cluster. That is, the *RepositoryName* attribute of the queue manager is not the specified cluster name, or the namelist specified by the *RepositoryNamelist* attribute does not contain the cluster name.

Programmer response

Reissue the command with the correct values or on the correct queue manager.

3090 (0C12) (RC3090): MQRCCF_CLUSTER_Q_USAGE_ERROR

Explanation

Queue cannot be a cluster queue.

The command was rejected because it would have resulted in a cluster queue also being a transmission queue, which is not permitted, or because the queue in question cannot be a cluster queue.

Programmer response

Ensure that the command specifies either:

- The *Usage* parameter with a value of MQUS_NORMAL, or
- The *ClusterName* and *ClusterNamelist* parameters with values of blanks.
- A *QName* parameter with a value that is not one of these reserved queues:
  - SYSTEM.CHANNEL.INITQ
  - SYSTEM.CHANNEL.SYNCQ
  - SYSTEM.CLUSTER.COMMAND.QUEUE
  - SYSTEM.CLUSTER.REPOSITORY.QUEUE
  - SYSTEM.COMMAND.INPUT
  - SYSTEM.QSG.CHANNEL.SYNCQ
  - SYSTEM.QSG.TRANSMIT.QUEUE

3091 (0C13) (RC3091): MQRCCF_ACTION_VALUE_ERROR

Explanation

Action value not valid.

The value specified for *Action* is not valid. There is only one valid value.

Programmer response

Specify MQACT_FORCE_REMOVE as the value of the *Action* parameter.

3092 (0C14) (RC3092): MQRCCF_COMMS_LIBRARY_ERROR

Explanation

Library for requested communications protocol could not be loaded.

The library needed for the requested communications protocol could not be loaded.

Programmer response

Install the library for the required communications protocol, or specify a communications protocol that has already been installed.

3093 (0C15) (RC3093): MQRCCF_NETBIOS_NAME_ERROR

Explanation

NetBIOS listener name not defined.

The NetBIOS listener name is not defined.

Programmer response

Add a local name to the configuration file and retry the operation.

3094 (0C16) (RC3094): MQRCCF_BROKER_COMMAND_FAILED

Explanation

The broker command failed to complete.

A broker command was issued but it failed to complete.

Programmer response

Diagnose the problem using the provided information and issue a corrected command.

For more information, look at the IBM MQ error logs.

3095 (0C17) (RC3095): MQRCCF_CFST_CONFLICTING_PARM

Explanation

Conflicting parameters.

The command was rejected because the parameter identified in the error response was in conflict with another parameter in the command.

Programmer response

Consult the description of the parameter identified to ascertain the nature of the conflict, and the correct command.

3096 (0C18) (RC3096): MQRCCF_PATH_NOT_VALID

Explanation

Path not valid.

The path specified was not valid.

Programmer response

Specify a valid path.

3097 (0C19) (RC3097): MQRCCF_PARM_SYNTAX_ERROR

Explanation

Syntax error found in parameter.

The parameter specified contained a syntax error.

Programmer response

Check the syntax for this parameter.

3098 (0C1A) (RC3098): MQRCCF_PWD_LENGTH_ERROR

Explanation

Password length error.

The password string length is rounded up by to the nearest eight bytes. This rounding causes the total length of the *SSLCryptoHardware* string to exceed its maximum.

Programmer response

Decrease the size of the password, or of earlier fields in the *SSLCryptoHardware* string.

3150 (0C4E) (RC3150): MQRCCF_FILTER_ERROR

Explanation

Filter not valid. This could be because either:

1. In an inquire command message, the specification of a filter is not valid.
2. In a publish/subscribe command message, the content-based filter expression supplied in the publish/subscribe command message contains invalid syntax, and cannot be used.

Programmer response

1. Correct the specification of the filter parameter structure in the inquire command message.
2. Correct the syntax of the filter expression in the publish/subscribe command message. The filter expression is the value of the *Filter* tag in the *psc* folder in the MQRFH2 structure. See the *Websphere MQ Integrator V2 Programming Guide* for details of valid syntax.

3151 (0C4F) (RC3151): MQRCCF_WRONG_USER

Explanation

Wrong user.

A publish/subscribe command message cannot be executed on behalf of the requesting user because the subscription that it would update is already owned by a different user. A subscription can be updated or deregistered only by the user that originally registered the subscription.

Programmer response

Ensure that applications that need to issue commands against existing subscriptions are running under the user identifier that originally registered the subscription. Alternatively, use different subscriptions for different users.

3152 (0C50) (RC3152): MQRCCF_DUPLICATE_SUBSCRIPTION

Explanation

The subscription already exists.

A matching subscription already exists.

Programmer response

Either modify the new subscription properties to distinguish it from the existing subscription or deregister the existing subscription. Then reissue the command.

3153 (0C51) (RC3153): MQRCCF_SUB_NAME_ERROR

Explanation

The subscription name parameter is in error.

Either the subscription name is of an invalid format or a matching subscription already exists with no subscription name.

Programmer response

Either correct the subscription name or remove it from the command and reissue the command.

3154 (0C52) (RC3154): MQRCCF_SUB_IDENTITY_ERROR

Explanation

The subscription identity parameter is in error.

Either the supplied value exceeds the maximum length allowed or the subscription identity is not currently a member of the subscription's identity set and a Join registration option was not specified.

Programmer response

Either correct the identity value or specify a Join registration option to add this identity to the identity set for this subscription.

3155 (0C53) (RC3155): MQRCCF_SUBSCRIPTION_IN_USE

Explanation

The subscription is in use.

An attempt to modify or deregister a subscription was attempted by a member of the identity set when they were not the only member of this set.

Programmer response

Reissue the command when you are the only member of the identity set. To avoid the identity set check and force the modification or deregistration remove the subscription identity from the command message and reissue the command.

3156 (0C54) (RC3156): MQRCCF_SUBSCRIPTION_LOCKED

Explanation

The subscription is locked.

The subscription is currently exclusively locked by another identity.

Programmer response

Wait for this identity to release the exclusive lock.

3157 (0C55) (RC3157): MQRCCF_ALREADY_JOINED

Explanation

The identity already has an entry for this subscription.

A Join registration option was specified but the subscriber identity was already a member of the subscription's identity set.

Programmer response

None. The command completed, this reason code is a warning.

3160 (0C58) (RC3160): MQRCCF_OBJECT_IN_USE

Explanation

Object in use by another command.

A modification of an object was attempted while the object was being modified by another command.

Programmer response

Retry the command.

3161 (0C59) (RC3161): MQRCCF_UNKNOWN_FILE_NAME

Explanation

File not defined to CICS.

A file name parameter identifies a file that is not defined to CICS.

Programmer response

Provide a valid file name or create a CSD definition for the required file.

3162 (0C5A) (RC3162): MQRCCF_FILE_NOT_AVAILABLE

Explanation

File not available to CICS.

A file name parameter identifies a file that is defined to CICS, but is not available.

Programmer response

Check that the CSD definition for the file is correct and enabled.

3163 (0C5B) (RC3163): MQRCCF_DISC_RETRY_ERROR

Explanation

Disconnection retry count not valid.

The *DiscRetryCount* value was not valid.

Programmer response

Specify a valid count.

3164 (0C5C) (RC3164): MQRCCF_ALLOC_RETRY_ERROR

Explanation

Allocation retry count not valid.

The *AllocRetryCount* value was not valid.

Programmer response

Specify a valid count.

3165 (0C5D) (RC3165): MQRCCF_ALLOC_SLOW_TIMER_ERROR

Explanation

Allocation slow retry timer value not valid.

The *AllocRetrySlowTimer* value was not valid.

Programmer response

Specify a valid timer value.

3166 (0C5E) (RC3166): MQRCCF_ALLOC_FAST_TIMER_ERROR

Explanation

Allocation fast retry timer value not valid.

The *AllocRetryFastTimer* value was not valid.

Programmer response

Specify a valid value.

3167 (0C5F) (RC3167): MQRCCF_PORT_NUMBER_ERROR

Explanation

Port number value not valid.

The *PortNumber* value was not valid.

Programmer response

Specify a valid port number value.

3168 (0C60) (RC3168): MQRCCF_CHL_SYSTEM_NOT_ACTIVE

Explanation

Channel system is not active.

An attempt was made to start a channel while the channel system was inactive.

Programmer response

Activate the channel system before starting a channel.

3169 (0C61) (RC3169): MQRCCF_ENTITY_NAME_MISSING

Explanation

Entity name required but missing.

A parameter specifying entity names must be supplied.

Programmer response

Specify the required parameter.

3170 (0C62) (RC3170): MQRCCF_PROFILE_NAME_ERROR

Explanation

Profile name not valid.

A profile name is not valid. Profile names might include wildcard characters or might be given explicitly. If you give an explicit profile name, then the object identified by the profile name must exist. This error might also occur if you specify more than one double asterisk in a profile name.

Programmer response

Specify a valid name.

3171 (0C63) (RC3171): MQRCCF_AUTH_VALUE_ERROR

Explanation

Authorization value not valid.

A value for the *AuthorizationList* or *AuthorityRemove* or *AuthorityAdd* parameter was not valid.

Programmer response

Specify a valid value.

3172 (0C64) (RC3172): MQRCCF_AUTH_VALUE_MISSING

Explanation

Authorization value required but missing.

A parameter specifying authorization values must be supplied.

Programmer response

Specify the required parameter.

3173 (0C65) (RC3173): MQRCCF_OBJECT_TYPE_MISSING

Explanation

Object type value required but missing.

A parameter specifying the object type must be supplied.

Programmer response

Specify the required parameter.

3174 (0C66) (RC3174): MQRCCF_CONNECTION_ID_ERROR

Explanation

Error in connection id parameter.

The *ConnectionId* specified was not valid.

Programmer response

Specify a valid connection id.

3175 (0C67) (RC3175): MQRCCF_LOG_TYPE_ERROR

Explanation

Log type not valid.

The log type value specified was not valid.

Programmer response

Specify a valid log type value.

3176 (0C68) (RC3176): MQRCCF_PROGRAM_NOT_AVAILABLE

Explanation

Program not available.

A request to start or stop a service failed because the request to start the program failed. This could be because the program could not be found at the specified location, or that insufficient system resources are available currently to start it.

Programmer response

Check that the correct name is specified in the definition of the service, and that the program is in the appropriate libraries, before retrying the request.

3177 (0C69) (RC3177): MQRCCF_PROGRAM_AUTH_FAILED

Explanation

Program not available.

A request to start or stop a service failed because the user does not have sufficient access authority to start the program at the specified location.

Programmer response

Correct the progam name and location, and the user's authority, before retrying the request.

3200 (0C80) (RC3200): MQRCCF_NONE_FOUND

Explanation

No items found matching request criteria.

An Inquire command found no items that matched the specified name and satisfied any other criteria requested.

3201 (0C81) (RC3201): MQRCCF_SECURITY_SWITCH_OFF

Explanation

Security refresh or reverification not processed, security switch set OFF.

Either

- a Reverify Security command was issued, but the subsystem security switch is off, so there are no internal control tables to flag for reverification; or
- a Refresh Security command was issued, but the security switch for the requested class or the subsystem security switch is off.

The switch in question might be returned in the message (with parameter identifier MQIACF_SECURITY_SWITCH).

3202 (0C82) (RC3202): MQRCCF_SECURITY_REFRESH_FAILED

Explanation

Security refresh did not take place.

A SAF RACROUTE REQUEST=STAT call to your external security manager (ESM) returned a non-zero return code. In consequence, the requested security refresh could not be done. The security item affected might be returned in the message (with parameter identifier MQIACF_SECURITY_ITEM).

Possible causes of this problem are:

- The class is not installed
- The class is not active
- The external security manager (ESM) is not active

- The RACF z/OS router table is incorrect

Programmer response

For information about resolving the problem, see the explanations of messages CSQH003I and CSQH004I.

3203 (0C83) (RC3203): MQRCCF_PARM_CONFLICT

Explanation

Incompatible parameters or parameter values.

The parameters or parameter values for a command are incompatible. One of the following occurred:

- A parameter was not specified that is required by another parameter or parameter value.
- A parameter or parameter value was specified that is not allowed with some other parameter or parameter value.
- The values for two specified parameters were not both blank or non-blank.
- The values for two specified parameters were incompatible.
- The specified value is inconsistent with the configuration.

The parameters in question might be returned in the message (with parameter identifiers MQIACF_PARAMETER_ID).

Programmer response

Reissue the command with correct parameters and values.

3204 (0C84) (RC3204): MQRCCF_COMMAND_INHIBITED

Explanation

Commands not allowed at present time.

The queue manager cannot accept commands at the present time, because it is restarting or terminating, or because the command server is not running.

3205 (0C85) (RC3205): MQRCCF_OBJECT_BEING_DELETED

Explanation

Object is being deleted.

The object specified on a command is in the process of being deleted, so the command is ignored.

3207 (0C87) (RC3207): MQRCCF_STORAGE_CLASS_IN_USE

Explanation

Storage class is active or queue is in use.

The command for a local queue involved a change to the *StorageClass* value, but there are messages on the queue, or other threads have the queue open.

Programmer response

Remove the messages from the queue, or wait until any other threads have closed the queue.

3208 (0C88) (RC3208): MQRCCF_OBJECT_NAME_RESTRICTED

Explanation

Incompatible object name and type.

The command used a reserved object name with an incorrect object type or subtype. The object is only allowed to be of a predetermined type, as listed in the explanation of message CSQM108I.

3209 (0C89) (RC3209): MQRCCF_OBJECT_LIMIT_EXCEEDED

Explanation

Local queue limit exceeded.

The command failed because no more local queues could be defined. There is an implementation limit of 524 287 for the total number of local queues that can exist. For shared queues, there is a limit of 512 queues in a single coupling facility structure.

Programmer response

Delete any existing queues that are no longer required.

3210 (0C8A) (RC3210): MQRCCF_OBJECT_OPEN_FORCE

Explanation

Object is in use, but could be changed specifying *Force* as MQFC_YES.

The object specified is in use. This could be because it is open through the API, or for certain parameter changes, because there are messages currently on the queue. The requested changes can be made by specifying *Force* as MQFC_YES on a Change command.

Programmer response

Wait until the object is not in use. Alternatively specify *Force* as MQFC_YES for a change command.

3211 (0C8B) (RC3211): MQRCCF_DISPOSITION_CONFLICT

Explanation

Parameters are incompatible with disposition.

The parameters or parameter values for a command are incompatible with the disposition of an object. One of the following occurred:

- A value specified for the object name or other parameter is not allowed for a local queue with a disposition that is shared or a model queue used to create a dynamic queue that is shared.
- A value specified for a parameter is not allowed for an object with such disposition.
- A value specified for a parameter must be non-blank for an object with such disposition.
- The *CommandScope* and *QSGDisposition* or *ChannelDisposition* parameter values are incompatible.

- The action requested for a channel cannot be performed because it has the wrong disposition.

The parameter and disposition in question may be returned in the message (with parameter identifiers MQIACF_PARAMETER_ID and MQIA_QSG_DISP).

Programmer response

Reissue the command with correct parameters and values.

3212 (0C8C) (RC3212): MQRCCF_Q_MGR_NOT_IN_QSG

Explanation

Queue manager is not in a queue-sharing group.

The command or its parameters are not allowed when the queue manager is not in a queue-sharing group. The parameter in question might be returned in the message (with parameter identifier MQIACF_PARAMETER_ID).

Programmer response

Reissue the command correctly.

3213 (0C8D) (RC3213): MQRCCF_ATTR_VALUE_FIXED

Explanation

Parameter value cannot be changed.

The value for a parameter cannot be changed. The parameter in question might be returned in the message (with parameter identifier MQIACF_PARAMETER_ID).

Programmer response

To change the parameter, the object must be deleted and then created again with the new value.

3215 (0C8F) (RC3215): MQRCCF_NAMELIST_ERROR

Explanation

Namelist is empty or wrong type.

A namelist used to specify a list of clusters has no names in it or does not have type MQNT_CLUSTER or MQNT_NONE.

Programmer response

Reissue the command specifying a namelist that is not empty and has a suitable type.

3217 (0C91) (RC3217): MQRCCF_NO_CHANNEL_INITIATOR

Explanation

Channel initiator not active.

The command requires the channel initiator to be started.

3218 (0C93) (RC3218): MQRCCF_CHANNEL_INITIATOR_ERROR

Explanation

Channel initiator cannot be started, or no suitable channel initiator is available.

This might occur because of the following reasons:

- The channel initiator cannot be started because:
  - It is already active.
  - There are insufficient system resources.
  - The queue manager was shutting down.
- The shared channel cannot be started because there was no suitable channel initiator available for any active queue manager in the queue-sharing group. This could be because:
  - No channel initiators are running.
  - The channel initiators that are running are too busy to allow any channel, or a channel of the particular type, to be started.

3222 (0C96) (RC3222): MQRCCF_COMMAND_LEVEL_CONFLICT

Explanation

Incompatible queue manager command levels.

Changing the *CFLevel* parameter of a CF structure, or deleting a CF structure, requires that all queue managers in the queue-sharing group have a command level of at least 530. Some of the queue managers have a level less than 530.

3223 (0C97) (RC3223): MQRCCF_Q_ATTR_CONFLICT

Explanation

Queue attributes are incompatible.

The queues involved in a Move Queue command have different values for one or more of these attributes: *DefinitionType*, *HardenGetBackout*, *Usage*. Messages cannot be moved safely if these attributes differ.

3224 (0C98) (RC3224): MQRCCF_EVENTS_DISABLED

Explanation

Events not enabled.

The command required performance or configuration events to be enabled.

Programmer response

Use the Change Queue manager command to enable the events if required.

3225 (0C99) (RC3225): MQRCCF_COMMAND_SCOPE_ERROR

Explanation

Queue-sharing group error.

While processing a command that used the *CommandScope* parameter, an error occurred while trying to send data to the coupling facility.

Programmer response

Notify your system programmer.

3226 (0C9A) (RC3226): MQRCCF_COMMAND_REPLY_ERROR

Explanation

Error saving command reply information.

While processing a command that used the *CommandScope* parameter, or a command for the channel initiator, an error occurred while trying to save information about the command.

Programmer response

The most likely cause is insufficient storage. If the problem persists, you may need to restart the queue manager after making more storage available.

3227 (0C9B) (RC3227): MQRCCF_FUNCTION_RESTRICTED

Explanation

Restricted command or parameter value used.

The command, or the value specified for one of its parameters, is not allowed because the installation and customization options chosen do not allow all functions to be used. The parameter in question might be returned in the message (with parameter identifier MQIACF_PARAMETER_ID).

3228 (0C9C) (RC3228): MQRCCF_PARM_MISSING

Explanation

Required parameter not specified.

The command did not specify a parameter or parameter value that was required. It might be for one of the following reasons:

- A parameter that is always required.
- A parameter that is one of a set of two or more alternative required parameters.
- A parameter that is required because some other parameter was specified.
- A parameter that is a list of values which has too few values.

The parameter in question might be returned in the message (with parameter identifier MQIACF_PARAMETER_ID).

Programmer response

Reissue the command with correct parameters and values.

3229 (0C9D) (RC3229): MQRCCF_PARM_VALUE_ERROR

Explanation

Parameter value invalid.

The value specified for a parameter was not acceptable. It might be for one of the following reasons:

- Outside the acceptable numeric range for the parameter.
- Not one of a list of acceptable values for the parameter.
- Using characters that are invalid for the parameter.
- Completely blank, when such is not allowed for the parameter.
- A filter value that is invalid for the parameter being filtered.

The parameter in question might be returned in the message (with parameter identifier MQIACF_PARAMETER_ID).

Programmer response

Reissue the command with correct parameters and values.

3230 (0C9E) (RC3230): MQRCCF_COMMAND_LENGTH_ERROR

Explanation

Command exceeds allowable length.

The command is so large that its internal form has exceeded the maximum length allowed. The size of the internal form of the command is affected by both the length, and the complexity of the command.

3231 (0C9F) (RC3231): MQRCCF_COMMAND_ORIGIN_ERROR

Explanation

Command issued incorrectly.

The command cannot be issued using command server. This is an internal error.

Programmer response

Notify your system programmer.

3232 (0CA0) (RC3232): MQRCCF_LISTENER_CONFLICT

Explanation

Address conflict for listener.

A listener was already active for a port and IP address combination that conflicted with the *Port* and *IPAddress* values specified by a Start Channel Listener or Stop Channel Listener command. The *Port* and *IPAddress* value combination specified must match a combination for which the listener is active. It cannot be a superset or a subset of that combination.

Programmer response

Reissue the command with correct values, if required.

3233 (0CA1) (RC3233): MQRCCF_LISTENER_STARTED

Explanation

Listener is started.

An attempt was made to start a listener, but it is already active for the requested `TransportType`, `InboundDisposition`, `Port`, and `IPAddress` values. The requested parameter values might be returned in the message, if applicable (with parameter identifiers MQIACH_XMIT_PROTOCOL_TYPE, MQIACH_INBOUND_DISP, MQIACH_PORT_NUMBER, MQCACH_IP_ADDRESS).

3234 (0CA2) (RC3234): MQRCCF_LISTENER_STOPPED

Explanation

Listener is stopped.

An attempt was made to stop a listener, but it is not active or already stopping for the requested `TransportType`, `InboundDisposition`, `Port`, and `IPAddress` values. The requested parameter values might be returned in the message, if applicable (with parameter identifiers MQIACH_XMIT_PROTOCOL_TYPE, MQIACH_INBOUND_DISP, MQIACH_PORT_NUMBER, MQCACH_IP_ADDRESS).

3235 (0CA3) (RC3235): MQRCCF_CHANNEL_ERROR

Explanation

Channel command failed.

A channel command failed because of an error in the channel definition, or at the remote end of the channel, or in the communications system. An error identifier value *nnn* may be returned in the message (with parameter identifier MQIACF_ERROR_ID).

Programmer response

For information about the error, see the explanation of the corresponding error message. Error *nnn* generally corresponds to message CSQX *nnn*, although there are some exceptions. z/OS For more information, see Distributed queuing message codes.

3236 (0CA4) (RC3236): MQRCCF_CF_STRUC_ERROR

Explanation

CF structure error.

A command could not be processed because of a coupling facility or CF structure error. It might be:

- A Backup CF Structure or Recover CF Structure command when the status of the CF structure is unsuitable. In this case, the CF structure status might be returned in the message together with the CF structure name (with parameter identifiers MQIACF_CF_STRUC_STATUS and MQCA_CF_STRUC_NAME).
- A command could not access an object because of an error in the coupling facility information, or because a CF structure has failed. In this case, the name of the object involved might be returned in the message (with parameter identifier MQCA_Q_NAME, for example).
- A command involving a shared channel could not access the channel status or synchronization key information.

Programmer response

In the case of a Backup CF Structure or Recover CF Structure command, take action appropriate to the CF structure status reported.

In other cases, check for error messages on the console log that might relate to the problem. Check whether the coupling facility structure has failed and check that Db2 is available.

3237 (0CA5) (RC3237): MQRCCF_UNKNOWN_USER_ID

Explanation

User identifier not found.

A user identifier specified in a Reverify Security command was not valid because there was no entry found for it in the internal control table. This could be because the identifier was entered incorrectly in the command, or because it was not in the table (for example, because it had timed-out). The user identifier in question might be returned in the message (with parameter identifier MQCACF_USER_IDENTIFIER).

3238 (0CA6) (RC3238): MQRCCF_UNEXPECTED_ERROR

Explanation

Unexpected or severe error.

An unexpected or severe error or other failure occurred. A code associated with the error might be returned in the message (with parameter identifier MQIACF_ERROR_ID).

Programmer response

Notify your system programmer.

3239 (0CA7) (RC3239): MQRCCF_NO_XCF_PARTNER

Explanation

MQ is not connected to the XCF partner.

The command involving the IMS bridge cannot be processed because MQ is not connected to the XCF partner. The group and member names of the XCF partner in question might be returned in the message (with parameter identifiers MQCA_XCF_GROUP_NAME and MQCA_XCF_MEMBER_NAME).

3240 (0CA8) (RC3240): MQRCCF_CFGR_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFGR *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3241 (0CA9) (RC3241): MQRCCF_CFIF_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFIF *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3242 (0CAA) (RC3242): MQRCCF_CFIF_OPERATOR_ERROR

Explanation

Parameter count not valid.

The MQCFIF *Operator* field value was not valid.

Programmer response

Specify a valid operator value.

3243 (0CAB) (RC3243): MQRCCF_CFIF_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFIF *Parameter* field value was not valid, or specifies a parameter that cannot be filtered, or that is also specified as a parameter to select a subset of objects.

Programmer response

Specify a valid parameter identifier.

3244 (0CAC) (RC3244): MQRCCF_CFSF_FILTER_VAL_LEN_ERR

Explanation

Filter value length not valid.

The MQCFSF *FilterValueLength* field value was not valid.

Programmer response

Specify a valid length.

3245 (0CAD) (RC3245): MQRCCF_CFSF_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFSF *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3246 (0CAE) (RC3246): MQRCCF_CFSF_OPERATOR_ERROR

Explanation

Parameter count not valid.

The MQCFSF `Operator` field value was not valid.

Programmer response

Specify a valid operator value.

3247 (0CAF) (RC3247): MQRCCF_CFSF_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFSF `Parameter` field value was not valid.

Programmer response

Specify a valid parameter identifier.

3248 (0CB0) (RC3248): MQRCCF_TOO_MANY_FILTERS

Explanation

Too many filters.

The command contained more than the maximum permitted number of filter structures.

Programmer response

Specify the command correctly.

3249 (0CB1) (RC3249): MQRCCF_LISTENER_RUNNING

Explanation

Listener is running.

An attempt was made to perform an operation on a listener, but it is currently active.

Programmer response

Stop the listener if required.

3250 (0CB2) (RC3250): MQRCCF_LSTR_STATUS_NOT_FOUND

Explanation

Listener status not found.

For Inquire Listener Status, no listener status is available for the specified listener. This might indicate that the listener has not been used.

Programmer response

None, unless this is unexpected, in which case consult your systems administrator.

3251 (0CB3) (RC3251): MQRCCF_SERVICE_RUNNING

Explanation

Service is running.

An attempt was made to perform an operation on a service, but it is currently active.

Programmer response

Stop the service if required.

3252 (0CB4) (RC3252): MQRCCF_SERV_STATUS_NOT_FOUND

Explanation

Service status not found.

For Inquire Service Status, no service status is available for the specified service. This might indicate that the service has not been used.

Programmer response

None, unless this is unexpected, in which case consult your systems administrator.

3253 (0CB5) (RC3253): MQRCCF_SERVICE_STOPPED

Explanation

Service is stopped.

An attempt was made to stop a service, but it is not active or already stopping.

3254 (0CB6) (RC3254): MQRCCF_CFBS_DUPLICATE_PARM

Explanation

Duplicate parameter.

Two MQCFBS structures with the same parameter identifier were present.

Programmer response

Check for and remove duplicate parameters.

3255 (0CB7) (RC3255): MQRCCF_CFBS_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFBS *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3256 (0CB8) (RC3256): MQRCCF_CFBS_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFBS *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3257 (0CB9) (RC3257): MQRCCF_CFBS_STRING_LENGTH_ERR

Explanation

String length not valid.

The MQCFBS *StringLength* field value was not valid. The value was negative or greater than the maximum permitted length of the parameter specified in the *Parameter* field.

Programmer response

Specify a valid string length for the parameter.

3258 (0CBA) (RC3258): MQRCCF_CFGR_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFGR *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3259 (0CBB) (RC3259): MQRCCF_CFGR_PARM_COUNT_ERROR

Explanation

Parameter count not valid.

The MQCFGR *ParameterCount* field value was not valid. The value was negative or greater than the maximum permitted for the parameter identifier specified in the *Parameter* field.

Programmer response

Specify a valid count for the parameter.

3260 (0CBC) (RC3260): MQRCCF_CONN_NOT_STOPPED

Explanation

Connection not stopped.

The Stop Connection command could not be executed, so the connection was not stopped.

3261 (0CBD) (RC3261): MQRCCF_SERVICE_REQUEST_PENDING

Explanation

A Suspend or Resume Queue Manager command was issued, or a Refresh Security command, but such a command is currently in progress.

Programmer response

Wait until the current request completes, then reissue the command if necessary.

3262 (0CBE) (RC3262): MQRCCF_NO_START_CMD

Explanation

No start command.

The service cannot be started because no start command is specified in the service definition.

Programmer response

Correct the definition of the service.

3263 (0CBF) (RC3263): MQRCCF_NO_STOP_CMD

Explanation

No stop command.

The service cannot be stopped because no stop command is specified in the service definition.

Programmer response

Correct the definition of the service.

3264 (0CC0) (RC3264): MQRCCF_CFBF_LENGTH_ERROR

Explanation

Structure length not valid.

The MQCFBF *StrucLength* field value was not valid.

Programmer response

Specify a valid structure length.

3265 (0CC1) (RC3265): MQRCCF_CFBF_PARM_ID_ERROR

Explanation

Parameter identifier is not valid.

The MQCFBF *Parameter* field value was not valid.

Programmer response

Specify a valid parameter identifier.

3266 (0CC2) (RC3266): MQRCCF_CFBF_FILTER_VAL_LEN_ERR

Explanation

Filter value length not valid.

The MQCFBF *FilterValueLength* field value was not valid.

Programmer response

Specify a valid length.

3267 (0CC3) (RC3267): MQRCCF_CFBF_OPERATOR_ERROR

Explanation

Parameter count not valid.

The MQCFBF *Operator* field value was not valid.

Programmer response

Specify a valid operator value.

3268 (0CC4) (RC3268): MQRCCF_LISTENER_STILL_ACTIVE

Explanation

Listener still active.

An attempt was made to stop a listener, but it failed and the listener is still active. For example, the listener might still have active channels.

Programmer response

Wait for the active connections to the listener to complete before trying the request again.

3269 (0CC5) (RC3269): MQRCCF_DEF_XMIT_Q_CLUS_ERROR

Explanation

The specified queue is not allowed to be used as the default transmission queue because it is reserved for use exclusively by clustering.

Programmer response

Change the value of the Default Transmission Queue, and try the command again.

3300 (0CE4) (RC3300): MQRCCF_TOPICSTR_ALREADY_EXISTS

Explanation

The topic string specified already exists in another topic object.

Programmer response

Verify that the topic string used is correct.

3301 (0CE5) (RC3301): MQRCCF_SHARING_CONVS_ERROR

Explanation

An invalid value has been given for SharingConversations parameter in the Channel definition

Programmer response

Correct the value used in the PCF SharingConversations (MQCFIN) parameter; see Change, Copy, and Create Channel for more information.

3302 (0CE6) (RC3302): MQRCCF_SHARING_CONVS_TYPE

Explanation

SharingConversations parameter is not allowed for this channel type.

Programmer response

See Change, Copy, and Create Channel to ensure that the channel type is compatible with the SharingConversations parameter.

3303 (0CE7) (RC3303): MQRCCF_SECURITY_CASE_CONFLICT

Explanation

A Refresh Security PCF command was issued, but the case currently in use differs from the system setting and if refreshed would result in the set of classes using different case settings.

Programmer response

Check that the class used is set up correctly and that the system setting is correct. If a change in case setting is required, issue the REFRESH SECURITY(*) command to change all classes.

3305 (0CE9) (RC3305): MQRCCF_TOPIC_TYPE_ERROR

Explanation

An Inquire or Delete Topic PCF command was issued with an invalid TopicType parameter.

Programmer response

Correct the TopicType parameter and reissue the command. For more details on the TopicType, see Change, Copy, and Create Topic.

3306 (0CEA) (RC3306): MQRCCF_MAX_INSTANCES_ERROR

Explanation

An invalid value was given for the maximum number of simultaneous instances of a server-connection channel (MaxInstances) for the channel definition.

Programmer response

See Change, Copy, and Create Channel for more information and correct the PCF application.

3307 (0CEB) (RC3307): MQRCCF_MAX_INSTS_PER_CLNT_ERR

Explanation

An invalid value was given for the MaxInstancesPerClient property.

Programmer response

See Change, Copy, and Create Channel for the range of values and correct the application.

3308 (0CEC) (RC3308): MQRCCF_TOPIC_STRING_NOT_FOUND

Explanation

When processing an Inquire Topic Status command, the topic string specified did not match any topic nodes in the topic tree.

Programmer response

Verify the topic string is correct.

3309 (0CED) (RC3309): MQRCCF_SUBSCRIPTION_POINT_ERR

Explanation

The Subscription point was not valid. Valid subscription points are the topic strings of the topic objects listed in the SYSTEM.QPUBSUB.SUBPOINT.NAMELIST.

Programmer response

Use a subscription point that matches the topic string of a topic object listed in the SYSTEM.QPUBSUB.SUBPOINT.NAMELIST (or remove the subscription point parameter and this uses the default subscription point)

3311 (0CEF) (RC2432): MQRCCF_SUB_ALREADY_EXISTS

Explanation

When processing a Copy or Create Subscription command, the target *Subscription* identifier exists.

Programmer response

If you are trying to copy an existing subscription, ensure that the *ToSubscriptionName* parameter contains a unique value. If you are trying to create a Subscription ensure that the combination of the *SubName* parameter, and *TopicObject* parameter or *TopicString* parameter are unique.

3314 (0CF2) (RC3314): MQRCCF_DURABILITY_NOT_ALLOWED

Explanation

An MQSUB call using the MQSO_DURABLE option failed. This can be for one of the following reasons:

- The topic subscribed to is defined as DURSUB(NO).
- The queue named SYSTEM.DURABLE.SUBSCRIBER.QUEUE is not available.
- The topic subscribed to is defined as both MCAST(ONLY) and DURSUB(YES) (or DURSUB(ASPARENT) and the parent is DURSUB(YES)).

Completion Code

MQCC_FAILED

Programmer Response

Durable subscriptions are stored on the SYSTEM.DURABLE.SUBSCRIBER.QUEUE. Ensure that this queue is available for use. Possible reasons for failure include the queue being full, the queue being put inhibited, the queue not existing, or (on z/OS) the pageset the queue is defined to use doesn't exist.

If the topic subscribed to is defined as DURSUB(NO) either alter the administrative topic node to use DURSUB(YES) or use the MQSO_NON_DURABLE option instead.

If the topic subscribed to is defined as MCAST(ONLY) when using IBM MQ Multicast messaging, alter the topic to use DURSUB(NO).

3317 (0CF5) (RC3317): MQRCCF_INVALID_DESTINATION

Explanation

The Subscription or Topic object used in a Change, Copy, Create or Delete PCF command is invalid.

Programmer response

Investigate and correct the required parameters for the specific command you are using. For more details, see Change, Copy, and Create Subscription.

3318 (0CF6) (RC3318): MQRCCF_PUBSUB_INHIBITED

Explanation

MQSUB, MQOPEN, MQPUT and MQPUT1 calls are currently inhibited for all publish/subscribe topics, either by means of the queue manager attribute PSMODE or because processing of publish/subscribe state at queue manager start-up has failed, or has not yet completed.

Completion Code

MQCC_FAILED

Programmer response

If this queue manager does not intentionally inhibit publish/subscribe, investigate any error messages that describe the failure at queue manager start-up, or wait until start-up processing completes. You can use the DISPLAY PUBSUB command to check the status of the publish/subscribe engine to ensure it is ready for use, and additionally on z/OS you will receive an information message CSQM076I.

3326 (0CFE) (RC3326): MQRCCF_CHLAUTH_TYPE_ERROR

Explanation

Channel authentication record type not valid.

The **`type`** parameter specified on the **`set`** command was not valid.

Programmer response

Specify a valid type.

3327 (0CFF) (RC3327): MQRCCF_CHLAUTH_ACTION_ERROR

Explanation

Channel authentication record action not valid.

The **`action`** parameter specified on the **`set`** command was not valid.

Programmer response

Specify a valid action.

3335 (0D07) (RC3335): MQRCCF_CHLAUTH_USRSRC_ERROR

Explanation

Channel authentication record user source not valid.

The **`user source`** parameter specified on the **`set`** command was not valid.

Programmer response

Specify a valid user source.

3336 (0D08) (RC3336): MQRCCF_WRONG_CHLAUTH_TYPE

Explanation

Parameter not allowed for this channel authentication record type.

The parameter is not allowed for the type of channel authentication record being set. Refer to the description of the parameter in error to determine the types of record for which this parameter is valid.

Programmer response
Remove the parameter.

3337 (0D09) (RC3337): MQRCCF_CHLAUTH_ALREADY_EXISTS

Explanation
Channel authentication record already exists

An attempt was made to add a channel authentication record, but it already exists.

Programmer response
Specify action as MQACT_REPLACE.

3338 (0D0A) (RC3338): MQRCCF_CHLAUTH_NOT_FOUND

Explanation
Channel authentication record not found.

The specified channel authentication record does not exist.

Programmer response
Specify a channel authentication record that exists.

3339 (0D0B) (RC3339): MQRCCF_WRONG_CHLAUTH_ACTION

Explanation
Parameter not allowed for this action on a channel authentication record.

The parameter is not allowed for the action being applied to a channel authentication record. Refer to the description of the parameter in error to determine the actions for which this parameter is valid.

Programmer response
Remove the parameter.

3340 (0D0C) (RC3340): MQRCCF_WRONG_CHLAUTH_USERSRC

Explanation
Parameter not allowed for this channel authentication record user source value.

The parameter is not allowed for a channel authentication record with the value that the **`user source`** field contains. Refer to the description of the parameter in error to determine the values of user source for which this parameter is valid.

Programmer response
Remove the parameter.

3341 (0D0D) (RC3341): MQRCCF_CHLAUTH_WARN_ERROR

Explanation
Channel authentication record **`warn`** value not valid.

The **warn** parameter specified on the **set** command was not valid.

Programmer response
Specify a valid value for **warn**.

3342 (0D0E) (RC3342): MQRCCF_WRONG_CHLAUTH_MATCH

Explanation
Parameter not allowed for this channel authentication record **match** value.

The parameter is not allowed for an **inquire channel authentication record** command with the value that the **match** field contains. Refer to the description of the parameter in error to find the values of **match** for which this parameter is valid.

Programmer response
Remove the parameter.

3343 (0D0F) (RC3343): MQRCCF_IPADDR_RANGE_CONFLICT

Explanation
A channel authentication record contained an IP address with a range that overlapped an existing range. A range must be a superset or subset of any existing ranges for the same channel profile name, or completely separate.

Programmer response
Specify a range that is a superset or subset of an existing range, or is completely separate to all existing ranges.

3344 (0D10) (RC3344): MQRCCF_CHLAUTH_MAX_EXCEEDED

Explanation
A channel authentication record was set taking the total number of entries for that type on a single channel profile over the maximum number allowed.

Programmer response
Remove some channel authentication records to make room.

3345 (0D11) (RC3345): MQRCCF_IPADDR_ERROR

Explanation
A channel authentication record contained an invalid IP address, or invalid wildcard pattern to match against IP addresses.

Programmer response
Specify a valid IP address or pattern.
Related information
Generic IP addresses

3346 (0D12) (RC3346): MQRCCF_IPADDR_RANGE_ERROR

Explanation

A channel authentication record contained an IP address with a range that was invalid, for example, the lower number is higher than or equal to the upper number of the range.

Programmer response

Specify a valid range in the IP address.

3347 (0D13) (RC3347): MQRCCF_PROFILE_NAME_MISSING

Explanation

Profile name missing.

A profile name was required for the command but none was specified.

Programmer response

Specify a valid profile name.

3348 (0D14) (RC3348): MQRCCF_CHLAUTH_CLNTUSER_ERROR

Explanation

Channel authentication record **`client user`** value not valid.

The **`client user`** value contains a wildcard character, which is not allowed.

Programmer response

Specify a valid value for the client user field.

3349 (0D15) (RC3349): MQRCCF_CHLAUTH_NAME_ERROR

Explanation

Channel authentication record channel name not valid.

When a channel authentication record specifies an IP address to block, the **`channel name`** value must be a single asterisk (*).

Programmer response

Enter a single asterisk in the channel name.

3350 (0D16) (RC3350): MQRCCF_CHLAUTH_RUNCHECK_ERROR

Runcheck command is using generic values.

Explanation

An Inquire Channel Authentication Record command using MQMATCH_RUNCHECK was issued, but one or more of the input fields on the command were provided with generic values, which is not allowed.

Programmer response

Enter non-generic values for channel name, address, one of the client user ID or remote queue manager and SSL Peer Name if used.

3353 (0D19) (RC3353): MQRCCF_SUITE_B_ERROR

Invalid values have been specified.

Explanation

An invalid combination of values has been specified for the **MQIA_SUITE_B_STRENGTH** parameter.

Programmer response

Review the combination entered and retry with appropriate values.

3363 (0D23) (RC3363): MQRCCF_CLUS_XMIT_Q_USAGE_ERROR

Explanation

If the local queue attribute **CLCHNAME** is set, the attribute **USAGE** must be set to XMITQ.

On z/OS, if the local queue attribute **CLCHNAME** is set, the attribute **INDXTYPE** must be set to **CORRELID**, and the transmission queue must not be a shared queue.

The **CLCHNAME** attribute is a generic cluster-sender channel name. It identifies the cluster-sender channel that transfers messages in a transmission queue to another queue manager.

Programmer response

Modify the application to set the **CLCHNAME** to blanks, or not set the **CLCHNAME** attribute at all, on queues other than transmission queues.

On z/OS, ensure that the transmission queue is indexed by correlation ID and the queue is not a shared queue.

3364 (0D24) (RC3364): MQRCCF_CERT_VAL_POLICY_ERROR

Explanation

An invalid certificate validation policy value was specified for the **MQIA_CERT_VAL_POLICY** attribute. The specified value is unknown or is not supported on the current platform.

Programmer response

Review the value specified and try again with an appropriate certificate validation policy.

3366 (0D26) (RC3366): MQRCCF_REVDNS_DISABLED

Explanation

A runcheck command completed successfully returning the records to be used. However, some Channel Authentication Records exist containing hostnames, and hostname reverse lookup is currently disabled, so these records will not have been matched against. This reason code is returned as an MQCC_WARNING.

Programmer response

If reverse lookup is correctly disabled, even though some Channel Authentication Records exist containing hostnames, this warning can be ignored.

If channel authentication records containing hostnames should be being matched against, and therefore reverse lookup of hostname should not currently be disabled, issue a Change Queue Manager command to re-enable it.

If reverse lookup for hostnames is correctly disabled and there should not be any Channel Authentication Records containing hostnames, issue a Set Channel Authentication Record to remove them.

3370 (0D2A) (RC3370): MQRCCF_CHLAUTH_CHKCLI_ERROR

Explanation

Channel authentication record check client not valid.

The check client parameter specified on the set command was not valid.

Programmer response

Specify a valid user source.

V 8.0.0.2 3377 (0D31) (RC3377): MQRCCF_TOPIC_RESTRICTED

Explanation

This error can occur when creating or modifying a topic object. One or more attributes of the topic object are not supported on an IBM MQ administrative topic.

Programmer response

Modify the configuration to adhere to the documented restrictions.

4001 (0FA1) (RC4001): MQRCCF_OBJECT_ALREADY_EXISTS

Explanation

Object already exists.

An attempt was made to create an object, but the object already existed and the *Replace* parameter was not specified as MQRP_YES.

Programmer response

Specify *Replace* as MQRP_YES, or use a different name for the object to be created.

4002 (0FA2) (RC4002): MQRCCF_OBJECT_WRONG_TYPE

Explanation

Object has wrong type or disposition.

An object already exists with the same name but a different subtype or disposition from that specified by the command.

Programmer response

Ensure that the specified object is the same subtype and disposition.

4003 (0FA3) (RC4003): MQRCCF_LIKE_OBJECT_WRONG_TYPE

Explanation

New and existing objects have different subtype.

An attempt was made to create an object based on the definition of an existing object, but the new and existing objects had different subtypes.

Programmer response

Ensure that the new object has the same subtype as the one on which it is based.

4004 (0FA4) (RC4004): MQRCCF_OBJECT_OPEN

Explanation

Object is open.

An attempt was made to operate on an object that was in use.

Programmer response

Wait until the object is not in use, and then retry the operation. Alternatively specify *Force* as MQFC_YES for a change command.

4005 (0FA5) (RC4005): MQRCCF_ATTR_VALUE_ERROR

Explanation

Attribute value not valid or repeated.

One or more of the attribute values specified are not valid or are repeated. The error response message contains the failing attribute selectors (with parameter identifier MQIACF_PARAMETER_ID).

Programmer response

Specify the attribute values correctly.

4006 (0FA6) (RC4006): MQRCCF_UNKNOWN_Q_MGR

Explanation

Queue manager not known.

The queue manager specified was not known.

Programmer response

Specify the name of the queue manager to which the command is sent, or blank.

4007 (0FA7) (RC4007): MQRCCF_Q_WRONG_TYPE

Explanation

Action not valid for the queue of specified type.

An attempt was made to perform an action on a queue of the wrong type.

Programmer response

Specify a queue of the correct type.

4008 (0FA8) (RC4008): MQRCCF_OBJECT_NAME_ERROR

Explanation

Name not valid.

An object or other name name was specified using characters that were not valid.

Programmer response

Specify only valid characters for the name.

4009 (0FA9) (RC4009): MQRCCF_ALLOCATE_FAILED

Explanation

Allocation failed.

An attempt to allocate a conversation to a remote system failed. The error might be due to an entry in the channel definition that is not valid, or it might be that the listening program at the remote system is not running.

Programmer response

Ensure that the channel definition is correct, and start the listening program if necessary. If the error persists, consult your systems administrator.

4010 (0FAA) (RC4010): MQRCCF_HOST_NOT_AVAILABLE

Explanation

Remote system not available.

An attempt to allocate a conversation to a remote system was unsuccessful. The error might be transitory, and the allocate might succeed later. This reason can occur if the listening program at the remote system is not running.

Programmer response

Ensure that the listening program is running, and retry the operation.

4011 (0FAB) (RC4011): MQRCCF_CONFIGURATION_ERROR

Explanation

Configuration error.

There was a configuration error in the channel definition or communication subsystem, and allocation of a conversation was not possible. This might be caused by one of the following:

- For LU 6.2, either the *ModeName* or the *TpName* is incorrect. The *ModeName* must match that on the remote system, and the *TpName* must be specified. (On IBM i, these are held in the communications Side Object.)
- For LU 6.2, the session might not be established.
- For TCP, the *ConnectionName* in the channel definition cannot be resolved to a network address. This might be because the name has not been correctly specified, or because the name server is not available.
- The requested communications protocol might not be supported on the platform.

Programmer response

Identify the error and take appropriate action.

4012 (0FAC) (RC4012): MQRCCF_CONNECTION_REFUSED

Explanation

Connection refused.

The attempt to establish a connection to a remote system was rejected. The remote system might not be configured to allow a connection from this system.

- For LU 6.2 either the user ID or the password supplied to the remote system is incorrect.
- For TCP the remote system might not recognize the local system as valid, or the TCP listener program might not be started.

Programmer response

Correct the error or restart the listener program.

4013 (0FAD) (RC4013): MQRCCF_ENTRY_ERROR

Explanation

Connection name not valid.

The connection name in the channel definition could not be resolved into a network address. Either the name server does not contain the entry, or the name server was not available.

Programmer response

Ensure that the connection name is correctly specified and that the name server is available.

4014 (0FAE) (RC4014): MQRCCF_SEND_FAILED

Explanation

Send failed.

An error occurred while sending data to a remote system. This might be caused by a communications failure.

Programmer response

Consult your systems administrator.

4015 (0FAF) (RC4015): MQRCCF_RECEIVED_DATA_ERROR

Explanation

Received data error.

An error occurred while receiving data from a remote system. This might be caused by a communications failure.

Programmer response

Consult your systems administrator.

4016 (0FB0) (RC4016): MQRCCF_RECEIVE_FAILED

Explanation

Receive failed.

The receive operation failed.

Programmer response

Correct the error and retry the operation.

4017 (0FB1) (RC4017): MQRCCF_CONNECTION_CLOSED

Explanation

Connection closed.

An error occurred while receiving data from a remote system. The connection to the remote system has unexpectedly terminated.

Programmer response

Contact your systems administrator.

4018 (0FB2) (RC4018): MQRCCF_NO_STORAGE

Explanation

Not enough storage available.

Insufficient storage is available.

Programmer response

Consult your systems administrator.

4019 (0FB3) (RC4019): MQRCCF_NO_COMMS_MANAGER

Explanation

Communications manager not available.

The communications subsystem is not available.

Programmer response

Ensure that the communications subsystem has been started.

4020 (0FB4) (RC4020): MQRCCF_LISTENER_NOT_STARTED

Explanation

Listener not started.

The listener program could not be started. Either the communications subsystem has not been started, or the number of current channels using the communications subsystem is the maximum allowed, or there are too many jobs waiting in the queue.

Programmer response

Ensure the communications subsystem is started or retry the operation later. Increase the number of current channels allowed, if appropriate.

4024 (0FB8) (RC4024): MQRCCF_BIND_FAILED

Explanation

Bind failed.

The bind to a remote system during session negotiation has failed.

Programmer response

Consult your systems administrator.

4025 (0FB9) (RC4025): MQRCCF_CHANNEL_INDOUBT

Explanation

Channel in-doubt.

The requested operation cannot complete because the channel is in doubt.

Programmer response

Examine the status of the channel, and either restart a channel to resolve the in-doubt state, or resolve the channel.

4026 (0FBA) (RC4026): MQRCCF_MQCONN_FAILED

Explanation

MQCONN call failed.

Programmer response

Check whether the queue manager is active.

4027 (0FBB) (RC4027): MQRCCF_MQOPEN_FAILED

Explanation

MQOPEN call failed.

Programmer response

Check whether the queue manager is active, and the queues involved are correctly set up.

4028 (0FBC) (RC4028): MQRCCF_MQGET_FAILED

Explanation

MQGET call failed.

Programmer response

Check whether the queue manager is active, and the queues involved are correctly set up, and enabled for MQGET.

4029 (0FBD) (RC4029): MQRCCF_MQPUT_FAILED

Explanation

MQPUT call failed.

Programmer response

Check whether the queue manager is active, and the queues involved are correctly set up, and not inhibited for puts.

4030 (0FBE) (RC4030): MQRCCF_PING_ERROR

Explanation

Ping error.

A ping operation can only be issued for a sender or server channel. If the local channel is a receiver channel, you must issue the ping from a remote queue manager.

Programmer response

Reissue the ping request for a different channel of the correct type, or for a receiver channel from a different queue manager.

4031 (0FBF) (RC4031): MQRCCF_CHANNEL_IN_USE

Explanation

Channel in use.

An attempt was made to perform an operation on a channel, but the channel is currently active.

Programmer response

Stop the channel or wait for it to terminate.

4032 (0FC0) (RC4032): MQRCCF_CHANNEL_NOT_FOUND

Explanation

Channel not found.

The channel specified does not exist.

Programmer response

Specify the name of a channel which exists.

4033 (0FC1) (RC4033): MQRCCF_UNKNOWN_REMOTE_CHANNEL

Explanation

Remote channel not known.

There is no definition of the referenced channel at the remote system.

Programmer response

Ensure that the local channel is correctly defined. If it is, add an appropriate channel definition at the remote system.

4034 (0FC2) (RC4034): MQRCCF_REMOTE_QM_UNAVAILABLE

Explanation

Remote queue manager not available.

The channel cannot be started because the remote queue manager is not available.

Programmer response

Start the remote queue manager.

4035 (0FC3) (RC4035): MQRCCF_REMOTE_QM_TERMINATING

Explanation

Remote queue manager terminating.

The channel is ending because the remote queue manager is terminating.

Programmer response

Restart the remote queue manager.

4036 (0FC4) (RC4036): MQRCCF_MQINQ_FAILED

Explanation

MQINQ call failed.

Programmer response

Check whether the queue manager is active.

4037 (0FC5) (RC4037): MQRCCF_NOT_XMIT_Q

Explanation

Queue is not a transmission queue.

The queue specified in the channel definition is not a transmission queue, or is in use.

Programmer response

Ensure that the queue is specified correctly in the channel definition, and that it is correctly defined to the queue manager.

4038 (0FC6) (RC4038): MQRCCF_CHANNEL_DISABLED

Explanation

Channel disabled.

An attempt was made to use a channel, but the channel was disabled (that is, stopped).

Programmer response

Start the channel.

4039 (0FC7) (RC4039): MQRCCF_USER_EXIT_NOT_AVAILABLE

Explanation

User exit not available.

The channel was terminated because the user exit specified does not exist.

Programmer response

Ensure that the user exit is correctly specified and the program is available.

4040 (0FC8) (RC4040): MQRCCF_COMMIT_FAILED

Explanation

Commit failed.

An error was received when an attempt was made to commit a unit of work.

Programmer response

Consult your systems administrator.

4041 (0FC9) (RC4041): MQRCCF_WRONG_CHANNEL_TYPE

Explanation

Parameter not allowed for this channel type.

The parameter is not allowed for the type of channel being created, copied, or changed. Refer to the description of the parameter in error to determine the types of channel for which the parameter is valid

Programmer response

Remove the parameter.

4042 (0FCA) (RC4042): MQRCCF_CHANNEL_ALREADY_EXISTS

Explanation

Channel already exists.

An attempt was made to create a channel but the channel already existed and *Replace* was not specified as MQRP_YES.

Programmer response

Specify *Replace* as MQRP_YES or use a different name for the channel to be created.

4043 (0FCB) (RC4043): MQRCCF_DATA_TOO_LARGE

Explanation

Data too large.

The data to be sent exceeds the maximum that can be supported for the command.

Programmer response

Reduce the size of the data.

4044 (0FCC) (RC4044): MQRCCF_CHANNEL_NAME_ERROR

Explanation

Channel name error.

The *ChannelName* parameter contained characters that are not allowed for channel names.

Programmer response

Specify a valid name.

4045 (0FCD) (RC4045): MQRCCF_XMIT_Q_NAME_ERROR

Explanation

Transmission queue name error.

The *XmitQName* parameter contains characters that are not allowed for queue names. This reason code also occurs if the parameter is not present when a sender or server channel is being created, and no default value is available.

Programmer response

Specify a valid name, or add the parameter.

4047 (0FCF) (RC4047): MQRCCF_MCA_NAME_ERROR

Explanation

Message channel agent name error.

The *MCAName* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4048 (0FD0) (RC4048): MQRCCF_SEND_EXIT_NAME_ERROR

Explanation

Channel send exit name error.

The *SendExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4049 (0FD1) (RC4049): MQRCCF_SEC_EXIT_NAME_ERROR

Explanation

Channel security exit name error.

The *SecurityExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4050 (0FD2) (RC4050): MQRCCF_MSG_EXIT_NAME_ERROR

Explanation

Channel message exit name error.

The *MsgExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4051 (0FD3) (RC4051): MQRCCF_RCV_EXIT_NAME_ERROR

Explanation

Channel receive exit name error.

The *ReceiveExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4052 (0FD4) (RC4052): MQRCCF_XMIT_Q_NAME_WRONG_TYPE

Explanation

Transmission queue name not allowed for this channel type.

The *XmitQName* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4053 (0FD5) (RC4053): MQRCCF_MCA_NAME_WRONG_TYPE

Explanation

Message channel agent name not allowed for this channel type.

The *MCAName* parameter is only allowed for sender, server or requester channel types.

Programmer response

Remove the parameter.

4054 (0FD6) (RC4054): MQRCCF_DISC_INT_WRONG_TYPE

Explanation

Disconnection interval not allowed for this channel type.

The *DiscInterval* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4055 (0FD7) (RC4055): MQRCCF_SHORT_RETRY_WRONG_TYPE

Explanation

Short retry parameter not allowed for this channel type.

The *ShortRetryCount* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4056 (0FD8) (RC4056): MQRCCF_SHORT_TIMER_WRONG_TYPE

Explanation

Short timer parameter not allowed for this channel type.

The *ShortRetryInterval* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4057 (0FD9) (RC4057): MQRCCF_LONG_RETRY_WRONG_TYPE

Explanation

Long retry parameter not allowed for this channel type.

The *LongRetryCount* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4058 (0FDA) (RC4058): MQRCCF_LONG_TIMER_WRONG_TYPE

Explanation

Long timer parameter not allowed for this channel type.

The *LongRetryInterval* parameter is only allowed for sender or server channel types.

Programmer response

Remove the parameter.

4059 (0FDB) (RC4059): MQRCCF_PUT_AUTH_WRONG_TYPE

Explanation

Put authority parameter not allowed for this channel type.

The *PutAuthority* parameter is only allowed for receiver or requester channel types.

Programmer response

Remove the parameter.

4061 (0FDD) (RC4061): MQRCCF_MISSING_CONN_NAME

Explanation

Connection name parameter required but missing.

The *ConnectionName* parameter is required for sender or requester channel types, but is not present.

Programmer response

Add the parameter.

4062 (0FDE) (RC4062): MQRCCF_CONN_NAME_ERROR

Explanation

Error in connection name parameter.

The *ConnectionName* parameter contains one or more blanks at the start of the name.

Programmer response

Specify a valid connection name.

4063 (0FDF) (RC4063): MQRCCF_MQSET_FAILED

Explanation

MQSET call failed.

Programmer response

Check whether the queue manager is active.

4064 (0FE0) (RC4064): MQRCCF_CHANNEL_NOT_ACTIVE

Explanation

Channel not active.

An attempt was made to stop a channel, but the channel was already stopped.

Programmer response

No action is required.

4065 (0FE1) (RC4065): MQRCCF_TERMINATED_BY_SEC_EXIT

Explanation

Channel terminated by security exit.

A channel security exit terminated the channel.

Programmer response

Check that the channel is attempting to connect to the correct queue manager, and if so that the security exit is specified correctly, and is working correctly, at both ends.

4067 (0FE3) (RC4067): MQRCCF_DYNAMIC_Q_SCOPE_ERROR

Explanation

Dynamic queue scope error.

The *Scope* attribute of the queue is to be MQSCO_CELL, but this is not allowed for a dynamic queue.

Programmer response

Predefine the queue if it is to have cell scope.

4068 (0FE4) (RC4068): MQRCCF_CELL_DIR_NOT_AVAILABLE

Explanation

Cell directory is not available.

The *Scope* attribute of the queue is to be MQSCO_CELL, but no name service supporting a cell directory has been configured.

Programmer response

Configure the queue manager with a suitable name service.

4069 (0FE5) (RC4069): MQRCCF_MR_COUNT_ERROR

Explanation

Message retry count not valid.

The *MsgRetryCount* value was not valid.

Programmer response

Specify a value in the range 0-999 999 999.

4070 (0FE6) (RC4070): MQRCCF_MR_COUNT_WRONG_TYPE

Explanation

Message-retry count parameter not allowed for this channel type.

The *MsgRetryCount* parameter is allowed only for receiver and requester channels.

Programmer response

Remove the parameter.

4071 (0FE7) (RC4071): MQRCCF_MR_EXIT_NAME_ERROR

Explanation

Channel message-retry exit name error.

The *MsgRetryExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4072 (0FE8) (RC4072): MQRCCF_MR_EXIT_NAME_WRONG_TYPE

Explanation

Message-retry exit parameter not allowed for this channel type.

The *MsgRetryExit* parameter is allowed only for receiver and requester channels.

Programmer response

Remove the parameter.

4073 (0FE9) (RC4073): MQRCCF_MR_INTERVAL_ERROR

Explanation

Message retry interval not valid.

The *MsgRetryInterval* value was not valid.

Programmer response

Specify a value in the range 0-999 999 999.

4074 (0FEA) (RC4074): MQRCCF_MR_INTERVAL_WRONG_TYPE

Explanation

Message-retry interval parameter not allowed for this channel type.

The *MsgRetryInterval* parameter is allowed only for receiver and requester channels.

Programmer response

Remove the parameter.

4075 (0FEB) (RC4075): MQRCCF_NPM_SPEED_ERROR

Explanation

Nonpersistent message speed not valid.

The *NonPersistentMsgSpeed* value was not valid.

Programmer response

Specify MQNPMS_NORMAL or MQNPMS_FAST.

4076 (0FEC) (RC4076): MQRCCF_NPM_SPEED_WRONG_TYPE

Explanation

Nonpersistent message speed parameter not allowed for this channel type.

The *NonPersistentMsgSpeed* parameter is allowed only for sender, receiver, server, requester, cluster sender, and cluster receiver channels.

Programmer response

Remove the parameter.

4077 (0FED) (RC4077): MQRCCF_HB_INTERVAL_ERROR

Explanation

Heartbeat interval not valid.

The *HeartbeatInterval* value was not valid.

Programmer response

Specify a value in the range 0-999 999.

4078 (0FEE) (RC4078): MQRCCF_HB_INTERVAL_WRONG_TYPE

Explanation

Heartbeat interval parameter not allowed for this channel type.

The *HeartbeatInterval* parameter is allowed only for receiver and requester channels.

Programmer response

Remove the parameter.

4079 (0FEF) (RC4079): MQRCCF_CHAD_ERROR

Explanation

Channel automatic definition error.

The *ChannelAutoDef* value was not valid.

Programmer response

Specify MQCHAD_ENABLED or MQCHAD_DISABLED.

4080 (0FF0) (RC4080): MQRCCF_CHAD_WRONG_TYPE

Explanation

Channel automatic definition parameter not allowed for this channel type.

The *ChannelAutoDef* parameter is allowed only for receiver and server-connection channels.

Programmer response

Remove the parameter.

4081 (0FF1) (RC4081): MQRCCF_CHAD_EVENT_ERROR

Explanation

Channel automatic definition event error.

The *ChannelAutoDefEvent* value was not valid.

Programmer response

Specify MQEVR_ENABLED or MQEVR_DISABLED.

4082 (0FF2) (RC4082): MQRCCF_CHAD_EVENT_WRONG_TYPE

Explanation

Channel automatic definition event parameter not allowed for this channel type.

The *ChannelAutoDefEvent* parameter is allowed only for receiver and server-connection channels.

Programmer response

Remove the parameter.

4083 (0FF3) (RC4083): MQRCCF_CHAD_EXIT_ERROR

Explanation

Channel automatic definition exit name error.

The *ChannelAutoDefExit* value contained characters that are not allowed for program names on the platform in question.

Programmer response

Specify a valid name.

4084 (0FF4) (RC4084): MQRCCF_CHAD_EXIT_WRONG_TYPE

Explanation

Channel automatic definition exit parameter not allowed for this channel type.

The *ChannelAutoDefExit* parameter is allowed only for receiver and server-connection channels.

Programmer response

Remove the parameter.

4085 (0FF5) (RC4085): MQRCCF_SUPPRESSED_BY_EXIT

Explanation

Action suppressed by exit program.

An attempt was made to define a channel automatically, but this was inhibited by the channel automatic definition exit. The *AuxErrorDataInt1* parameter contains the feedback code from the exit indicating why it inhibited the channel definition.

Programmer response

Examine the value of the *AuxErrorDataInt1* parameter, and take any action that is appropriate.

4086 (0FF6) (RC4086): MQRCCF_BATCH_INT_ERROR

Explanation

Batch interval not valid.

The batch interval specified was not valid.

Programmer response

Specify a valid batch interval value.

4087 (0FF7) (RC4087): MQRCCF_BATCH_INT_WRONG_TYPE

Explanation

Batch interval parameter not allowed for this channel type.

The *BatchInterval* parameter is allowed only for sender and server channels.

Programmer response

Remove the parameter.

4088 (0FF8) (RC4088): MQRCCF_NET_PRIORITY_ERROR

Explanation

Network priority value is not valid.

Programmer response

Specify a valid value.

4089 (0FF9) (RC4089): MQRCCF_NET_PRIORITY_WRONG_TYPE

Explanation

Network priority parameter not allowed for this channel type.

The *NetworkPriority* parameter is allowed for sender and server channels only.

Programmer response

Remove the parameter.

4090 (0FFA) (RC4090): MQRCCF_CHANNEL_CLOSED

Explanation

Channel closed.

The channel was closed prematurely. This can occur because a user stopped the channel while it was running, or a channel exit decided to close the channel.

Programmer response

Determine the reason that the channel was closed prematurely. Restart the channel if required.

4092 (0FFC) (RC4092): MQRCCF_SSL_CIPHER_SPEC_ERROR

Explanation

SSL cipher specification not valid.

The *SSLCipherSpec* specified is not valid.

Programmer response

Specify a valid cipher specification.

4093 (0FFD) (RC4093): MQRCCF_SSL_PEER_NAME_ERROR

Explanation

SSL peer name not valid.

The *SSLPeerName* specified is not valid.

Programmer response

Specify a valid peer name.

4094 (0FFE) (RC4094): MQRCCF_SSL_CLIENT_AUTH_ERROR

Explanation

SSL client authentication not valid.

The *SSLClientAuth* specified is not valid.

Programmer response

Specify a valid client authentication.

4095 (0FFF) (RC4095): MQRCCF_RETAINED_NOT_SUPPORTED

Explanation

Retained messages used on restricted stream.

An attempt has been made to use retained messages on a publish/subscribe stream defined to be restricted to JMS usage. JMS does not support the concept of retained messages and the request is rejected.

Programmer response

Either modify the application not to use retained messages, or modify the broker *JmsStreamPrefix* configuration parameter so that this stream is not treated as a JMS stream.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes

IBM MQ can use Secure Sockets Layer (SSL) with the various communication protocols. Use this topic to identify the error codes that can be returned by SSL.

The table in this appendix documents the return codes, in decimal form, from the Secure Sockets Layer (SSL) that can be returned in messages from the distributed queuing component.

Table 1. SSL return codes

Return code (decimal)	Explanation
1	Handle is not valid.
3	An internal error has occurred.
4	Insufficient storage is available
5	Handle is in the incorrect state.
6	Key label is not found.
7	No certificates available.
8	Certificate validation error.
9	Cryptographic processing error.
10	ASN processing error.
11	LDAP processing error.
12	An unexpected error has occurred.
102	Error detected while reading key database or SAF key ring.

Table 1. SSL return codes (continued)

Return code (decimal)	Explanation
103	Incorrect key database record format.
106	Incorrect key database password.
109	No certificate authority certificates.
201	No key database password supplied.
202	Error detected while opening the key database.
203	Unable to generate temporary key pair
204	Key database password is expired.
302	Connection is active.
401	Certificate is expired or is not valid yet.
402	No SSL cipher specifications.
403	No certificate received from partner.
405	Certificate format is not supported.
406	Error while reading or writing data.
407	Key label does not exist.
408	Key database password is not correct.
410	SSL message format is incorrect.
411	Message authentication code is incorrect.
412	SSL protocol or certificate type is not supported.
413	Certificate signature is incorrect.
414	Certificate is not valid.
415	SSL protocol violation.
416	Permission denied.
417	Self-signed certificate cannot be validated.
420	Socket closed by remote partner.
421	SSL V2 cipher is not valid.
422	SSL V3 cipher is not valid.
427	LDAP is not available.
428	Key entry does not contain a private key.
429	SSL V2 header is not valid.
431	Certificate is revoked.
432	Session renegotiation is not allowed.
433	Key exceeds allowable export size.
434	Certificate key is not compatible with cipher suite.
435	Certificate authority is unknown.

Table 1. SSL return codes (continued)

Return code (decimal)	**Explanation**
436	Certificate revocation list cannot be processed.
437	Connection closed.
438	Internal error reported by remote partner.
439	Unknown alert received from remote partner.
501	Buffer size is not valid.
502	Socket request would block.
503	Socket read request would block.
504	Socket write request would block.
505	Record overflow.
601	Protocol is not SSL V3 or TLS V1.
602	Function identifier is not valid.
701	Attribute identifier is not valid.
702	The attribute has a negative length, which is invalid.
703	The enumeration value is invalid for the specified enumeration type.
704	Invalid parameter list for replacing the SID cache routines.
705	The value is not a valid number.
706	Conflicting parameters were set for additional certificate validation
707	The AES cryptographic algorithm is not supported.
708	The PEERID does not have the correct length.
1501	GSK_SC_OK
1502	GSK_SC_CANCEL
1601	The trace started successfully.
1602	The trace stopped successfully.
1603	No trace file was previously started so it cannot be stopped.
1604	Trace file already started so it cannot be started again.
1605	Trace file cannot be opened. The first parameter of gsk_start_trace() must be a valid full path filename.

In some cases, the secure sockets library reports a certificate validation error in an AMQ9633 error message. Table 2 lists the certificate validation errors that can be returned in messages from the distributed queuing component.

Table 2. Certificate validation errors.

A table listing return codes and explanations for certificate validation errors that can be returned in messages from the distributed queuing component.

Return code (decimal)	**Explanation**
575001	Internal error
575002	ASN error due to a malformed certificate
575003	Cryptographic error
575004	Key database error
575005	Directory error
575006	Invalid implementation library
575008	No appropriate validator
575009	The root CA is not trusted
575010	No certificate chain was built
575011	Digital signature algorithm mismatch
575012	Digital signature mismatch
575013	X.509 version does not allow Key IDs
575014	X.509 version does not allow extensions
575015	Unknown X.509 certificate version
575016	The certificate validity range is invalid
575017	The certificate is not yet valid
575018	The certificate has expired
575019	The certificate contains unknown critical extensions
575020	The certificate contains duplicate extensions
575021	The issuers directory name does not match the issuer's issuer
575022	The Authority Key ID serial number value does not match the serial number of the issuer
575023	The Authority Key ID and Subject Key ID do not match
575024	Unrecognized issuer alternative name
575025	The certificate Basic Constraints forbid use as a CA
575026	The certificate has a non-zero Basic Constraints path length but is not a CA
575027	The certificate Basic Constraints maximum path length was exceeded
575028	The certificate is not permitted to sign other certificates
575029	The certificate is not signed by a CA
575030	Unrecognized Subject Alternative Name
575031	The certificate chain is invalid
575032	The certificate is revoked
575033	Unrecognized CRL distribution point

Table 2. Certificate validation errors.

A table listing return codes and explanations for certificate validation errors that can be returned in messages from the distributed queuing component.

(continued)

Return code (decimal)	**Explanation**
575034	Name chaining failed
575035	Certificate is not in a chain
575036	The CRL is not yet valid
575037	The CRL has expired
575038	The certificate version does not allow critical extensions
575039	Unknown CRL distribution points
575040	No CRLs for CRL distribution points
575041	Indirect CRLs are not supported
575042	Missing issuing CRL distribution point name
575043	Distribution points do not match
575044	No available CRL data source
575045	CA Subject name is null
575046	Distinguished names do not chain
575047	Missing Subject Alternative Name
575048	Unique ID mismatch
575049	Name not permitted
575050	Name excluded
575051	CA certificate is missing Critical Basic Constraints
575052	Name constraints are not critical
575053	Name constraints minimum subtree value if set is not zero
575054	Name constraints maximum subtree value if set is not allowed
575055	Unsupported name constraint
575056	Empty policy constraints
575057	Bad certificate policies
575058	Certificate policies not acceptable
575059	Bad acceptable certificate policies
575060	Certificate policy mappings are critical
575061	Revocation status could not be determined
575062	Extended key usage error
575063	Unknown OCSP version
575064	Unknown OCSP response

Table 2. Certificate validation errors.

A table listing return codes and explanations for certificate validation errors that can be returned in messages from the distributed queuing component.

(continued)

Return code (decimal)	Explanation
575065	Bad OCSP key usage extension
575066	Bad OCSP nonce
575067	Missing OCSP nonce
575068	No OCSP client available
575069	Policy not critical
575070	OCSP old but good
575071	OCSP old but revoked
575072	Incorrect curve
575073	Incorrect key size
575074	Incorrect Sigalg

A possible explanation for an error message being issued with return code 575074 is that is that the sole CipherSpec chosen by the client demands the use of a server certificate with an Elliptic curve signature, but the server certificate was using an RSA signature. The reverse could be true, that is, the server has an elliptic curve certificate but the sole CipherSpec proposed was RSA.

Related reference

"API completion and reason codes" on page 43
For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call.

"PCF reason codes" on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

"WCF custom channel exceptions" on page 320
Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Related information

Diagnostic messages: AMQ4000-9999

z/OS IBM MQ for z/OS messages, completion, and reason codes

WCF custom channel exceptions

Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Reading a message

For each message, this information is provided:

- The message identifier, in two parts:
    1. The characters "WCFCH" which identify the message as being from the WCF custom channel for IBM MQ

2. A four-digit decimal code followed by the character 'E'

- The text of the message.
- An explanation of the message giving further information.
- The response required from the user. In some cases, particularly for information messages, the response required might be "none".

Message variables

Some messages display text or numbers that vary according to the circumstances causing the message to occur; these circumstances are known as *message variables*. The message variables are indicated as {0}, {1}, and so on.

In some cases a message might have variables in the Explanation or Response. Find the values of the message variables by looking in the error log. The complete message, including the Explanation and the Response, is recorded there.

The following message types are described:

“WCFCH0001E-0100E: General/State messages” on page 321
“WCFCH0101E-0200E: URI Properties messages” on page 323
“WCFCH0201E-0300E: Factory/Listener messages” on page 324
“WCFCH0301E-0400E: Channel messages” on page 325
“WCFCH0401E-0500E: Binding messages” on page 326
“WCFCH0501E-0600E: Binding properties messages” on page 328
“WCFCH0601E-0700E: Async operations messages” on page 328

Related reference

“API completion and reason codes” on page 43
For each call, a completion code and a reason code are returned by the queue manager or by an exit routine, to indicate the success or failure of the call.

“PCF reason codes” on page 244
Reason codes might be returned by a broker in response to a command message in PCF format, depending on the parameters used in that message.

“Secure Sockets Layer (SSL) and Transport Layer Security (TLS) return codes” on page 315
IBM MQ can use Secure Sockets Layer (SSL) with the various communication protocols. Use this topic to identify the error codes that can be returned by SSL.

“WCF custom channel exceptions” on page 320
Diagnostic messages are listed in this topic in numeric order, grouped according to the part of the WCF custom channel from which they originate.

Related information

IBM MQ AMQ messages

z/OS IBM MQ for z/OS messages, completion, and reason codes

WCFCH0001E-0100E: General/State messages

Use the following information to understand WCFCH0001E-0100E general/state messages.

WCFCH0001E
An object cannot be opened because its state is '{0}'.

Explanation
An internal error has occurred.

Response
Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM MQ support web page, or the IBM SupportAssistant web

page, to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0002E
An object cannot be closed because its state is '{0}'.

Explanation
An internal error has occurred.

Response
Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM MQ support web page, or the IBM SupportAssistant web page, to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0003E
An object cannot be used because its state is '{0}'.

Explanation
An internal error has occurred.

Response
Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM MQ support web page, or the IBM SupportAssistant web page, to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0004E
The specified 'Timeout' value '{0}' is out of range.

Explanation
The value is out of range, it must be greater than or equal to 'TimeSpan.Zero'.

Response
Specify a value which is in range or, to disable Timeout, specify a 'TimeSpan.MaxValue' value.

WCFCH0005E
The operation did not complete within the specified time of '{0}' for endpoint address '{1}'.

Explanation
A timeout occurred.

Response
Investigate the cause for the timeout.

WCFCH0006E
The parameter '{0}' is not of the expected type '{1}'

Explanation
A parameter with an unexpected type has been passed to a method call.

Response
Review the exception stack trace for further information.

WCFCH0007E
The parameter '{0}' must not be null.

Explanation
A method has been called with a required parameter set to a null value.

Response
Modify the application to provide a value for this parameter.

WCFCH0008E
An error occurred while processing an operation for endpoint address '{0}'.

Explanation
The operation failed to complete.

Response
Review the linked exceptions and stack trace for further information.

WCFCH0101E-0200E: URI Properties messages

Use the following information to understand WCFCH0101E-0200E URI properties messages.

WCFCH0101E
The endpoint URI must start with the valid character string '{0}'.

Explanation
The endpoint URI is incorrect, it must start with a valid character string.

Response
Specify an endpoint URI which starts with a valid character string.

WCFCH0102E
The endpoint URI must contain a '{0}' parameter with a value.

Explanation
The endpoint URI is incorrect, a parameter and its value are missing.

Response
Specify an endpoint URI with a value for this parameter.

WCFCH0103E
The endpoint URI must contain a '{0}' parameter with a value of '{1}'.

Explanation
The endpoint URI is incorrect, the parameter must contain the correct value.

Response
Specify an endpoint URI with a correct parameter and value.

WCFCH0104E
The endpoint URI contains a '{0}' parameter with an invalid value of '{1}'.

Explanation
The endpoint URI is incorrect, a valid parameter value must be specified.

Response
Specify an endpoint URI with a correct value for this parameter.

WCFCH0105E
The endpoint URI contains a '{0}' parameter with an invalid queue or queue manager name.

Explanation
The endpoint URI is incorrect, a valid queue and queue manager name must be specified.

Response
Specify an endpoint URI with valid values for the queue and the queue manager.

WCFCH0106E
The '{0}' property is a required property and must appear as the first property in the endpoint URI.

Explanation
The endpoint URI is incorrect, a parameter is either missing or in the wrong position.

Response
Specify an endpoint URI which contains this property as the first parameter.

WCFCH0107E
The property '{1}' cannot be used when the binding property is set to '{0}'.

Explanation
The endpoint URI connectionFactory parameter is incorrect, an invalid combination of properties has been used.

Response
Specify an endpoint URI connectionFactory which contains a valid combination of properties or binding.

WCFCH0109E
Property '{1}' must also be specified when property '{0}' is specified.

Explanation
The endpoint URI connectionFactory parameter is incorrect, it contains an invalid combination of properties.

Response
Specify an endpoint URI connectionFactory which contains a valid combination of properties.

WCFCH0110E
Property '{0}' has an invalid value '{1}'.

Explanation
The endpoint URI connectionFactory parameter is incorrect, the property does not contain a valid value.

Response
Specify an endpoint URI connectionFactory which contains a valid value for the property.

WCFCH0111E
The value '{0}' is not supported for the binding mode property. XA operations are not supported.

Explanation
The endpoint URI connectionFactory parameter is incorrect, the binding mode is not supported.

Response
Specify an endpoint URI connectionFactory which contains a valid value for the binding mode.

WCFCH0112E
The endpoint URI '{0}' is badly formatted.

Explanation
The endpoint URI must follow the format described in the documentation.

Response
Review the endpoint URI to ensure that it contains a valid value.

WCFCH0201E-0300E: Factory/Listener messages

Use the following information to understand WCFCH0201E-0300E factory/listener messages.

WCFCH0201E
Channel shape '{0}' is not supported.

Explanation
The users application or the WCF service contract has requested a channel shape which is not supported.

Response
Identify and use a channel shape which is supported by the channel.

WCFCH0202E
'{0}' MessageEncodingBindingElements have been specified.

Explanation
The WCF binding configuration used by an application contains more than one message encoder.

Response
Specify no more then 1 MessageEncodingBindingElement in the binding configuration.

WCFCH0203E
The endpoint URI address for the service listener must be used exactly as provided.

Explanation
The binding information for the endpoint URI address must specify a value of 'Explicit' for the 'listenUriMode' parameter.

Response
Change the parameter value to 'Explicit'.

WCFCH0204E
SSL is not supported for managed client connections [endpoint URI: '{0}'].

Explanation
The endpoint URI specifies an SSL connection type which is only supported for unmanaged client connections.

Response
Modify the channels binding properties to specify an unmanaged client connection mode.

WCFCH0301E-0400E: Channel messages

Use the following information to understand WCFCH0301E-0400E channel messages.

WCFCH0301E
The URI scheme '{0}' is not supported.

Explanation
The requested endpoint contains a URI scheme which is not supported by the channel.

Response
Specify a valid scheme for the channel.

WCFCH0302E
The received message '{0}' was not a JMS bytes or a JMS text message.

Explanation
A message has been received but it is not of the correct type. It must be either a JMS bytes message or a JMS text message.

Response
Check the origin and contents of the message and determine the cause for it being incorrect.

WCFCH0303E
'ReplyTo' destination missing.

Explanation
A reply cannot be sent because the original request does not contain a 'ReplyTo' destination.

Response
Investigate the reason for the missing destination value.

WCFCH0304E
The connection attempt to queue manager '{0}' failed for endpoint '{1}'

Explanation
The queue manager could not be contacted at the given address.

Response
Review the linked exception for further details.

WCFCH0305E
The connection attempt to the default queue manager failed for endpoint '{0}'

Explanation
The queue manager could not be contacted at the given address.

Response
Review the linked exception for further details.

WCFCH0306E
An error occurred while attempting to receive data from endpoint '{0}'

Explanation
The operation could not be completed.

Response
Review the linked exception for further details.

WCFCH0307E
An error occurred while attempting to send data for endpoint '{0}'

Explanation
The operation could not be completed.

Response
Review the linked exception for further details.

WCFCH0308E
An error occurred while attempting to close the channel for endpoint '{0}'

Explanation
The operation could not be completed.

Response
Review the linked exception for further details.

WCFCH0309E
An error occurred while attempting to open the channel for endpoint '{0}'

Explanation
The operation could not be completed.

Response
The endpoint might be down, unavailable, or unreachable, review the linked exception for further details.

WCFCH0310E
The timeout '{0}' was exceeded while attempting to receive data from endpoint '{0}'

Explanation
The operation did not complete in the time allowed.

Response
Review the system status and configuration and increase the timeout if required.

WCFCH0311E
The timeout '{0}' was exceeded while attempting to send data for endpoint '{0}'

Explanation
The operation did not complete in the time allowed.

Response
Review the system status and configuration and increase the timeout if required.

WCFCH0312E
The timeout '{0}' was exceeded while attempting to close the channel for endpoint '{0}'

Explanation
The operation did not complete in the time allowed.

Response
Review the system status and configuration and increase the timeout if required.

WCFCH0313E
The timeout '{0}' was exceeded while attempting to open the channel for endpoint '{0}'

Explanation
The operation did not complete in the time allowed.

Response
The endpoint might be down, unavailable, or unreachable, review the system status and configuration and increase the timeout if required.

WCFCH0401E-0500E: Binding messages

Use the following information to understand WCFCH0401E-0500E binding messages.

WCFCH0401E
No context.

Explanation
An internal error has occurred.

Response

Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM Support Portal for IBM MQ (see https://www.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_MQ), or the IBM Support Assistant (at https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant), to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0402E

Channel type '{0}' is not supported.

Explanation

The users application or the WCF service contract has requested a channel shape which is not supported.

Response

Identify and use a channel shape which is supported by the channel.

WCFCH0403E

No exporter.

Explanation

An internal error has occurred.

Response

Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM Support Portal for IBM MQ (see https://www.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_MQ), or the IBM Support Assistant (at https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant), to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0404E

The WS-Addressing version '{0}' is not supported.

Explanation

The addressing version specified is not supported.

Response

Specify an addressing version which is supported.

WCFCH0405E

No importer.

Explanation

An internal error has occurred.

Response

Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM Support Portal for IBM MQ (see https://www.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_MQ), or the IBM Support Assistant (at https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant), to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0406E

Endpoint 'Binding' value missing.

Explanation

An internal error has occurred.

Response

Use the standard facilities supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM Support Portal for IBM MQ (see https://www.ibm.com/support/entry/portal/Overview/Software/WebSphere/WebSphere_MQ), or the IBM Support Assistant (at https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant), to see if a solution is already available. If you cannot find a match, contact your IBM support center. Do not discard these files until the problem has been resolved.

WCFCH0501E-0600E: Binding properties messages

Use the following information to understand WCFCH0501E-0600E binding properties messages.

WCFCH0501E
The binding property '{0}' has an invalid value '{1}'.

Explanation
An invalid value has been specified for a binding property.

Response
Specify a valid value for the property.

WCFCH0601E-0700E: Async operations messages

Use the following information to understand WCFCH0601E-0700E async operations messages.

WCFCH0601E
The async result parameter '{0}' object is not valid for this call.

Explanation
An invalid async result object has been provided.

Response
Specify a valid value for the parameter.

Contacting IBM Software Support

You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

About this task

The IBM MQ Support pages within the IBM Support Site are:

- IBM i distributed IBM MQ for Multiplatforms Support web page
- z/OS IBM MQ for z/OS Support web page

To receive notifications about IBM MQ fixes, troubleshooting and other news, you can subscribe to notifications.

If you cannot resolve an issue yourself and need help from IBM Support, you can open a case. Follow the steps in this topic to fully describe the problem and contact IBM Software Support.

For more information about IBM Support, including how to register for support, see the IBM Support Guide.

Procedure

1. Determine the business severity level for the problem.

   When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the effect on your business of the problem that you are reporting. Use the following criteria:

Severity	Effect on business
Severity 1	**Critical** effect on business: You are unable to use the program, resulting in a critical effect on operations. This condition requires an immediate solution.
Severity 2	**Significant** effect on business: The program is usable but is severely limited.
Severity 3	**Some** effect on business: The program is usable with less significant features (not critical to operations) unavailable.
Severity 4	**Minimal** effect on business: The problem has little effect on operations, or a reasonable workaround to the problem has been implemented.

When deciding the severity of the problem, take care not to understate it, or to overstate it. The support center procedures depend on the severity level so that the most appropriate use can be made of the center's skills and resources. A severity level 1 problem is normally dealt with immediately.

2. Describe the problem and gather background information.

   You might find the information you need in your own in-house tracking system for problems.

   Be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you to solve the problem efficiently. To save time, know the answers to these questions:

   - What was the source of the problem within your system software; that is, the program that seems to be the cause of the problem.
   - What software versions were you running when the problem occurred?
   - Do you have logs, traces, and messages that are related to the problem symptoms?
   - Can the problem be re-created? If so, what steps led to the failure?
   - Have any changes been made to the system? For example:
     - Hardware changes
     - Operating system upgrades
     - Networking software updates
     - Changes in the level of licensed programs
     - PTFs applied
     - Additional features used
     - Application programs changed
     - Unusual operator action
     - z/OS Regenerations
   - Are you currently using a workaround for this problem? If so, be prepared to explain it when you report the problem.
3. Open a case with IBM Software Support (https://www.ibm.com/mysupport/s/createrecord/NewCase).

What to do next

You might be asked to give values from a formatted dump or trace table, or to carry out some special activity, for example to set a trap, or to use trace with a specific type of selectivity, and then to report the results. You will be given guidance by the support center on how to obtain this information.

You can inquire any time at your support center on how your PMR is progressing, particularly if it is a problem of high severity.

How your problem is then progressed depends on its nature. The representative who handles the problem gives you guidance.

First Failure Support Technology (FFST

First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM support personnel to diagnose the problem.

First Failure Data Capture (FFDC) provides an automated snapshot of the system environment when an internal event occurs. In the case of an error, this snapshot is used by IBM support personnel to provide a better understanding of the state of the system and IBM MQ when the problem occurred.

The information about an event is contained in an FFST file. In IBM MQ, FFST files have a file type of FDC. FFST files do not always indicate an error. An FFST might be informational.

Monitoring and housekeeping

Here are some tips to help you with managing FFST events:

- Monitor FFST events for your system, and ensure that appropriate and timely remedial action is taken when an event occurs. In some cases, the FDC files might be expected and can therefore be ignored, for example FFST events that arise when IBM MQ processes are ended by the user. By appropriate monitoring, you can determine which events are expected, and which events are not.
- FFST events are also produced for events outside IBM MQ. For example, if there is a problem with the IO subsystem or network, this problem can be reported in an FDC type file. These types of event are outside the control of IBM MQ and you might need to engage third parties to investigate the root cause.
- Ensure that good housekeeping of FFST files is carried out. The files must be archived and the directory or folder must be cleared to ensure that only the most recent and relevant FDC files are available, should the support team need them.

Use the information in the following links to find out the names, locations, and contents of FFST files in different platforms.

- "FFST: IBM MQ classes for JMS" on page 331
- "FFST: IBM MQ for Windows" on page 334
- "FFST: IBM MQ for UNIX and Linux systems" on page 336
- IBM i "FFST: IBM MQ for IBM i" on page 338
- "FFST: IBM WebSphere MQ for HP Integrity NonStop Server" on page 340

Related concepts

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Troubleshooting overview" on page 7
Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

"Using logs" on page 341
There are a variety of logs that you can use to help with problem determination and troubleshooting.

"Problem determination on z/OS" on page 389
IBM MQ for z/OS, CICS, Db2, and IMS produce diagnostic information which can be used for problem determination.

Related tasks

"Using trace" on page 350
You can use different types of trace to help you with problem determination and troubleshooting.

"Contacting IBM Software Support" on page 328

You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

FFST: IBM MQ classes for JMS

Describes the name, location, and contents of the First Failure Support Technology (FFST) files that are generated by the IBM MQ classes for JMS.

When using the IBM MQ classes for JMS, FFST information is recorded in a file in a directory that is called `FFDC`, which by default is a subdirectory of the current working directory for the IBM MQ classes for JMS application that was running when the FFST was generated. If the property com.ibm.msg.client.commonservices.trace.outputName has been set in the IBM MQ classes for JMS configuration file, the FFDC directory is a subdirectory of the directory that the property points to. For information about the IBM MQ classes for JMS , see The IBM MQ classes for JMS configuration file.

An FFST file contains one FFST record. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records typically indicate either a configuration problem with the system or an internal error within the IBM MQ classes for JMS .

FFST files are named `JMSC` *nnnn*`.FDC`, where *nnnn* starts at 1. If the full file name already exists, this value is incremented by one until a unique FFST file name is found.

An instance of an IBM MQ classes for JMS application writes FFST information to multiple FFST files. If multiple errors occur during a single execution of the application, each FFST record is written to a different FFST file.

Sections of an FFST record

An FFST record that is generated by the IBM MQ classes for JMS contains the following sections:

The header
A header, indicating the time when the FFST record was created, the platform that the IBM MQ classes for JMS application is running on, and the internal method that was being called. The header also contains a probe identifier, which uniquely identifies the place within the IBM MQ classes for JMS that generated the FFST record.

Data
Some internal data that is associated with the FFST record.

Version information
Information about the version of the IBM MQ classes for JMS being used by the application that generated the FFST record.

Stack Trace
The Java stack trace for the thread that generated the FFST record.

Property Store Contents
A list of all of the Java system properties that have been set on the Java Runtime Environment that the IBM MQ classes for JMS application is running in.

WorkQueueMananger Contents
Information about the internal thread pool that is used by the IBM MQ classes for JMS .

Runtime properties
Details about the amount of memory and the number of processors available on the system where the IBM MQ classes for JMS application is running.

Component Manager Contents
Some information about the internal components that are loaded by the IBM MQ classes for JMS .

Provider Specific information
Information about all of the active JMS Connections, JMS Sessions, MessageProducer, and MessageConsumer objects currently being used by the IBM MQ classes for JMS application that was running when the FFST was generated. This information includes the name of the queue manager that JMS Connections and JMS Sessions are connected to, and the name of the IBM MQ queue or topic objects that are being used by MessageProducers and MessageConsumers.

All Thread information

Details about the state of all of the active threads in the Java Runtime Environment that the IBM MQ classes for JMS application was running in when the FFST record was generated. The name of each thread is shown, together with a Java stack trace for every thread.

Example FFST log file

A typical FFST log is shown in Figure 1 on page 333

```
----------------------------------START FFST------------------------------------
c:\JBoss-6.0.0\bin\FFDC\JMSCC0007.FDC PID:4472

JMS Common Client First Failure Symptom Report


Product     :- IBM MQ classes for JMS
Date/Time    :- Mon Feb 03 14:14:46 GMT 2014
System time   :- 1391436886081
Operating System :- Windows Server 2008
UserID      :- pault
Java Vendor   :- IBM Corporation
Java Version   :- 2.6

Source Class   :- com.ibm.msg.client.commonservices.j2se.wmqsupport.PropertyStoreImpl
Source Method  :- getBooleanProperty(String)
ProbeID     :- XS002005
Thread      :- name=pool-1-thread-3 priority=5 group=workmanager-threads
ccl=BaseClassLoader@ef1c3794{vfs:///C:/JBoss-6.0.0/server/default/deploy/basicMDB.ear}

Data
----

|  name :- com.ibm.mq.connector.performJavaEEContainerChecks

Version information
-------------------

Java Message Service Client
7.5.0.2
p750-002-130627
Production

IBM MQ classes for Java Message Service
7.5.0.2
p750-002-130627
Production

IBM MQ JMS Provider
7.5.0.2
p750-002-130627
Production

Common Services for Java Platform, Standard Edition
7.5.0.2
p750-002-130627
Production



Stack trace
-----------

Stack trace to show the location of the FFST call
|  FFST Location :- java.lang.Exception
|      at com.ibm.msg.client.commonservices.trace.Trace.getCurrentPosition(Trace.java:1972)
|      at com.ibm.msg.client.commonservices.trace.Trace.createFFSTString(Trace.java:1911)
|      at com.ibm.msg.client.commonservices.trace.Trace.ffstInternal(Trace.java:1800)
|      at com.ibm.msg.client.commonservices.trace.Trace.ffst(Trace.java:1624)
|      at com.ibm.msg.client.commonservices.j2se.propertystore.PropertyStoreImpl.getBooleanProperty(
PropertyStoreImpl.java:322)
|      at com.ibm.msg.client.commonservices.propertystore.PropertyStore.getBooleanPropertyObject(Pr
opertyStore.java:302)
|      at com.ibm.mq.connector.outbound.ConnectionWrapper.jcaMethodAllowed(ConnectionWrapper.java:510)
|      at com.ibm.mq.connector.outbound.ConnectionWrapper.setExceptionListener(ConnectionWrapper.java:244)
|      at com.ibm.basicMDB.MDB.onMessage(MDB.java:45)
...

Property Store Contents
-----------------------

All currently set properties
|  awt.toolkit                           :- sun.awt.windows.WToolkit
|  catalina.ext.dirs                        :- C:\JBoss-6.0.0\server\default\lib
|  catalina.home                          :- C:\JBoss-6.0.0\server\default
|  com.ibm.cpu.endian                        :- little
|  com.ibm.jcl.checkClassPath                    :-
|  com.ibm.mq.connector.performJavaEEContainerChecks        :- false
|  com.ibm.oti.configuration                   :- scar
|  com.ibm.oti.jcl.build                      :- 20131013_170512
|  com.ibm.oti.shared.enabled                    :- false
|  com.ibm.oti.vm.bootstrap.library.path              :- C:\Program
Files\IBM\Java70\jre\bin\compressedrefs;C:\Program Files\IBM\Java70\jre\bin
|  com.ibm.oti.vm.library.version                  :- 26
|  com.ibm.system.agent.path                   :- C:\Program
Files\IBM\Java70\jre\bin
|  com.ibm.util.extralibs.properties                 :-
|  com.ibm.vm.bitmode                        :- 64
|  com.ibm.zero.version                       :- 2
|  console.encoding                        :- Cp850
|  file.encoding                         :- Cp1252
|  file.encoding.pkg                        :- sun.io
...


WorkQueueMananger Contents
--------------------------

|  Current ThreadPool size   :- 2
|  Maintain ThreadPool size   :- false
|  Maximum ThreadPool size   :- -1
|  ThreadPool inactive timeout :- 0

Runtime properties
------------------

|  Available processors     :- 4
|  Free memory in bytes (now)  :- 54674936
|  Max memory in bytes     :- 536870912
|  Total memory in bytes (now) :- 235012096

Component Manager Contents
--------------------------

Common Services Components:
|  CMVC        :- p750-002-130627
|  Class Name     :- class com.ibm.msg.client.commonservices.j2se.J2SEComponent
|  Component Name   :- com.ibm.msg.client.commonservices.j2se
|  Component Title  :- Common Services for Java Platform, Standard Edition
|  Factory Class   :- class com.ibm.msg.client.commonservices.j2se.CommonServicesImplementation
|  Version      :- 7.5.0.2
|  inPreferenceTo[0] :- com.ibm.msg.client.commonservices.j2me

Messaging Provider Components:
|  CMVC       :- p750-002-130627
|  Class Name    :- class com.ibm.msg.client.wmq.factories.WMQComponent
|  Component Name  :- com.ibm.msg.client.wmq
|  Component Title :- IBM MQ JMS Provider
|  Factory Class  :- class com.ibm.msg.client.wmq.factories.WMQFactoryFactory
|  Version     :- 7.5.0.2



Provider Specific Information
-----------------------------
```

The information in the header, Data, and Stack Trace sections of the FFST record are used by IBM to assist in problem determination. In many cases, there is little that the system administrator can do when an FFST record is generated, apart from raising problems through the IBM Support Center.

Suppressing FFST records

An FFST file that is generated by the IBM MQ classes for JMS contain one FFST record. If a problem occurs multiple times during the execution of an IBM MQ classes for JMS application, multiple FFST files with the same probe identifier are generated. This might not be desirable. The property com.ibm.msg.client.commonservices.ffst.suppress can be used to suppress the production of FFST files. This property must be set in the IBM MQ classes for JMS configuration file used by the application, and can take the following values:

0: Output all FFDC files (default).

-1: Output only the first FFST file for a probe identifier.

integer: Suppress all FFST files for a probe identifier except those files that are a multiple of this number.

FFST: IBM MQ for Windows

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for Windows systems.

In IBM MQ for Windows, FFST information is recorded in a file in the `C:\Program Files\IBM\WebSphere MQ\errors` directory.

An FFST file contains one or more records. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records typically indicate either a configuration problem with the system or an IBM MQ internal error.

FFST files are named `AMQ` *`nnnnn.mm`*`.FDC`, where:

nnnnn
: Is the ID of the process reporting the error

mm
: Starts at 0. If the full file name already exists, this value is incremented by one until a unique FFST file name is found. An FFST file name can already exist if a process is reused.

An instance of a process will write all FFST information to the same FFST file. If multiple errors occur during a single execution of the process, an FFST file can contain many records.

When a process writes an FFST record it also sends a record to the Event Log. The record contains the name of the FFST file to assist in automatic problem tracking. The Event log entry is made at the application level.

A typical FFST log is shown in Figure 2 on page 335.

```
+-----------------------------------------------------------------------------+
| WebSphere MQ First Failure Symptom Report                                   |
| =========================================                                   |
|                                                                             |
| Date/Time         :- Mon January 28 2008 21:59:06 GMT                       |
| UTC Time/Zone     :- 1201539869.892015 0 GMT                                |
| Host Name         :- 99VXY09 (Windows XP Build 2600: Service Pack 1)        |
| PIDS              :- 5724H7200                                              |
| LVLS              :- 7.0.0.0                                                |
| Product Long Name :- WebSphere MQ for Windows                               |
| Vendor            :- IBM                                                    |
| Probe Id          :- HL010004                                               |
| Application Name  :- MQM                                                    |
| Component         :- hlgReserveLogSpace                                     |
| SCCS Info         :- lib/logger/amqhlge0.c, 1.26                            |
| Line Number       :- 246                                                    |
| Build Date        :- Jan 25 2008                                            |
| CMVC level        :- p000-L050202                                           |
| Build Type        :- IKAP - (Production)                                    |
| UserID            :- IBM_User                                               |
| Process Name      :- C:\Program Files\IBM\WebSphere MQ\bin\amqzlaa0.exe     |
| Process           :- 00003456                                               |
| Thread            :- 00000030                                               |
| QueueManager      :- qmgr2                                                  |
| ConnId(1) IPCC    :- 162                                                    |
| ConnId(2) QM      :- 45                                                     |
| Major Errorcode   :- hrcE_LOG_FULL                                          |
| Minor Errorcode   :- OK                                                     |
| Probe Type        :- MSGAMQ6709                                             |
| Probe Severity    :- 2                                                      |
| Probe Description :- AMQ6709: The log for the Queue manager is full.        |
| FDCSequenceNumber :- 0                                                      |
+-----------------------------------------------------------------------------+

MQM Function Stack
zlaMainThread
zlaProcessMessage
zlaProcessMQIRequest
zlaMQPUT
zsqMQPUT
kpiMQPUT
kqiPutIt
kqiPutMsgSegments
apiPutMessage
aqmPutMessage
aqhPutMessage
aqqWriteMsg
aqqWriteMsgData
aqlReservePutSpace
almReserveSpace
hlgReserveLogSpace
xcsFFST

MQM Trace History
-------------} hlgReserveLogSpace rc=hrcW_LOG_GETTING_VERY_FULL
-------------{ xllLongLockRequest
-------------} xllLongLockRequest rc=OK

...
```

Figure 2. Sample IBM MQ for Windows First Failure Symptom Report

The Function Stack and Trace History are used by IBM to assist in problem determination. In many cases there is little that the system administrator can do when an FFST record is generated, apart from raising problems through the IBM Support Center.

In certain circumstances a small dump file can be generated in addition to an FFST file and placed in the `C:\Program Files\IBM\WebSphere MQ\errors` directory. A dump file will have the same name as the FFST file, in the form `AMQnnnnn.mm.dmp`. These files can be used by IBM to assist in problem determination.

First Failure Support Technology (FFST) files and Windows clients

The files are produced already formatted and are in the errors subdirectory of the IBM MQ MQI client installation directory.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or an IBM MQ internal error.

The files are named `AMQnnnnn.mm.FDC`, where:

- `nnnnn` is the process ID reporting the error
- `mm` is a sequence number, normally 0

When a process creates an FFST it also sends a record to the system log. The record contains the name of the FFST file to assist in automatic problem tracking.

The system log entry is made at the "user.error" level.

First Failure Support Technology is explained in detail in First Failure Support Technology (FFST).

FFST: IBM MQ for UNIX and Linux systems

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for UNIX and Linux systems.

For IBM MQ on UNIX and Linux systems, FFST information is recorded in a file in the `/var/mqm/errors` directory.

An FFST file contains one or more records. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records indicate either a configuration problem with the system or an IBM MQ internal error.

FFST files are named `AMQ` *`nnnnn.mm`*`.FDC`, where:

nnnnn
Is the ID of the process reporting the error

mm
Starts at 0. If the full file name already exists, this value is incremented by one until a unique FFST file name is found. An FFST file name can already exist if a process is reused.

An instance of a process will write all FFST information to the same FFST file. If multiple errors occur during a single execution of the process, an FFST file can contain many records.

In order to read the contents of a FFST file, you must be either the creator of the file, or a member of the mqm group.

When a process writes an FFST record, it also sends a record to syslog. The record contains the name of the FFST file to assist in automatic problem tracking. The syslog entry is made at the *user.error* level. See the operating-system documentation about `syslog.conf` for information about configuring this.

Some typical FFST data is shown in Figure 3 on page 337.

```
+-----------------------------------------------------------------------------+
|                                                                             |
| WebSphere MQ First Failure Symptom Report                                   |
| =========================================                                   |
|                                                                             |
| Date/Time         :- Mon January 28 2008 21:59:06 GMT                       |
| UTC Time/Zone     :- 1201539869.892015 0 GMT                                |
| Host Name         :- mqperfh2 (HP-UX B.11.23)                               |
| PIDS              :- 5724H7202                                              |
| LVLS              :- 7.0.0.0                                                |
| Product Long Name :- WebSphere MQ for HP-UX                                 |
| Vendor            :- IBM                                                    |
| Probe Id          :- XC034255                                               |
| Application Name  :- MQM                                                    |
| Component         :- xcsWaitEventSem                                        |
| SCCS Info         :- lib/cs/unix/amqxerrx.c, 1.204                          |
| Line Number       :- 6262                                                   |
| Build Date        :- Jan 25 2008                                            |
| CMVC level        :- p000-L050203                                           |
| Build Type        :- IKAP - (Production)                                    |
| UserID            :- 00000106 (mqperf)                                      |
| Program Name      :- amqzmuc0                                               |
| Addressing mode   :- 64-bit                                                 |
| Process           :- 15497                                                  |
| Thread            :- 1                                                      |
| QueueManager      :- CSIM                                                   |
| ConnId(2) QM      :- 4                                                      |
| Major Errorcode   :- OK                                                     |
| Minor Errorcode   :- OK                                                     |
| Probe Type        :- INCORROUT                                              |
| Probe Severity    :- 4                                                      |
| Probe Description :- AMQ6109: An internal WebSphere MQ error has occurred.  |
| FDCSequenceNumber :- 0                                                      |
|                                                                             |
+-----------------------------------------------------------------------------+

MQM Function Stack
amqzmuc0
xcsWaitEventSem
xcsFFST

MQM Trace History
Data: 0x00003c87
--} xcsCheckProcess rc=OK
--{ xcsRequestMutexSem
--} xcsRequestMutexSem rc=OK

...
```

Figure 3. FFST report for IBM MQ for UNIX systems

The Function Stack and Trace History are used by IBM to assist in problem determination. In many cases there is little that the system administrator can do when an FFST report is generated, apart from raising problems through the IBM Support Center.

However, there are some problems that the system administrator might be able to solve. If the FFST shows *out of resource* or *out of space on device* descriptions when calling one of the IPC functions (for example, `semop` or `shmget`), it is likely that the relevant kernel parameter limit has been exceeded.

If the FFST report shows a problem with `setitimer` , it is likely that a change to the kernel timer parameters is needed.

To resolve these problems, increase the IPC limits, rebuild the kernel, and restart the machine.

First Failure Support Technology (FFST) files and UNIX and Linux clients

FFST logs are written when a severe IBM MQ error occurs. They are written to the directory `/var/mqm/errors`.

These are normally severe, unrecoverable errors and indicate either a configuration problem with the system or an IBM MQ internal error.

The files are named `AMQnnnnn.mm.FDC`, where:

- nnnnn is the process id reporting the error
- `mm` is a sequence number, normally 0

When a process creates an FFST it also sends a record to the system log. The record contains the name of the FFST file to assist in automatic problem tracking.

The system log entry is made at the "user.error" level.

First Failure Support Technology is explained in detail in First Failure Support Technology (FFST).

FFST: IBM MQ for IBM i

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for IBM i systems.

For IBM i, FFST information is recorded in a stream file in the `/QIBM/UserData/mqm/errors` directory.

These errors are normally severe, unrecoverable errors, and indicate either a configuration problem with the system or an IBM MQ internal error.

The stream files are named `AMQ` *`nnnnn.mm`*`.FDC`, where:

nnnnn	Is the ID of the process reporting the error
mm	Is a sequence number, normally 0

A copy of the job log of the failing job is written to a file with the same name as the .FDC file. The file name ends with .JOB.

Some typical FFST data is shown in the following example.

```
-------------------------------------------------------------------------------
| WebSphere MQ First Failure Symptom Report                                   |
| =========================================                                   |
|                                                                             |
| Date/Time         :- Mon January 28 2008 21:59:06 GMT                       |
| UTC Time/Zone     :- 1201539869.892015 0 GMT                                |
| Host Name         :- WINAS12B.HURSLEY.IBM.COM                               |
| PIDS              :- 5733A38                                                |
| LVLS              :- 520                                                    |
| Product Long Name :- WebSphere MQ for IBMi                                  |
| Vendor            :- IBM                                                    |
| Probe Id          :- XY353001                                               |
| Application Name  :- MQM                                                    |
| Component         :- xehAS400ConditionHandler                               |
| Build Date        :- Feb 25 2008                                            |
| UserID            :- 00000331 (MAYFCT)                                      |
| Program Name      :- STRMQM_R  MAYFCT                                       |
| Job Name          :- 020100/MAYFCT/STRMQM_R                                 |
| Activation Group  :- 101 (QMQM) (QMQM/STRMQM_R)                             |
| Process           :- 00001689                                               |
| Thread            :- 00000001                                               |
| QueueManager      :- TEST.AS400.OE.P                                        |
| Major Errorcode   :- STOP                                                   |
| Minor Errorcode   :- OK                                                     |
| Probe Type        :- HALT6109                                               |
| Probe Severity    :- 1                                                      |
| Probe Description :- 0                                                      |
| Arith1            :- 1 1                                                    |
| Comment1          :- 00d0                                                   |
------------------------------------------------------------------------------

MQM Function Stack
lpiSPIMQConnect
zstMQConnect
ziiMQCONN
ziiClearUpAgent
xcsTerminate
xlsThreadInitialization
```

```
xcsConnectSharedMem
xstConnSetInSPbyHandle
xstConnSharedMemSet
xcsFFST

MQM Trace History
<-- xcsCheckProcess rc=xecP_E_INVALID_PID
-->
xcsCheckProcess
<-- xcsCheckProcess rc=xecP_E_INVALID_PID
-->
xlsThreadInitialization
-->
xcsConnectSharedMem
-->
xcsRequestThreadMutexSem
<-- xcsRequestThreadMutexSem rc=OK
-->
xihGetConnSPDetailsFromList
<-- xihGetConnSPDetailsFromList rc=OK
-->
xstCreateConnExtentList
<-- xstCreateConnExtentList rc=OK
-->
xstConnSetInSPbyHandle
-->
xstSerialiseSPList
-->
xllSpinLockRequest
<-- xllSpinLockRequest rc=OK
<-- xstSerialiseSPList rc=OK
-->
xstGetSetDetailsFromSPByHandle
<-- xstGetSetDetailsFromSPByHandle rc=OK
-->
xstConnSharedMemSet
-->
xstConnectExtent
-->
xstAddConnExtentToList
<-- xstAddConnExtentToList rc=OK
<-- xstConnectExtent rc=OK
-->
xcsBuildDumpPtr
-->
xcsGetMem
<-- xcsGetMem rc=OK
<-- xcsBuildDumpPtr rc=OK
-->
xcsBuildDumpPtr
<-- xcsBuildDumpPtr rc=OK
-->
xcsBuildDumpPtr
<-- xcsBuildDumpPtr rc=OK
-->
xcsFFST

Process Control Block
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :8bba0:0:6d  E7C9C8D7 000004E0 00000699 00000000  XIHP...\...r....
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :8bbb0:1:6d  00000000 00000002 00000000 00000000  ................
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :8bbc0:2:6d  80000000 00000000 EC161F7C FC002DB0  ...........@...¢
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :8bbd0:3:6d  80000000 00000000 EC161F7C FC002DB0  ...........@...¢
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :8bbe0:4:6d  00000000 00000000 00000000 00000000  ................

Thread Control Block
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :1db0:20:6d  E7C9C8E3 00001320 00000000 00000000  XIHT............
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :1dc0:21:6d  00000001 00000000 00000000 00000000  ................
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :1dd0:22:6d  80000000 00000000 DD13C17B 81001000  ..........A#a...
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :1de0:23:6d  00000000 00000046 00000002 00000001  ................
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :1df0:24:6d  00000000 00000000 00000000 00000000  ................


RecoveryIndex
SPP:0000 :1aefSTRMQM_R MAYFCT  020100 :2064:128:6d  00000000                 ....
```

Note:

1. The `MQM Trace History` section is a log of the 200 most recent function trace statements, and is recorded in the FFST report regardless of any TRCMQM settings.

2. The queue manager details are recorded only for jobs that are connected to a queue manager subpool.
3. When the failing component is `xehAS400ConditionHandler`, additional data is logged in the errors directory giving extracts from the job log relating to the exception condition.

The function stack and trace history are used by IBM to assist in problem determination. In most cases, there is little that the system administrator can do when an FFST report is generated, apart from raising problems through the IBM Support Center.

FFST: IBM WebSphere MQ for HP Integrity NonStop Server

Describes the name, location, and contents of the First Failure Support Technology™ (FFST™) files for HP Integrity NonStop Server systems.

In IBM MQ client for HP Integrity NonStop Server systems, FFST information is recorded in a file in the `<mqpath>/var/mqm/errors` directory.

An FFST file contains one or more records. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records indicate either a configuration problem with the system or an IBM MQ internal error.

FFST files are named `AMQ.nnn.xx.ppp.qq.FDC`, where:

nnn
The name of the process that is reporting the error.

xx
The processor number on which the process is running.

ppp
The PIN of the process that you are tracing.

qq
A sequence that starts at 0. If the full file name exists, this value is incremented by one until a unique FFST file name is found. An FFST file name can exist if a process is reused.

Each field can contain fewer or more digits than shown in the example.

An instance of a process writes all FFST information to the same FFST file. If multiple errors occur during a single execution of the process, an FFST file can contain many records.

To read the contents of an FFST file, you must be either the creator of the file, or a member of the mqm group.

When a process writes an FFST record, it also creates an EMS event.

Figure 4 on page 341 shows a typical FFST report for an IBM MQ client on a HP Integrity NonStop Server system:

```
+-----------------------------------------------------------------------------+
|                                                                             |
| WebSphere MQ First Failure Symptom Report                                   |
| =========================================                                   |
|                                                                             |
| Date/Time         :- Mon April 29 2013 10:21:26 EDT                         |
| UTC Time          :- 1367245286.105303                                      |
| UTC Time Offset   :- -240 (EST)                                            |
| Host Name         :- MYHOST                                                 |
| Operating System  :- HP NonStop J06.14, NSE-AB 069194                       |
|                                                                             |
| PIDS              :- 5724H7222                                              |
| LVLS              :- 7.1.0.0                                                |
| Product Long Name :- WebSphere MQ for HP NonStop Server                     |
| Vendor            :- IBM                                                    |
| Installation Path :- /home/cmarti/client/opt/mqm                            |
| Probe Id          :- MQ000020                                               |
| Application Name  :- MQM                                                    |
| Component         :- Unknown                                                |
| SCCS Info         :- S:/cmd/trace/amqxdspa.c,                               |
| Line Number       :- 3374                                                   |
| Build Date        :- Apr 24 2013                                            |
| Build Level       :- D20130424-1027                                         |
| Build Type        :- ICOL - (Development)                                   |
| File Descriptor   :- 6                                                      |
| Effective UserID  :- 11329 (MQM.CMARTI)                                     |
| Real UserID       :- 11329 (MQM.CMARTI)                                     |
| Program Name      :- dspmqtrc                                               |
| Addressing mode   :- 32-bit                                                 |
| LANG              :-                                                        |
| Process           :- 1,656 $Y376 OSS 469762429                              |
| Thread(n)         :- 1                                                      |
| UserApp           :- FALSE                                                  |
| Last HQC          :- 0.0.0-0                                                |
| Last HSHMEMB      :- 0.0.0-0                                                |
| Major Errorcode   :- krcE_UNEXPECTED_ERROR                                  |
| Minor Errorcode   :- OK                                                     |
| Probe Type        :- INCORROUT                                              |
| Probe Severity    :- 2                                                      |
| Probe Description :- AMQ6125: An internal WebSphere MQ error has occurred.  |
| FDCSequenceNumber :- 0                                                      |
| Comment1          :- AMQ.3.520.sq_tc.0.TRC                                  |
| Comment2          :- Unrecognised hookID:0x3 at file offset 0x4b84          |
|                                                                             |
+-----------------------------------------------------------------------------+

MQM Function Stack
xcsFFST

MQM Trace History
{ xppInitialiseDestructorRegistrations
} xppInitialiseDestructorRegistrations rc=OK
{ xcsGetEnvironmentInteger
-{ xcsGetEnvironmentString

...
```

Figure 4. Sample FFST data

The Function Stack and Trace History are used by IBM to help problem determination. In many cases, there is little that the system administrator can do when an FFST report is generated, apart from raising problems through the IBM Support Center. However, there are some problems that the system administrator might be able to solve, for example, if the FFST report shows `Out of resource` or `Out of space on device`.

For more information about FFST, see " First Failure Support Technology (FFST" on page 330.

Using logs

There are a variety of logs that you can use to help with problem determination and troubleshooting.

Use the following links to find out about the logs available for your platform and how to use them:

- Windows Linux UNIX “Error logs on Windows, UNIX and Linux systems” on page 342
- IBM i “Error logs on IBM i” on page 346
- “Error logs on HP Integrity NonStop Server” on page 345

z/OS On z/OS error messages are written to:

- The z/OS system console
- The channel-initiator job log

It is possible to suppress or exclude some messages on both distributed and z/OS systems.

For details of suppressing some messages on distributed systems, see “Suppressing channel error messages from error logs” on page 349.

z/OS On z/OS, if you are using the z/OS message processing facility to suppress messages, the console messages can be suppressed. See IBM MQ for z/OS concepts for more information.

z/OS For information about error messages, console logs, and dumps on IBM MQ for z/OS, see Problem determination on z/OS.

Related concepts

“Troubleshooting and support” on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

“Troubleshooting overview” on page 7
Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

“ First Failure Support Technology (FFST” on page 330
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM support personnel to diagnose the problem.

Related tasks

“Using trace” on page 350
You can use different types of trace to help you with problem determination and troubleshooting.

Error logs on Windows, UNIX and Linux systems

About error log files, and an example.

At installation time, an `errors` subdirectory is created in the `/var/mqm` file path under UNIX and Linux systems, and in the installation directory, for example `C:\Program Files\IBM\WebSphere MQ\` file path under Windows systems. The `errors` subdirectory can contain up to three error log files named:

- AMQERR01.LOG
- AMQERR02.LOG
- AMQERR03.LOG

For more information about directories where log files are stored, see “Error log directories on UNIX, Linux, and Windows” on page 344.

After you have created a queue manager, it creates three error log files when it needs them. These files have the same names as those files in the system error log directory. That is, AMQERR01, AMQERR02, and AMQERR03, and each has a default capacity of 2 MB (2 097 152 bytes). The capacity can be altered in the `Extended` queue manager properties page from the IBM MQ Explorer, or in the `QMErrorLog` stanza in the qm.ini file. These files are placed in the `errors` subdirectory in the queue manager data directory that you selected when you installed IBM MQ or created your queue manager. The default location for the `errors` subdirectory is `/var/mqm/qmgrs/` *qmname* file path under UNIX and Linux systems, and `C:\Program Files\IBM\WebSphere MQ\qmgrs\` *`qmname`* `\errors` file path under Windows systems.

As error messages are generated, they are placed in AMQERR01. When AMQERR01 gets bigger than 2 MB (2 097 152 bytes) it is copied to AMQERR02. Before the copy, AMQERR02 is copied to AMQERR03.LOG. The previous contents, if any, of AMQERR03 are discarded.

The latest error messages are thus always placed in AMQERR01, the other files being used to maintain a history of error messages.

All messages relating to channels are also placed in the appropriate error files belonging to the queue manager, unless the queue manager is unavailable, or its name is unknown. In which case, channel-related messages are placed in the system error log directory.

To examine the contents of any error log file, use your usual system editor.

An example of an error log

Figure 5 on page 343 shows an extract from an IBM MQ error log:

```
17/11/2004 10:32:29 - Process(2132.1) User(USER_1) Program(runmqchi.exe)
Host(HOST_1) Installation(Installation1)
VRMF(7.1.0.0) QMgr (A.B.C)
AMQ9542: Queue manager is ending.

EXPLANATION:
The program will end because the queue manager is quiescing.
ACTION:
None.
----- amqrimna.c : 931 -------------------------------------------------------
```

Figure 5. Sample IBM MQ error log

Operator messages

Operator messages identify normal errors, typically caused directly by users doing things like using parameters that are not valid on a command. Operator messages are national-language enabled, with message catalogs installed in standard locations.

These messages are written to the associated window, if any. In addition, some operator messages are written to the AMQERR01.LOG file in the queue manager directory, and others to the equivalent file in the system error log directory.

Error log access restrictions

Certain error log directories and error logs have access restrictions.

To gain the following access permissions, a user or application must be a member of the mqm group:

- Read and write access to all queue manager error log directories.
- Read and write access to all queue manager error logs.
- Write access to the system error logs.

If an unauthorized user or application attempts to write a message to a queue manager error log directory, the message is redirected to the system error log directory.

Ignoring error codes under UNIX and Linux systems

On UNIX and Linux systems, if you do not want certain error messages to be written to a queue manager error log, you can specify the error codes that are to be ignored using the QMErrorLog stanza.

For more information, see Queue manager error logs.

Ignoring error codes under Windows systems

On Windows systems, the error message is written to both the IBM MQ error log and the Windows Application Event Log. The error messages written to the Application Event Log includes messages of error severity, warning severity, and information severity. If you do not want certain error messages to be written to the Windows Application Event Log, you can specify the error codes that are to be ignored in the Windows registry.

Use the following registry key:

```
HKLM\Software\IBM\WebSphere MQ\Installation\MQ_INSTALLATION_NAME\IgnoredErrorCodes
```

where *MQ_INSTALLATION_NAME* is the installation name associated with a particular installation of IBM MQ.

The value that you set it to is an array of strings delimited by the NULL character, with each string value relating to the error code that you want ignored from the error log. The complete list is terminated with a NULL character, which is of type REG_MULTI_SZ.

For example, if you want IBM MQ to exclude error codes AMQ3045, AMQ6055, and AMQ8079 from the Windows Application Event Log, set the value to:

```
AMQ3045\0AMQ6055\0AMQ8079\0\0
```

The list of messages you want to exclude is defined for all queue managers on the machine. Any changes you make to the configuration will not take effect until each queue manager is restarted.

Related concepts

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Using logs" on page 341
There are a variety of logs that you can use to help with problem determination and troubleshooting.

"Problem determination on z/OS" on page 389
IBM MQ for z/OS, CICS, Db2, and IMS produce diagnostic information which can be used for problem determination.

Related tasks

"Using trace" on page 350
You can use different types of trace to help you with problem determination and troubleshooting.

Related reference

"Error logs on IBM i" on page 346
Use this information to understand the IBM MQ for IBM i error logs.

Error log directories on UNIX, Linux, and Windows

IBM MQ uses a number of error logs to capture messages concerning its own operation of IBM MQ, any queue managers that you start, and error data coming from the channels that are in use. The location of the error logs depends on whether the queue manager name is known and whether the error is associated with a client.

The location the error logs are stored in depends on whether the queue manager name is known and whether the error is associated with a client. *MQ_INSTALLATION_PATH* represents the high level directory where IBM MQ is installed.

- If the queue manager name is known, the location of the error log is shown in Table 3 on page 345.

Table 3. Queue manager error log directory

Platform	Directory
UNIX and Linux systems	`/var/mqm/qmgrs/` *qmname* `/errors`
Windows systems	*MQ_INSTALLATION_PATH*`\QMGRS\` *qmname* `\ERRORS\AMQERR01.LOG`

- If the queue manager name is not known, the location of the error log is shown in Table 4 on page 345.

Table 4. System error log directory

Platform	Directory
UNIX and Linux systems	`/var/mqm/errors`
Windows systems	*MQ_INSTALLATION_PATH*`\QMGRS\@SYSTEM\ERRORS\AMQERR01.LOG`

- If an error has occurred with a client application, the location of the error log on the client is shown in Table 5 on page 345.

Table 5. Client error log directory

Platform	Directory
UNIX and Linux systems	`/var/mqm/errors`
Windows systems	`MQ_DATA_PATH\ERRORS\AMQERR01.LOG`

In IBM MQ for Windows, an indication of the error is also added to the Application Log, which can be examined with the Event Viewer application provided with Windows systems.

Early errors

There are a number of special cases where these error logs have not yet been established and an error occurs. IBM MQ attempts to record any such errors in an error log. The location of the log depends on how much of a queue manager has been established.

If, because of a corrupt configuration file for example, no location information can be determined, errors are logged to an errors directory that is created at installation time on the root directory (`/var/mqm` or `C:\Program Files\IBM\WebSphere MQ`).

If IBM MQ can read its configuration information, and can access the value for the Default Prefix, errors are logged in the errors subdirectory of the directory identified by the Default Prefix attribute. For example, if the default prefix is `C:\Program Files\IBM\WebSphere MQ`, errors are logged in `C:\Program Files\IBM\WebSphere MQ\errors`.

For further information about configuration files, see Changing IBM MQ and queue manager configuration information.

Note: Errors in the Windows Registry are notified by messages when a queue manager is started.

Error logs on HP Integrity NonStop Server

Use this information to understand the IBM MQ client on HP Integrity NonStop Server error logs, together with an example.

At installation time, an errors subdirectory is created in the `<mqpath>/var/mqm` file path. The errors subdirectory can contain up to three error log files named:

- `AMQERR01.LOG`
- `AMQERR02.LOG`
- `AMQERR03.LOG`

As error messages are generated, they are written to AMQERR01.LOG. When AMQERR01.LOG gets bigger than 2 MB (2 097 152 bytes), it is copied to AMQERR02.LOG. Before the copy, AMQERR02.LOG is copied to AMQERR03.LOG. The previous contents, if any, of AMQERR03.LOG are discarded.

The latest error messages are therefore always placed in AMQERR01.LOG. The other log files are used to maintain a history of error messages.

To examine the contents of any error log file, use your system editor. The contents of the log files can read by any user, but write access requires the user to be a member of the mqm group.

An example of an error log

Figure 6 on page 346 shows an extract from an IBM MQ error log:

```
04/30/13 06:18:22 - Process(320406477.1) User(MYUSER) Program(nssfcps_c)
                    Host(myhost)
                    VRMF(7.1.0.0)
AMQ9558: The remote channel 'SYSTEM.DEF.SVRCONN' on host 'hostname
(x.x.x.x)(1414)' is not currently available.

EXPLANATION:
The channel program ended because an instance of channel 'SYSTEM.DEF.SVRCONN'
could not be started on the remote system. This could be for one of the
following reasons:

The channel is disabled.

The remote system does not have sufficient resources to run another instance of
the channel.

In the case of a client-connection channel, the limit on the number of
instances configured for the remote server-connection channel was reached.

ACTION:
Check the remote system to ensure that the channel is able to run. Try the
operation again.
----- cmqxrfpt.c : 504 --------------------------------------------------------
```

Figure 6. Sample IBM MQ error log

IBM i Error logs on IBM i

Use this information to understand the IBM MQ for IBM i error logs.

By default, only members of the QMQMADM group can access error logs. To give users access to error logs, who are not members of this group, set **ValidateAuth** to *No* and grant those users *PUBLIC authority. See Filesystem for more information.

IBM MQ uses a number of error logs to capture messages concerning the operation of IBM MQ itself, any queue managers that you start, and error data coming from the channels that are in use.

At installation time, a /QIBM/UserData/mqm/errors subdirectory is created in the IFS.

The location of the error logs depends on whether the queue manager name is known.

In the IFS:

- If the queue manager name is known and the queue manager is available, error logs are located in:

```
/QIBM/UserData/mqm/qmgrs/qmname/errors
```

- If the queue manager is not available, error logs are located in:

```
/QIBM/UserData/mqm/errors
```

You can use the system utility EDTF to browse the errors directories and files. For example:

```
EDTF '/QIBM/UserData/mqm/errors'
```

Alternatively, you can use option 23 against the queue manager from the WRKMQM panel.

The errors subdirectory can contain up to three error log files named:

- `AMQERR01.LOG`
- `AMQERR02.LOG`
- `AMQERR03.LOG`

After you have created a queue manager, three error log files are created when they are needed by the queue manager. These files have the same names as the `/QIBM/UserData/mqm/errors` ones, that is AMQERR01, AMQERR02, and AMQERR03, and each has a capacity of 2 MB (2 097 152 bytes). The files are placed in the errors subdirectory of each queue manager that you create, that is `/QIBM/UserData/mqm/qmgrs/`*`qmname`*`/errors`.

As error messages are generated, they are placed in AMQERR01. When AMQERR01 gets bigger than 2 MB (2 097 152 bytes), it is copied to AMQERR02. Before the copy, AMQERR02 is copied to AMQERR03.LOG. The previous contents, if any, of AMQERR03 are discarded.

The latest error messages are thus always placed in AMQERR01, the other files being used to maintain a history of error messages.

All messages relating to channels are also placed in the appropriate errors files of the queue manager, unless the name of their queue manager is unknown or the queue manager is unavailable. When the queue manager name is unavailable or its name cannot be determined, channel-related messages are placed in the `/QIBM/UserData/mqm/errors` subdirectory.

To examine the contents of any error log file, use your system editor, EDTF, to view the stream files in the IFS.

Note:

1. Do not change ownership of these error logs.
2. If any error log file is deleted, it is automatically re-created when the next error message is logged.

Early errors

There are a number of special cases where the error logs have not yet been established and an error occurs. IBM MQ attempts to record any such errors in an error log. The location of the log depends on how much of a queue manager has been established.

If, because of a corrupted configuration file, for example, no location information can be determined, errors are logged to an errors directory that is created at installation time.

If both the IBM MQ configuration file and the DefaultPrefix attribute of the AllQueueManagers stanza are readable, errors are logged in the errors subdirectory of the directory identified by the DefaultPrefix attribute.

Operator messages

Operator messages identify normal errors, typically caused directly by users doing things like using parameters that are not valid on a command. Operator messages are national language enabled, with message catalogs installed in standard locations.

These messages are written to the job log, if any. In addition, some operator messages are written to the `AMQERR01.LOG` file in the queue manager directory, and others to the `/QIBM/UserData/mqm/errors` directory copy of the error log.

An example IBM MQ error log

Figure 7 on page 348 shows a typical extract from an IBM MQ error log.

```
************Beginning of data**************
07/19/02  11:15:56 AMQ9411: Repository manager ended normally.

EXPLANATION:
Cause . . . . . :   The repository manager ended normally.
Recovery  . . . :   None.
Technical Description . . . . . . . . :   None.
-------------------------------------------------------------------------------
07/19/02  11:15:57 AMQ9542: Queue manager is ending.

EXPLANATION:
Cause . . . . . :   The program will end because the queue manager is quiescing.
Recovery  . . . :   None.
Technical Description . . . . . . . . :   None.
----- amqrimna.c : 773 --------------------------------------------------------

07/19/02  11:16:00 AMQ8004: WebSphere MQ queue manager 'mick' ended.
EXPLANATION:
Cause . . . . . :   WebSphere MQ queue manager 'mick' ended.
Recovery  . . . :   None.
Technical Description . . . . . . . . :   None.
-------------------------------------------------------------------------------
07/19/02  11:16:48 AMQ7163: WebSphere MQ job number 18429 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18429 the CCSID is 37. The job name is
                    582775/MQUSER/AMQZXMA0.
Recovery  . . . :   None
-------------------------------------------------------------------------------
07/19/02  11:16:49 AMQ7163: WebSphere MQ job number 18430 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18430 the CCSID is 0. The job name is
                    582776/MQUSER/AMQZFUMA.
Recovery  . . . :   None
-------------------------------------------------------------------------------
07/19/02  11:16:49 AMQ7163: WebSphere MQ job number 18431 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18431 the CCSID is 37. The job name is
                    582777/MQUSER/AMQZXMAX.
Recovery  . . . :   None
-------------------------------------------------------------------------------
07/19/02  11:16:50 AMQ7163: WebSphere MQ job number 18432 started.

EXPLANATION:
Cause . . . . . :   This job has started to perform work for Queue Manager
                    mick, The job's PID is 18432 the CCSID is 37. The job name is
                    582778/MQUSER/AMQALMPX.
Recovery  . . . :   None
-------------------------------------------------------------------------------
```

Figure 7. Extract from an IBM MQ error log

Related concepts

"Error logs on Windows, UNIX and Linux systems" on page 342
About error log files, and an example.

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Using logs" on page 341
There are a variety of logs that you can use to help with problem determination and troubleshooting.

"Problem determination on z/OS" on page 389

IBM MQ for z/OS, CICS, Db2, and IMS produce diagnostic information which can be used for problem determination.

Related tasks

"Using trace" on page 350
You can use different types of trace to help you with problem determination and troubleshooting.

distributed Suppressing channel error messages from error logs

You can prevent selected messages from being sent to the error logs for a specified time interval, for example if your IBM MQ system produces a large number of information messages that fill the error logs.

About this task

There are two ways of suppressing messages for a given time interval:

- By using SuppressMessage and SuppressInterval in the QMErrorLog stanza in the `qm.ini` file.
- By using the environment variables MQ_CHANNEL_SUPPRESS_MSGS and MQ_CHANNEL_SUPPRESS_INTERVAL.

Procedure

- To suppress messages for a given time interval by using the QMErrorLog stanza in the `qm.ini` file, specify the messages that are to be written to the queue manager error log once only during a given time interval with SuppressMessage, and specify the time interval for which the messages are to be suppressed with SuppressInterval.
  For example, to suppress the messages AMQ9999, AMQ9002, AMQ9209 for 30 seconds, include the following information in the QMErrorLog stanza of the `qm.ini` file:

  ```
  SuppressMessage=9001,9002,9202
  SuppressInterval=30
  ```

  Windows Linux Alternatively, instead of editing the `qm.ini` file directly, you can use the Extended Queue Manager properties page in MQ Explorer to exclude and suppress messages.
- To suppress messages for a given time interval by using the environment variables **MQ_CHANNEL_SUPPRESS_MSGS** and **MQ_CHANNEL_SUPPRESS_MSGS**, complete the following steps:
  a) Specify the messages that are to be suppressed with **MQ_CHANNEL_SUPPRESS_MSGS**.

     You can include up to 20 channel error message codes in a comma-separated list. There is no comprehensive list of message ids that can be included in the **MQ_CHANNEL_SUPPRESS_MSGS** environment variable. However, the message ids must be channel messages (that is AMQ9xxx: messages).

     The following examples are for messages AMQ9999, AMQ9002, AMQ9209.

     – Linux UNIX On UNIX and Linux:

     ```
     export MQ_CHANNEL_SUPPRESS_MSGS=9999,9002,9209
     ```

     – Windows On Windows:

     ```
     set MQ_CHANNEL_SUPPRESS_MSGS=9999,9002,9209
     ```

  b) Specify the time interval for which the messages are to be suppressed with **MQ_CHANNEL_SUPPRESS_INTERVAL**.

     The default value is `60,5` which means that after the first five occurrences of a given message in a 60 second interval, any further occurrences of that message are suppressed until the end of that 60 second interval. A value of `0,0` means always suppress. A value of `0,`*n* where *n* > 0 means never suppress.

Related information
QMErrorLog stanza on UNIX, Linux, and Windows
QMErrorLog stanza on IBM i
Environment variables
Queue manager properties

Using trace

You can use different types of trace to help you with problem determination and troubleshooting.

About this task

Use this information to find out about the different types of trace, and how to run trace for your platform.

- "Using trace on Windows" on page 350
- "Using trace on UNIX and Linux systems" on page 352
- IBM i "Using trace on IBM MQ server on IBM i" on page 356
- IBM i "Using trace on IBM MQ client on IBM i" on page 359
- z/OS "Using trace for problem determination on z/OS" on page 361
- "Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
- "Tracing IBM MQ classes for JMS applications" on page 373
- "Tracing IBM MQ classes for Java applications" on page 377
- "Tracing the IBM MQ Resource Adapter" on page 381
- "Tracing additional IBM MQ Java components" on page 383
- "Controlling trace in a running process by using IBM MQ classes for Java and IBM MQ classes for JMS" on page 386

Related concepts
"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Troubleshooting overview" on page 7
Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

"Using logs" on page 341
There are a variety of logs that you can use to help with problem determination and troubleshooting.

" First Failure Support Technology (FFST" on page 330
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM support personnel to diagnose the problem.

Related tasks
"Contacting IBM Software Support" on page 328
You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

Using trace on Windows

Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

Windows uses the following commands for the client trace facility:

strmqtrc
  to start tracing

endmqtrc
  to end tracing

The output files are created in the `MQ_DATA_PATH/trace` directory.

Trace files on IBM MQ for Windows

Trace files are named `AMQ` *ppppp*. *qq*`.TRC` where the variables are:

ppppp
  The ID of the process reporting the error.

qq
  A sequence number, starting at 0. If the full file name exists, this value is incremented by one until a unique trace file name is found. A trace file name can exist if a process is reused.

Note:

1. The process identifier can contain fewer, or more, digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

To format or view a trace file, you must be either the creator of the trace file, or a member of the mqm group.

SSL trace files have the names `AMQ.SSL.TRC` and `AMQ.SSL.TRC.1`. You cannot format SSL trace files; send them unchanged to IBM support.

How to start and stop a trace

Enable or modify tracing using the **strmqtrc** control command (see strmqtrc). To stop tracing, use the **endmqtrc** control command (see endmqtrc).

In IBM MQ for Windows systems, you can also start and stop tracing using the MQ Explorer, as follows:

1. Start the MQ Explorer from the **Start** menu.
2. In the Navigator View, right-click the **IBM MQ** tree node, and select **Trace...**. The Trace Dialog is displayed.
3. Click **Start** or **Stop** as appropriate.

Selective component tracing

Use the `-t` and `-x` options to control the amount of trace detail to record. By default, all trace points are enabled. You can specify the points that you do not want to trace using the `-x` option. So if, for example, you want to trace only data flowing over communications networks, use:

```
strmqtrc -x all -t comms
```

For detailed information about the trace command, see strmqtrc.

Selective process tracing

Use the `-p` option of the **strmqtrc** command control to restrict trace generation to specified named processes. For example, to trace all threads that result from any running process called amqxxx.exe, use the following command:

```
strmqtrc -p amqxxx.exe
```

For detailed information about the trace command, see strmqtrc.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace on IBM MQ server on IBM i" on page 356
Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Using trace on UNIX and Linux systems

Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

UNIX and Linux systems use the following commands for the IBM MQ MQI client trace facility:

strmqtrc
to start tracing

endmqtrc
to end tracing

dspmqtrc <filename>
to display a formatted trace file

The trace facility uses a number of files, which are:

- One file for each entity being traced, in which trace information is recorded
- One additional file on each machine, to provide a reference for the shared memory used to start and end tracing
- One file to identify the semaphore used when updating the shared memory

Files associated with trace are created in a fixed location in the file tree, which is `/var/mqm/trace`.

All client tracing takes place to files in this directory.

You can handle large trace files by mounting a temporary file system over this directory.

On AIX you can use AIX system trace in addition to using the strmqtrc and endmqtrc commands. For more information, see "Tracing with the AIX system trace" on page 354.

Trace files on IBM MQ for UNIX and Linux systems

Trace files are created in the directory `/var/mqm/trace`.

Note: You can accommodate the production of large trace files by mounting a temporary file system over the directory that contains your trace files. Alternatively, rename the trace directory and create the symbolic link `/var/mqm/trace` to a different directory.

Trace files are named `AMQ` *ppppp*. *qq*.TRC where the variables are:

ppppp
The ID of the process reporting the error.

qq

A sequence number, starting at 0. If the full file name exists, this value is incremented by one until a unique trace file name is found. A trace file name can exist if a process is reused.

Note:

1. The process identifier can contain fewer, or more, digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

To format or view a trace file, you must be either the creator of the trace file, or a member of the mqm group.

SSL trace files have the names `AMQ.SSL.TRC` and `AMQ.SSL.TRC.1`. You cannot format SSL trace files; send them unchanged to IBM support.

How to start and stop a trace

In IBM MQ for UNIX and Linux systems, you enable or modify tracing using the **strmqtrc** control command (see strmqtrc). To stop tracing, you use the **endmqtrc** control command (see endmqtrc). On IBM MQ for Linux (x86 and x86-64 platforms) systems, you can alternatively use the MQ Explorer to start and stop tracing. However, you can trace only everything using the function provided, equivalent to using the commands `strmqtrc -e` and `endmqtrc -e`.

Trace output is unformatted; use the **dspmqtrc** control command to format trace output before viewing. For example, to format all trace files in the current directory use the following command:

```
dspmqtrc *.TRC
```

For detailed information about the control command, **dspmqtrc**, see dspmqtrc.

Selective component tracing on IBM MQ for UNIX and Linux systems

Use the `-t` and `-x` options to control the amount of trace detail to record. By default, all trace points are enabled. Specify the points you do not want to trace using the `-x` option. If, for example, you want to trace, for queue manager QM1, only output data associated with using Secure Sockets Layer (SSL) channel security, use:

```
strmqtrc -m QM1 -t ssl
```

For detailed information about the trace command, see strmqtrc.

Selective component tracing on IBM MQ for AIX

Use the environment variable MQS_TRACE_OPTIONS to activate the high detail and parameter tracing functions individually.

Because MQS_TRACE_OPTIONS enables tracing to be active without high detail and parameter tracing functions, you can use it to reduce the effect on performance and trace size when you are trying to reproduce a problem with tracing switched on.

Only set the environment variable MQS_TRACE_OPTIONS if you have been instructed to do so by your service personnel.

Typically MQS_TRACE_OPTIONS must be set in the process that starts the queue manager, and before the queue manager is started, or it is not recognized. Set MQS_TRACE_OPTIONS before tracing starts. If it is set after tracing starts it is not recognized.

Selective process tracing on IBM MQ for UNIX and Linux systems

Use the `-p` option of the **`strmqtrc`** command control to restrict trace generation to specified named processes. For example, to trace all threads that result from any running process called `amqxxx`, use the following command:

```
strmqtrc -p amqxxx
```

For detailed information about the trace command, see strmqtrc.

Related concepts

"Using trace on IBM MQ server on IBM i" on page 356
Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **`runmqakm`** tracing and **`strmqikm`** (iKeyman) and **`runmqckm`** (iKeycmd) tracing.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related reference

"Using trace on Windows" on page 350
Use the **`strmqtrc`** and **`endmqtrc`** commands or the MQ Explorer interface to start and end tracing.

Tracing with the AIX system trace

In addition to the IBM MQ trace, IBM MQ for AIX users can use the standard AIX system trace.

AIX system tracing is a two-step process:

1. Gathering the data
2. Formatting the results

IBM MQ uses two trace hook identifiers:

X'30D'
This event is recorded by IBM MQ on entry to or exit from a subroutine.

X'30E'
This event is recorded by IBM MQ to trace data such as that being sent or received across a communications network.

Trace provides detailed execution tracing to help you to analyze problems. IBM service support personnel might ask for a problem to be re-created with trace enabled. The files produced by trace can be **very** large so it is important to qualify a trace, where possible. For example, you can optionally qualify a trace by time and by component.

There are two ways to run trace:

1. Interactively.

   The following sequence of commands runs an interactive trace on the program `myprog` and ends the trace.

   ```
   trace -j30D,30E -o trace.file
   ->!myprog
   ->q
   ```

2. Asynchronously.

   The following sequence of commands runs an asynchronous trace on the program `myprog` and ends the trace.

   ```
   trace -a -j30D,30E -o trace.file
   myprog
   trcstop
   ```

You can format the trace file with the command:

```
trcrpt -t MQ_INSTALLATION_PATH/lib/amqtrc.fmt trace.file > report.file
```

`MQ_INSTALLATION_PATH` represents the high-level directory in which IBM MQ is installed.

`report.file` is the name of the file where you want to put the formatted trace output.

Note: All IBM MQ activity on the machine is traced while the trace is active.

Using trace on HP Integrity NonStop Server

Use the **`strmqtrc`** and **`endmqtrc`** commands to start and end tracing, and **`dspmqtrc`** to display a trace file.

Use the following commands on the IBM MQ client for HP Integrity NonStop Server system to use the IBM MQ client trace facility:

`strmqtrc`
: To start tracing

`endmqtrc`
: To end tracing

`dspmqtrc <filename>`
: To display a formatted trace file

The trace facility creates a file for each entity that is being traced. The trace files are created in a fixed location, which is `<mqpath>/var/mqm/trace`. You can handle large trace files by mounting a temporary file system over this directory.

Trace files are named `AMQ.nnn.xx.ppp.qq.TRC` where:

nnn
: The name of the process.

xx
: The processor number on which the process is running.

ppp
: The PIN of the process that you are tracing.

qq
: A sequence number, starting at 0. If the full file name exists, this value is incremented by one until a unique trace file name is found. A trace file name can exist if a process is reused.

Note:

1. Each field can contain fewer, or more, digits than shown in the example.
2. There is one trace file for each process that is running as part of the entity that is being traced.

Trace files are created in a binary format. To format or view a trace file use the **`dspmqtrc`** command, you must be either the creator of the trace file, or a member of the mqm group. For example, to format all trace files in the current directory use the following command:

```
dspmqtrc *.TRC
```

For more information about the control command **dspmqtrc**, see dspmqtrc.

How to start and stop a trace

On IBM MQ client for HP Integrity NonStop Server systems, you can enable or modify tracing by using the **strmqtrc** control command, for more information, see strmqtrc. To stop tracing, use the **endmqtrc** control command, for more information, see endmqtrc.

The control commands **strmqtrc** and **endmqtrc** affect tracing only for those processes that are running in one specific processor. By default, this processor is the same as the one in your OSS shell. To enable or end tracing for processes that are running in another processor, you must precede the **strmqtrc** or **endmqtrc** commands with `run -cpu=n` at an OSS shell command prompt, where n is the processor number. Here is an example of how to enter the **strmqtrc** command at an OSS shell command prompt:

```
run -cpu=2 strmqtrc
```

This command enables tracing for all processes that are running in processor 2.

The `-m` option to select a queue manager is not relevant for use on the IBM MQ client for HP Integrity NonStop Server . Specifying the `-m` option produces an error.

Use the `-t` and `-x` options to control the amount of trace detail to record. By default, all trace points are enabled. Specify the points that you do not want to trace by using the `-x` option.

IBM i Using trace on IBM MQ server on IBM i

Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

There are two stages in using trace:

1. Decide whether you want early tracing. Early tracing lets you trace the creation and startup of queue managers. Note, however, that early trace can easily generate large amounts of trace, because it is implemented by tracing all jobs for all queue managers. To enable early tracing, use TRCMQM with the TRCEARLY parameter set to *YES.
2. Start tracing work using `TRCMQM` **ON*. To stop the trace, you have two options:
   - `TRCMQM` **OFF*, to stop collecting trace records for a queue manager. The trace records are written to files in the `/QIBM/UserData/mqm/trace` directory.
   - `TRCMQM` **END*, to stop collecting trace records for all queue managers and to disable early trace. This option ignores the value of the TRCEARLY parameter.

Specify the level of detail you want, using the TRCLEVEL parameter set to one of the following values:

DFT
For minimum-detail level for flow processing trace points.

DETAIL
For high-detail level for flow processing trace points.

PARMS
For default-detail level for flow processing trace points.

Specify the type of trace output you want, using the OUTPUT parameter set to one of the following values:

***MQM**
Collect binary IBM MQ trace output in the directory specified by the TRCDIR parameter. This value is the default value.

***MQMFMT**
Collect formatted IBM MQ trace output in the directory specified by the TRCDIR parameter.

***PEX**
Collect Performance Explorer (PEX) trace output

***ALL**
Collect both IBM MQ unformatted trace and PEX trace output

Selective trace

You can reduce the amount of trace data being saved, improving runtime performance, using the command `TRCMQM` with `F4=prompt`, then F9 to customize the TRCTYPE and EXCLUDE parameters:

TRCTYPE
Specifies the type of trace data to store in the trace file. If you omit this parameter, all trace points except those trace points specified in EXCLUDE are enabled.

EXCLUDE
Specifies the type of trace data to omit from the trace file. If you omit this parameter, all trace points specified in TRCTYPE are enabled.

The options available on both TRCTYPE and EXCLUDE are:

***ALL (TRCTYPE only)**
All the trace data as specified by the following keywords is stored in the trace file.

trace-type-list
You can specify more than one option from the following keywords, but each option can occur only once.

***API**
Output data for trace points associated with the MQI and major queue manager components.

***CMTRY**
Output data for trace points associated with comments in the IBM MQ components.

***COMMS**
Output data for trace points associated with data flowing over communications networks.

***CSDATA**
Output data for trace points associated with internal data buffers in common services.

***CSFLOW**
Output data for trace points associated with processing flow in common services.

***LQMDATA**
Output data for trace points associated with internal data buffers in the local queue manager.

***LQMFLOW**
Output data for trace points associated with processing flow in the local queue manager.

***OTHDATA**
Output data for trace points associated with internal data buffers in other components.

***OTHFLOW**
Output data for trace points associated with processing flow in other components.

***RMTDATA**
Output data for trace points associated with internal data buffers in the communications component.

***RMTFLOW**
Output data for trace points associated with processing flow in the communications component.

***SVCDATA**
Output data for trace points associated with internal data buffers in the service component.

***SVCFLOW**
Output data for trace points associated with processing flow in the service component.

***VSNDATA**
Output data for trace points associated with the version of IBM MQ running.

Wrapping trace

Use the MAXSTG parameter to wrap trace, and to specify the maximum size of storage to be used for the collected trace records.

The options are:

DFT
Trace wrapping is not enabled. For each job, trace data is written to a file with the suffix .TRC until tracing is stopped.

maximum-K-bytes
Trace wrapping is enabled. When the trace file reaches its maximum size, it is renamed with the suffix .TRS, and a new trace file with suffix .TRC is opened. Any existing .TRS file is deleted. Specify a value in the range 1 through 16 000.

Formatting trace output

To format any trace output:

- Enter the QShell
- Enter the command

```
/QSYS.LIB/QMQM.LIB/DSPMQTRC.PGM [-t Format] [-h] [-s]
[-o OutputFileName] InputFileName
```

where:

InputFileName
Is a required parameter specifying the name of the file containing the unformatted trace. For example `/QIBM/UserData/mqm/trace/AMQ12345.TRC`.

-t *FormatTemplate*
Specifies the name of the template file containing details of how to display the trace. The default value is `/QIBM/ProdData/mqm/lib/amqtrc.fmt`.

-h
Omit header information from the report.

-s
Extract trace header and put to stdout.

-o *output_filename*
The name of the file into which to write formatted data.

You can also specify `dspmqtrc *` to format all trace.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related reference

"Using trace on Windows" on page 350

Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

IBM i Using trace on IBM MQ client on IBM i

On IBM i, there is no Control Language (CL) command to capture the trace when using a standalone IBM MQ MQI client. STRMQTRC and ENDMQTRC programs can be used to enable and disable the trace.

Example for start trace:

```
CALL PGM(QMQM/STRMQTRC) PARM('-e' '-t' 'all' '-t' 'detail')
Where -e option requests early tracing of all the process -t option for trace type
```

To end the trace

```
CALL PGM(QMQM/ENDMQTRC) PARM('-e')
```

- Optional parameters:

  -t *TraceType*
  The points to trace and the amount of trace detail to record. By default all trace points are enabled and a default-detail trace is generated.

  Alternatively, you can supply one or more of the options in Table 1. For each *TraceType* value you specify, including -t all, specify either -t parms or -t detail to obtain the appropriate level of trace detail. If you do not specify either -t parms or -t detail for any particular trace type, only a default-detail trace is generated for that trace type.

  If you supply multiple trace types, each must have its own -t flag. You can include any number of -t flags, if each has a valid trace type associated with it.

  It is not an error to specify the same trace type on multiple -t flags.

  See the following table for allowed values for *TraceType*.

Table 6. TraceType values

Value	**Description**
all	Output data for every trace point in the system (the default). Using *all* activates tracing at default detail level.
api	Output data for trace points associated with the message queue interface (MQI) and major queue manager components.
commentary	Output data for trace points associated with comments in the IBM MQ components.
comms	Output data for trace points associated with data flowing over communications networks.
csdata	Output data for trace points associated with internal data buffers in common services.
csflows	Output data for trace points associated with processing flow in common services.
detail	Activate tracing at high-detail level for flow processing trace points.
lqmdata	Output data for trace points associated with internal data buffers in the local queue manager.
lqmflows	Output data for trace points associated with processing flow in the local queue manager.
otherdata	Output data for trace points associated with internal data buffers in other components.

Table 6. TraceType values (continued)

Value	Description
otherfl ows	Output data for trace points associated with processing flow in other components.
parms	Activate tracing at default-detail level for flow processing trace points.
remote data	Output data for trace points associated with internal data buffers in the communications component.
remote flows	Output data for trace points associated with processing flow in the communications component.
service data	Output data for trace points associated with internal data buffers in the service component.
service flows	Output data for trace points associated with processing flow in the service component.
version data	Output data for trace points associated with the version of IBM MQ running.

-x *TraceType*
: The points not to trace. By default all trace points are enabled and a default-detail trace is generated. The *TraceType* values you can specify are the same as the values listed for the -t flag in Table 1.

  You can use the -x flag with *TraceType* values to exclude those trace points you do not want to record. Excluding specified trace points is useful in reducing the amount of trace produced.

  If you supply multiple trace types, each must have its own -x flag. You can include any number of -x flags, if each has a valid *TraceType* associated with it.

-s
: Reports the tracing options that are currently in effect. You must use this parameter on its own with no other parameters.

  A limited number of slots are available for storing trace commands. When all slots are in use, then no more trace commands can be accepted unless they replace an existing slot. Slot numbers are not fixed, so if the command in slot number 0 is removed, for example by an **endmqtrc** command, then all the other slots move up, with slot 1 becoming slot 0, for example. An asterisk (*) in a field means that no value is defined, and is equivalent to the asterisk wildcard.

-l *MaxSize*
: The maximum size of a trace file (AMQppppp.qq.TRC) in megabytes (MB). For example, if you specify a *MaxSize* of 1, the size of the trace is limited to 1 MB.

  When a trace file reaches the specified maximum, it is renamed to AMQppppp.qq.TRS and a new AMQppppp.qq.TRC file is started. If a previous copy of an AMQppppp.qq.TRS file exists, it is deleted.

  The highest value that *MaxSize* can be is 2048 MB.

-e
: Requests early tracing of all processes

For more details see the **strmqtrc** command

- To end the trace:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM [-e] [-a]
```

where:

-e
: Ends early tracing of all processes.

  Using **endmqtrc** with no parameters has the same effect as **endmqtrc -e**. You cannot specify the -e flag with the -m flag, the -i flag, or the -p flag.

-a
: Ends all tracing.

For more details see the endmqtrc **endmqtrc** command

- To display a formatted trace file:

```
/QSYS.LIB/QMQM.LIB/DSPMQTRC.pgm
```

To examine FFST files, see the First Failure Support Technology (FFST) for IBM MQ for IBM i.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related reference

"Using trace on Windows" on page 350
Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

z/OS Using trace for problem determination on z/OS

There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

The trace facilities available with IBM MQ for z/OS are:

- The user parameter (or API) trace
- The IBM internal trace used by the support center
- The channel initiator trace
- The line trace

Use the following links to find out how to collect and interpret the data produced by the user parameter trace, and describes how to produce the IBM internal trace for use by the IBM support center. There is also information about the other trace facilities that you can use with IBM MQ.

- Controlling the GTF for your z/OS system
- Controlling the IBM MQ trace for each queue manager subsystem for which you want to collect data
- "Formatting and identifying the control block information" on page 364
- "Interpreting the trace information" on page 365

If trace data is not produced, check the following:

- Was the GTF started correctly, specifying EIDs 5E9, 5EA, and 5EE on the USRP option?

- Was the START TRACE(GLOBAL) command entered correctly, and were the relevant classes specified?

For more information about other trace options available on z/OS, see "Other types of trace" on page 367.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace on IBM MQ server on IBM i" on page 356
Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related reference

"Using trace on Windows" on page 350
Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

The MQI call and user parameter, and z/OS generalized trace facility (GTF)

Use this topic to understand how to control GTF and IBM MQ trace.

You can obtain information about MQI calls and user parameters passed by some IBM MQ calls on entry to, and exit from, IBM MQ. To do this, use the global trace in conjunction with the z/OS generalized trace facility (GTF).

Controlling the GTF

Use this topic to understand how to start and stop the GTF.

- Starting the GTF
- Stopping the GTF

Starting the GTF

When you start the GTF, specify the USRP option. You are prompted to enter a list of event identifiers (EIDs). The EIDs used by IBM MQ are:

5E9
To collect information about control blocks on entry to IBM MQ

5EA
To collect information about control blocks on exit from IBM MQ

Sometimes, if an error occurs that you cannot solve yourself, you might be asked by your IBM support center to supply other, internal, trace information for them to analyze. The additional type of trace is:

5EE
To collect information internal to IBM MQ

You can also use the JOBNAMEP option, specifying the batch, CICS, IMS, or TSO job name, to limit the trace output to specific jobs. Figure 8 on page 363 illustrates sample startup for the GTF, specifying the four EIDs, and a jobname. The lines shown in bold type **like this** are the commands that you enter at the console; the other lines are prompts and responses.

For more information about starting the GTF trace, see the *MVS Diagnosis: Tools and Service Aids* manual.

```
START GTFxx.yy
 #HASP100 GTFxx.yy ON STCINRDR
 #HASP373 GTFxx.yy STARTED
*01 AHL100A SPECIFY TRACE OPTIONS
 R 01,TRACE=JOBNAMEP,USRP
 TRACE=JOBNAMEP,USRP
 IEE600I REPLY TO 01 IS;TRACE=JOBNAMEP,USRP
*02 ALH101A SPECIFY TRACE EVENT KEYWORDS - JOBNAME=,USR=
 R 02,JOBNAME=(xxxxMSTR,xxxxCHIN,zzzzzzzz),USR=(5E9,5EA,5EE)
 JOBNAME=(xxxxMSTR,xxxxCHIN,zzzzzzzz),USR=(5E9,5EA,5EE)
 IEE600I REPLY TO 02 IS;JOBNAME=(xxxxMSTR,xxxxCHIN,zzzzzzzz),USR=(5E9,5EA,5EE)
*03 ALH102A CONTINUE TRACE DEFINITION OR REPLY END
 R 03,END
 END
 IEE600I REPLY TO 03 IS;END
 AHL103I TRACE OPTIONS SELECTED-USR=(5E9,5EA,5EE)
 AHL103I JOBNAME=(xxxxMSTR,xxxxCHIN,zzzzzzzz)
*04 AHL125A RESPECIFY TRACE OPTIONS OR REPLY U
 R 04,U
 U
 IEE600I REPLY TO 04 IS;U
 AHL031I GTF INITIALIZATION COMPLETE

where:

     xx is the name of the GTF procedure to use (optional), and yy is an
     identifier for this occurrence of GTF trace.

     xxxx is the name of the queue manager and zzzzzzzz is
     a batch job or CICS region name. Up to 5 job names can be listed.
```

Figure 8. Example startup of GTF to use with the IBM MQ trace

When using GTF, specify the primary job name (CHINIT, CICS, or batch) in addition to the queue manager name (xxxxMSTR).

Stopping the GTF

When you stop the GTF, you must specify the additional identifier (**yy**) used at startup. Figure 9 on page 363 illustrates a sample stop command for the GTF. The commands shown in bold type **like this** are the commands that you enter at the console.

```
 STOP yy
```

Figure 9. Example of GTF Stop command to use with the IBM MQ trace

Related information

Generating IBM MQ GTF trace on IBM z/OS

Controlling the trace within IBM MQ

IBM MQ for z/OS trace is controlled using MQSC commands. Use this topic to understand how to control the trace, and the type of trace information that is output.

Use the START TRACE command, specifying type GLOBAL to start writing IBM MQ records to the GTF. You must also specify dest(GTF), for example in the following command:

```
/cpf start trace(G)class(2,3)dest(GTF)
```

To define the events that you want to produce trace data for, use one or more of the following classes:

CLASS	Event traced
2	Record the MQI call and MQI parameters when a completion code other than MQRC_NONE is detected.
3	Record the MQI call and MQI parameters on entry to and exit from the queue manager.

After the trace has started, you can display information about, alter the properties of, and stop, the trace with the following commands:

- DISPLAY TRACE
- ALTER TRACE
- STOP TRACE

To use any of the trace commands, you must have one of the following:

- Authority to issue start and stop trace commands (trace authority)
- Authority to issue the display trace command (display authority)

Note:

1. The trace commands can also be entered through the initialization input data sets.
2. The trace information produced will also include details of syncpoint flows - for example PREPARE and COMMIT.

For information about these commands, see MQSC commands.

Formatting and identifying the control block information

After capturing a trace, the output must be formatted and the IBM MQ control blocks identified.

- Formatting the information
- Identifying the control blocks associated with IBM MQ
- Identifying the event identifier associated with the control block

Formatting the information

To format the user parameter data collected by the global trace, use either the batch job shown in Figure 10 on page 365 or the IPCS `GTFTRACE USR( xxx )` command, where *xxx* is:

5E9
  To format information about control blocks on entry to IBM MQ MQI calls

5EA
  To format information about control blocks on exit from IBM MQ MQI calls

5EE
  To format information about IBM MQ internals

You can also specify the JOBNAME(*jobname*) parameter to limit the formatted output to specific jobs.

```
//S1 EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=4096K
//IPCSPARM DD DSN=SYS1.PARMLIB,DISP=SHR
//IPCSDDIR DD DSN=thlqual.ipcs.dataset.directory,DISP=SHR
//SYSTSPRT DD SYSOUT=*,DCB=(LRECL=137)
//IPCSTOC  DD SYSOUT=*
//GTFIN    DD DSN=gtf.trace,DISP=SHR
//SYSTSIN  DD *
IPCS
SETDEF FILE(GTFIN) NOCONFIRM
GTFTRACE USR(5E9,5EA,5EE)
/*
//STEPLIB  DD  DSN=thlqual.SCSQAUTH,DISP=SHR
```

Figure 10. Formatting the GTF output in batch

Identifying the control blocks associated with IBM MQ

The format identifier for the IBM MQ trace is D9. This value appears at the beginning of each formatted control block in the formatted GTF output, in the form:

```
USRD9
```

Identifying the event identifier associated with the control block

The trace formatter inserts one of the following messages at the top of each control block. These indicate whether the data was captured on entry to or exit from IBM MQ

- CSQW072I ENTRY: MQ user parameter trace
- CSQW073I EXIT: MQ user parameter trace

Related concepts

"Controlling the GTF" on page 362
Use this topic to understand how to start and stop the GTF.

Interpreting the trace information

The GTFTRACE produced by IBM MQ can be examined to determine possible errors with invalid addresses, invalid control blocks, and invalid data.

When you look at the data produced by the GTFTRACE command, consider the following points:

- If the control block consists completely of zeros, it is possible that an error occurred while copying data from the user's address space. This might be because an invalid address was passed.
- If the first part of the control block contains non-null data, but the rest consists of zeros, it is again possible that an error occurred while copying data from the user's address space, for example, the control block was not placed entirely within valid storage. This might also be due to the control block not being initialized correctly.
- If the error occurred on exit from IBM MQ, it is possible that IBM MQ might not write the data to the user's address space. The data displayed is the version that it was attempting to copy to the user's address space.

The following tables show details of the control blocks that are traced.

Table 7 on page 366 illustrates which control blocks are traced for different MQI calls.

Table 7. Control blocks traced for IBM MQ MQI calls

MQI call	Entry	Exit
MQCB	MQCBD, MQMD, MQGMO	MQCBD, MQMD, MQGMO
MQCLOSE	None	None
MQGET	MQMD, MQGMO	MQMD, MQGMO, and the first 256 bytes of message data
MQINQ	Selectors (if *SelectorCount* is greater than 0)	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)
MQOPEN	MQOD	MQOD
MQPUT	MQMD, MQPMO, and the first 256 bytes of message data	MQMD, MQPMO, and the first 256 bytes of message data
MQPUT1	MQMD, MQOD, MQPMO, and the first 256 bytes of message data	MQMD, MQOD, MQPMO, and the first 256 bytes of message data
MQSET	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)	Selectors (if *SelectorCount* is greater than 0) Integer attributes (if *IntAttrCount* is greater than 0) Character attributes (if *CharAttrLength* is greater than 0)
MQSTAT	MQSTS	MQSTS
MQSUB	MQSD, MQSD.ObjectString, MQSD.SubName, MQSD.SubUserData, MQSD.SelectionString, MQSD.ResObjectString	MQSD, MQSD.ObjectString, MQSD.SubName, MQSD.SubUserData, MQSD.SelectionString, MQSD.ResObjectString
MQSUBRQ	MQSRO	MQSRO

Note: In the special case of an MQGET call with the WAIT option, a double entry is seen if there is no message available at the time of the MQGET request, but a message subsequently becomes available before the expiry of any time interval specified.

This is because, although the application has issued a single MQGET call, the adapter is performing the wait on behalf of the application and when a message becomes available it reissues the call. So in the trace it appears as a second MQGET call.

Information about specific fields of the queue request parameter list is also produced in some circumstances. The fields in this list are identified as follows:

Identifier	Description
Action	Requested action
BufferL	Buffer length
CBD	Address of callback descriptor
CompCode	Completion code

Identifier	Description
CharAttL	Character attributes length
DataL	Data length
Hobj	Object handle
Hsub	Subscription handle
IntAttC	Count of integer attributes
pObjDesc	Object descriptor
Oper	Operation
Options	Options
pBuffer	Address of buffer
pCharAtt	Address of character attributes
pCTLO	Address of control callback options
pECB	Address of ECB used in get
pGMO	Address of get message options
pIntAtt	Address of integer attributes
pMsgDesc	Address of message descriptor
pPMO	Address of put message options
pSD	Address of subscription descriptor
pSelect	Address of selectors
pSRQOpt	Address of subscription request options
pSTS	Address of status structure
Reason	Reason code
RSVn	Reserved for IBM
SelectC	Selector count
Thread	Thread
Type	Requested type
UOWInfo	Information about the unit of work
Userid	CICS or IMS user ID, for batch or TSO this is zero

Other types of trace

There are other trace facilities available for problem determination. Use this topic to investigate channel initiator trace, line trace, CICS adapter trace, SSL trace, and z/OS trace.

It can be helpful to use the following trace facilities with IBM MQ.

- The channel initiator trace
- The line trace
- The CICS adapter trace
- System SSL trace

- z/OS z/OS traces

The channel initiator trace

See Figure 15 on page 402 for information about how to get a dump of the channel initiator address space. Note that dumps produced by the channel initiator do not include trace data space. The trace data space, which is called CSQXTRDS, contains trace information. You can request this by specifying it on a slip trap or when you use the dump command.

You can run the trace using the START TRACE command. You can also set this trace to start automatically using the TRAXSTR queue manager attribute. For more information about how to do this, see ALTER QMGR.

You can display this trace information by entering the IPCS command:

```
LIST 1000. DSPNAME(CSQXTRDS)
```

You can format it using the command:

```
CTRACE COMP(CSQXssnm)
```

where *ssnm* is the subsystem name.

The line trace

A wrap-around line trace exists for each channel. This trace is kept in a 4 KB buffer for each channel in the channel initiator address space. Trace is produced for each channel, so it is ideal for problems where a channel appears to be hung, because information can be collected about the activity of this channel long after the normal trace has wrapped.

The line trace is always active; you cannot turn it off. It is available for both LU 6.2 and TCP channels and should reduce the number of times a communications trace is required.

You can view the trace as unformatted trace that is written to CSQSNAP. You can display the trace by following these steps:

1. Ensure that the CHIN procedure has a SNAP DD statement.
2. Start a CHIN trace, specifying IFCID 202 as follows:

   ```
   START TRACE(CHINIT) CLASS(4) IFCID(202)
   ```

3. Display the channel status for those channels for which the line trace is required:

   ```
   DISPLAY CHSTATUS(channel) SAVED
   ```

   This dumps the current line for the selected channels to CSQSNAP. See “Snap dumps” on page 417 for further information.

   Note:

   a. The addresses of the storage dump are incorrect because the CSQXFFST mechanism takes a copy of the storage before writing it to CSQSNAP.
   b. The dump to CSQSNAP is only produced the first time you run the DISPLAY CHSTATUS SAVED command. This is to prevent getting dumps each time you run the command.

      To obtain another dump of line trace data, you must stop and restart the current trace.

i) You can use a selective STOP TRACE command to stop just the trace that was started to gather the line trace data. To do this, note the TRACE NUMBER assigned to the trace as shown in this example:

```
+ssid START TRACE(CHINIT) CLASS(4) IFCID(202)
     CSQW130I +ssid 'CHINIT' TRACE STARTED, ASSIGNED TRACE NUMBER 01
```

ii) To stop the trace, issue the following command:

```
+ssid STOP TRACE(CHINIT) TNO(01)
```

iii) You can then enter another START TRACE command with a DISPLAY CHSTATUS SAVED command to gather more line trace data to CSQSNAP.

4. The line trace buffer is unformatted. Each entry starts with a clock, followed by a time stamp, and an indication of whether this is an OUTBOUND or INBOUND flow. Use the time stamp information to find the earliest entry.

The CICS adapter trace

The CICS adapter writes entries to the CICS trace if your trace number is set to a value in the range 0 through 199 (decimal), and if either:

- CICS user tracing is enabled, or
- CICS internal/auxiliary trace is enabled

You can enable CICS tracing in one of two ways:

- Dynamically, using the CICS-supplied transaction CETR
- By ensuring that the USERTR parameter in the CICS system initialization table (SIT) is set to YES

For more information about enabling CICS trace, see the *CICS Problem Determination Guide*.

The CICS trace entry originating from the CICS adapter has a value AP0 *000*, where *000* is the hexadecimal equivalent of the decimal value of the CICS adapter trace number you specified.

The trace entries are shown in “CICS adapter trace entries” on page 369.

System SSL trace

You can collect System SSL trace using the SSL Started Task. The details of how to set up this task are in the *System Secure Sockets Layer Programming* documentation, SC24-5901. A trace file is generated for each SSLTASK running in the CHINIT address space.

z/OS traces

z/OS

z/OS traces, which are common to all products operating as formal subsystems of z/OS, are available for use with IBM MQ. For information about using and interpreting this trace facility, see the *MVS Diagnosis: Tools and Service Aids* manual.

CICS adapter trace entries

Use this topic as a reference for CICS adapter trace entries.

The CICS trace entry for these values is AP0 xxx (where xxx is the hexadecimal equivalent of the trace number you specified when the CICS adapter was enabled). These trace entries are all issued by CSQCTRUE, except CSQCTEST, which is issued by CSQCRST and CSQCDSP.

Table 8. CICS adapter trace entries

Name	Description	Trace sequence	Trace data
CSQCABNT	Abnormal termination	Before issuing END_THREAD ABNORMAL to IBM MQ. This is due to the end of the task and therefore an implicit backout could be performed by the application. A ROLLBACK request is included in the END_THREAD call in this case.	Unit of work information. You can use this information when finding out about the status of work. (For example, it can be verified against the output produced by the DISPLAY THREAD command, or the log print utility.)
CSQCAUID	Bridge security	Before validating bridge user password or PassTicket.	User ID.
CSQCBACK	Syncpoint backout	Before issuing BACKOUT to IBM MQ. This is due to an explicit backout request from the application.	Unit of work information.
CSQCCONX	MQCONNX	Before issuing `MQCONNX` to IBM MQ.	Connection tag.
CSQCCCRC	Completion code and reason code	After unsuccessful return from API call.	Completion code and reason code.
CSQCCOMM	Syncpoint commit	Before issuing COMMIT to IBM MQ. This can be due to a single-phase commit request or the second phase of a two-phase commit request. The request is due to a explicit syncpoint request from the application.	Unit of work information.
CSQCDCFF	IBM use only		
CSQCDCIN	IBM use only		
CSQCDCOT	IBM use only		
CSQCEXER	Execute resolve	Before issuing EXECUTE_RESOLVE to IBM MQ.	The unit of work information of the unit of work issuing the EXECUTE_RESOLVE. This is the last in-doubt unit of work in the resynchronization process.
CSQCGETW	GET wait	Before issuing CICS wait.	Address of the ECB to be waited on.
CSQCGMGD	GET message data	After successful return from `MQGET` .	Up to 40 bytes of the message data.
CSQCGMGH	GET message handle	Before issuing `MQGET` to IBM MQ.	Object handle.
CSQCGMGI	Get message ID	After successful return from `MQGET` .	Message ID and correlation ID of the message.
CSQCHCER	Hconn error	Before issuing any MQ verb.	Connection handle.
CSQCINDL	In-doubt list	After successful return from the second INQUIRE_INDOUBT.	The in-doubt units of work list.
CSQCINDO	IBM use only		

Table 8. CICS adapter trace entries (continued)

Name	Description	Trace sequence	Trace data
CSQCINDS	In-doubt list size	After successful return from the first INQUIRE_INDOUBT and the in-doubt list is not empty.	Length of the list; divided by 64 gives the number of in-doubt units of work.
CSQCINDW	Syncpoint in doubt	During syncpoint processing, CICS is in doubt as to the disposition of the unit of work.	Unit of work information.
CSQCINQH	INQ handle	Before issuing `MQINQ` to IBM MQ.	Object handle.
CSQCLOSH	CLOSE handle	Before issuing `MQCLOSE` to IBM MQ.	Object handle.
CSQCLOST	Disposition lost	During the resynchronization process, CICS informs the adapter that it has been cold started so no disposition information regarding the unit of work being resynchronized is available.	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCNIND	Disposition not in doubt	During the resynchronization process, CICS informs the adapter that the unit of work being resynchronized should not have been in doubt (that is, perhaps it is still running).	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCNORT	Normal termination	Before issuing END_THREAD NORMAL to IBM MQ. This is due to the end of the task and therefore an implicit syncpoint commit might be performed by the application. A COMMIT request is included in the END_THREAD call in this case.	Unit of work information.
CSQCOPNH	OPEN handle	After successful return from `MQOPEN` .	Object handle.
CSQCOPNO	OPEN object	Before issuing `MQOPEN` to IBM MQ.	Object name.
CSQCPMGD	PUT message data	Before issuing `MQPUT` to IBM MQ.	Up to 40 bytes of the message data.
CSQCPMGH	PUT message handle	Before issuing `MQPUT` to IBM MQ.	Object handle.
CSQCPMGI	PUT message ID	After successful `MQPUT` from IBM MQ.	Message ID and correlation ID of the message.
CSQCPREP	Syncpoint prepare	Before issuing PREPARE to IBM MQ in the first phase of two-phase commit processing. This call can also be issued from the distributed queuing component as an API call.	Unit of work information.
CSQCP1MD	PUTONE message data	Before issuing `MQPUT1` to IBM MQ.	Up to 40 bytes of data of the message.
CSQCP1MI	PUTONE message ID	After successful return from `MQPUT1` .	Message ID and correlation ID of the message.

Table 8. CICS adapter trace entries (continued)

Name	Description	Trace sequence	Trace data
CSQCP1ON	PUTONE object name	Before issuing MQPUT1 to IBM MQ.	Object name.
CSQCRBAK	Resolved backout	Before issuing RESOLVE_ROLLBACK to IBM MQ.	Unit of work information.
CSQCRCMT	Resolved commit	Before issuing RESOLVE_COMMIT to IBM MQ.	Unit of work information.
CSQCRMIR	RMI response	Before returning to the CICS RMI (resource manager interface) from a specific invocation.	Architected RMI response value. Its meaning depends of the type of the invocation. To determine the type of invocation, look at previous trace entries produced by the CICS RMI component.
CSQCRSYN	Resync	Before the resynchronization process starts for the task.	Unit of work ID known to CICS for the unit of work being resynchronized.
CSQCSETH	SET handle	Before issuing MQSET to IBM MQ.	Object handle.
CSQCTASE	IBM use only		
CSQCTEST	Trace test	Used in EXEC CICS ENTER TRACE call to verify the trace number supplied by the user or the trace status of the connection.	No data.

Tracing TLS: runmqakm, strmqikm, and runmqckm functions

How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

strmqikm and runmqckm trace

To request **strmqikm** tracing, run the **strmqikm** command for your platform with the following -D flags.

On UNIX, Linux, and Windows:

```
strmqikm -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

To request **runmqckm** tracing, run the **runmqckm** command for your platform with the following -D flags.

On UNIX, Linux, and Windows:

```
runmqckm -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

strmqikm and **runmqckm** write three trace files to the directory from which you start them, so consider starting iKeyman or **runmqckm** from the trace directory to which the runtime TLS trace is written: `/var/mqm/trace` on UNIX and Linux systems and *MQ_INSTALLATION_PATH*`/trace` on Windows. *MQ_INSTALLATION_PATH* represents the high-level directory in which IBM MQ is installed.

The trace file generated by **strmqikm** and **runmqckm** has the following format:

```
debugTrace. n
```

where *n* is an incrementing number starting at 0.

runmqakm trace

To request **runmqakm** tracing, run the **runmqakm** command with the following flags:

```
runmqakm -trace filename
```

where *filename* is the name of the trace file to create. You cannot format the **runmqakm** trace file. Send it unchanged to IBM support. The **runmqakm** trace file is a binary file and, if it is transferred to IBM support via FTP, it must be transferred in binary transfer mode.

Runtime TLS trace

On UNIX, Linux, and Windows systems, you can independently request trace information for **strmqikm**, **runmqckm**, the runtime TLS functions, or a combination of these.

The runtime TLS trace files have the names AMQ.TLS.TRC and AMQ.TLS.TRC.1 and the TLS trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format any of the TLS trace files; send them unchanged to IBM support. The TLS trace files are binary files and, if they are transferred to IBM support via FTP, they must be transferred in binary transfer mode.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace on IBM MQ server on IBM i" on page 356
Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related reference

"Using trace on Windows" on page 350
Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

Tracing IBM MQ classes for JMS applications

The trace facility in IBM MQ classes for JMS is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

If you are asked to provide trace output to investigate an issue, use one of the options mentioned below:

- If the issue is easy to recreate, then collect an IBM MQ classes for JMS trace by using a Java System Property. For more information, see "Collecting an IBM MQ classes for JMS trace by using a Java system property" on page 374.
- If an application needs to run for a period of time before the issue occurs, collect an IBM MQ classes for JMS trace by using the IBM MQ classes for JMS configuration file. For more information, see "Collecting an IBM MQ classes for JMS trace by using the IBM MQ classes for JMS configuration file" on page 375.
- To generate a trace from an application that is currently running, collect the IBM MQ classes for JMS trace dynamically by using the traceControl utility. For more information, see "Collecting an IBM MQ classes for JMS trace dynamically by using the traceControl utility " on page 376.

If you are unsure which option to use, contact your IBM Support representative and they will be able to advise you on the best way to collect trace for the issue that you are seeing.

If a severe or unrecoverable error occurs, First Failure Support Technology (FFST) information is recorded in a file with a name of the format `JMSCC` *xxxx*`.FDC` where *xxxx* is a four-digit number. This number is incremented to differentiate `.FDC` files.

`.FDC` files are always written to a subdirectory called `FFDC`. The subdirectory is in one of two locations, depending on whether trace is active:

Trace is active, and *traceOutputName* is set
The FFDC directory is created as a subdirectory of the directory to which the trace file is being written.

Trace is not active or *traceOutputName* is not set
The FFDC directory is created as a subdirectory of the current working directory.

For more information about FFST in IBM MQ classes for JMS, see "FFST: IBM MQ classes for JMS" on page 331.

The JSE common services uses `java.util.logging` as its trace and logging infrastructure. The root object of this infrastructure is the `LogManager`. The log manager has a `reset` method that closes all handlers and sets the log level to `null`, which in effect turns off all the trace. If your application or application server calls `java.util.logging.LogManager.getLogManager().reset()`, it closes all trace, which might prevent you from diagnosing any problems. To avoid closing all trace, create a `LogManager` class with an overridden `reset()` method that does nothing, as shown in the following example:

```
package com.ibm.javaut.tests;
import java.util.logging.LogManager;
public class JmsLogManager extends LogManager {
    // final shutdown hook to ensure that the trace is finally shutdown
    // and that the lock file is cleaned-up
    public class ShutdownHook extends Thread{
        public void run(){
            doReset();
        }
    }
        public JmsLogManager(){
        // add shutdown hook to ensure final cleanup
        Runtime.getRuntime().addShutdownHook(new ShutdownHook());
    }
        public void reset() throws SecurityException {
        // does nothing
    }
    public void doReset(){
        super.reset();
    }
        }
```

The shutdown hook is necessary to ensure that trace is properly shut down when the JVM finishes. To use the modified log manager instead of the default one, add a system property to the JVM startup:

```
java -Djava.util.logging.manager=com. mycompany.logging.LogManager ...
```

Collecting an IBM MQ classes for JMS trace by using a Java system property

For issues that can be reproduced in a short amount of time, IBM MQ classes for JMS trace should be collected by setting a Java system property when starting the application.

About this task

To collect a trace by using a Java system property, complete the following steps.

Procedure

- Run the application that is going to be traced by using the following command:

```
java -Dcom.ibm.msg.client.commonservices.trace.status=ON application_name
```

By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in:

- For IBM MQ classes for JMS for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_`*`%PID%`*`.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.cl`*`%u`*`.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.cl`*`%u`*`.trc`.

where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.

The application stops writing information to the trace file when it is stopped.

If the application has to run for a long period of time before the issue that the trace is being collected for occurs, then the trace file could potentially be very large. In this situation, consider collecting trace by using the IBM MQ classes for JMS configuration file (see "Collecting an IBM MQ classes for JMS trace by using the IBM MQ classes for JMS configuration file" on page 375). When enabling trace in this way, it is possible to control the amount of trace data that the IBM MQ classes for JMS generate.

Collecting an IBM MQ classes for JMS trace by using the IBM MQ classes for JMS configuration file

If an application must run for a long period of time before an issue occurs, IBM MQ classes for JMS trace should be collected by using the IBM MQ classes for JMS configuration file. The configuration file allows you to specify various options to control the amount of trace data that is collected.

About this task

To collect a trace by using the IBM MQ classes for JMS configuration file, complete the following steps.

Procedure

1. Create an IBM MQ classes for JMS configuration file.

   For more information about this file, see The IBM MQ classes for JMS configuration file.
2. Edit the IBM MQ classes for JMS configuration file so that the property **`com.ibm.msg.client.commonservices.trace.status`** is set to the value `ON`.
3. Optional: Edit the other properties that are listed in the IBM MQ classes for JMS configuration file Java Standard Edition Trace Settings.
4. Run the IBM MQ classes for JMS application by using the following command:

   ```
   java -Dcom.ibm.msg.client.config.location=config_file_url
   application_name
   ```

   where *config_file_url* is a uniform resource locator (URL) that specifies the name and location of the IBM MQ classes for JMS configuration file. URLs of the following types are supported: `http`, `file`, `ftp`, and `jar`.

Here is an example of a Java command:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myjms.config
MyAppClass
```

This command identifies the IBM MQ classes for JMS configuration file as the file `D:\mydir\myjms.config` on the local Windows system.

By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in:

- For IBM MQ classes for JMS for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_%PID%.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_%PID%.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_%PID%.cl%u.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.cl%u.trc`.

where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.

To change the name of the trace file, and the location where it is written, ensure that the IBM MQ classes for JMS configuration file that the application uses contains an entry for the property **com.ibm.msg.client.commonservices.trace.outputName**. The value for the property can be either of the following:

- The name of the trace file that is created in the application's working directory.
- The fully qualified name of the trace file, including the directory in which the file is created.

For example, to configure the IBM MQ classes for JMS to write trace information for an application to a file called `C:\Trace\trace.trc`, the IBM MQ classes for JMS configuration file that the application uses needs to contain the following entry:

```
com.ibm.msg.client.commonservices.trace.outputName=C:\Trace\trace.trc
```

Collecting an IBM MQ classes for JMS trace dynamically by using the traceControl utility

The traceControl utility that is shipped with the IBM MQ classes for JMS allows trace to be collected from a running application. This can be very useful if IBM Support need to see a trace from an application once an issue has occurred, or if trace needs to be collected from a critical application that cannot be stopped.

About this task

For more information about the traceControl utility, see "Controlling trace in a running process by using IBM MQ classes for Java and IBM MQ classes for JMS" on page 386.

To collect a trace by using the traceControl utility, complete the following steps.

Procedure

1. Bring up a command prompt, and navigate to the directory *MQ_INSTALLATION_PATH*`\java\lib`.
2. Run the command:

   ```
   java -jar com.ibm.mq.traceControl -list
   ```

   This command brings up a list of all of the Java processes on the system.
3. Identify the process identifier for the IBM MQ classes for JMS application that needs to be traced, and run the command:

   ```
   java -jar com.ibm.mq.traceControl -i process identifier -enable
   ```

   Trace is now turned on for the application.

   By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in:

   - For IBM MQ classes for JMS for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_`*`%PID%`*`.trc`.
   - V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.trc`.
   - V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.trc`.
   - V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the JAR file `com.ibm.mqjms.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.cl`*`%u`*`.trc`.
   - V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.cl`*`%u`*`.trc`.

   where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.
4. To turn trace off, run the command:

   ```
   java -jar com.ibm.mq.traceControl -i process identifier -disable
   ```

Tracing IBM MQ classes for Java applications

The trace facility in IBM MQ classes for Java is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue, use one of the options mentioned below:

- If the issue is easy to recreate, then collect an IBM MQ classes for Java trace by using a Java System Property. For more information, see "Collecting an IBM MQ classes for Java trace by using a Java system property" on page 378.
- If an application needs to run for a period of time before the issue occurs, collect an IBM MQ classes for Java trace by using the IBM MQ classes for Java configuration file. For more information, see "Collecting an IBM MQ classes for Java trace by using the IBM MQ classes for Java configuration file" on page 379.
- To generate a trace from an application that is currently running, collect the IBM MQ classes for Java trace dynamically by using the traceControl utility. For more information, see "Collecting an IBM MQ classes for Java trace dynamically by using the traceControl utility " on page 380.

If you are unsure which option to use, contact your IBM Support representative and they will be able to advise you on the best way to collect trace for the issue you are seeing.

If a severe or unrecoverable error occurs, First Failure Support Technology (FFST) information is recorded in a file with a name of the format `JAVACC` *xxxx*`.FDC` where *xxxx* is a four-digit number. It is incremented to differentiate `.FDC` files.

`.FDC` files are always written to a subdirectory called `FFDC`. The subdirectory is in one of two locations, depending on whether trace is active:

Trace is active, and *traceOutputName* is set
The FFDC directory is created as a subdirectory of the directory to which the trace file is being written.

Trace is not active or *traceOutputName* is not set
The FFDC directory is created as a subdirectory of the current working directory.

The JSE common services uses `java.util.logging` as its trace and logging infrastructure. The root object of this infrastructure is the `LogManager`. The log manager has a `reset` method, which closes all handlers and sets the log level to `null`, which in effect turns off all the trace. If your application or application server calls `java.util.logging.LogManager.getLogManager().reset()`, it closes all trace, which might prevent you from diagnosing any problems. To avoid closing all trace, create a `LogManager` class with an overridden `reset()` method that does nothing, as in the following example.

```
package com.ibm.javaut.tests;
import java.util.logging.LogManager;
public class JmsLogManager extends LogManager {
        // final shutdown hook to ensure that the trace is finally shutdown
        // and that the lock file is cleaned-up
        public class ShutdownHook extends Thread{
                public void run(){
                        doReset();
                }
        }
                public JmsLogManager(){
                // add shutdown hook to ensure final cleanup
                Runtime.getRuntime().addShutdownHook(new ShutdownHook());
        }
                public void reset() throws SecurityException {
                // does nothing
        }
        public void doReset(){
                super.reset();
        }
        }
```

The shutdown hook is necessary to ensure that trace is properly shutdown when the JVM finishes. To use the modified log manager instead of the default one, add a system property to the JVM startup:

```
java -Djava.util.logging.manager=com. mycompany.logging.LogManager ...
```

Collecting an IBM MQ classes for Java trace by using a Java system property

For issues that can be reproduced in a short amount of time, IBM MQ classes for Java trace should be collected by setting a Java system property when starting the application.

About this task

To collect a trace by using a Java system property, complete the following steps.

Procedure

- Run the application that is going to be traced by using the following command:

  ```
  java -Dcom.ibm.msg.client.commonservices.trace.status=ON application_name
  ```

By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in:

- For IBM MQ classes for Java for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_%PID%.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.cl%u.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.cl%u.trc`.

where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.

The application stops writing information to the trace file when it is stopped.

If the application has to run for a long period of time before the issue that the trace is being collected for occurs, then the trace file could potentially be very large. In this situation, consider collecting trace by using the IBM MQ classes for Java configuration file (see "Collecting an IBM MQ classes for Java trace by using the IBM MQ classes for Java configuration file" on page 379). When enabling trace in this way, it is possible to control the amount of trace data that the IBM MQ classes for Java generates.

Collecting an IBM MQ classes for Java trace by using the IBM MQ classes for Java configuration file

If an application must run for a long period of time before an issue occurs, IBM MQ classes for Java trace should be collected by using the IBM MQ classes for Java configuration file. The configuration file allows you to specify various options to control the amount of trace data that is collected.

About this task

To collect a trace by using the IBM MQ classes for Java configuration file, complete the following steps.

Procedure

1. Create an IBM MQ classes for Java configuration file.

   For more information about this file, see The IBM MQ classes for Java configuration file.
2. Edit the IBM MQ classes for Java configuration file so that the property **`com.ibm.msg.client.commonservices.trace.status`** is set to the value `ON`.
3. Optional: Edit the other properties that are listed in the IBM MQ classes for Java configuration file Java Standard Edition Trace Settings.
4. Run the IBM MQ classes for Java application by using the following command:

   ```
   java -Dcom.ibm.msg.client.config.location=config_file_url
   application_name
   ```

   where *config_file_url* is a uniform resource locator (URL) that specifies the name and location of the IBM MQ classes for Java configuration file. URLs of the following types are supported: `http`, `file`, `ftp`, and `jar`.

Here is an example of a Java command:

```
java -Dcom.ibm.msg.client.config.location=file:/D:/mydir/myJava.config
MyAppClass
```

This command identifies the IBM MQ classes for Java configuration file as the file `D:\mydir\myJava.config` on the local Windows system.

By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in:

- For IBM MQ classes for Java for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_`*`%PID%`*`.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.trc`.
- V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_`*`%PID%`*`.cl`*`%u`*`.trc`.
- V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_`*`%PID%`*`.cl`*`%u`*`.trc`.

where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.

To change the name of the trace file, and the location where it is written, ensure that the IBM MQ classes for Java configuration file that the application uses contains an entry for the property **`com.ibm.msg.client.commonservices.trace.outputName`**. The value for the property can be either of the following:

- The name of the trace file that is created in the application's working directory.
- The fully qualified name of the trace file, including the directory in which the file is created.

For example, to configure the IBM MQ classes for Java to write trace information for an application to a file called `C:\Trace\trace.trc`, the IBM MQ classes for Java configuration file that the application uses needs to contain the following entry:

```
com.ibm.msg.client.commonservices.trace.outputName=C:\Trace\trace.trc
```

Collecting an IBM MQ classes for Java trace dynamically by using the traceControl utility

The traceControl utility that is shipped with the IBM MQ classes for Java allows trace to be collected from a running application. This can be very useful if IBM Support need to see a trace from an application once an issue has occurred, or if trace needs to be collected from a critical application that cannot be stopped.

About this task

For more information about the traceControl utility, see "Controlling trace in a running process by using IBM MQ classes for Java and IBM MQ classes for JMS" on page 386.

To collect a trace by using the traceControl utility, complete the following steps.

Procedure

1. Bring up a command prompt, and navigate to the directory `MQ_INSTALLATION_PATH\java\lib`.
2. Run the command:

   ```
   java -jar com.ibm.mq.traceControl -list
   ```

   This command brings up a list of all of the Java processes on the system.
3. Identify the process identifier for the IBM MQ classes for Java application that needs to be traced, and run the command:

   ```
   java -jar com.ibm.mq.traceControl -i process identifier -enable
   ```

   Trace is now turned on for the application.

   By default, trace information is written to a trace file in the current working directory of the application. The name of the trace file depends upon the environment that the application is running in: :

   - For IBM MQ classes for Java for Version 8.0.0, Fix Pack 6 or earlier, trace is written to a file called `mqjms_%PID%.trc`.
   - V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.trc`.
   - V 8.0.0.7 From Version 8.0.0, Fix Pack 7, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.trc`.
   - V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the JAR file `com.ibm.mq.jar`, trace is written to a file called `mqjava_%PID%.cl%u.trc`.
   - V 8.0.0.15 From Version 8.0.0, Fix Pack 15, if the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.allclient.jar`, trace is written to a file called `mqjavaclient_%PID%.cl%u.trc`.

   where *%PID%* is the process identifier of the application that is being traced, and *%u* is a unique number to differentiate files between threads running trace under different Java classloaders.
4. To turn trace off, run the command:

   ```
   java -jar com.ibm.mq.traceControl -i process identifier -disable
   ```

Tracing the IBM MQ Resource Adapter

The ResourceAdapter object encapsulates the global properties of the IBM MQ resource adapter. To enable trace of the IBM MQ resource adapter, properties need to be defined in the ResourceAdapter object.

The ResourceAdapter object has two sets of properties:

- Properties associated with diagnostic tracing
- Properties associated with the connection pool managed by the resource adapter

The way you define these properties depends on the administration interfaces provided by your application server.

Table 9 on page 382 lists the properties of the ResourceAdapter object that are associated with diagnostic tracing.

Table 9. Properties of the ResourceAdapter object that are associated with diagnostic tracing

Name of property	Type	Default value	Description
traceEnabled	String	false	A flag to enable or disable diagnostic tracing. If the value is false, tracing is turned off.
traceLevel	String	3	The level of detail in a diagnostic trace. The value can be in the range 0, which produces no trace, to 10, which provides the most detail. See Table 10 on page 382 for a description of each level. If trace is enabled, traceLevel should be set to the value 10, unless otherwise specified by IBM Support.
logWriterEnabled	String	true	A flag to enable or disable the sending of a diagnostic trace to a LogWriter object provided by the application server. If the value is true, the trace is sent to a LogWriter object. If the value is false, any LogWriter object provided by the application server is not used.

Table 10 on page 382 describes the levels of detail for diagnostic tracing.

Table 10. The levels of detail for diagnostic tracing

Level number	Level of detail
0	No trace.
1	The trace contains error messages.
3	The trace contains error and warning messages.
6	The trace contains error, warning, and information messages.
8	The trace contains error, warning, and information messages, and entry and exit information for methods.
9	The trace contains error, warning, and information messages, entry and exit information for methods, and diagnostic data.
10	The trace contains all trace information.

Note: Any level that is not included in this table is equivalent to the next lowest level. For example, specifying a trace level of 4 is equivalent to specifying a trace level of 3. However, the levels that are not included might be used in future releases of the IBM MQ resource adapter, so it is better to avoid using these levels.

If diagnostic tracing is turned off, error and warning messages are written to the system error stream. If diagnostic tracing is turned on, error messages are written to the system error stream and to the trace destination, but warning messages are written only to the trace destination. However, the trace contains warning messages only if the trace level is 3 or higher. By default, the trace destination is the current working directory, but if the logWriterEnabled property is set, the trace is sent to the application server.

In general, the ResourceAdapter object requires no administration. However, to enable diagnostic tracing on UNIX and Linux systems for example, you can set the following properties:

```
traceEnabled:    true
traceLevel:      10
```

These properties have no effect if the resource adapter has not been started, which is the case, for example, when applications using IBM MQ resources are running only in the client container. In this situation, you can set the properties for diagnostic tracing as Java Virtual Machine (JVM) system properties. You can set the properties by using the -D flag on the **java** command, as in the following example:

```
java ... -DtraceEnabled=true -DtraceLevel=10
```

You do not need to define all the properties of the ResourceAdapter object. Any properties left unspecified take their default values. In a managed environment, it is better not to mix the two ways of specifying properties. If you do mix them, the JVM system properties take precedence over the properties of the ResourceAdapter object.

Tracing additional IBM MQ Java components

For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Diagnostic information in this context consists of trace, first-failure data capture (FFDC) and error messages.

You can choose to have this information produced using IBM MQ facilities or the facilities of IBM MQ classes for Java or IBM MQ classes for JMS, as appropriate. Generally use the IBM MQ diagnostic facilities if they are available on the local system.

You might want to use the Java diagnostics in the following circumstances:

- On a system on which queue managers are available, if the queue manager is managed separately from the software you are running.
- To reduce performance effect of IBM MQ trace.

To request and configure diagnostic output, two system properties are used when starting an IBM MQ Java process:

- System property com.ibm.mq.commonservices specifies a standard Java property file, which contains a number of lines which are used to configure the diagnostic outputs. Each line of code in the file is free-format, and is terminated by a new line character.
- System property com.ibm.mq.commonservices.diagid associates trace and FFDC files with the process which created them.

For information about using the com.ibm.mq.commonservices properties file to configure diagnostics information, see "Using com.ibm.mq.commonservices" on page 384.

For instructions on locating trace information and FFDC files, see "Java trace and FFDC files" on page 385.

Related concepts

"Using trace on UNIX and Linux systems" on page 352
Use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file

"Using trace on IBM MQ server on IBM i" on page 356
Use the TRCMQM command to start and stop tracing and specify the type of trace that you require.

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"Tracing TLS: runmqakm, strmqikm, and runmqckm functions" on page 372
How to trace Transport Layer Security (TLS), and request **runmqakm** tracing and **strmqikm** (iKeyman) and **runmqckm** (iKeycmd) tracing.

Related reference

"Using trace on Windows" on page 350

Use the **strmqtrc** and **endmqtrc** commands or the MQ Explorer interface to start and end tracing.

Using com.ibm.mq.commonservices

The com.ibm.mq.commonservices properties file contains the following entries relating to the output of diagnostics from the Java components of IBM MQ.

Note that case is significant in all these entries:

Diagnostics.Java= *options*
Which components are traced using Java trace. Options are one or more of *explorer*, *soap*, and *wmqjavaclasses*, separated by commas, where "explorer" refers to the diagnostics from the IBM MQ Explorer, "soap" refers to the diagnostics from the running process within IBM MQ Transport for SOAP, and "wmqjavaclasses" refers to the diagnostics from the underlying IBM MQ Java classes. By default no components are traced.

Diagnostics.Java.Trace.Detail= *high|medium|low*
Detail level for Java trace. The *high* and *medium* detail levels match those used in IBM MQ tracing but *low* is unique to Java trace. This property is ignored if Diagnostics.Java is not set. The default is *medium*.

Diagnostics.Java.Trace.Destination.File= *enabled|disabled*
Whether Java trace is written to a file. This property is ignored if Diagnostics.Java is not set. The default is *disabled*.

Diagnostics.Java.Trace.Destination.Console= *enabled|disabled*
Whether Java trace is written to the system console. This property is ignored if Diagnostics.Java is not set. The default is *disabled*.

Diagnostics.Java.Trace.Destination.Pathname= *dirname*
The directory to which Java trace is written. This property is ignored if Diagnostics.Java is not set or Diagnostics.Java.Trace.Destination.File=disabled. On UNIX and Linux systems, the default is /var/mqm/trace if it is present, otherwise the Java console (System.err). On Windows, the default is the system console.

Diagnostics.Java.FFDC.Destination.Pathname= *dirname*
The directory to which Java FFDC output is written. The default is the current working directory.

Diagnostics.Java.Errors.Destination.Filename= *filename*
The fully qualified file name to which Java error messages are written. The default is AMQJAVA.LOG in the current working directory.

An example of a com.ibm.mq.commonservices properties file is given in Figure 11 on page 385. Lines beginning with the number sign (#) are treated as comments.

```
#
# Java diagnostics for WebSphere MQ Transport for SOAP
# and the WebSphere MQ Java Classes are both enabled
#
Diagnostics.Java=soap,wmqjavaclasses
#
# High detail Java trace
#
Diagnostics.Java.Trace.Detail=high
#
# Java trace is written to a file and not to the console.
#
Diagnostics.Java.Trace.Destination.File=enabled
Diagnostics.Java.Trace.Destination.Console=disabled
#
# Directory for Java trace file
#
Diagnostics.Java.Trace.Destination.Pathname=c:\\tracedir
#
# Directory for First Failure Data Capture
#
Diagnostics.Java.FFDC.Destination.Pathname=c:\\ffdcdir
#
# Directory for error logging
#
Diagnostics.Java.Errors.Destination.Filename=c:\\errorsdir\\SOAPERRORS.LOG
#
```

Figure 11. Sample com.ibm.mq.commonservices properties file

A sample properties file, WMQSoap_RAS.properties, is also supplied as part of the " Java messaging and SOAP transport" install option.

Java trace and FFDC files

File name conventions for Java trace and FFDC files.

When Java trace is generated for IBM MQ Transport for SOAP, it is written to a file with a name of the format `AMQ. *diagid*. *counter*.TRC`. Here, *diagid* is the value of the system property com.ibm.mq.commonservices.diagid associated with this Java process, as described earlier in this section, and *counter* is an integer greater than or equal to 0. All letters in the name are in uppercase, matching the naming convention used for normal IBM MQ trace.

If com.ibm.mq.commonservices.diagid is not specified, the value of *diagid* is the current time, in the format `YYYYMMDDhhmmssmmm`.

When Java trace is generated for the MQ Explorer, it is written to file with a name of the format `AMQYYYYMMDDHHmmssmmm.TRC.n`. Each time MQ Explorer trace is run, the trace facility renames all previous trace files by incrementing the file suffix `.n` by one. The trace facility then creates a new file with the suffix `.0` that is always the latest.

The IBM MQ Java classes trace file has a name based on the equivalent IBM MQ Transport for SOAP Java trace file. The name differs in that it has the string .JC added before the .TRC string, giving a format of `AMQ. *diagid*. *counter*.JC.TRC`.

When Java FFDC is generated for the MQ Explorer or for IBM MQ Transport for SOAP, it is written to a file with a name of the format `AMQ. *diagid*. *counter*.FDC` where *diagid* and *counter* are as described for Java trace files.

Java error message output for the MQ Explorer and for IBM MQ Transport for SOAP is written to the file specified by *Diagnostics.Java.Errors.Destination.Filename* for the appropriate Java process. The format of these files matches closely the format of the standard IBM MQ error logs.

When a process is writing trace information to a file, it appends to a single trace output file for the lifetime of the process. Similarly, a single FFDC output file is used for the lifetime of a process.

All trace output is in the UTF-8 character set.

Controlling trace in a running process by using IBM MQ classes for Java and IBM MQ classes for JMS

The IBM MQ classes for Java and IBM MQ classes for JMS register a Standard MBean that allows suitable Java Management Extensions (JMX) tools to control certain aspects of trace behavior for a client process.

Principles

As an alternative to the well-known general-purpose tools like `jconsole` you can use a command-line tool in the form of an executable JAR file to access these facilities.

The JAR file is called `com.ibm.mq.traceControl.jar` and is stored in the `java/lib` subdirectory of the IBM MQ installation.

Note: Depending on configuration, JMX tools can be used either locally (on the same system as the process) or remotely. The local case is discussed initially.

Finding the process

To control a process, you must establish a JMX connection it. To control a process locally, you must specify its identifier.

To display a summary of running Java processes with their identifiers, run the executable JAR file with the option `-list`. This option produces a list of identifiers and descriptions for the processes that are found.

Examining trace status

When you have found the identifier for the relevant process, run the executable JAR file with the options `-i identifier -status`, where 'identifier' is the identifier of the process you want to change. These options display the status, either `enabled` or `disabled` , for the process, and the information about where the process is running, the name of the trace file, and a tree that represents the inclusion and exclusion of packages in trace.

Enabling and disabling trace

To enable trace for a process, run the executable JAR file with the options `-i identifier -enable`.

To disable trace for a process, run the executable JAR file with the options `-i identifier -disable`.

Note: You can choose only one option from the set `-status, -enable`, and `-disable`.

Including and excluding packages

To include a package in trace for a process, run the executable JAR file with the options `-i identifier -ip` *package_name*, where *package_name* is the name of your package.

To exclude a package from trace for a process, run the executable JAR file with the options `-i identifier -ep` *package_name*.

Note: You can use multiple `-ip` and `-ep` options. These options are not checked for consistency.

When you specify a package for exclusion or inclusion, the handling of packages that have matching prefixes is not affected. For example, excluding the package com.ibm.mq.jms from trace would not exclude com.ibm.mq, com.ibm.msq.client.jms, or com.ibm.mq.remote.api, but it would exclude com.ibm.mq.jms.internal.

```
C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -list
10008 : 'MQSample'
9004 : ' MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -list'

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : false
User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
```

```
Trace File Name : mqjms.trc
Package Include/Exclude tree
root - Included

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -enable
Enabling trace
Tracing enabled : true

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : true
User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
Trace File Name : mqjms_10008.cl0.trc
Package Include/Exclude tree
root - Included

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -ip
com.ibm.mq.jms
Adding 'com.ibm.mq.jms' to the list of packages included in trace


C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : true
User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
Trace File Name : mqjms_10008.cl0.trc
Package Include/Exclude tree
root - Included
com - Included
ibm - Included
mq - Included
jms - Included

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -ip
com.acme.banana -ep com.acme.banana.split -ip com.acme.banana.shake
Adding 'com.acme.banana' to the list of packages included in trace
Adding 'com.acme.banana.shake' to the list of packages included in trace
Adding 'com.acme.banana.split' to the list of packages excluded from trace

C:>java -jar MQ_INSTALLATION_PATH/java/lib/com.ibm.mq.traceControl.jar -i 10008 -status
Tracing enabled : true User Directory : C:\Users\IBM_ADMIN\RTCworkspace\sandpit
Trace File Name : mqjms_10008.cl0.trc
Package Include/Exclude tree
root - Included
com - Included
acme - Included
banana - Included
shake - Included
split - Excluded
ibm - Included
mq - Included
jms - Included
```

The package inclusion-exclusion tree

The tracing mechanism for IBM MQ classes for Java and IBM MQ classes for JMS tracks the inclusion and exclusion of packages by means of a tree structure, starting from a root node. In the tree structure each node represents one element of a package name, identified by the package name element and containing a trace status which can be either `Included` or `Excluded`. For example the package com.ibm.mq would be represented by three nodes identified by the strings `com`, `ibm`, and `mq`.

Initially, the tree usually contains entries to include most packages, but the header and pcf packages are excluded as they generate a lot of noise. So the initial tree will look something like

```
root - Included
com - Included
ibm - Included
mq - Included
headers - Excluded
pcf - Excluded
```

When the trace facility is determining whether to include or exclude a package, it matches leading portions of the package name to the nodes in the tree as far as possible and takes the status of the last matched node. At the initial state of the tree, the packages com.ibm.msg.client and com.ibm.mq.jms would be included, as the last nodes in the tree that matches them (com->ibm and com->ibm->mq respectively) are marked as *Included*. Conversely, the package com.ibm.headers.internal would be excluded as the last matching node in the tree (com->ibm->mq->headers) is marked as *Excluded*.

As further changes are made to the tree by using the `com.ibm.mq.TraceControl.jar`, it is important to remember that inclusion or exclusion only affects a package and child packages. So, given the initial state that is shown previously, specifying `-ep com.ibm.mq.jms`, would update the tree to look like:

```
root - Included
com - Included
ibm - Included
mq - Included
headers - Excluded
jms - Excluded
pcf - Excluded
```

Which would exclude packages com.ibm.mq.jms, and com.ibm.mq.jms.internal, without affecting packages outside the com.ibm.mq.jms.* hierarchy.

If `-ip com.ibm.mq.jms.admin` is specified next, the tree would look like:

```
root - Included
com - Included
ibm - Included
mq - Included
headers - Excluded
jms - Excluded
admin - Included
pcf - Excluded
```

Which would still exclude packages com.ibm.mq.jms, com.ibm.mq.jms.internal, but now the packages com.ibm.mq.jms.admin, and com.ibm.mq.jms.admin.internal are included in trace.

Connecting remotely

You can connect remotely only if the process was started with a JMX agent that is enabled for remote connection, and that uses the `-Dcom.sun.management.jmxremote.port=port_number` system setting.

After you have started with this system setting, you can run the executable JAR file with the options `-h` *`host_name`* `-p` *`port_number`* in place of the `-i` *`identifier`* option, where *host_name* is the name of the host you wish to connect to and *port_number* is the name of the port to be used.

Note: You must ensure that you take appropriate steps to minimize security risks by enabling SSL for the connection. See the Oracle documentation on JMX for further details https://www.oracle.com.

Limitations

The following limitations exist:

- For non-IBM JVMs, the tool must be started with `tools.jar` added to its class path. The command that is on these platforms is :

  ```
  java -cp <MQ_INSTALL_DIR>/java/lib/com.ibm.mq.traceControl.jar;<JAVA_HOME>/lib/tools.jar
  com.ibm.msg.client.commonservices.trace.TraceController
  ```

- Local attach is controlled by user ID. The tool must be run under the same ID as the process that is to be controlled.

V 8.0.0.14 Enabling dynamic tracing of LDAP client library code

From IBM MQ 8.0.0, Fix Pack 14, it is possible to switch LDAP client trace on and off without also stopping or starting the queue manager.

About this task

Before Version 8.0.0, Fix Pack 14, it was not possible to switch the LDAP client trace on and off without also stopping or starting the queue manager.

From Version 8.0.0, Fix Pack 14, you can switch LDAP client trace on with the **strmqtrc** command and off with the **endmqtrc** command without needing to stop or start the queue manager. To enable this behavior, it is also necessary to set an environment variable **AMQ_LDAP_TRACE** to a non-null value.

When the **AMQ_LDAP_TRACE** is set to a non-null value, and the LDAP functionality is used, some queue manager processes create zero length files under `/var/mqm/trace`. When the trace is then switched on using the **strmqtrc** command, some trace information is written to these files. Later, when trace is switched off with the **endmqtrc** command, trace information ceases to be written to the files, but handles to the files remain open until the queue manager ends.

UNIX On UNIX platforms, filesystem space cannot be released completely simply by unlinking these files with the **rm** command. This is a side-effect from the fact that the handles remain open. Therefore, a queue manager end should be performed, whenever disk space in `/var/mqm/trace` needs to be released.

Procedure

- Set the environment variable AMQ_LDAP_TRACE to a non-null value.
- Use the **strmqtrc** command to switch the trace on:

```
strmqtrc -m QMNAME -t servicedata
```

- Use the **endmqtrc** command to switch the trace off.

z/OS Problem determination on z/OS

IBM MQ for z/OS, CICS, Db2, and IMS produce diagnostic information which can be used for problem determination.

This section contains information about the following topics:

- The recovery actions attempted by the queue manager when a problem is detected.
- IBM MQ for z/OS abends, and the information produced when an abend occurs.
- The diagnostic information produced by IBM MQ for z/OS, and additional sources of useful information.

The type of information provided to help with problem determination and application debugging depends on the type of error encountered, and the way your subsystem is set up.

See the following links for more information about problem determination and diagnostic information on IBM MQ for z/OS:

- "IBM MQ for z/OS performance constraints" on page 390
- "IBM MQ for z/OS recovery actions" on page 392
- "IBM MQ for z/OS abends" on page 393
- "Diagnostic information produced on IBM MQ for z/OS" on page 396
- "Other sources of information" on page 398
- "Diagnostic aids for CICS" on page 399
- "Diagnostic aids for IMS" on page 399

- “Diagnostic aids for Db2” on page 399
- “IBM MQ for z/OS dumps” on page 400
- “Dealing with performance problems on z/OS” on page 419
- “Dealing with incorrect output” on page 425

Related concepts

“Troubleshooting overview” on page 7
Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

“Using logs” on page 341
There are a variety of logs that you can use to help with problem determination and troubleshooting.

“ First Failure Support Technology (FFST” on page 330
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM support personnel to diagnose the problem.

Related tasks

“Using trace” on page 350
You can use different types of trace to help you with problem determination and troubleshooting.

IBM MQ for z/OS performance constraints

Use this topic to investigate z/OS resources that can cause performance constraints.

There are a number of decisions to be made when customizing IBM MQ for z/OS that can affect the way your systems perform. These decisions include:

- The size and placement of data sets
- The allocation of buffers
- The distribution of queues among page sets, and Coupling Facility structures
- The number of tasks that you allow to access the queue manager at any one time

Log buffer pools

Insufficient log buffers can cause applications to wait until a log buffer is available, which can affect IBM MQ performance. RMF reports might show heavy I/O to volumes that hold log data sets.

There are three parameters you can use to tune log buffers. The most important is OUTBUFF. If the log manager statistic QJSTWTB is greater than 0, increase the size of the log buffer. This parameter controls the number of buffers to be filled before they are written to the active log data sets (in the range 1 - 256). Commits and out-of-syncpoint processing of persistent messages cause log buffers to be written out to the log. As a result this parameter might have little effect except when processing large messages, and the number of commits or out of sync point messages is low. These parameters are specified in the CSQ6LOGP macro (see Using CSQ6LOGP for details), and the significant ones are:

OUTBUFF
: This parameter controls the size of the output buffer (in the range 40 KB through 4000 KB).

WRTHRSH
: This parameter controls the number of buffers to be filled before they are written to the active log data sets (in the range 1 through 256).

You must also be aware of the LOGLOAD parameter of the CSQ6SYSP macro. This parameter specifies the number of log records that are written between checkpoint records. The range is 200 through 16 000 000 but a typical value for a large system is 500 000. If a value is too small you receive frequent checkpoints, which consume processor time and can cause additional disk I/O.

Buffer pool size

There is a buffer pool associated with each page set. You can specify the number of buffers in the buffer pool using the DEFINE BUFFPOOL command.

Incorrect specification of buffer pool size can adversely affect IBM MQ performance. The smaller the buffer pool, the more frequently physical I/O is required. RMF might show heavy I/O to volumes that hold page sets. For buffer pools with only short-lived messages the buffer manager statistics QPSTSLA, QPSTSOS, and QPSTRIO must typically be zero. For other buffer pools, QPSTSOS and QPSTSTLA must be zero.

Distribution of data sets on available DASD

The distribution of page data sets on DASD can have a significant effect on the performance of IBM MQ.

Place log data sets on low usage volumes with log *n* and log *n+1* on different volumes. Ensure that dual logs are placed on DASD on different control units and that the volumes are not on the same physical disk.

Distribution of queues on page sets

The distribution of queues on page sets can affect performance. This change in performance can be indicated by poor response times experienced by transactions using specific queues that reside on heavily used page sets. RMF reports might show heavy I/O to volumes containing the affected page sets.

You can assign queues to specific page sets by defining storage class (STGCLASS) objects specifying a particular page set, and then defining the STGCLASS parameter in the queue definition. It is a good idea to define heavily used queues on different page sets in this way.

Distribution of queues on Coupling Facility structures

The distribution of queues on Coupling Facility structures can affect performance.

A queue-sharing group can connect to up to 64 Coupling Facility structures, one of which must be the administration structure. You can use the remaining 63 Coupling Facility structures for IBM MQ data with each structure holding up to 512 queues. If you need more than one Coupling Facility structure, separate the queues across several structures based on the function of the queue.

There are some steps you can take to maximize efficiency:

- Delete any Coupling Facility structures you no longer require.
- Place all the queues used by an application on the same Coupling Facility to make application processing efficient.
- If work is particularly performance sensitive, choose a faster Coupling Facility structure.

Consider that if you lose a Coupling Facility structure, you lose any non-persistent messages stored in it. The loss of these non-persistent messages can cause consistency problems if queues are spread across various Coupling Facility structures. To use persistent messages, you must define the Coupling Facility structures with at least CFLEVEL(3) and RECOVER(YES).

Limitation of concurrent threads

The number of tasks accessing the queue manager can also affect performance, particularly if there are other constraints, such as storage, or there are many tasks accessing a few queues. The symptoms can be heavy I/O against one or more page sets, or poor response times from tasks known to access the same queues. The number of threads in IBM MQ is limited to 32767 for both TSO and Batch.

In a CICS environment, you can use CICS MAXTASK to limit concurrent access.

Using the IBM MQ trace for administration

Although you might have to use specific traces on occasion, using the trace facility has a negative effect on the performance of your systems.

Consider what destination you want your trace information sent to. Using the internal trace table saves I/O, but it is not large enough for traces that produce large volumes of data.

The statistics trace gathers information at intervals. The intervals are controlled by the STATIME parameter of the CSQ6SYSP macro, described in Using CSQ6SYSP. An accounting trace record is produced when the task or channel ends, which might be after many days.

You can limit traces by class, resource manager identifier (RMID), and instrumentation facility identifier (IFCID) to reduce the volume of data collected. See START TRACE for more information.

z/OS IBM MQ for z/OS recovery actions

Use this topic to understand some of the recovery actions for user detected and queue manager detected errors.

IBM MQ for z/OS can recover from program checks caused by incorrect user data. A completion and reason code are issued to the caller. These codes are documented in IBM MQ for z/OS messages, completion, and reason codes.

Program errors

Program errors might be associated with user application program code or IBM MQ code, and fall into two categories:

- User detected errors
- Subsystem detected errors

User detected errors

User detected errors are detected by the user (or a user-written application program) when the results of a service request are not as expected (for example, a nonzero completion code). The collection of problem determination data cannot be automated because detection occurs after the IBM MQ function has completed. Rerunning the application with the IBM MQ user parameter trace facility activated can provide the data needed to analyze the problem. The output from this trace is directed to the *generalized trace facility* (GTF).

You can turn the trace on and off using an operator command. See "Using trace for problem determination on z/OS" on page 361 for more information.

Queue manager detected errors

The queue manager detects errors such as:

- A program check
- A data set filling up
- An internal consistency error

IBM MQ analyzes the error and takes the following actions:

- If the problem was caused by a user or application error (such as an invalid address being used), the error is reflected back to the application by completion and reason codes.

- If the problem was not caused by a user or application error (for example, all available DASD has been used, or the system detected an internal inconsistency), IBM MQ recovers if possible, either by sending completion and reason codes to the application, or if this is not possible, by stopping the application.
- If IBM MQ cannot recover, it terminates with a specific reason code. An SVC dump is typically taken recording information in the *system diagnostic work area* (SDWA) and *variable recording area* (VRA) portions of the dump, and an entry is made in SYS1.LOGREC.

IBM MQ for z/OS abends

Abends can occur in WebSphere for z/OS or other z/OS systems. Use this topic to understand the IBM MQ system abend codes and how to investigate abends which occur in CICS, IMS, and z/OS.

IBM MQ for z/OS uses two system abend completion codes, X'5C6' and X'6C6'. These codes identify:

- Internal errors encountered during operation
- Diagnostic information for problem determination
- Actions initiated by the component involved in the error

X'5C6'

An X'5C6' abend completion code indicates that IBM MQ has detected an internal error and has terminated an internal task (TCB) or a user-connected task abnormally. Errors associated with a X'5C6' abend completion code might be preceded by a z/OS system code, or by internal errors.

Examine the diagnostic material generated by the X'5C6' abend to determine the source of the error that actually resulted in a subsequent task or subsystem termination.

X'6C6'

An X'6C6' abend completion code indicates that IBM MQ has detected a severe error and has terminated the queue manager abnormally. When a X'6C6' is issued, IBM MQ has determined that continued operation could result in the loss of data integrity. Errors associated with a X'6C6' abend completion code might be preceded by a z/OS system error, one or more X'5C6' abend completion codes, or by error message CSQV086E indicating abnormal termination of IBM MQ.

Table 11 on page 393 summarizes the actions and diagnostic information available to IBM MQ for z/OS when these abend completion codes are issued. Different pieces of this information are relevant in different error situations. The information produced for a particular error depends upon the specific problem. For more information about the z/OS services that provide diagnostic information, see "Diagnostic information produced on IBM MQ for z/OS" on page 396.

Table 11. Abend completion codes

	X'5C6'	**X'6C6'**
Explanation	• Error during IBM MQ normal operation	• Severe error; continued operation might jeopardize data integrity
System action	• Internal IBM MQ task is abended • Connected user task is abended	• The entire IBM MQ subsystem is abended • User task with an active IBM MQ connection might be abnormally terminated with a X'6C6' code • Possible MEMTERM (memory termination) of connected allied address space
Diagnostic information	• SVC dump • SYS1.LOGREC entry • VRA data entries	• SYS1.LOGREC • VRA data entries

Table 11. Abend completion codes (continued)

	X'5C6'	**X'6C6'**
Associated reason codes	• IBM MQ abend reason code • Associated z/OS system codes	• Subsystem termination reason code • z/OS system completion codes and X'5C6' codes that precede the X'6C6' abend
Location of accompanying codes	• SVC dump title • Message CSQW050I • Register 15 of SDWA section 'General Purpose Registers at Time of Error' • SYS1.LOGREC entries • VRA data entries	• SYS1.LOGREC • VRA data entries • Message CSQV086E, which is sent to z/OS system operator

Related concepts

"Dealing with program abends" on page 394
Abends can occur with applications and other z/OS systems. Use this topic to investigate the various types of abends that can occur when using IBM MQ for z/OS.

"CICS, IMS, and z/OS abends" on page 395
Use this topic to investigate abends from CICS, IMS, and z/OS.

"Diagnostic information produced on IBM MQ for z/OS" on page 396
Use this topic to investigate some of the diagnostic information produced by z/OS that can be useful in problem determination and understand how to investigate error messages, dumps, console logs, job output, symptom strings, and queue output.

"IBM MQ for z/OS dumps" on page 400
Use this topic for information about the use of dumps in problem determination. It describes the steps you should take when looking at a dump produced by an IBM MQ for z/OS address space.

Dealing with program abends

Abends can occur with applications and other z/OS systems. Use this topic to investigate the various types of abends that can occur when using IBM MQ for z/OS.

Types of abend

Program abends can be caused by applications failing to check, and respond to, reason codes from IBM MQ. For example, if a message has not been received, using fields that would have been set up in the message for calculation might cause X'0C4' or X'0C7' abends (ASRA abends in CICS).

The following pieces of information indicate a program abend:

- Error messages from IBM MQ in the console log
- CICS error messages
- CICS transaction dumps
- IMS region dumps
- IMS messages on user or master terminal
- Program dump information in batch or TSO output
- Abend messages in batch job output
- Abend messages on the TSO screen

If you have an abend code, see one of the following manuals for an explanation of the cause of the abend:

- For IBM MQ for z/OS abends (abend codes X'5C6' and X'6C6'), see IBM MQ for z/OS messages, completion, and reason codes
- For batch abends, the *MVS System Codes* manual
- For CICS abends, the *CICS Messages and Codes* manual
- For IMS abends, the *IMS/ESA® Messages and Codes* manual
- For Db2 abends, the *Db2 Messages and Codes* manual
- For RRS abends, the *MVS System Messages* manual
- For XES abends, the *MVS System Messages* manual

Batch abends

Batch abends cause an error message containing information about the contents of registers to be displayed in the syslog. TSO abends cause an error message containing similar information to be produced on the TSO screen. A SYSUDUMP is taken if there is a SYSUDUMP DD statement for the step (see "IBM MQ for z/OS dumps" on page 400).

CICS transaction abends

CICS transaction abends are recorded in the CICS CSMT log, and a message is produced at the terminal (if there is one). A CICS AICA abend indicates a possible loop. See "Dealing with loops" on page 424 for more information. If you have a CICS abend, using CEDF and the CICS trace might help you to find the cause of the problem. See the *CICS Problem Determination Guide* for more information.

IMS transaction abends

IMS transaction abends are recorded on the IMS master terminal, and an error message is produced at the terminal (if there is one). If you have an IMS abend, see the *IMS/ESA Diagnosis Guide and Reference* manual.

CICS, IMS, and z/OS abends

Use this topic to investigate abends from CICS, IMS, and z/OS.

CICS abends

A CICS abend message is sent to the terminal, if the application is attached to one, or to the CSMT log. CICS abend codes are explained in the *CICS Messages and Codes* manual.

The CICS adapter issues abend reason codes beginning with the letter Q (for example, QDCL). These codes are documented in IBM MQ for z/OS messages, completion, and reason codes

IMS abends

An IMS application might abend in one of the following circumstances:

- A normal abend.
- An IMS pseudo abend, with an abend code such as U3044 resulting from an error in an ESAF exit program.
- Abend 3051 or 3047, when the REO (region error option) has been specified as "Q" or "A", and an IMS application attempts to reference a non-operational external subsystem, or when resources are unavailable at the time when a thread is created.

An IMS message is sent to the user terminal or job output, and the IMS master terminal. The abend might be accompanied by a region dump.

z/OS abends

During IBM MQ operation, an abend might occur with a z/OS system completion code. If you receive a z/OS abend, see the appropriate z/OS publication.

z/OS Diagnostic information produced on IBM MQ for z/OS

Use this topic to investigate some of the diagnostic information produced by z/OS that can be useful in problem determination and understand how to investigate error messages, dumps, console logs, job output, symptom strings, and queue output.

IBM MQ for z/OS functional recovery routines use z/OS services to provide diagnostic information to help you in problem determination.

The following z/OS services provide diagnostic information:

SVC dumps
The IBM MQ abend completion code X'5C6' uses the z/OS SDUMP service to create SVC dumps. The content and storage areas associated with these dumps vary, depending on the specific error and the state of the queue manager at the time the error occurred.

SYS1.LOGREC

Entries are requested in the SYS1.LOGREC data set at the time of the error using the z/OS SETRP service. The following are also recorded in SYS1.LOGREC:

- Subsystem abnormal terminations
- Secondary abends occurring in a recovery routine
- Requests from the recovery termination manager

Variable recording area (VRA) data
Data entries are added to the VRA of the SDWA by using a z/OS VRA defined key. VRA data includes a series of diagnostic data entries common to all IBM MQ for z/OS abend completion codes. Additional information is provided during initial error processing by the invoking component recovery routine, or by the recovery termination manager.

IBM MQ for z/OS provides unique messages that, together with the output of dumps, are aimed at providing sufficient data to allow diagnosis of the problem without having to try to reproduce it. This is known as first failure data capture.

Error messages

IBM MQ produces an error message when a problem is detected. IBM MQ diagnostic messages begin with the prefix CSQ. Each error message generated by IBM MQ is unique; that is, it is generated for one and only one error. Information about the error can be found in IBM MQ for z/OS messages, completion, and reason codes.

The first three characters of the names of IBM MQ modules are also usually CSQ. The exceptions to this are modules for C++ (IMQ), and the header files (CMQ). The fourth character uniquely identifies the component. Characters five through eight are unique within the group identified by the first four characters.

Make sure that you have some documentation on application messages and codes for programs that were written at your installation, as well as viewing IBM MQ for z/OS messages, completion, and reason codes

There might be some instances when no message is produced, or, if one is produced, it cannot be communicated. In these circumstances, you might have to analyze a dump to isolate the error to a particular module. For more information about the use of dumps, see "IBM MQ for z/OS dumps" on page 400.

Dumps

Dumps are an important source of detailed information about problems. Whether they are as the result of an abend or a user request, they allow you to see a snapshot of what was happening at the moment the dump was taken. "IBM MQ for z/OS dumps" on page 400 contains guidance about using dumps to locate problems in your IBM MQ system. However, because they only provide a snapshot, you might need to use them with other sources of information that cover a longer period of time, such as logs.

Snap dumps are also produced for specific types of error in handling MQI calls. The dumps are written to the CSQSNAP DD.

Console logs and job output

You can copy console logs into a permanent data set, or print them as required. If you are only interested in specific events, you can select which parts of the console log to print.

Job output includes output produced from running the job, as well as that from the console. You can copy this output into permanent data sets, or print it as required. You might need to collect output for all associated jobs, for example CICS, IMS, and IBM MQ.

Symptom strings

Symptom strings display important diagnostic information in a structured format. When a symptom string is produced, it is available in one or more of the following places:

- On the z/OS system console
- In SYS1.LOGREC
- In any dump taken

Figure 12 on page 397 shows an example of a symptom string.

```
PIDS/ 5655R3600 RIDS/CSQMAIN1 AB/S6C6 PRCS/0E30003
```

Figure 12. Sample symptom string

The symptom string provides a number of keywords that you can use to search the IBM software support database. If you have access to one of the optional search tools, you can search the database yourself. If you report a problem to the IBM support center, you are often asked to quote the symptom string.

Although the symptom string is designed to provide keywords for searching the database, it can also give you a lot of information about what was happening at the time the error occurred, and it might suggest an obvious cause or a promising area to start your investigation.

Queue information

You can display information about the status of queues by using the operations and control panels. Alternatively you can enter the DISPLAY QUEUE and DISPLAY QSTATUS commands from the z/OS console.

Note: If the command was issued from the console, the response is copied to the console log, allowing the documentation to be kept together compactly.

Related concepts
"Using trace for problem determination on z/OS" on page 361

There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

“Other sources of information” on page 398
Use this topic to investigate other sources of information for problem determination.

“Diagnostic aids for CICS” on page 399
You can use the CICS diagnostic transactions to display information about queue manager tasks, and MQI calls. Use this topic to investigate these facilities.

“Diagnostic aids for IMS” on page 399
Use this topic to investigate IMS diagnostic facilities.

“Diagnostic aids for Db2” on page 399
Use this topic to investigate references for Db2 diagnostic tools.

Other sources of information

Use this topic to investigate other sources of information for problem determination.

You might find the following items of documentation useful when solving problems with IBM MQ for z/OS.

- Your own documentation
- Documentation for the products you are using
- Source listings and link-edit maps
- Change log
- System configuration charts
- Information from the DISPLAY CONN command

Your own documentation

Your own documentation is the collection of information produced by your organization about what your system and applications should do, and how they are supposed to do it. How much of this information you need depends on how familiar you are with the system or application in question, and could include:

- Program descriptions or functional specifications
- Flowcharts or other descriptions of the flow of activity in a system
- Change history of a program
- Change history of your installation
- Statistical and monitoring profile showing average inputs, outputs, and response times

Documentation for the products you are using

The documentation for the product you are using are the InfoCenters in the IBM MQ library, and in the libraries for any other products you use with your application.

Make sure that the level of any documentation you refer to matches the level of the system you are using. Problems often arise through using either obsolete information, or information about a level of a product that is not yet installed.

Source listings and link-edit maps

Include the source listings of any applications written at your installation with your set of documentation. (They can often be the largest single element of documentation.) Make sure that you include the relevant output from the linkage editor with your source listings to avoid wasting time trying to find your way through a load module with an out-of-date link map. Be sure to include the JCL at the beginning of your listings, to show the libraries that were used and the load library the load module was placed in.

Change log

The information in the change log can tell you of changes made in the data processing environment that might have caused problems with your application program. To get the most out of your change log, include the data concerning hardware changes, system software (such as z/OS and IBM MQ) changes, application changes, and any modifications made to operating procedures.

System configuration charts

System configuration charts show what systems are running, where they are running, and how the systems are connected to each other. They also show which IBM MQ, CICS, or IMS systems are test systems and which are production systems.

Information from the DISPLAY CONN command

The DISPLAY CONN command provides information about which applications are connected to a queue manager, and information to help you to diagnose those that have a long-running unit of work. You could collect this information periodically and check it for any long-running units of work, and display the detailed information about that connection.

Diagnostic aids for CICS

You can use the CICS diagnostic transactions to display information about queue manager tasks, and MQI calls. Use this topic to investigate these facilities.

You can use the CKQC transaction (the CICS adapter control panels) to display information about queue manager tasks, and what state they are in (for example, a GET WAIT). See Administering IBM MQ for z/OS for more information about CKQC.

The application development environment is the same as for any other CICS application, and so you can use any tools normally used in that environment to develop IBM MQ applications. In particular, the *CICS execution diagnostic facility* (CEDF) traps entry to and exit from the CICS adapter for each MQI call, as well as trapping calls to all CICS API services. Examples of the output produced by this facility are given in Examples of CEDF output.

The CICS adapter also writes trace entries to the CICS trace. These entries are described in "CICS adapter trace entries" on page 369.

Additional trace and dump data is available from the CICS region. These entries are as described in the *CICS Problem Determination Guide*.

Diagnostic aids for IMS

Use this topic to investigate IMS diagnostic facilities.

The application development environment is the same as for any other IMS application, and so any tools normally used in that environment can be used to develop IBM MQ applications.

Trace and dump data is available from the IMS region. These entries are as described in the *IMS/ESA Diagnosis Guide and Reference* manual.

Diagnostic aids for Db2

Use this topic to investigate references for Db2 diagnostic tools.

Refer to the following manuals for help in diagnosing Db2 problems:

- *Db2 for z/OS Diagnosis Guide and Reference*
- *Db2 Messages and Codes*

z/OS IBM MQ for z/OS dumps

Use this topic for information about the use of dumps in problem determination. It describes the steps you should take when looking at a dump produced by an IBM MQ for z/OS address space.

How to use dumps for problem determination

When solving problems with your IBM MQ for z/OS system, you can use dumps in two ways:

- To examine the way IBM MQ processes a request from an application program.

  To do this, you typically need to analyze the whole dump, including control blocks and the internal trace.
- To identify problems with IBM MQ for z/OS itself, under the direction of IBM support center personnel.

Use the instructions in the following topics to get and process a dump:

- "Getting a dump" on page 400
- "Using the z/OS DUMP command" on page 401
- "Processing a dump using the IBM MQ for z/OS dump display panels" on page 403
- "Processing a dump using line mode IPCS" on page 407
- "Processing a dump using IPCS in batch" on page 414

The dump title might provide sufficient information in the abend and reason codes to resolve the problem. You can see the dump title in the console log, or by using the z/OS command `DISPLAY DUMP,TITLE`. The format of the dump title is explained in "Analyzing the dump and interpreting dump titles" on page 415. For information about the IBM MQ for z/OS abend codes, see "IBM MQ for z/OS abends" on page 393, and abend reason codes are documented in IBM MQ for z/OS messages, completion, and reason codes.

If there is not enough information about your problem in the dump title, format the dump to display the other information contained in it.

See the following topics for information about different types of dumps:

- "SYSUDUMP information" on page 416
- "Snap dumps" on page 417
- "SYS1.LOGREC information" on page 418
- "SVC dumps" on page 418

Related concepts

"Using trace for problem determination on z/OS" on page 361
There are different trace options that can be used for problem determination with IBM MQ. Use this topic to understand the different options and how to control trace.

"IBM MQ for z/OS abends" on page 393
Abends can occur in WebSphere for z/OS or other z/OS systems. Use this topic to understand the IBM MQ system abend codes and how to investigate abends which occur in CICS, IMS, and z/OS.

"Diagnostic information produced on IBM MQ for z/OS" on page 396
Use this topic to investigate some of the diagnostic information produced by z/OS that can be useful in problem determination and understand how to investigate error messages, dumps, console logs, job output, symptom strings, and queue output.

Getting a dump

Use this topic to understand the different dump types for problem determination.

The following table shows information about the types of dump used with IBM MQ for z/OS and how they are initiated. It also shows how the dump is formatted:

Table 12. Types of dump used with IBM MQ for z/OS

Dump type	Data set	Output type	Formatted by	Caused by
SVC	Defined by system	Machine readable	IPCS in conjunction with an IBM MQ for z/OS verb exit	z/OS or IBM MQ for z/OS functional recovery routine detecting error, or the operator entering the z/OS DUMP command
SYSUDUMP	Defined by JCL (SYSOUT=A)	Formatted	Normally SYSOUT=A	An abend condition (only taken if there is a SYSUDUMP DD statement for the step)
Snap	Defined by JCL CSQSNAP (SYSOUT=A)	Formatted	Normally SYSOUT=A	Unexpected MQI call errors reported to adapters, or FFST information from the channel initiator
Stand-alone	Defined by installation (tape or disk)	Machine readable	IPCS in conjunction with an IBM MQ for z/OS verb exit	Operator IPL of the stand-alone dump program

IBM MQ for z/OS recovery routines request SVC dumps for most X'5C6' abends. The exceptions are listed in "SVC dumps" on page 418. SVC dumps issued by IBM MQ for z/OS are the primary source of diagnostic information for problems.

If the dump is initiated by the IBM MQ subsystem, information about the dump is put into area called the *summary portion*. This contains information that the dump formatting program can use to identify the key components.

For more information about SVC dumps, see the *MVS Diagnosis: Tools and Service Aids* manual.

Using the z/OS DUMP command

To resolve a problem, IBM can ask you to create a dump file of the queue manager address space, channel initiator address space, or coupling facilities structures. Use this topic to understand the commands to create these dump files.

You might be asked to create dump file for any or several of the following items for IBM to resolve the problem:

- Main IBM MQ address space
- Channel initiator address space
- Coupling facility application structure
- Coupling facility administration structure for your queue-sharing group

Figure 13 on page 402 through to Figure 17 on page 403 show examples of the z/OS commands to do this, assuming a subsystem name of CSQ1.

```
DUMP COMM=(MQ QUEUE MANAGER DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=(CSQ1MSTR,BATCH),CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=CSQ1MSTR,CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),END
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQ QUEUE MANAGER MAIN DUMP
```

Figure 13. Dumping the IBM MQ queue manager and application address spaces

```
DUMP COMM=(MQ QUEUE MANAGER DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=(CSQ1MSTR),CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=CSQ1MSTR,CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),END
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQ QUEUE MANAGER DUMP
```

Figure 14. Dumping the IBM MQ queue manager address space

```
DUMP COMM=(MQ CHIN DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=CSQ1CHIN,CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=CSQ1CHIN,CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),CONT
*03 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),CONT
R 03,DSPNAME=('CSQ1CHIN'.CSQXTRDS),END
IEE600I REPLY TO 03 IS;DSPNAME='CSQ1CHIN'.CSQXTRDS,END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQ CHIN DUMP
```

Figure 15. Dumping the channel initiator address space

```
DUMP COMM=(MQ MSTR & CHIN DUMP)
*01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,JOBNAME=(CSQ1MSTR,CSQ1CHIN),CONT
*02 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 01 IS;JOBNAME=(CSQ1MSTR,CSQ1CHIN),CONT
R 02,SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),CONT
*03 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
 IEE600I REPLY TO 02 IS;SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM),CONT
R 03,DSPNAME=('CSQ1CHIN'.CSQXTRDS),END
IEE600I REPLY TO 03 IS;DSPNAME=('CSQ1CHIN'.CSQXTRDS),END
IEA794I SVC DUMP HAS CAPTURED: 869
DUMPID=001 REQUESTED BY JOB (*MASTER*)
DUMP TITLE=MQ MSTR & CHIN DUMP
```

Figure 16. Dumping the IBM MQ queue manager and channel initiator address spaces

```
DUMP COMM=('MQ APPLICATION STRUCTURE 1 DUMP')
01 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
R 01,STRLIST=(STRNAME=QSG1APPLICATION1,(LISTNUM=ALL,ADJUNCT=CAPTURE,ENTRYDATA=UNSER))
IEE600I REPLY TO 01 IS;STRLIST=(STRNAME=QSG1APPLICATION1,(LISTNUM=
IEA794I SVC DUMP HAS CAPTURED: 677
DUMPID=057 REQUESTED BY JOB (*MASTER*)
DUMP TITLE='MQ APPLICATION STRUCTURE 1 DUMP'
```

Figure 17. Dumping a coupling facility structure

Processing a dump using the IBM MQ for z/OS dump display panels

You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

IBM MQ for z/OS provides a set of panels to help you process dumps. The following section describes how to use these panels:

1. From the IPCS PRIMARY OPTION MENU, select **ANALYSIS - Analyze dump contents** (option 2).

   The IPCS MVS ANALYSIS OF DUMP CONTENTS panel is displayed.
2. Select **COMPONENT - MVS component data** (option 6).

   The IPCS MVS DUMP COMPONENT DATA ANALYSIS panel is displayed. The appearance of the panel depends on the products installed at your installation, but will be similar to the panel shown in IPCS MVS Dump Component Data Analysis panel:

```
---------------- IPCS MVS DUMP COMPONENT DATA ANALYSIS -------------
OPTION ===>                                               SCROLL ===

To display information, specify "S option name" or enter S to the
left of the option desired.  Enter ? to the left of an option to
display help regarding the component support.

  Name     Abstract
  ALCWAIT  Allocation wait summary
  AOMDATA  AOM analysis
  ASMCHECK Auxiliary storage paging activity
  ASMDATA  ASM control block analysis
  AVMDATA  AVM control block analysis
  COMCHECK Operator communications data
  CSQMAIN  WebSphere MQ dump formatter panel interface
  CSQWDMP  WebSphere MQ dump formatter
  CTRACE   Component trace summary
  DAEDATA  DAE header data
  DIVDATA  Data-in-virtual storage
```

   Figure 18. IPCS MVS Dump Component Data Analysis panel
3. Select **CSQMAIN IBM MQ dump formatter panel interface** by typing s beside the line and pressing Enter.

   If this option is not available, it is because the member CSQ7IPCS is not present; you should see Configuring z/OS for more information about installing the IBM MQ for z/OS dump formatting member.

   Note: If you have already used the dump to do a preliminary analysis, and you want to reexamine it, select **CSQWDMP IBM MQ dump formatter** to display the formatted contents again, using the default options.
4. The IBM MQ for z/OS - DUMP ANALYSIS menu is displayed. Use this menu to specify the action that you want to perform on a system dump.

```
---------------IBM WebSphere MQ for z/OS - DUMP ANALYSIS----------------
 COMMAND ===>



      1 Display all dump titles 00 through 99
      2 Manage the dump inventory
      3 Select a dump

      4 Display address spaces active at time of dump
      5 Display the symptom string
      6 Display the symptom string and other related data
      7 Display LOGREC data from the buffer in the dump
      8 Format and display the dump

      9 Issue IPCS command or CLIST



(c) Copyright IBM Corporation 1993, 2022. All rights reserved.

  F1=Help    F3=Exit   F12=Cancel
```

5. Before you can select a particular dump for analysis, the dump you require must be present in the dump inventory. To ensure that this is so, perform the following steps:

   a. If you do not know the name of the data set containing the dump, specify option 1 - **Display all dump titles xx through xx**.

      This displays the dump titles of all the dumps contained in the SYS1.DUMP data sets (where xx is a number in the range 00 through 99). You can limit the selection of data sets for display by using the xx fields to specify a range of data set numbers.

      If you want to see details of all available dump data sets, set these values to 00 and 99.

      Use the information displayed to identify the dump you want to analyze.

   b. If the dump has not been copied into another data set (that is, it is in one of the SYS1.DUMP data sets), specify option 2 - **Manage the dump inventory**

      The dump inventory contains the dump data sets that you have used. Because the SYS1.DUMP data sets are reused, the name of the dump that you identified in step “5.a” on page 404 might be in the list displayed. However, this entry refers to the previous dump that was stored in this data set, so delete it by typing DD next to it and pressing Enter. Then press F3 to return to the DUMP ANALYSIS MENU.

6. Specify option 3 - **Select a dump**, to select the dump that you want to work with. Type the name of the data set containing the dump in the Source field, check that NOPRINT and TERMINAL are specified in the Message Routing field (this is to ensure that the output is directed to the terminal), and press Enter. Press F3 to return to the DUMP ANALYSIS MENU.

7. Having selected a dump to work with, you can now use the other options on the menu to analyze the data in different parts of the dump:

   - To display a list of all address spaces active at the time the dump was taken, select option 4.
   - To display the symptom string, select option 5.
   - To display the symptom string and other serviceability information, including the variable recording area of the system diagnostic work area (SDWA), select option 6.
   - To format and display the data contained in the in-storage LOGREC buffer, select option 7.

     It could be that the abend that caused the dump was not the original cause of the error, but was caused by an earlier problem. To determine which LOGREC record relates to the cause of the problem, go to the bottom of the data set, type `FIND ERRORID: PREV`, and press Enter. The header of the latest LOGREC record is displayed, for example:

```
JOBNAME: NONE-FRR
 ERRORID: SEQ=00081  CPU=0040  ASID=0033  TIME=14:42:47.1

SEARCH ARGUMENT ABSTRACT

   PIDS/5655R3600 RIDS/CSQRLLM1#L RIDS/CSQRRHSL AB/S05C6
   PRCS/00D10231 REGS/0C1F0 RIDS/CSQVEUS2#R

   SYMPTOM             DESCRIPTION
   -------             -----------
   PIDS/5655R3600      PROGRAM ID: 5655R3600
.
.
.
```

Note the program identifier (if it is not 5655R3600, the problem was not caused by IBM MQ for z/OS and you could be looking at the wrong dump). Also note the value of the TIME field. Repeat the command to find the previous LOGREC record, and note the value of the TIME field again. If the two values are close to each other (say, within about one or two tenths of a second), they could both relate to the same problem.

- To format and display the dump, select option 8. The FORMAT AND DISPLAY THE DUMP panel is displayed:

```
---------IBM MQ for z/OS - FORMAT AND DISPLAY DUMP--------
COMMAND ===>

1 Display the control blocks and trace
2 Display just the control blocks
3 Display just the trace


Options:

Use the summary dump? . . . . . . . . . . . . . . __ 1 Yes
2 No


Subsystem name (required if summary dump not used) ____


Address space identifier or ALL. . . . . . . . . . ALL_



F1=Help  F3=Exit  F12=Cancel
```

- Use this panel to format your selected system dump. You can choose to display control blocks, data produced by the internal trace, or both, which is the default.

  Note: You cannot do this for dumps from the channel initiator, or for dumps of coupling facility structures.

  – To display the whole of the dump, that is:
    - The dump title
    - The variable recording area (VRA) diagnostic information report
    - The save area trace report
    - The control block summary
    - The trace table

    select option 1.
  – To display the information listed for option 1, without the trace table, select option 2.
  – To display the information listed for option 1, without the control blocks, select option 3.

You can also use the following options:

– **Use the Summary Dump?**

  Use this field to specify whether you want IBM MQ to use the information contained in the summary portion when formatting the selected dump. The default setting is YES.

  Note: If a summary dump has been taken, it might include data from more than one address space.

– **Subsystem name**

  Use this field to identify the subsystem with the dump data you want to display. This is only required if there is no summary data (for example, if the operator requested the dump), or if you have specified NO in the **Use the summary dump?** field.

  If you do not know the subsystem name, type `IPCS SELECT ALL` at the command prompt, and press Enter to display a list of all the jobs running at the time of the error. If one of the jobs has the word ERROR against it in the SELECTION CRITERIA column, make a note of the name of that job. The job name is of the form *xxxx* MSTR, where *xxxx* is the subsystem name.

```
IPCS OUTPUT STREAM ------------------------
COMMAND ===>
ASID JOBNAME ASCBADDR SELECTION CRITERIA
---- -------- -------- ------------------
0001 *MASTER* 00FD4D80 ALL
0002 PCAUTH  00F8AB80 ALL
0003 RASP   00F8C100 ALL
0004 TRACE  00F8BE00 ALL
0005 GRS   00F8BC00 ALL
0006 DUMPSRV 00F8DE00 ALL
0008 CONSOLE 00FA7E00 ALL
0009 ALLOCAS 00F8D780 ALL
000A SMF   00FA4A00 ALL
000B VLF   00FA4800 ALL
000C LLA   00FA4600 ALL
000D JESM   00F71E00 ALL
001F MQM1MSTR 00FA0680 ERROR ALL
```

  If no job has the word ERROR against it in the SELECTION CRITERIA column, select option 0 - DEFAULTS on the main IPCS Options Menu panel to display the IPCS Default Values panel. Note the address space identifier (ASID) and press F3 to return to the previous panel. Use the ASID to determine the job name; the form is *xxxx* MSTR, where *xxxx* is the subsystem name.

  The following command shows which ASIDs are in the dump data set:

```
LDMP DSN('SYS1.DUMPxx') SELECT(DUMPED) NOSUMMARY
```

  This shows the storage ranges dumped for each address space.

  Press F3 to return to the FORMAT AND DISPLAY THE DUMP panel, and type this name in the **Subsystem name** field.

– **Address space identifier**

  Use this field if the data in a dump comes from more than one address space. If you only want to look at data from a particular address space, specify the identifier (ASID) for that address space.

  The default value for this field is ALL, which displays information about all the address spaces relevant to the subsystem in the dump. Change this field by typing the 4-character ASID over the value displayed.

  Note: Because the dump contains storage areas common to all address spaces, the information displayed might not be relevant to your problem if you specify the address space identifier incorrectly. In this case, return to this panel, and enter the correct address space identifier.

Related concepts

"Processing a dump using line mode IPCS" on page 407
Use the IPCS commands to format a dump.

"Processing a dump using IPCS in batch" on page 414
Use this topic to understand how IBM MQ dumps can be formatted by IPCS commands in batch mode.

"Analyzing the dump and interpreting dump titles" on page 415
Use this topic to understand how dump titles are formatted, and how to analyze a dump.

Processing a dump using line mode IPCS

Use the IPCS commands to format a dump.

To format the dump using line mode IPCS commands, select the dump required by issuing the command:

```
SETDEF DSN('SYS1.DUMP xx ')
```

(where SYS1.DUMP *xx* is the name of the data set containing the dump). You can then use IPCS subcommands to display data from the dump.

See the following topics for information on how to format different types of dumps using IPCS commands:

- "Formatting an IBM MQ for z/OS dump" on page 407
- "Formatting a dump from the channel initiator" on page 413

Related concepts

"Processing a dump using the IBM MQ for z/OS dump display panels" on page 403
You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

"Processing a dump using IPCS in batch" on page 414
Use this topic to understand how IBM MQ dumps can be formatted by IPCS commands in batch mode.

"Analyzing the dump and interpreting dump titles" on page 415
Use this topic to understand how dump titles are formatted, and how to analyze a dump.

Formatting an IBM MQ for z/OS dump

Use this topic to understand how to format a queue manager dump using line mode IPCS commands.

The IPCS VERBEXIT CSQWDMP invokes the IBM MQ for z/OS dump formatting program (CSQWDPRD), and enables you to format an SVC dump to display IBM MQ data. You can restrict the amount of data that is displayed by specifying parameters.

IBM Service Personnel might require dumps of your coupling facility administration structure and application structures for your queue-sharing group, with dumps of queue managers in the queue-sharing group, to aid problem diagnosis. For information on formatting a coupling facility list structure, and the STRDATA subcommand, see the *MVS IPCS Commands* book.

Note: This section describes the parameters required to extract the necessary data. Separate operands by commas, not blanks. A blank that follows any operand in the control statement terminates the operand list, and any subsequent operands are ignored. Table 13 on page 407 explains each keyword that you can specify in the control statement for formatting dumps.

Table 13. Keywords for the IBM MQ for z/OS dump formatting control statement

Keyword	**Description**
SUBSYS= *aaaa*	Use this keyword if the summary dump portion is not available, or not to be used, to give the name of the subsystem to format information for. *aaaa* is a 1 through 4-character subsystem name.
ALL (default)	All control blocks and the trace table.

Table 13. Keywords for the IBM MQ for z/OS dump formatting control statement (continued)

Keyword	**Description**
AA	Data is displayed for all IBM MQ for z/OS control blocks in all address spaces.
DIAG=Y	Print diagnostic information. Use only under guidance from IBM service personnel. DIAG=N (suppresses the formatting of diagnostic information) is the default.
EB= *nnnnnnnn*	Only the trace points associated with this EB thread are displayed (the format of this keyword is EB= *nnnnnnnn* where *nnnnnnnn* is the 8-digit address of an EB thread that is contained in the trace). You must use this in conjunction with the TT keyword.
LG	All control blocks.
PTF=Y, LOAD= *load module name*	A list of PTFs at the front of the report (from MEPL). PTF=N (suppresses the formatting of such a list) is the default. The optional load subparameter allows you to specify the name of a load module, up to a maximum of 8 characters, for which to format a PTF report.
SA= *hhhh*	The control blocks for a specified address space. Use either of the following formats: • SA= *hh* or • SA= *hhhh* where *h* represents a hexadecimal digit.
SG	A subset of system-wide control blocks.
TT ,HANDLES=x ,LOCKS=x ,INSYNCS=x ,URINFO=ALL/LONG	Format trace table Indicate threads with greater than x handles Indicate threads with greater than x locks Indicate threads with greater than x insync operations Show UR info for ALL threads or for long-running threads

Table 14 on page 408details the dump formatting keywords that you can use to format the data relating to individual resource managers.

You cannot use these keywords in conjunction with any of the keywords inTable 13 on page 407.

Table 14. Resource manager dump formatting keywords

Keyword	**What is formatted**

BMC=1
BMC=2(*buffer pool number*)
BMC=3(xx/yyyyyy)
BMC=4(xx/yyyyyy)
	Buffer manager data. BMC=1 formats control blocks of all buffers.
	BMC=2 formats data relating to the buffer identified in the 2-digit *buffer pool number*.
	BMC=3 and BMC=4 display a page from a pageset, if the page is present in a buffer. (The difference between BMC=3 and BMC=4 is the route taken to the page.)

BUFL= *<nnnnnnnnnnn>*	Storage access buffer allocation sz.

Table 14. Resource manager dump formatting keywords (continued)

Keyword	**What is formatted**
CALLD=Y =W	Show arrow for call depth in TT. and indent trace entry.
CALLTIME=Y	Print call time on exit trace.
CB= *(<addr>/[<strmodel>])*	Format address as IBM MQ block.
CBF=1	CBF report level 1.
CCB=S	Show the Composite Capability Block (CCB) for system EBs in TT.
CFS=1	CFS report level 1.
CFS=2	CFS report level 2.

CHLAUTH=1/2
 	
ONAM=<20 chars>
	CHLAUTH report level.
	
	The optional ONAM subparameter allows you to specify the object name, up to a maximum of 20 characters, to limit data printed to objects starting with characters in ONAM.

CLUS=1	Cluster report including the cluster repository known on the queue manager.
CLUS=2	Cluster report showing cluster registrations.
CLXQ=1	Cluster XMITQ report level 1.

CLXQ=2
 	
ONAM=<20 chars>
	Cluster XMITQ report level 2.
	
	The optional ONAM subparameter allows you to specify the object name, up to a maximum of 20 characters, to limit data printed to objects starting with characters in ONAM.

CMD=0/1/2	Command trace table display level.
D=1/2/3	Detail level for some reports.
Db2=1	Db2 report level 1.

DMC=1,
 	
ONAM=<48 chars>
	DMC report level 1.
	
	The optional ONAM subparameter allows you to specify the object name, up to a maximum of 48 characters, to limit data printed to objects starting with characters in ONAM.

DMC=2,
 	
ONAM=<48 chars>
	DMC report level 2.
	
	The optional ONAM subparameter allows you to limit the objects printed to those with names beginning with the characters specified in ONAM (up to a maximum of 48 characters).

DMC=3,
 	
ONAM=<48 chars>
	DMC report level 3.
	
	The optional ONAM subparameter allows you to limit the objects printed to those with names beginning with the characters specified in ONAM (up to a maximum of 48 characters).

GR=1	Group indoubt report level 1.
IMS=1	IMS report level 1

Table 15. Resource manager dump formatting keywords (J -P)

Keyword	What is formatted
JOBNAME= *xxxxxxxx*	Job name
LKM=1	LKM report level 1.
LKM=2/3, ,NAME=<up to 48 chars> ,NAMEX= *xxxxxxxxxxxxxxxx* ,NAMESP=1/2/3/4/5/6/7/8 ,TYPE=DMCP/QUALNM/TOPIC/ STGCLASS ,QUAL=GET/PUT/CRE/DFXQ/ PGSYNC/CHGCNT/ DELETE/EXPIRE LKM=3 LKM=4 ,JOBNAME= *xxxxxxxx* ,ASID= *xxxx*	LKM report level 2/3. Name (character) Name (Hex) Namespace Lock type Lock qualification LKM report level 3 LKM report level 4
LMC=1	LMC report level 1.
MAXTR= *nnnnnnnnn*	Max trace entries to format
MHASID= *xxxx*	Message handle ASID for properties
MMC=1 OBJ=MQLO/MQSH/MQRO/ MQAO/MQMO/MCHL/ MNLS/MSTC/MPRC/ : ' MAUT ONAM	MMC report level 1 Object type The optional ONAM subparameter allows you to limit the objects printed to those with names beginning with the characters specified in ONAM (up to a maximum of 48 characters).

MMC=2
ONAM=<48 chars>
	MMC report level 2
	The optional ONAM subparameter allows you to limit the objects printed to those with names beginning with the characters specified in ONAM (up to a maximum of 48 characters).

MSG=*nnnnnnnnnnnnnnnn*
MASID=*xxxx*
LEN=*xxxxxxxx*
MSGD=S/D
	Format the message at pointer.
	MASID allows storage in other address spaces.
	LEN limits amount of storage to format.
	MSGD controls level of detail.

MSGD=S/D	Message details in DMC=3, BMC=3/4, PSID reports. The parameter controls level of details, S is summary and D is detailed.
MSGH = *nnnnnnnnnnnnnnnn*	Message handle

Table 15. Resource manager dump formatting keywords (J -P) (continued)

Keyword	**What is formatted**
MT	Message properties trace
MQVCX	MQCHARVs in hexadecimal format
PROPS= *nnnnnnnnnnnnnnnn*	Message properties pointer
PSID= *nnnnnnnn*	Pageset to format page
PSTRX	Properties strings in hex format

Table 16. Resource manager dump formatting keywords (R -Z)

Keyword	**What is formatted**
RPR= *nnnnnnnn*	Page or record to format
SHOWDEL	Show deleted records for DMC=3
SMC=1/2/3	Storage manager
TC= * A E 0	TT data char format, concatenated print all in suitable character set always print ASCII always print EBCDIC never print either
TFMT=H/M	Time format - human or STCK
THR= *nnnnnnnn*	Thread address
THR=*/2/3	Set thread report level
TOP=1	TOP report level 1
TOP=2	TOP report level 2
TOP= *nnnnnnnnnnnnnnnn* /TSTR=<48 chars> /TSTRX=<hex 1208 str>	Tnode 64bit address or Topic string (wildcard with % at start or end)' This will be converted EBCDIC to ASCII, but only invariant characters Hexadecimal of topic string in 1208 always wildcard character at start.
TOP=3	TOP report level 3
TOP=4	TOP report level 4

TSEG=M(RU)/Q(P64)
I(NTERPOLATE)
F(WD)
D(EBUG)
	Search process for 64-bit trace
	Guess missing TSEG address or addresses
	Force forward sort
	Debug search process

TSEG=(M,Q,I,F,D)	Specify multiple TSEG options
W=0/1/2/3	TT width format
XA=1	XA report level 1
ZMH = *nnnnnnnnnnnnnnnn*	ZST message handle

If the dump is initiated by the operator, there is no information in the summary portion of the dump.Table 17 on page 412shows additional keywords that you can use in the CSQWDMP control statement.

Table 17. Summary dump keywords for the IBM MQ for z/OS dump formatting control statement

Keyword	**Description**
SUBSYS= *aaaa*	Use this keyword if the summary dump portion is not available, or not to be used, to give the name of the subsystem to format information for. *aaaa* is a 1 through 4-character subsystem name.
SUMDUMP=NO	Use this keyword if the dump has a summary portion, but you do not want to use it. (You would usually only do this if so directed by your IBM support center.)

The following list shows some examples of how to use these keywords:

- For default formatting of all address spaces, using information from the summary portion of the dump, use:

```
VERBX CSQWDMP
```

- To display the trace table from a dump of subsystem named MQMT, which was initiated by an operator (and so does not have a summary portion) use:

```
VERBX CSQWDMP 'TT,SUBSYS=MQMT'
```

- To display all the control blocks and the trace table from a dump produced by a subsystem abend, for an address space with ASID (address space identifier) 1F, use:

```
VERBX CSQWDMP 'TT,LG,SA=1F'
```

- To display the portion of the trace table from a dump associated with a particular EB thread, use:

```
VERBX CSQWDMP 'TT,EB= nnnnnnnn '
```

- To display message manager 1 report for local non-shared queue objects with a name begins with 'ABC' use:

```
VERBX CSQWDMP 'MMC=1,ONAM=ABC,Obj=MQLO'
```

Table 18 on page 412shows some other commands that are used frequently for analyzing dumps. For more information about these subcommands, see the *MVS IPCS Commands* manual.

Table 18. IPCS subcommands used for dump analysis

Subcommand	**Description**
STATUS	To display data usually examined during the initial part of the problem determination process.
STRDATA LISTNUM(ALL) ENTRYPOS(ALL) DETAIL	To format coupling facility structure data.
VERBEXIT LOGDATA	To format the in-storage LOGREC buffer records present before the dump was taken. LOGDATA locates the LOGREC entries that are contained in the LOGREC recording buffer and invokes the EREP program to format and print the LOGREC entries. These entries are formatted in the style of the normal detail edit report.

Table 18. IPCS subcommands used for dump analysis (continued)

Subcommand	Description
VERBEXIT TRACE	To format the system trace entries for all address spaces.
VERBEXIT SYMPTOM	To format the symptom strings contained in the header record of a system dump such as stand-alone dump, SVC dump, or an abend dump requested with a SYSUDUMP DD statement.
VERBEXIT GRSTRACE	To format diagnostic data from the major control blocks for global resource serialization.
VERBEXIT SUMDUMP	To locate and display the summary dump data that an SVC dump provides.
VERBEXIT DAEDATA	To format the dump analysis and elimination (DAE) data for the dumped system.

Related concepts

"Formatting a dump from the channel initiator" on page 413
Use this topic to understand how to format a channel initiator dump using line mode IPCS commands.

Formatting a dump from the channel initiator

Use this topic to understand how to format a channel initiator dump using line mode IPCS commands.

The IPCS VERBEXIT CSQXDPRD enables you to format a channel initiator dump. You can select the data that is formatted by specifying keywords.

This section describes the keywords that you can specify.

Table 19 on page 413 describes the keywords that you can specify with CSQXDPRD.

Table 19. Keywords for the IPCS VERBEXIT CSQXDPRD

Keyword	What is formatted
SUBSYS= *aaaa*	The control blocks of the channel initiator associated with the named subsystem. It is required for all new formatted dumps.
CHST=1, CNAM= *channel name*, DUMP=S|F|C	All channel information. The optional CNAM subparameter allows you to specify the name of a channel, up to a maximum of 20 characters, for which to format details. The optional DUMP subparameter allows you to control the extent of formatting, as follows: • Specify DUMP=S (for "short") to format the first line of the hexadecimal dump of the channel data. • Specify DUMP=F (for "full") to format all lines of the data. • Specify DUMP=C (for "compressed˺) to suppress the formatting of all duplicate lines in the data containing only X'00'. This is the default option
CHST=2, CNAM= *channel name*,	A summary of all channels, or of the channel specified by the CNAM keyword. See CHST=1 for details of the CNAM subparameter.
CHST=3, CNAM= *channel name*,	Data provided by CHST=2 and a program trace, line trace and formatted semaphore table print of all channels in the dump. See CHST=1 for details of the CNAM subparameter.

Table 19. Keywords for the IPCS VERBEXIT CSQXDPRD (continued)

Keyword	What is formatted
CLUS=1	Cluster report including the cluster repository known on the queue manager.
CLUS=2	Cluster report showing cluster registrations.

CTRACE=S|F,
 	
DPRO= *nnnnnnnn*,
 	
TCB= *nnnnnnn*
	Select either a short (CTRACE=S) or full (CTRACE=F) CTRACE.
	
	The optional DPRO subparameter allows you to specify a CTRACE for the DPRO specified.
	
	The optional TCB subparameter allows you to specify a CTRACE for the job specified.

DISP=1, DUMP=S|F|C	Dispatcher report See CHST=1 for details of the DUMP subparameter.
BUF=1	Buffer report
XSMF=1	Format channel initiator SMF data that is available in a dump.

Related concepts

"Formatting an IBM MQ for z/OS dump" on page 407

Use this topic to understand how to format a queue manager dump using line mode IPCS commands.

Processing a dump using IPCS in batch

Use this topic to understand how IBM MQ dumps can be formatted by IPCS commands in batch mode.

To use IPCS in batch, insert the required IPCS statements into your batch job stream (see Figure 19 on page 414).

Change the data set name (DSN=) on the DUMP00 statement to reflect the dump you want to process, and insert the IPCS subcommands that you want to use.

```
//**************************************************
//*   RUNNING IPCS IN A BATCH JOB                  *
//**************************************************
//MQMDMP   EXEC  PGM=IKJEFT01,REGION=5120K
//STEPLIB  DD    DSN=mqm.library-name,DISP=SHR
//SYSTSPRT DD    SYSOUT=*
//IPCSPRNT DD    SYSOUT=*
//IPCSDDIR DD    DSN=dump.directory-name,DISP=OLD
//DUMP00   DD    DSN=dump.name,DISP=SHR
//SYSTSIN  DD    *
  IPCS NOPARM TASKLIB(SCSQLOAD)
  SETDEF PRINT TERMINAL DDNAME(DUMP00) NOCONFIRM
  **************************************************
  * INSERT YOUR IPCS COMMANDS HERE, FOR EXAMPLE:   *
  VERBEXIT LOGDATA
  VERBEXIT SYMPTOM
  VERBEXIT CSQWDMP 'TT,SUBSYS=QMGR'
  **************************************************

  CLOSE   ALL
END
/*
```

Figure 19. Sample JCL for printing dumps through IPCS in the z/OS environment

Related concepts

"Processing a dump using the IBM MQ for z/OS dump display panels" on page 403

You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

"Processing a dump using line mode IPCS" on page 407
Use the IPCS commands to format a dump.

"Analyzing the dump and interpreting dump titles" on page 415
Use this topic to understand how dump titles are formatted, and how to analyze a dump.

Analyzing the dump and interpreting dump titles

Use this topic to understand how dump titles are formatted, and how to analyze a dump.

- Analyzing the dump
- Dump title variation with PSW and ASID

Analyzing the dump

The dump title includes the abend completion and reason codes, the failing load module and CSECT names, and the release identifier. For more information on the dump title see Dump title variation with PSW and ASID

The formats of SVC dump titles vary slightly, depending on the type of error.

Figure 20 on page 415 shows an example of an SVC dump title. Each field in the title is described after the figure.

```
ssnm,ABN=5C6-00D303F2,U=AUSER,C=R3600. 710.LOCK-CSQL1GET,
 M=CSQGFRCV,LOC=CSQLLPLM.CSQL1GET+0246
```

Figure 20. Sample SVC dump title

`ssnm,ABN=compltn-reason`

- `ssnm` is the name of the subsystem that issued the dump.
- `compltn` is the 3-character hexadecimal abend completion code (in this example, X'5C6'), prefixed by U for user abend codes.
- `reason` is the 4-byte hexadecimal reason code (in this example, X'00D303F2').

Note: The abend and reason codes might provide sufficient information to resolve the problem. See the IBM MQ for z/OS messages, completion, and reason codes for an explanation of the reason code.

`U=userid`

- `userid` is the user identifier of the user (in this example, AUSER). This field is not present for channel initiators.

`C=compid.release.comp-function`

- `compid` is the last 5 characters of the component identifier. The value R3600 uniquely identifies IBM MQ for z/OS.
- `release` is a 3-digit code indicating the version, release, and modification level of IBM MQ for z/OS (in this example, 710).
- `comp` is an acronym for the component in control at the time of the abend (in this example, LOCK).
- `function` is the name of a function, macro, or routine in control at the time of abend (in this example, CSQL1GET). This field is not always present.

`M=module`

- `module` is the name of the FRR or ESTAE recovery routine (in this example, CSQGFRCV). This field is not always present.

  Note: This is not the name of the module where the abend occurred; that is given by `LOC`.

`LOC=loadmod.csect+csect_offset`

- `loadmod` is the name of the load module in control at the time of the abend (in this example, CSQLLPLM). This might be represented by an asterisk if it is unknown.
- `csect` is the name of the CSECT in control at the time of abend (in this example, CSQL1GET).
- `csect_offset` is the offset within the failing CSECT at the time of abend (in this example, 0246).

Note: The value of `csect_offset` might vary if service has been applied to this CSECT, so do not use this value when building a keyword string to search the IBM software support database.

Dump title variation with PSW and ASID

Some dump titles replace the load module name, CSECT name, and CSECT offset with the PSW (program status word) and ASID (address space identifier). Figure 21 on page 416 illustrates this format.

```
ssnm,ABN=compltn-reason,U=userid,C=compid.release.comp-function,
 M=module,PSW=psw_contents,ASID=address_space_id
```

Figure 21. Dump title with PSW and ASID

`psw_contents`

- The PSW at the time of the error (for example, X'077C100000729F9C').

`address_space_id`

- The address space in control at the time of the abend (for example, X'0011'). This field is not present for a channel initiator.

Related concepts

"Processing a dump using the IBM MQ for z/OS dump display panels" on page 403
You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

"Processing a dump using line mode IPCS" on page 407
Use the IPCS commands to format a dump.

"Processing a dump using IPCS in batch" on page 414
Use this topic to understand how IBM MQ dumps can be formatted by IPCS commands in batch mode.

SYSUDUMP information

The z/OS system can create SYSUDUMPs, which can be used as part of problem determination. This topic shows a sample SYSUDUMP output and gives a reference to the tools for interpreting SYSUDUMPs.

SYSUDUMP dumps provide information useful for debugging batch and TSO application programs. For more information about SYSUDUMP dumps, see the *MVS Diagnosis: Tools and Service Aids* manual.

Figure 22 on page 417 shows a sample of the beginning of a SYSUDUMP dump.

```
JOB MQMBXBA1  STEP TSOUSER  TIME 102912   DATE 001019   ID = 000  CPUID = 632202333081
PAGE 00000001

COMPLETION CODE      SYSTEM = 0C1      REASON CODE = 00000001

PSW AT ENTRY TO ABEND  078D1000 000433FC        ILC 2  INTC 000D

PSW LOAD MODULE = BXBAAB01  ADDRESS = 000433FC  OFFSET = 0000A7F4

ASCB: 00F56400
+0000 ASCB..... ASCB      FWDP..... 00F60180  BWDP..... 0047800  CMSF..... 019D5A30
SVRB..... 008FE9E0
+0014 SYNC..... 00000D6F  IOSP..... 00000000  TNEW..... 00D18F0  CPUS..... 00000001
ASID..... 0066
+0026 R026..... 0000      LL5...... 00        HLHI..... 01       DPHI..... 00
DP....... 9D
+002C TRQP..... 80F5D381  LDA...... 7FF154E8  RSMF..... 00       R035..... 0000
TRQI..... 42
+0038 CSCB..... 00F4D048  TSB...... 00B61938  EJST..... 0000001  8C257E00

+0048 EWST..... 9CCDE747  76A09480            JSTL..... 00141A4  ECB...... 808FEF78
UBET..... 9CCDE740
 .
 .
 .
ASSB: 01946600
+0000 ASSB..... ASSB      VAFN..... 00000000  EVST..... 0000000  00000000

+0010 VFAT..... 00000000  00000000            RSV...... 000      XMCC..... 0000
XMCT.....00000000
+0020 VSC...... 00000000  NVSC..... 0000004C  ASRR..... 0000000  R02C..... 00000000
00000000 00000000
+0038           00000000  00000000

*** ADDRESS SPACE SWITCH EVENT MASK OFF (ASTESSEM = 0) ***

TCB: 008D18F0
+0000 RBP...... 008FE7D8  PIE...... 00000000  DEB...... 00B1530  TIO...... 008D4000
CMP......805C6000
+0014 TRN...... 40000000  MSS...... 7FFF7418  PKF...... 80       FLGS..... 01000000  00
+0022 LMP...... FF        DSP...... FE        LLS...... 00D1A88  JLB...... 00011F18
JPQ......00000000
+0030 GPR0-3... 00001000  008A4000  00000000  00000000
+0040 GPR4-7... 00FDC730  008A50C8  00000002  80E73F04
+0050 GPR8-11.. 81CC4360  008A6754  008A67B4  00000008
```

Figure 22. Sample beginning of a SYSUDUMP

Snap dumps

Snap dump data sets are controlled by z/OS JCL command statements. Use this topic to understand the CSQSNAP DD statement.

Snap dumps are always sent to the data set defined by the CSQSNAP DD statement. They can be issued by the adapters or the channel initiator.

- Snap dumps are issued by the batch, CICS, IMS, or RRS adapter when an unexpected error is returned by the queue manager for an MQI call. A full dump is produced containing information about the program that caused the problem.

  For a snap dump to be produced, the CSQSNAP DD statement must be in the batch application JCL, CICS JCL, or IMS dependent region JCL.

- Snap dumps are issued by the channel initiator in specific error conditions instead of a system dump. The dump contains information relating to the error. Message CSQX053E is also issued at the same time.

  To produce a snap dump, the CSQSNAP DD statement must be in the channel initiator started-task procedure.

SYS1.LOGREC information

Use this topic to understand how the z/OS SYS1.LOGREC information can assist with problem determination.

IBM MQ for z/OS and SYS1.LOGREC

The SYS1.LOGREC data set records various errors that different components of the operating system encounter. For more information about using SYS1.LOGREC records, see the *MVS Diagnosis: Tools and Service Aids* manual.

IBM MQ for z/OS recovery routines write information in the *system diagnostic work area* (SDWA) to the SYS1.LOGREC data set when retry is attempted, or when percolation to the next recovery routine occurs. Multiple SYS1.LOGREC entries can be recorded, because two or more retries or percolations might occur for a single error.

The SYS1.LOGREC entries recorded near the time of abend might provide valuable historical information about the events leading up to the abend.

Finding the applicable SYS1.LOGREC information

To obtain a SYS1.LOGREC listing, either:

- Use the EREP service aid, described in the *MVS Diagnosis: Tools and Service Aids* manual to format records in the SYS1.LOGREC data set.
- Specify the VERBEXIT LOGDATA keyword in IPCS.
- Use option 7 on the DUMP ANALYSIS MENU (refer to "Processing a dump using the IBM MQ for z/OS dump display panels" on page 403).

Only records available in storage when the dump was requested are included. Each formatted record follows the heading `*****LOGDATA*****`.

SVC dumps

Use this topic to understand how to suppress SVC dumps, and reasons why SVC dumps are not produced.

When SVC dumps are not produced

Under some circumstances, SVC dumps are not produced. Generally, dumps are suppressed because of time or space problems, or security violations. The following list summarizes other reasons why SVC dumps might not be produced:

- The z/OS *serviceability level indication processing* (SLIP) commands suppressed the abend.

  The description of IEACMD00 in the *MVS Initialization and Tuning Reference* manual lists the defaults for SLIP commands executed at IPL time.
- The abend reason code was one that does not require a dump to determine the cause of abend.
- SDWACOMU or SDWAEAS (part of the system diagnostic work area, SDWA) was used to suppress the dump.

Suppressing IBM MQ for z/OS dumps using z/OS DAE

You can suppress SVC dumps that duplicate previous dumps. The *MVS Diagnosis: Tools and Service Aids* manual gives details about using z/OS *dump analysis and elimination* (DAE).

To support DAE, IBM MQ for z/OS defines two *variable recording area* (VRA) keys and a minimum symptom string. The two VRA keys are:

- KEY VRADAE (X'53'). No data is associated with this key.
- KEY VRAMINSC (X'52') DATA (X'08')

IBM MQ for z/OS provides the following data for the minimum symptom string in the *system diagnostic work area* (SDWA):

- Load module name
- CSECT name
- Abend code
- Recovery routine name
- Failing instruction area
- REG/PSW difference
- Reason code
- Component identifier
- Component subfunction

Dumps are considered duplicates for the purpose of suppressing duplicate dumps if eight (the X'08' from the VRAMINSC key) of the nine symptoms are the same.

Dealing with performance problems on z/OS

Use this topic to investigate performance problems in more detail.

Performance problems are characterized by the following:

- Poor response times in online transactions
- Batch jobs taking a long time to complete
- The transmission of messages is slow

Performance problems can be caused by many factors, from a lack of resource in the z/OS system as a whole, to poor application design.

The following topics present problems and suggested solutions, starting with problems that are relatively simple to diagnose, such as DASD contention, through problems with specific subsystems, such as IBM MQ and CICS or IMS.

- "IBM MQ for z/OS system considerations" on page 419
- "CICS constraints" on page 420
- "Dealing with applications that are running slowly or have stopped on z/OS" on page 420

Remote queuing problems can be due to network congestion and other network problems. They can also be caused by problems at the remote queue manager.

Related concepts

"Making initial checks" on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

"Dealing with incorrect output" on page 425
Incorrect output can be missing, unexpected, or corrupted information. Read this topic to investigate further.

IBM MQ for z/OS system considerations

The z/OS system is an area that requires examination when investigating performance problems.

You might already be aware that your z/OS system is under stress because these problems affect many subsystems and applications.

You can use the standard monitoring tools such as Resource Monitoring Facility (RMF) to monitor and diagnose these problems. They might include:

- Constraints on storage (paging)

- Constraints on processor cycles
- Constraints on DASD
- Channel path usage

Use normal z/OS tuning techniques to resolve these problems.

CICS constraints

CICS constraints can also have an adverse effect on IBM MQ performance. Use this topic for further information about CICS constraints.

Performance of IBM MQ tasks can be affected by CICS constraints. For example, your system might have reached MAXTASK, forcing transactions to wait, or the CICS system might be short on storage. For example, CICS might not be scheduling transactions because the number of concurrent tasks has been reached, or CICS has detected a resource problem. If you suspect that CICS is causing your performance problems (for example because batch and TSO jobs run successfully, but your CICS tasks time out, or have poor response times), see the *CICS Problem Determination Guide* and the *CICS Performance Guide*.

Note: CICS I/O to transient data extrapartition data sets uses the z/OS RESERVE command. This could affect I/O to other data sets on the same volume.

Dealing with applications that are running slowly or have stopped on z/OS

Waits and loops can exhibit similar symptoms. Use the links in this topic to help differentiate between waits and loops.

Waits and loops are characterized by unresponsiveness. However, it can be difficult to distinguish between waits, loops, and poor performance.

Any of the following symptoms might be caused by a wait or a loop, or by a badly tuned or overloaded system:

- An application that appears to have stopped running (if IBM MQ for z/OS is still responsive, this problem is probably caused by an application problem)
- An MQSC command that does not produce a response
- Excessive use of processor time

To perform the tests shown in these topics, you need access to the z/OS console, and to be able to issue operator commands.

- "Distinguishing between waits and loops" on page 420
- "Dealing with waits" on page 422
- "Dealing with loops" on page 424

Related concepts

"Making initial checks" on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

Distinguishing between waits and loops

Waits and loops on IBM MQ can present similar symptoms. Use this topic to help determine if you are experiencing a wait of a loop.

Because waits and loops can be difficult to distinguish, in some cases you need to carry out a detailed investigation before deciding which classification is right for your problem.

This section gives you guidance about choosing the best classification, and advice on what to do when you have decided on a classification.

Waits

For problem determination, a wait state is regarded as the state in which the execution of a task has been suspended. That is, the task has started to run, but has been suspended without completing, and has subsequently been unable to resume.

A problem identified as a wait in your system could be caused by any of the following:

- A wait on an MQI call
- A wait on a CICS or IMS call
- A wait for another resource (for example, file I/O)
- An ECB wait
- The CICS or IMS region waiting
- TSO waiting
- IBM MQ for z/OS waiting for work
- An apparent wait, caused by a loop
- Your task is not being dispatched by CICS or MVS due to higher priority work
- Db2 or RRS are inactive

Loops

A loop is the repeated execution of some code. If you have not planned the loop, or if you have designed it into your application but it does not terminate for some reason, you get a set of symptoms that vary depending on what the code is doing, and how any interfacing components and products react to it. In some cases, at first, a loop might be diagnosed as a wait or performance problem, because the looping task competes for system resources with other tasks that are not involved in the loop. However, a loop consumes resources but a wait does not.

An apparent loop problem in your system could be caused by any of the following:

- An application doing a lot more processing than usual and therefore taking much longer to complete
- A loop in application logic
- A loop with MQI calls
- A loop with CICS or IMS calls
- A loop in CICS or IMS code
- A loop in IBM MQ for z/OS

Symptoms of waits and loops

Any of the following symptoms could be caused by a wait, a loop, or by a badly tuned or overloaded system:

- Timeouts on MQGET WAITs
- Batch jobs suspended
- TSO session suspended
- CICS task suspended
- Transactions not being started because of resource constraints, for example CICS MAX task
- Queues becoming full, and not being processed
- System commands not accepted, or producing no response

Related concepts
"Dealing with waits" on page 422

Waits can occur in batch or TSO applications, CICS transactions, and other components. Use this topic to determine where waits can occur.

“Dealing with loops” on page 424
Loops can occur in different areas of a z/OS system. Use this topic to help determine where a loop is occurring.

Dealing with waits

Waits can occur in batch or TSO applications, CICS transactions, and other components. Use this topic to determine where waits can occur.

When investigating what appears to be a problem with tasks or subsystems waiting, it is necessary to take into account the environment in which the task or subsystem is running.

It might be that your z/OS system is generally under stress. In this case, there can be many symptoms. If there is not enough real storage, jobs experience waits at paging interrupts or swap-outs. Input/output (I/O) contention or high channel usage can also cause waits.

You can use standard monitoring tools, such as *Resource Monitoring Facility* (RMF) to diagnose such problems. Use normal z/OS tuning techniques to resolve them.

Is a batch or TSO program waiting?

Consider the following points:

Your program might be waiting on another resource
For example, a VSAM control interval (CI) that another program is holding for update.

Your program might be waiting for a message that has not yet arrived

This condition might be normal behavior if, for example, it is a server program that constantly monitors a queue.

Alternatively, your program might be waiting for a message that has arrived, but has not yet been committed.

Issue the DIS CONN(*) TYPE(HANDLE) command and examine the queues in use by your program.

If you suspect that your program has issued an MQI call that did not involve an MQGET WAIT, and control has not returned from IBM MQ, take an SVC dump of both the batch or TSO job, and the IBM MQ subsystem before canceling the batch or TSO program.

Also consider that the wait state might be the result of a problem with another program, such as an abnormal termination (see “Messages do not arrive when expected” on page 426), or in IBM MQ itself (see “Is IBM MQ waiting for z/OS ?” on page 423). Refer to “IBM MQ for z/OS dumps” on page 400 (specifically Figure 13 on page 402) for information about obtaining a dump.

If the problem persists, refer to “Contacting IBM Software Support” on page 328 for information about reporting the problem to IBM.

Is a CICS transaction waiting?

Consider the following points:

CICS might be under stress
This might indicate that the maximum number of tasks allowed (MAXTASK) has been reached, or a short on storage (SOS) condition exists. Check the console log for messages that might explain this (for example, SOS messages), or see the *CICS Problem Determination Guide*.

The transaction might be waiting for another resource
For example, this might be file I/O. You can use CEMT INQ TASK to see what the task is waiting for. If the resource type is MQSERIES your transaction is waiting on IBM MQ (either in an MQGET WAIT or a task switch). Otherwise see the *CICS Problem Determination Guide* to determine the reason for the wait.

The transaction might be waiting for IBM MQ for z/OS
: This might be normal, for example, if your program is a server program that waits for messages to arrive on a queue. Otherwise it might be the result of a transaction abend, for example (see "Messages do not arrive when expected" on page 426). If so, the abend is reported in the CSMT log.

The transaction might be waiting for a remote message
: If you are using distributed queuing, the program might be waiting for a message that has not yet been delivered from a remote system (for further information, refer to "Problems with missing messages when using distributed queuing" on page 427).

If you suspect that your program has issued an MQI call that did not involve an MQGET WAIT (that is, it is in a task switch), and control has not returned from IBM MQ, take an SVC dump of both the CICS region, and the IBM MQ subsystem before canceling the CICS transaction. Refer to "Dealing with loops" on page 424 for information about waits. Refer to "IBM MQ for z/OS dumps" on page 400 (specifically Figure 13 on page 402) for information about obtaining a dump.

If the problem persists, refer to "Contacting IBM Software Support" on page 328 for information about reporting the problem to IBM.

Is Db2 waiting?

If your investigations indicate that Db2 is waiting, check the following:

1. Use the Db2 -DISPLAY THREAD(*) command to determine if any activity is taking place between the queue manager and the Db2 subsystem.
2. Try and determine whether any waits are local to the queue manager subsystems or are across the Db2 subsystems.

Is RRS active?

- Use the D RRS command to determine if RRS is active.

Is IBM MQ waiting for z/OS ?

If your investigations indicate that IBM MQ itself is waiting, check the following:

1. Use the DISPLAY THREAD(*) command to check if anything is connected to IBM MQ.
2. Use SDSF DA, or the z/OS command `DISPLAY A,xxxxMSTR` to determine whether there is any processor usage (as shown in "Has your application or IBM MQ for z/OS stopped processing work?" on page 32).
   - If IBM MQ is using some processor time, reconsider other reasons why IBM MQ might be waiting, or consider whether this is actually a performance problem.
   - If there is no processor activity, check whether IBM MQ responds to commands. If you can get a response, reconsider other reasons why IBM MQ might be waiting.
   - If you cannot get a response, check the console log for messages that might explain the wait (for example, IBM MQ might have run out of active log data sets, and be waiting for offload processing).

If you are satisfied that IBM MQ has stalled, use the STOP QMGR command in both QUIESCE and FORCE mode to terminate any programs currently being executed.

If the STOP QMGR command fails to respond, cancel the queue manager with a dump, and restart. If the problem recurs, refer to "Contacting IBM Software Support" on page 328 for further guidance.

Related concepts

"Distinguishing between waits and loops" on page 420
Waits and loops on IBM MQ can present similar symptoms. Use this topic to help determine if you are experiencing a wait of a loop.

"Dealing with loops" on page 424

Loops can occur in different areas of a z/OS system. Use this topic to help determine where a loop is occurring.

Dealing with loops

Loops can occur in different areas of a z/OS system. Use this topic to help determine where a loop is occurring.

The following topics describe the various types of loop that you might encounter, and suggest some responses.

Is a batch application looping?

If you suspect that a batch or TSO application is looping, use the console to issue the z/OS command `DISPLAY JOBS,A` (for a batch application) or `DISPLAY TS,A` (for a TSO application). Note the CT values from the data displayed, and repeat the command.

If any task shows a significant increase in the CT value, it might be that the task is looping. You could also use SDSF DA, which shows you the percentage of processor that each address space is using.

Is a batch job producing a large amount of output?

An example of this behavior might be an application that browses a queue and prints the messages. If the browse operation has been started with BROWSE FIRST, and subsequent calls have not been reset to BROWSE NEXT, the application browses, and prints the first message on the queue repeatedly.

You can use SDSF DA to look at the output of running jobs if you suspect that it might be causing a problem.

Does a CICS region show heavy processor activity?

It might be that a CICS application is looping, or that the CICS region itself is in a loop. You might see AICA abends if a transaction goes into a tight (unyielding) loop.

If you suspect that CICS, or a CICS application is looping, see the *CICS Problem Determination Guide*.

Does an IMS region show heavy processor activity?

It might be that an IMS application is looping. If you suspect this behavior, see *IMS Diagnosis Guide and Reference* l.

Is the queue manager showing heavy processor activity?

Try to enter an MQSC DISPLAY command from the console. If you get no response, it is possible that the queue manager is looping. Follow the procedure shown in “Has your application or IBM MQ for z/OS stopped processing work?” on page 32 to display information about the processor time being used by the queue manager. If this command indicates that the queue manager is in a loop, take a memory dump, cancel the queue manager and restart.

If the problem persists, see “Contacting IBM Software Support” on page 328 for information about reporting the problem to IBM.

Is a queue, page set, or Coupling Facility structure filling up unexpectedly?

If so, it might indicate that an application is looping, and putting messages on to a queue. (It might be a batch, CICS, or TSO application.)

Identifying a looping application

In a busy system, it might be difficult to identify which application is causing the problem. If you keep a cross-reference of applications to queues, terminate any programs or transactions that might be putting messages on to the queue. Investigate these programs or transactions before using them again. (The most likely culprits are new, or changed applications; check your change log to identify them.)

Try issuing a DISPLAY QSTATUS command on the queue. This command returns information about the queue that might help to identify which application is looping.

Incorrect triggering definitions

It might be that a getting application has not been triggered because of incorrect object definitions, for example, the queue might be set to NOTRIGGER.

Distributed queuing

Using distributed queuing, a symptom of this problem might be a message in the receiving system indicating that MQPUT calls to the dead-letter queue are failing. This problem might be caused because the dead-letter queue has also filled up. The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message might not be put on to the target queue. See MQDLH - Dead-letter header for information about the dead-letter header structure.

Allocation of queues to page sets

If a particular page set frequently fills up, there might be a problem with the allocation of queues to page sets. See IBM MQ for z/OS performance constraints for more information.

Shared queues

Is the Coupling Facility structure full? The z/OS command DISPLAY CF displays information about Coupling Facility storage including the total amount, the total in use, and the total free control and non-control storage. The RMF Coupling Facility Usage Summary Report provides a more permanent copy of this information.

Are a task, and IBM MQ for z/OS, showing heavy processor activity?

In this case, a task might be looping on MQI calls (for example, browsing the same message repeatedly).

Related concepts

“Distinguishing between waits and loops” on page 420
Waits and loops on IBM MQ can present similar symptoms. Use this topic to help determine if you are experiencing a wait of a loop.

“Dealing with waits” on page 422
Waits can occur in batch or TSO applications, CICS transactions, and other components. Use this topic to determine where waits can occur.

Dealing with incorrect output

Incorrect output can be missing, unexpected, or corrupted information. Read this topic to investigate further.

The term "incorrect output˒ can be interpreted in many different ways, and its meaning for problem determination with this product documentation is explained in “Have you obtained incorrect output?” on page 40.

The following topics contains information about the problems that you could encounter with your system and classify as incorrect output:

- Application messages that do not arrive when you are expecting them
- Application messages that contain the wrong information, or information that has been corrupted

Additional problems that you might encounter if your application uses distributed queues are also described.

- “Messages do not arrive when expected” on page 426
- “Problems with missing messages when using distributed queuing” on page 427
- “Problems with getting messages when using message grouping” on page 429
- “Finding messages sent to a cluster queue” on page 429
- “Finding messages sent to the IBM MQ - IMS bridge” on page 429
- “Messages contain unexpected or corrupted information” on page 430

Related concepts

“Making initial checks” on page 8
There are some initial checks that you can make that may provide answers to common problems that you may have.

“Dealing with performance problems on z/OS” on page 419
Use this topic to investigate performance problems in more detail.

Messages do not arrive when expected

Missing messages can have different causes. Use this topic to investigate the causes further.

If messages do not arrive on the queue when you are expecting them, check for the following:

Has the message been put onto the queue successfully?

Did IBM MQ issue a return and reason code for the MQPUT, for example:

- Has the queue been defined correctly, for example is MAXMSGL large enough? (reason code 2030).
- Can applications put messages on to the queue (is the queue enabled for MQPUT calls)? (reason code 2051).
- Is the queue already full? This could mean that an application could not put the required message on to the queue (reason code 2053).

Is the queue a shared queue?

- Have Coupling Facility structures been defined successfully in the CFRM policy data set? Messages held on shared queues are stored inside a Coupling Facility.
- Have you activated the CFRM policy?

Is the queue a cluster queue?

If it is, there might be multiple instances of the queue on different queue managers. This means that the messages could be on a different queue manager.

- Did you want the message to go to a cluster queue?
- Is your application designed to work with cluster queues?
- Did the message get put to a different instance of the queue from that expected?

Check any cluster-workload exit programs to see that they are processing messages as intended.

Do your gets fail?

- Does the application need to take a syncpoint?

  If messages are being put or got within syncpoint, they are not available to other tasks until the unit of recovery has been committed.
- Is the time interval on the MQGET long enough?

  If you are using distributed processing, you should allow for reasonable network delays, or problems at the remote end.
- Was the message you are expecting defined as persistent?

If not, and the queue manager has been restarted, the message will have been deleted. Shared queues are an exception because nonpersistent messages survive a queue manager restart.

- Are you waiting for a specific message that is identified by a message or correlation identifier (*MsgId* or *CorrelId*)?

  Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful MQGET call sets both these values to that of the message got, so you might need to reset these values to get another message successfully.

  Also check if you can get other messages from the queue.
- Can other applications get messages from the queue?

  If so, has another application already retrieved the message?

  If the queue is a shared queue, check that applications on other queue managers are not getting the messages.

If you cannot find anything wrong with the queue, and the queue manager itself is running, make the following checks on the process that you expected to put the message on to the queue:

- Did the application get started?

  If it should have been triggered, check that the correct trigger options were specified.
- Is a trigger monitor running?
- Was the trigger process defined correctly (both to IBM MQ for z/OS and CICS or IMS)?
- Did it complete correctly?

  Look for evidence of an abend, for example, in the CICS log.
- Did the application commit its changes, or were they backed out?

  Look for messages in the CICS log indicating this.

If multiple transactions are serving the queue, they might occasionally conflict with one another. For example, one transaction might issue an MQGET call with a buffer length of zero to find out the length of the message, and then issue a specific MQGET call specifying the *MsgId* of that message. However, while this is happening, another transaction might have issued a successful MQGET call for that message, so the first application receives a completion code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multi-server environment must be designed to cope with this situation.

Have any of your systems suffered an outage? For example, if the message you were expecting should have been put on to the queue by a CICS application, and the CICS system went down, the message might be in doubt. This means that the queue manager does not know whether the message should be committed or backed out, and so has locked it until this is resolved when resynchronization takes place.

Note: The message is deleted after resynchronization if CICS decides to back it out.

Also consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If so, refer to "Messages contain unexpected or corrupted information" on page 430.

Problems with missing messages when using distributed queuing

Use this topic tom understand possible causes of missing messages when using distributed queuing.

If your application uses distributed queuing, consider the following points:

Has distributed queuing been correctly installed on both the sending and receiving systems?
Ensure that the instructions about installing the distributed queue management facility in Configuring z/OS have been followed correctly.

Are the links available between the two systems?
Check that both systems are available, and connected to IBM MQ for z/OS. Check that the LU 6.2 or TCP/IP connection between the two systems is active or check the connection definitions on any other systems that you are communicating with.

See Monitoring and performance for more information about trace-route messaging in a network.

Is the channel running?

- Issue the following command for the transmission queue:

  ```
  DISPLAY QUEUE (qname) IPPROCS
  ```

  If the value for IPPROCS is 0, this means that the channel serving this transmission queue is not running.
- Issue the following command for the channel:

  ```
  DISPLAY CHSTATUS (channel-name) STATUS MSGS
  ```

  Use the output produced by this command to check that the channel is serving the correct transmission queue and that it is connected to the correct target machine and port. You can determine whether the channel is running from the STATUS field. You can also see if any messages have been sent on the channel by examining the MSGS field.

  If the channel is in RETRYING state, this is probably caused by a problem at the other end. Check that the channel initiator and listener have been started, and that the channel has not been stopped. If somebody has stopped the channel, you need to start it manually.

Is triggering set on in the sending system?
Check that the channel initiator is running.

Does the transmission queue have triggering set on?
If a channel is stopped under specific circumstances, triggering can be set off for the transmission queue.

Is the message you are waiting for a reply message from a remote system?
Check the definitions of the remote system, as previously described, and check that triggering is activated in the remote system. Also check that the LU 6.2 connection between the two systems is not single session (if it is, you cannot receive reply messages).

Check that the queue on the remote queue manager exists, is not full, and accepts the message length. If any of these criteria are not fulfilled, the remote queue manager tries to put the message on the dead-letter queue. If the message length is longer than the maximum length that the channel permits, the sending queue manager tries to put the message on its dead-letter queue.

Is the queue already full?

This could mean that an application could not put the required message on to the queue. If this is so, check if the message has been put on to the dead-letter queue.

The dead-letter queue message header (dead-letter header structure) contains a reason or feedback code explaining why the message could not be put on to the target queue. See MQDLH - Dead-letter header for more information about the dead-letter header structure.

Is there a mismatch between the sending and receiving queue managers?
For example, the message length could be longer than the receiving queue manager can handle. Check the console log for error messages.

Are the channel definitions of the sending and receiving channels compatible?
For example, a mismatch in the wrap value of the sequence number stops the channel. See Distributed queuing and clusters.

Has data conversion been performed correctly?
If a message has come from a different queue manager, are the CCSIDs and encoding the same, or does data conversion need to be performed.

Has your channel been defined for fast delivery of nonpersistent messages?

If your channel has been defined with the NPMSPEED attribute set to FAST (the default), and the channel has stopped for some reason and then been restarted, nonpersistent messages might have been lost. See Nonpersistent message speed (NPMSPEED) for more information about fast messages.

Is a channel exit causing the messages to be processed in an unexpected way?

For example, a security exit might prevent a channel from starting, or an *ExitResponse* of MQXCC_CLOSE_CHANNEL might terminate a channel.

Problems with getting messages when using message grouping

Use this topic to understand some of the issues with getting messages when using message grouping.

Is the application waiting for a complete group of messages?

Ensure all the messages in the group are on the queue. If you are using distributed queuing, see "Problems with missing messages when using distributed queuing" on page 427. Ensure the last message in the group has the appropriate MsgFlags set in the message descriptor to indicate that it is the last message. Ensure the message expiry of the messages in the group is set to a long enough interval that they do not expire before they are retrieved.

If messages from the group have already been retrieved, and the get request is not in logical order, turn off the option to wait for a complete group when retrieving the other group messages.

If the application issues a get request in logical order for a complete group, and midway through retrieving the group it cannot find a message:

Ensure that no other applications are running against the queue and getting messages. Ensure that the message expiry of the messages in the group is set to a long enough interval that they do not expire before they are retrieved. Ensure that no one has issued the CLEAR QUEUE command. You can retrieve incomplete groups from a queue by getting the messages by group ID, without specifying the logical order option.

Finding messages sent to a cluster queue

Use this topic to understand some of the issues involved with finding messages sent to a cluster queue.

Before you can use the techniques described in these topics to find a message that did not arrive at a cluster queue, you need to determine the queue managers that host the queue to which the message was sent. You can determine this in the following ways:

- You can use the DISPLAY QUEUE command to request information about cluster queues.
- You can use the name of the queue and queue manager that is returned in the MQPMO structure.

  If you specified the MQOO_BIND_ON_OPEN option for the message, these fields give the destination of the message. If the message was not bound to a particular queue and queue manager, these fields give the name of the first queue and queue manager to which the message was sent. In this case, it might not be the ultimate destination of the message.

Finding messages sent to the IBM MQ - IMS bridge

Use this topic to understand possible causes for missing messages sent to the IBM MQ - IMS bridge.

If you are using the IBM MQ - IMS bridge, and your message has not arrived as expected, consider the following:

Is the IBM MQ - IMS bridge running?

Issue the following command for the bridge queue:

```
DISPLAY QSTATUS(qname) IPPROCS CURDEPTH
```

The value of IPPROCS should be 1; if it is 0, check the following:

- Is the queue a bridge queue?

- Is IMS running?
- Has OTMA been started?
- Is IBM MQ connected to OTMA?

  Note: There are two IBM MQ messages that you can use to establish whether you have a connection to OTMA. If message CSQ2010I is present in the job log of the task, but message CSQ2011I is not present, IBM MQ is connected to OTMA. This message also tells you to which IBM MQ system OTMA is connected. For more information about the content of these messages, see IBM MQ for z/OS messages, completion, and reason codes.

Within the queue manager there is a task processing each IMS bridge queue. This task gets from the queue, sends the request to IMS, and then does a commit. If persistent messages are used, then the commit requires disk I/O and so the process takes longer than for non-persistent messages. The time to process the get, send, and commit, limits the rate at which the task can process messages. If the task can keep up with the workload then the current depth is close to zero. If you find that the current depth is often greater than zero you might be able to increase throughput by using two queues instead of one.

Use the IMS command `/DIS OTMA` to check that OTMA is active.

If your messages are flowing to IMS, check the following:

- Use the IMS command `/DIS TMEMBER client TPIPE ALL` to display information about IMS Tpipes. From this you can determine the number of messages enqueued on, and dequeued from, each Tpipe. (Commit mode 1 messages are not usually queued on a Tpipe.)
- Use the IMS command `/DIS A` to show whether there is a dependent region available for the IMS transaction to run in.
- Use the IMS command `/DIS TRAN trancode` to show the number of messages queued for a transaction.
- Use the IMS command `/DIS PROG progname` to show if a program has been stopped.

Was the reply message sent to the correct place?

Issue the following command:

```
DISPLAY QSTATUS(*) CURDEPTH
```

Does the CURDEPTH indicate that there is a reply on a queue that you are not expecting?

Messages contain unexpected or corrupted information

Use this topic to understand some of the issues that can cause unexpected or corrupted output.

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following points:

Has your application, or the application that put the message on to the queue changed?

Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.

For example, a copybook formatting the message might have been changed, in which case, both applications have to be recompiled to pick up the changes. If one application has not been recompiled, the data will appear corrupt to the other.

Check that no external source of data, such as a VSAM data set, has changed. This could also invalidate your data if any necessary recompilations have not been done. Also check that any CICS maps and TSO panels that you are using for input of message data have not changed.

Is an application sending messages to the wrong queue?

Check that the messages your application is receiving are not intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.

If your application has used an alias queue, check that the alias points to the correct queue.

If you altered the queue to make it a cluster queue, it might now contain messages from different application sources.

Has the trigger information been specified correctly for this queue?

Check that your application should have been started, or should a different application have been started?

Has data conversion been performed correctly?

If a message has come from a different queue manager, are the CCSIDs and encoding the same, or does data conversion need to be performed.

Check that the *Format* field of the MQMD structure corresponds with the content of the message. If not, the data conversion process might not have been able to deal with the message correctly.

If these checks do not enable you to solve the problem, check your application logic, both for the program sending the message, and for the program receiving it.

Dealing with issues when capturing SMF data for the channel initiator (CHINIT)

Channel accounting and CHINIT statistics SMF data might not be captured for various reasons.

For more information, see:

Related information

Layout of SMF records for the channel initiator

Troubleshooting channel accounting data

Checks to carry out if channel accounting SMF data is not being produced for channels.

Procedure

1. Check that you have STATCHL set, either at the queue manager or the channel level.
   - A value of OFF at channel level means that data is not collected for this channel.
   - A value of OFF at queue manager level means data is not collected for channels with STATCHL(QMGR).
   - A value of NONE (only applicable at queue manager level) means data is not collected for all channels, regardless of their STATCHL setting.
2. For client channels check that STATCHL is set at the queue manager level.
3. For automatically defined cluster sender channels, check that the STATACLS is set.
4. Issue the display trace command. You need TRACE(A) CLASS(4) enabled for channel accounting data to be collected.
5. If the trace is enabled, SMF data is written:
   - On a timed interval, depending on the value of the STATIME system parameter. A value of zero means that the SMF statistics broadcast is used. Use the DIS SYSTEM command to display the value of STATIME.
   - If the SET SYSTEM command is issued to change the value of the STATIME system parameter.
   - When the CHINIT is shut down.
   - If the STOP TRACE(A) CLASS(4) is issued, any accounting data is written out.

6. SMF might hold the data in memory before writing it out to the SMF data sets or the SMF structure. Issue the MVS command **D SMF,O** and note the MAXDORM value. SMF can keep the data in memory for the MAXDORM period before writing it out.

Related information

Planning for channel initiator SMF data

Interpreting IBM MQ performance statistics

Troubleshooting CHINIT statistics data

Checks to carry out if CHINIT statistics SMF data is not being produced.

Procedure

1. Issue the display trace command. You need TRACE(S) CLASS(4) enabled for information about the CHINIT.
2. If the trace is enabled, SMF data is written:
   - On a timed interval, depending on the value of the STATIME system parameter. A value of zero means that the SMF statistics broadcast is used. Use the DIS SYSTEM command to display the value of STATIME.
   - If the SET SYSTEM command is issued to change the value of the STATIME system parameter.
   - When the CHINIT is shut down.
   - If the STOP TRACE(S) CLASS(4) is issued, any statistics data is written out.
3. SMF can hold the data in memory before writing it out to the SMF data sets or the SMF structure. Issue the MVS command **D SMF,O** and note the MAXDORM value. SMF can keep the data in memory for the MAXDORM period before writing it out.

Problem determination in DQM

Aspects of problem determination relating to distributed queue management (DQM) and suggested methods of resolving problems.

Some of the problems that are described are platform and installation specific. Where this is the case, it is made clear in the text.

IBM MQ provides a utility to assist with problem determination named **amqldmpa**. During the course of problem determination, your IBM service representative might ask you to provide output from the utility.

Your IBM service representative will provide you with the parameters you require to collect the appropriate diagnostic information, and information on how you send the data you record to IBM.

Attention: You should not rely on the format of the output from this utility, as the format is subject to change without notice.

Problem determination for the following scenarios is discussed:

- “Error message from channel control” on page 433
- “Ping” on page 433
- “Dead-letter queue considerations” on page 434
- “Validation checks” on page 434
- “In-doubt relationship” on page 434
- “Channel startup negotiation errors” on page 435
- “When a channel refuses to run” on page 435
- “Retrying the link” on page 437
- “Data structures” on page 438
- “User exit problems” on page 438

- “Disaster recovery” on page 438
- “Channel switching” on page 439
- “Connection switching” on page 439
- “Client problems” on page 439
- “Error logs” on page 440
- “Message monitoring” on page 441

Related concepts

“Troubleshooting and support” on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

“Making initial checks on Windows, UNIX and Linux systems” on page 9
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks on z/OS” on page 28
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Making initial checks on IBM i” on page 18
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

“Reason codes and exceptions” on page 43
You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

Related information

Configuring distributed queuing

Communications protocol return codes

Error message from channel control

Problems found during normal operation of the channels are reported to the system console and to the system log. In IBM MQ for Windows they are reported to the channel log. Problem diagnosis starts with the collection of all relevant information from the log, and analysis of this information to identify the problem.

However, this could be difficult in a network where the problem may arise at an intermediate system that is staging some of your messages. An error situation, such as transmission queue full, followed by the dead-letter queue filling up, would result in your channel to that site closing down.

In this example, the error message you receive in your error log will indicate a problem originating from the remote site, but may not be able to tell you any details about the error at that site.

You need to contact your counterpart at the remote site to obtain details of the problem, and to receive notification of that channel becoming available again.

Ping

Ping is useful in determining whether the communication link and the two message channel agents that make up a message channel are functioning across all interfaces.

Ping makes no use of transmission queues, but it does invoke some user exit programs. If any error conditions are encountered, error messages are issued.

To use ping, you can issue the MQSC command PING CHANNEL. On z/OS z/OS IBM i and i5/OS , you can also use the panel interface to select this option.

On UNIX platforms, IBM i i5/OS, and Windows, you can also use the MQSC command PING QMGR to test whether the queue manager is responsive to commands.

Dead-letter queue considerations

In some IBM MQ implementations the dead-letter queue is referred to as an *undelivered-message queue*.

If a channel ceases to run for any reason, applications will probably continue to place messages on transmission queues, creating a potential overflow situation. Applications can monitor transmission queues to find the number of messages waiting to be sent, but this would not be a normal function for them to carry out.

When this occurs in a message-originating node, and the local transmission queue is full, the application's PUT fails.

When this occurs in a staging or destination node, there are three ways that the MCA copes with the situation:

1. By calling the message-retry exit, if one is defined.
2. By directing all overflow messages to a *dead-letter queue* (DLQ), returning an exception report to applications that requested these reports.

   Note: In distributed-queuing management, if the message is too big for the DLQ, the DLQ is full, or the DLQ is not available, the channel stops and the message remains on the transmission queue. Ensure your DLQ is defined, available, and sized for the largest messages you handle.
3. By closing down the channel, if neither of the previous options succeeded.
4. By returning the undelivered messages back to the sending end and returning a full report to the reply-to queue (MQRC_EXCEPTION_WITH_FULL_DATA and MQRO_DISCARD_MSG).

If an MCA is unable to put a message on the DLQ:

- The channel stops
- Appropriate error messages are issued at the system consoles at both ends of the message channel
- The unit of work is backed out, and the messages reappear on the transmission queue at the sending channel end of the channel
- Triggering is disabled for the transmission queue

Validation checks

A number of validation checks are made when creating, altering, and deleting channels, and where appropriate, an error message returned.

Errors may occur when:

- A duplicate channel name is chosen when creating a channel
- Unacceptable data is entered in the channel parameter fields
- The channel to be altered is in doubt, or does not exist

In-doubt relationship

If a channel is in doubt, it is usually resolved automatically on restart, so the system operator does not need to resolve a channel manually in normal circumstances. See In-doubt channels for further information.

Channel startup negotiation errors

During channel startup, the starting end has to state its position and agree channel running parameters with the corresponding channel. It may happen that the two ends cannot agree on the parameters, in which case the channel closes down with error messages being issued to the appropriate error logs.

Shared channel recovery

The following table shows the types of shared-channel failure and how each type is handled.

Type of failure:	**What happens:**
Channel initiator communications subsystem failure	The channels dependent on the communications subsystem enter channel retry, and are restarted on an appropriate queue-sharing group channel initiator by a load-balanced start command.
Channel initiator failure	The channel initiator fails, but the associated queue manager remains active. The queue manager monitors the failure and initiates recovery processing.
Queue manager failure	The queue manager fails (failing the associated channel initiator). Other queue managers in the queue-sharing group monitor the event and initiate peer recovery.
Shared status failure	Channel state information is stored in Db2, so a loss of connectivity to Db2 becomes a failure when a channel state change occurs. Running channels can carry on running without access to these resources. On a failed access to Db2, the channel enters retry.

Shared channel recovery processing on behalf of a failed system requires connectivity to Db2 to be available on the system managing the recovery to retrieve the shared channel status.

When a channel refuses to run

If a channel refuses to run, there are a number of potential reasons.

Carry out the following checks:

- Check that DQM and the channels have been set up correctly. This is a likely problem source if the channel has never run. Reasons could be:
  - A mismatch of names between sending and receiving channels (remember that uppercase and lowercase letters are significant)
  - Incorrect channel types specified
  - The sequence number queue (if applicable) is not available, or is damaged
  - The dead-letter queue is not available
  - The sequence number wrap value is different on the two channel definitions
  - A queue manager or communication link is not available
  - A receiver channel might be in STOPPED state
  - The connection might not be defined correctly
  - There might be a problem with the communications software (for example, is TCP running?)
- It is possible that an in-doubt situation exists, if the automatic synchronization on startup has failed for some reason. This is indicated by messages on the system console, and the status panel may be used to show channels that are in doubt.

  The possible responses to this situation are:
  - Issue a Resolve channel request with Backout or Commit.

You need to check with your remote link supervisor to establish the number of the last-committed unit of work ID (LUWID) committed. Check this against the last number at your end of the link. If the remote end has committed a number, and that number is not yet committed at your end of the link, then issue a RESOLVE COMMIT command.

In all other cases, issue a RESOLVE BACKOUT command.

The effect of these commands is that backed out messages reappear on the transmission queue and are sent again, while committed messages are discarded.

If in doubt yourself, perhaps backing out with the probability of duplicating a sent message would be the safer decision.

– Issue a RESET CHANNEL command.

This command is for use when sequential numbering is in effect, and should be used with care. Its purpose is to reset the sequence number of messages and you should use it only after using the RESOLVE command to resolve any in-doubt situations.

- Windows IBM i z/OS UNIX When sequential numbering is being used, and a sender channel starts up after being reset, the sender channel takes two actions:
  – It tells the receiver channel that it has been reset.
  – It specifies the next message sequence number that is to be used by both the sender and receiver channels.
- If the status of a receiver end of the channel is STOPPED, it can be reset by starting the receiver end.

  Note: This does not start the channel, it merely resets the status. The channel must still be started from the sender end.

Triggered channels

If a triggered channel refuses to run, investigate the possibility of in-doubt messages here: “When a channel refuses to run” on page 435

Another possibility is that the trigger control parameter on the transmission queue has been set to NOTRIGGER by the channel. This happens when:

- There is a channel error.
- The channel was stopped because of a request from the receiver.
- The channel was stopped because of a problem on the sender that requires manual intervention.

After diagnosing and fixing the problem, start the channel manually.

An example of a situation where a triggered channel fails to start is as follows:

1. A transmission queue is defined with a trigger type of FIRST.
2. A message arrives on the transmission queue, and a trigger message is produced.
3. The channel is started, but stops immediately because the communications to the remote system are not available.
4. The remote system is made available.
5. Another message arrives on the transmission queue.
6. The second message does not increase the queue depth from zero to one, so no trigger message is produced (unless the channel is in RETRY state). If this happens, restart the channel manually.

On IBM MQ for z/OS, if the queue manager is stopped using MODE(FORCE) during channel initiator shutdown, it might be necessary to manually restart some channels after channel initiator restart.

Conversion failure

Another reason for the channel refusing to run could be that neither end is able to carry out necessary conversion of message descriptor data between ASCII and EBCDIC, and integer formats. In this instance, communication is not possible.

Network problems

There are a number of things to check if you are experiencing network problems.

When using LU 6.2, make sure that your definitions are consistent throughout the network. For example, if you have increased the RU sizes in your CICS Transaction Server for z/OS or Communications Manager definitions, but you have a controller with a small MAXDATA value in its definition, the session might fail if you attempt to send large messages across the network. A symptom of this problem might be that channel negotiation takes place successfully, but the link fails when message transfer occurs.

When using TCP, if your channels are unreliable and your connections break, you can set a KEEPALIVE value for your system or channels. You do this using the SO_KEEPALIVE option to set a system-wide value.

z/OS On IBM MQ for z/OS, you also have the following options:

- Use the Keepalive Interval channel attribute (KAINT) to set channel-specific keepalive values.
- Use the RCVTIME and RCVTMIN channel initiator parameters.

These options are discussed in Checking that the other end of the channel is still available, and Keepalive Interval (KAINT).

Adopting an MCA

The Adopt MCA function enables IBM MQ to cancel a receiver channel and to start a new one in its place.

For more information about this function, see Adopting an MCA.

Registration time for DDNS

When a group TCP/IP listener is started, it registers with DDNS. But there can be a delay until the address is available to the network. A channel that is started in this period, and which targets the newly registered generic name, fails with an 'error in communications configuration' message. The channel then goes into retry until the name becomes available to the network. The length of the delay is dependent on the name server configuration used.

Dial-up problems

IBM MQ supports connection over dial-up lines but you should be aware that with TCP, some protocol providers assign a new IP address each time you dial in. This can cause channel synchronization problems because the channel cannot recognize the new IP addresses and so cannot ensure the authenticity of the partner. If you encounter this problem, you need to use a security exit program to override the connection name for the session.

This problem does not occur when an IBM MQ for IBM i, UNIX systems, or Windows systems product is communicating with another product at the same level, because the queue manager name is used for synchronization instead of the IP address.

Retrying the link

An error scenario may occur that is difficult to recognize. For example, the link and channel may be functioning perfectly, but some occurrence at the receiving end causes the receiver to stop. Another unforeseen situation could be that the receiver system has run out of memory and is unable to complete a transaction.

You need to be aware that such situations can arise, often characterized by a system that appears to be busy but is not actually moving messages. You need to work with your counterpart at the far end of the link to help detect the problem and correct it.

Retry considerations

If a link failure occurs during normal operation, a sender or server channel program will itself start another instance, provided that:

1. Initial data negotiation and security exchanges are complete
2. The retry count in the channel definition is greater than zero

Note: For IBM i, UNIX systems, and Windows, to attempt a retry a channel initiator must be running. In platforms other than IBM MQ for IBM i, UNIX systems, and Windows systems, this channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel is using.

Shared channel recovery on z/OS

See "Shared channel recovery" on page 435, which includes a table that shows the types of shared-channel failure and how each type is handled.

Data structures

Data structures are needed for reference when checking logs and trace entries during problem diagnosis.

More information can be found in Channel-exit calls and data structures and Developing applications reference.

User exit problems

The interaction between the channel programs and the user-exit programs has some error-checking routines, but this facility can only work successfully when the user exits obey certain rules.

These rules are described in Channel-exit programs for messaging channels. When errors occur, the most likely outcome is that the channel stops and the channel program issues an error message, together with any return codes from the user exit. Any errors detected on the user exit side of the interface can be determined by scanning the messages created by the user exit itself.

You might need to use a trace facility of your host system to identify the problem.

Disaster recovery

Disaster recovery planning is the responsibility of individual installations, and the functions performed may include the provision of regular system 'snapshot' dumps that are stored safely off-site. These dumps would be available for regenerating the system, should some disaster overtake it. If this occurs, you need to know what to expect of the messages, and the following description is intended to start you thinking about it.

First a recap on system restart. If a system fails for any reason, it may have a system log that allows the applications running at the time of failure to be regenerated by replaying the system software from a syncpoint forward to the instant of failure. If this occurs without error, the worst that can happen is that message channel syncpoints to the adjacent system may fail on startup, and that the last batches of messages for the various channels will be sent again. Persistent messages will be recovered and sent again, nonpersistent messages may be lost.

If the system has no system log for recovery, or if the system recovery fails, or where the disaster recovery procedure is invoked, the channels and transmission queues may be recovered to an earlier state, and the messages held on local queues at the sending and receiving end of channels may be inconsistent.

Messages may have been lost that were put on local queues. The consequence of this happening depends on the particular IBM MQ implementation, and the channel attributes. For example, where strict message sequencing is in force, the receiving channel detects a sequence number gap, and the channel closes down for manual intervention. Recovery then depends upon application design, as in the worst case the sending application may need to restart from an earlier message sequence number.

Channel switching

A possible solution to the problem of a channel ceasing to run would be to have two message channels defined for the same transmission queue, but with different communication links. One message channel would be preferred, the other would be a replacement for use when the preferred channel is unavailable.

If triggering is required for these message channels, the associated process definitions must exist for each sender channel end.

To switch message channels:

- If the channel is triggered, set the transmission queue attribute NOTRIGGER.
- Ensure the current channel is inactive.
- Resolve any in-doubt messages on the current channel.
- If the channel is triggered, change the process attribute in the transmission queue to name the process associated with the replacement channel.

  In this context, some implementations allow a channel to have a blank process object definition, in which case you may omit this step as the queue manager will find and start the appropriate process object.
- Restart the channel, or if the channel was triggered, set the transmission queue attribute TRIGGER.

Connection switching

Another solution would be to switch communication connections from the transmission queues.

To do this:

- If the sender channel is triggered, set the transmission queue attribute NOTRIGGER.
- Ensure the channel is inactive.
- Change the connection and profile fields to connect to the replacement communication link.
- Ensure that the corresponding channel at the remote end has been defined.
- Restart the channel, or if the sender channel was triggered, set the transmission queue attribute TRIGGER.

Client problems

A client application may receive an unexpected error return code, for example:

- Queue manager not available
- Queue manager name error
- Connection broken

Look in the client error log for a message explaining the cause of the failure. There may also be errors logged at the server, depending on the nature of the failure.

Terminating clients

Even though a client has terminated, it is still possible for its surrogate process to be holding its queues open. Normally this will only be for a short time until the communications layer notifies that the partner has gone.

Error logs

IBM MQ error messages are placed in different error logs depending on the platform. There are error logs for:

- Windows
- UNIX systems
- z/OS

Error logs for Windows

IBM MQ for Windows uses a number of error logs to capture messages concerning the operation of IBM MQ itself, any queue managers that you start, and error data coming from the channels that are in use.

The location the error logs are stored in depends on whether the queue manager name is known and whether the error is associated with a client.

- If the queue manager name is known and the queue manager is available:

  ```
  <install directory>\QMGRS\QMgrName\ERRORS\AMQERR01.LOG
  ```

- If the queue manager is not available:

  ```
  <install directory>\QMGRS\@SYSTEM\ERRORS\AMQERR01.LOG
  ```

- If an error has occurred with a client application:

  ```
  <install directory>\ERRORS\AMQERR01.LOG
  ```

On Windows, you should also examine the Windows application event log for relevant messages.

Error logs on UNIX and Linux systems

IBM MQ on UNIX and Linux systems uses a number of error logs to capture messages concerning the operation of IBM MQ itself, any queue managers that you start, and error data coming from the channels that are in use. The location the error logs are stored in depends on whether the queue manager name is known and whether the error is associated with a client.

- If the queue manager name is known:

  ```
  /var/mqm/qmgrs/QMgrName/errors
  ```

- If the queue manager name is not known (for example when there are problems in the listener or SSL handshake):

  ```
  /var/mqm/errors
  ```

When a client is installed, and there is a problem in the client application, the following log is used:

- If an error has occurred with a client application:

  ```
  /var/mqm/errors/
  ```

Error logs on z/OS

Error messages are written to:

- The z/OS system console
- The channel-initiator job log

If you are using the z/OS message processing facility to suppress messages, the console messages might be suppressed. See Planning your IBM MQ environment on z/OS.

Message monitoring

If a message does not reach its intended destination, you can use the IBM MQ display route application, available through the control command `dspmqrte` , to determine the route a message takes through the queue manager network and its final location.

The IBM MQ display route application is described in the IBM MQ display route application section.

Channel authentication records troubleshooting

If you are having problems using channel authentication records, check whether the problem is described in the following information.

What address are you presenting to the queue manager?

The address that your channel presents to the queue manager depends on the network adapter being used. For example, if the CONNAME you use to get to the listener is "localhost", you present 127.0.0.1 as your address; if it is the real IP address of your computer, then that is the address you present to the queue manager. You might invoke different authentication rules for 127.0.0.1 and your real IP address.

Using BLOCKADDR with channel names

If you use SET CHLAUTH TYPE(BLOCKADDR), it must have the generic channel name CHLAUTH(*) and nothing else. You must block access from the specified addresses using any channel name.

CHLAUTH(*) on z/OS systems

z/OS On z/OS, a channel name including the asterisk (*) must be enclosed in quotation marks. This rule also applies to the use of a single asterisk to match all channel names. Thus, where you would specify CHLAUTH(*) on other platforms, on z/OS you must specify CHLAUTH('*').

Behavior of SET CHLAUTH command over queue manager restart

If the SYSTEM.CHLAUTH.DATA.QUEUE, has been deleted or altered in a way that it is no longer accessible i.e. PUT(DISABLED), the **`SET CHLAUTH`** command will only be partially successful. In this instance, **`SET CHLAUTH`** will update the in-memory cache, but will fail when hardening.

This means that although the rule put in place by the **`SET CHLAUTH`** command may be operable initially, the effect of the command will not persist over a queue manager restart. The user should investigate, ensuring the queue is accessible and then reissue the command (using **`ACTION(REPLACE)`**) before cycling the queue manager.

If the SYSTEM.CHLAUTH.DATA.QUEUE remains inaccessible at queue manager startup, the cache of saved rules cannot be loaded and all channels will be blocked until the queue and rules become accessible.

Maximum size of ADDRESS and ADDRLIST on z/OS systems

z/OS

On z/OS, the maximum size for the ADDRESS and ADDRLIST fields are 48 characters. Some IPv6 address patterns could be longer than this limit, for example '0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff'. In this case, you could use '*' instead.

If you want to use a pattern more than 48 characters long, try to express the requirement in a different way. For example, instead of specifying

'0001-fffe:0001-fffe:0001-fffe:0001-fffe:0001-fffe:0001-fffe:0001-fffe:0001-fffe' as the address pattern for a USERSRC(MAP), you could specify three rules:

- USERSRC(MAP) for all addresses (*)
- USERSRC(NOACCESS) for address '0000:0000:0000:0000:0000:0000:0000:0000'
- USERSRC(NOACCESS) for address 'ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff'

Commands troubleshooting

- **Scenario:** You receive errors when you use special characters in descriptive text for some commands.
- **Explanation:** Some characters, for example, back slash (\) and double quote (") characters have special meanings when used with commands.
- **Solution:** Precede special characters with a \, that is, enter \\ or \" if you want \ or " in your text. Not all characters are allowed to be used with commands. For more information about characters with special meanings and how to use them, see Characters with special meanings.

Distributed publish/subscribe troubleshooting

Use the advice given in the subtopics to help you to detect and deal with problems when you use publish/subscribe clusters or hierarchies.

Before you begin

If your problems relate to clustering in general, rather than to publish/subscribe messaging using clusters, see“Queue manager clusters troubleshooting” on page 473.

There are also some helpful troubleshooting tips in Design considerations for retained publications in publish/subscribe clusters.

Related information

Configuring a publish/subscribe cluster
Designing publish/subscribe clusters
Distributed publish/subscribe system queue errors

Routing for publish/subscribe clusters: Notes on behavior

Use the advice given here to help you to detect and deal with routing problems when you are using clustered publish/subscribe messaging.

For information about status checking and troubleshooting for any queue manager cluster, see “Queue manager clusters troubleshooting” on page 473.

- All clustered definitions of the same named topic object in a cluster must have the same **CLROUTE** setting. You can check the **CLROUTE** setting for all topics on all hosts in the cluster using the following MQSC command:

```
display tcluster(*) clroute
```

- The **CLROUTE** property has no effect unless the topic object specifies a value for the **CLUSTER** property.
- Check that you have spelled the cluster name correctly on your topic. You can define a cluster object such as a topic before defining the cluster. Therefore, when you define a cluster topic, no validation is done on the cluster name because it might not yet exist. Consequently, the product does not alert you to misspelt cluster names.
- When you set the **CLROUTE** property, if the queue manager knows of a clustered definition of the same object from another queue manager that has a different **CLROUTE** setting, the system generates an

MQRCCF_CLUSTER_TOPIC_CONFLICT exception. However, through near simultaneous object definition on different queue managers, or erratic connectivity with full repositories, differing definitions might be created. In this situation the full repository queue managers arbitrate, accepting one definition and reporting an error for the other one. To get more information about the conflict, use the following MQSC command to check the cluster state of all topics on all queue managers in the cluster:

```
display tcluster(*) clstate
```

A state of invalid, or pending (if this does not soon turn to active), indicates a problem. If an invalid topic definition is detected, identify the incorrect topic definition and remove it from the cluster. The full repositories have information about which definition was accepted and which was rejected, and the queue managers that created the conflict have some indication of the nature of the problem. See also CLSTATE in DISPLAY TOPIC.

- Setting the **CLROUTE** parameter at a point in the topic tree causes the entire branch beneath it to route topics in that way. You cannot change the routing behavior of a sub-branch of this branch. For this reason, defining a topic object for a lower or higher node in the topic tree with a different **CLROUTE** setting is rejected with an MQRCCF_CLUSTER_TOPIC_CONFLICT exception.
- You can use the following MQSC command to check the topic status of all the topics in the topic tree:

```
display tpstatus('#')
```

  If you have a large number of branches in the topic tree, the previous command might display status for an inconveniently large number of topics. If that is the case, you can instead display a manageably small branch of the tree, or an individual topic in the tree. The information displayed includes the topic string, cluster name and cluster route setting. It also includes the publisher count and subscription count (number of publishers and subscribers), to help you judge whether the number of users of this topic is as you expect.
- Changing the cluster routing of a topic in a cluster is a significant change to the publish/subscribe topology. After a topic object has been clustered (through setting the **CLUSTER** property) you cannot change the value of the **CLROUTE** property. The object must be un-clustered (**CLUSTER** set to ' ') before you can change the value. Un-clustering a topic converts the topic definition to a local topic, which results in a period during which publications are not delivered to subscriptions on remote queue managers; this should be considered when performing this change. See The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager. If you try to change the value of the **CLROUTE** property while it is clustered, the system generates an MQRCCF_CLROUTE_NOT_ALTERABLE exception.
- For topic host routing, you can explore alternative routes through the cluster by adding and removing the same cluster topic definition on a range of cluster queue managers. To stop a given queue manager from acting as a topic host for your cluster topic, either delete the topic object, or use the PUB(DISABLED) setting to quiesce message traffic for this topic, as discussed in Special handling for the PUB parameter. Do not un-cluster the topic by setting the **CLUSTER** property to ' ', because removing the cluster name converts the topic definition to a local topic, and prevents the clustering behavior of the topic when used from this queue manager. See The effect of defining a non-cluster topic with the same name as a cluster topic from another queue manager.
- You cannot change the cluster of a sub-branch of the topic tree when the branch has already been clustered to a different cluster and **CLROUTE** is set to TOPICHOST. If such a definition is detected at define time, the system generates an MQRCCF_CLUSTER_TOPIC_CONFLICT exception. Similarly, inserting a newly clustered topic definition at a higher node for a different cluster generates an exception. Because of the clustering timing issues previously described, if such an inconsistency is later detected, the queue manager issues errors to the queue manager log.

Related information

Configuring a publish/subscribe cluster

Designing publish/subscribe clusters

Checking proxy subscription locations

A proxy subscription enables a publication to flow to a subscriber on a remote queue manager. If your subscribers are not getting messages that are published elsewhere in the queue manager network, check that your proxy subscriptions are where you expect them to be.

Missing proxy subscriptions can show that your application is not subscribing on the correct topic object or topic string, or that there is a problem with the topic definition, or that a channel is not running or is not configured correctly.

To show proxy subscriptions, use the following MQSC command:

```
display sub(*) subtype(proxy)
```

Proxy subscriptions are used in all distributed publish/subscribe topologies (hierarchies and clusters). For a topic host routed cluster topic, a proxy subscription exists on each topic hosts for that topic. For a direct routed cluster topic, the proxy subscription exists on every queue manager in the cluster. Proxy subscriptions can also be made to exist on every queue manager in the network by setting the `proxysub(force)` attribute on a topic.

See also Subscription performance in publish/subscribe networks.

Resynchronization of proxy subscriptions

Under normal circumstances, queue managers automatically ensure that the proxy subscriptions in the system correctly reflect the subscriptions on each queue manager in the network. Should the need arise, you can manually resynchronize a queue manager's local subscriptions with the proxy subscriptions that it propagated across the network using the **REFRESH QMGR TYPE(PROXYSUB)** command. However, you should do so only in exceptional circumstances.

When to manually resynchronize proxy subscriptions

When a queue manager is receiving subscriptions that it should not be sent, or not receiving subscriptions that it should receive, you should consider manually resynchronizing the proxy subscriptions. However, resynchronization temporarily creates a sudden additional proxy subscription load on the network, originating from the queue manager where the command is issued. For this reason, do not manually resynchronize unless IBM MQ service, IBM MQ documentation, or error logging instructs you to do so.

You do not need to manually resynchronize proxy subscriptions if automatic revalidation by the queue manager is about to occur. Typically, a queue manager revalidates proxy subscriptions with affected directly-connected queue managers at the following times:

- When forming a hierarchical connection
- When modifying the **PUBSCOPE** or **SUBSCOPE** or **CLUSTER** attributes on a topic object
- When restarting the queue manager

Sometimes a configuration error results in missing or extraneous proxy subscriptions:

- Missing proxy subscriptions can be caused if the closest matching topic definition is specified with **Subscription scope** set to `Queue Manager` or with an empty or incorrect cluster name. Note that **Publication scope** does not prevent the sending of proxy subscriptions, but does prevent publications from being delivered to them.
- Extraneous proxy subscriptions can be caused if the closest matching topic definition is specified with **Proxy subscription behavior** set to `Force`.

When configuration errors cause these problems, manual resynchronization does not resolve them. In these cases, amend the configuration.

The following list describes the exceptional situations in which you should manually resynchronize proxy subscriptions:

- After issuing a **REFRESH CLUSTER** command on a queue manager in a publish/subscribe cluster.

- When messages in the queue manager error log tell you to run the **REFRESH QMGR TYPE(REPOS)** command.
- When a queue manager cannot correctly propagate its proxy subscriptions, perhaps because a channel has stopped and all messages cannot be queued for transmission, or because operator error has caused messages to be incorrectly deleted from the SYSTEM.CLUSTER.TRANSMIT.QUEUE queue.
- When messages are incorrectly deleted from other system queues.
- When a **DELETE SUB** command is issued in error on a proxy subscription.
- As part of disaster recovery.

How to manually resynchronize proxy subscriptions

First rectify the original problem (for example by restarting the channel), then issue the following command on the queue manager:

```
REFRESH QMGR TYPE(PROXYSUB)
```

When you issue this command, the queue manager sends, to each of its directly-connected queue managers, a list of its own topic strings for which proxy subscriptions should exist. The directly-connected queue managers then update their held proxy subscriptions to match the list. Next, the directly-connected queue managers send back to the originating queue manager a list of their own topic strings for which proxy subscriptions should exist, and the originating queue manager updates its held proxy subscriptions accordingly.

Important usage notes:

- Publications missed due to proxy subscriptions not being in place are not recovered for the affected subscriptions.
- Resynchronization requires the queue manager to start channels to other queue managers. If you are using direct routing in a cluster, or you are using topic host routing and this command is issued on a topic host queue manager, the queue manager will start channels to all other queue managers in the cluster, even those that have not performed publish/subscribe work. Therefore the queue manager that you are refreshing must have enough capability to cope with communicating with every other queue manager in the cluster.
- z/OS If this command is issued on z/OS when the CHINIT is not running, the command is queued up and processed when the CHINIT starts.

Related information

Checking that async commands for distributed networks have finished
REFRESH CLUSTER considerations for publish/subscribe clusters

Loop detection in a distributed publish/subscribe network

In a distributed publish/subscribe network, it is important that publications and proxy subscriptions cannot loop, because this would result in a flooded network with connected subscribers receiving multiple copies of the same original publication.

The proxy subscription aggregation system described in Proxy subscriptions in a publish/subscribe network does not prevent the formation of a loop, although it will prevent the perpetual looping of proxy subscriptions. Because the propagation of publications is determined by the existence of proxy subscriptions, they can enter a perpetual loop. Websphere MQ V7.0 uses the following technique to prevent publications from perpetually looping:

As publications move around a publish/subscribe topology each queue manager adds a unique fingerprint to the message header. Whenever a publish/subscribe queue manager receives a publication from another publish/subscribe queue manager, the fingerprints held in the message header are checked. If its own fingerprint is already present, the publication has fully circulated around a loop, so the queue manager discards the message, and adds an entry to the error log.

Note: Within a loop, publications are propagated in both directions around the loop, and each queue manager within the loop receives both publications before the originating queue manager discards the looped publications. This results in subscribing applications receiving duplicate copies of publications until the loop is broken.

Loop detection fingerprint format

The loop detection fingerprints are inserted into an RFH2 header or flow as part of the V8.0 protocol. An RFH2 programmer needs to understand the header and pass on the fingerprint information intact. earlier versions of IBM Integration Bus use RFH1 headers which do not contain the fingerprint information.

```
<ibm>
  <Rfp>uuid1</Rfp>
  <Rfp>uuid2</Rfp>
  <Rfp>uuid3</Rfp>
  . . .
</ibm>
```

<ibm> is the name of the folder that holds the list of routing fingerprints containing the unique user identifier (uuid) of each queue manager that has been visited.

Every time that a message is published by a queue manager, it adds its uuid into the <ibm> folder using the <Rfp> (routing fingerprint) tag. Whenever a publication is received, IBM MQ uses the message properties API to iterate through the <Rfp> tags to see if that particular uuid value is present. Because of the way that the WebSphere Platform Messaging component of IBM MQ attaches to IBM Integration Bus through a channel and RFH2 subscription when using the queued publish/subscribe interface, IBM MQ also creates a fingerprint when it receives a publication by that route.

The goal is to not deliver any RFH2 to an application if it is not expecting any, simply because we have added in our fingerprint information.

Whenever an RFH2 is converted into message properties, it will also be necessary to convert the <ibm> folder; this removes the fingerprint information from the RFH2 that is passed on or delivered to applications that have used the IBM MQ V7.0, or later, API.

JMS applications do not see the fingerprint information, because the JMS interface does not extract that information from the RFH2, and therefore does not hand it on to its applications.

The Rfp message properties are created with `propDesc.CopyOptions = MQCOPY_FORWARD and MQCOPY_PUBLISH`. This has implications for applications receiving and then republishing the same message. It means that such an application can continue the chain of routing fingerprints by using `PutMsgOpts.Action = MQACTP_FORWARD`, but must be coded appropriately to remove its own fingerprint from the chain. By default the application uses `PutMsgOpts.Action = MQACTP_NEW` and starts a new chain.

IBM MQ client for HP Integrity NonStop Server troubleshooting

Provides information to help you to detect and deal with problems when you are using the IBM MQ client for HP Integrity NonStop Server.

Toggling between the use of IBM MQ and TMF transactions on a single connection

If an IBM MQ client for HP Integrity NonStop Server application toggles between the use of IBM MQ and TMF transactions on a single connection, then IBM MQ operations such as MQPUT and MQGET might fail with a return code of "2072 (0818) (RC2072): MQRC_SYNCPOINT_NOT_AVAILABLE" on page 76. Errors and a first failure symptom report for the client application are generated in the IBM MQ client for HP Integrity NonStop Server errors directory.

This error occurs because mixed TMF and IBM MQ transactions on a single connection are not supported.

Use the standard facilities that are supplied with your system to record the problem identifier and to save any generated output files. Use either the IBM MQ Support site: https://www.ibm.com/support/home/, or the IBM Support Assistant (ISA): https://www.ibm.com/support/home/product/C100515X13178X21/other_software/ibm_support_assistant to check whether a solution is already available. If you are unable to find a solution, contact your IBM support center. Do not discard these files until the problem is resolved.

Java and JMS troubleshooting

Use the advice that is given here to help you to resolve common problems that can arise when you are using Java or JMS applications.

About this task

The subtopics in this section provide advice to help you to detect and deal with problems that you might encounter under the following circumstances:

- When using the IBM MQ resource adapter
- When connecting to a queue manager with a specified provider version

Related concepts

"Tracing IBM MQ classes for JMS applications" on page 373
The trace facility in IBM MQ classes for JMS is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

"Tracing the IBM MQ Resource Adapter" on page 381
The ResourceAdapter object encapsulates the global properties of the IBM MQ resource adapter. To enable trace of the IBM MQ resource adapter, properties need to be defined in the ResourceAdapter object.

"Tracing additional IBM MQ Java components" on page 383
For Java components of IBM MQ, for example the IBM MQ Explorer and the Java implementation of IBM MQ Transport for SOAP, diagnostic information is output using the standard IBM MQ diagnostic facilities or by Java diagnostic classes.

Related tasks

"Tracing IBM MQ classes for Java applications" on page 377
The trace facility in IBM MQ classes for Java is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

Related information

Using IBM MQ classes for JMS
Using the IBM MQ resource adapter
Using IBM MQ classes for Java

JMS provider version troubleshooting

Use the advice that is given here to help you to resolve common problems that can arise when you are connecting to a queue manager with a specified provider version.

JMS 2.0 function is not supported with this connection error

- **Error code:** JMSCC5008
- **Scenario:** You receive a `JMS 2.0 function is not supported with this connection` error.
- **Explanation:** The use of the JMS 2.0 functionality is only supported when connecting to an IBM MQ 8.0 queue manager that is using IBM MQ messaging provider Version 8 mode.
- **Solution:** Change the application to not use the JMS 2.0 function, or ensure that the application connects to an IBM MQ 8.0 queue manager that is using IBM MQ messaging provider Version 8 mode.

JMS 2.0 API is not supported with this connection error

- **Error code:** JMSCC5007
- **Scenario:** You receive a `JMS 2.0 API is not supported with this connection` error.
- **Explanation:** The use of the JMS 2.0 API is only supported when you are connecting to an IBM MQ Version 7 or Version 8 queue manager that is using IBM MQ messaging provider Normal or Version 8 mode. You might, for example, receive this error if you are attempting to connect to a Version 6 queue manager or if you are connecting by using migration mode. This typically happens if SHARECNV(0) or PROVIDER_VERSION=6 is specified.
- **Solution:** Change the application to not use the JMS 2.0 API, or ensure that the application connects to an IBM MQ Version 7 or Version 8 queue manager by using IBM MQ messaging provider Normal or Version 8 mode.

Queue manager command level did not match the requested provider version error

- **Error code:** JMSFMQ0003
- **Scenario:** You receive a `queue manager command level did not match the requested provider version`error.
- **Explanation:** The queue manager version that is specified in the provider version property on the connection factory is not compatible with the requested queue manager. For example, you might have specified PROVIDER_VERSION=8, and attempt to connect to a queue manager with a command level less than 800, such as 750.
- **Solution:** Modify the connection factory to connect to a queue manager that can support the provider version required.

For more information about provider version, see Rules for selecting the IBM MQ messaging provider mode.

Related information

Configuring the JMS **`PROVIDERVERSION`** property

PCF processing in JMS

IBM MQ Programmable Change Format (PCF) messages are a flexible, powerful way in which to query and modify attributes of a queue manager, and the PCF classes provided in the IBM MQ classes for Java provide a convenient way of accessing their functionality in a Java application. The functionality can also be accessed from IBM MQ classes for JMS, however there is a potential problem.

The common model for processing PCF responses in JMS

A common approach to processing PCF responses in JMS is to extract the bytes payload of the message, wrap it in a `DataInputStream` and pass it to the `com.ibm.mq.headers.pcf.PCFMessage` constructor.

```
Message m = consumer.receive(10000);
//Reconstitute the PCF response.
ByteArrayInputStream bais =
    new ByteArrayInputStream(((BytesMessage)m).getBody(byte[].class));
DataInput di = new DataInputStream(bais);
 PCFMessage pcfResponseMessage = new PCFMessage(di);
```

See Using the IBM MQ Headers package for some examples.

Unfortunately this is not a totally reliable approach for all platforms - in general the approach works for big-endian platforms, but not for little-endian platforms.

What is the problem?

The problem is that in parsing the message headers, the `PCFMessage` class has to deal with issues of numeric encoding - the headers contain length fields which are in some encoding, that is big-endian or little-endian.

If you pass a "pure" `DataInputStream` to the constructor, the `PCFMessage` class has no good indication of the encoding, and has to assume a default - quite possibly incorrectly.

If this situation arises, you will probably see a "MQRCCF_STRUCTURE_TYPE_ERROR" (reason code 3013) in the constructor:

```
com.ibm.mq.headers.MQDataException: MQJE001: Completion Code '2', Reason '3013'.
        at com.ibm.mq.headers.pcf.PCFParameter.nextParameter(PCFParameter.java:167)
        at com.ibm.mq.headers.pcf.PCFMessage.initialize(PCFMessage.java:854)
        at com.ibm.mq.headers.pcf.PCFMessage.<init>(PCFMessage.java:156)
```

This message almost invariably means that the encoding has been misinterpreted. The probable reason for this is that the data that has been read is little-endian data which has been interpreted as big-endian.

The solution

The way to avoid this problem is to pass the `PCFMessage` constructor something which will tell the constructor the numeric encoding of the data it is working with.

To do this, make an `MQMessage` from the data received.

The following code is an outline example of the code you might use.

Attention: The code is an outline example only and does not contain any error handling information.

```
      // get a response into a JMS Message
      Message receivedMessage = consumer.receive(10000);
      BytesMessage bytesMessage = (BytesMessage) receivedMessage;
      byte[] bytesreceived = new byte[(int) bytesMessage.getBodyLength()];
      bytesMessage.readBytes(bytesreceived);

      // convert to MQMessage then to PCFMessage
      MQMessage mqMsg = new MQMessage();
      mqMsg.write(bytesreceived);
      mqMsg.encoding = receivedMessage.getIntProperty("JMS_IBM_Encoding");
      mqMsg.format = receivedMessage.getStringProperty("JMS_IBM_Format");
      mqMsg.seek(0);

      PCFMessage pcfMsg = new PCFMessage(mqMsg);
```

JMS connection pool error handling

Connection pool error handling is carried out by various methods of a purge policy.

The connection pool purge policy comes into operation if an error is detected when an application is using a JMS connection to a JMS provider. The connection manager can either:

- Close only the connection that encountered the problem. This is known as the `FailingConnectionOnly` purge policy and is the default behavior.

  Any other connections created from the factory, that is, those in use by other applications, and those that are in the free pool of the factory, are left alone.
- Close the connection that encountered the problem, throw away any connections in the free pool of the factory, and mark any in-use connections as stale.

  The next time the application that is using the connection tries to perform a connection-based operation, the application receives a `StaleConnectionException`. For this behavior, set the purge policy to `Entire Pool`.

Purge policy - failing connection only

Use the example described in Examples of using the connection pool. Two MDBs are deployed into the application server, each one using a different listener port. The listener ports both use the `jms/CF1` connection factory.

After 600 seconds, you stop the first listener, and the connection that this listener port was using is returned to the connection pool.

If the second listener encounters a network error while polling the JMS destination, the listener port shuts down. Because the purge policy for the `jms/CF1` connection factory is set to `FailingConnectionOnly`, the connection manager throws away only the connection that was used by the second listener. The connection in the free pool remains where it is.

If you now restart the second listener, the connection manager passes the connection from the free pool to the listener.

Purge policy - entire pool

For this situation, assume that you have three MDBs installed into your application server, each one using its own listener port. The listener ports have created connections from the `jms/CF1` factory. After a period of time you stop the first listener, and its connection, `c1`, is put into the `jms/CF1` free pool.

When the second listener detects a network error, it shuts itself down and closes `c2`. The connection manager now closes the connection in the free pool. However, the connection being used by third listener remains.

What should you set the purge policy to?

As previously stated, the default value of the purge policy for JMS connection pools is `FailingConnectionOnly`.

However, setting the purge policy to `EntirePool` is a better option. In most cases, if an application detects a network error on its connection to the JMS provider, it is likely that all open connections created from the same connection factory have the same problem.

If the purge policy is set to `FailingConnectionOnly`, the connection manager leaves all of the connections in the free pool. The next time an application tries to create a connection to the JMS provider, the connection manager returns one from the free pool if there is one available. However, when the application tries to use the connection, it encounters the same network problem as the first application.

Now, consider the same situation with the purge policy set to `EntirePool`. As soon as the first application encounters the network problem, the connection manager discards the failing connection and closes all connections in the free pool for that factory.

When a new application starts up and tries to create a connection from the factory, the connection manager tries to create a new one, as the free pool is empty. Assuming that the network problem has been resolved, the connection returned to the application is valid.

Connection pool errors while trying to create a JMS Context

If an error occurs while you are trying to create a JMS Context, it is possible to determine from the error message if the top-level pool or lower-level pool had the issue.

How pools are used for Contexts

When using Connection and Sessions, there are pools for each type of object; a similar model is followed for Contexts.

A typical application that uses distributed transactions involves both messaging and non-messaging workloads in the same transaction.

Assuming that no work is currently working, and the application makes its first createConnection method call, a context facade or proxy is created in the equivalent of the connection pool (the top-level pool). Another object is created in the equivalent of the session pool. This second object encapsulates the underlying JMS Context (lower-level pool).

Pooling, as a concept, is used to permit an application to scale. Many threads are able to access a constrained set of resources. In this example, another thread will execute the createContext method call to get a context from the pool. Should other threads still be doing messaging work, then the top-level pool is expanded to provide an additional context for the requesting thread.

In the case where a thread requests a context and the messaging work has completed but the non-messaging work has not, so the transaction is not complete, the lower-level pool is expanded. The top-level context proxy remains assigned to the transaction until that transaction is resolved, so cannot be assigned to another transaction.

In the case of the lower pool becoming full, this means that the non-messaging work is taking potentially a long time.

In the case of the top-level pool becoming full, this means that the overall messaging work is taking a while and the pool should be expanded.

Identifying which pool an error originated from

You can determine the pool in which an error originated from the error message text:

- For the top-level pool, the message text is `Failed to create context`. This message means that the top-level pool is full of Context-proxy objects, all of which have currently running transactions that are performing messaging.
- For the lower-level pool, the message text is `Failed to set up new JMSContext`. This message means that although a connect-proxy is available, it is still necessary to wait for non-messaging work to complete.

Top-level pool example

```
************************[8/19/16 10:10:48:643 UTC] 000000a2
    LocalExceptio E CNTR0020E: EJB threw an unexpected (non-declared)  exception during
    invocation of method "onMessage" on bean
    "BeanId(SibSVTLiteMDB#SibSVTLiteMDBXA_RecoveryEJB_undeployed.jar#QueueReceiver,  null)".
    Exception data: javax.jms.JMSRuntimeException: Failed to create context
      at com.ibm.ejs.jms.JMSCMUtils.mapToJMSRuntimeException(JMSCMUtils.java:522)
      at
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContextInternal(JMSConnectionFactoryHandle.java:4
49)
      at
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContext(JMSConnectionFactoryHandle.java:335)
      at sib.test.svt.lite.mdb.xa.SVTMDBBase.sendReplyMessage(SVTMDBBase.java:554)
      at sib.test.svt.lite.mdb.xa.QueueReceiverBean.onMessage(QueueReceiverBean.java:128)
      at
sib.test.svt.lite.mdb.xa.MDBProxyQueueReceiver_37ea5ce9.onMessage(MDBProxyQueueReceiver_37ea5ce9.j
ava)
      at
com.ibm.mq.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:151)
      at com.ibm.mq.jms.MQSession$FacadeMessageListener.onMessage(MQSession.java:129)
      at com.ibm.msg.client.jms.internal.JmsSessionImpl.run(JmsSessionImpl.java:3236)
      at com.ibm.mq.jms.MQSession.run(MQSession.java:937)
      at com.ibm.mq.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
      at com.ibm.mq.connector.inbound.AbstractWorkImpl.run(AbstractWorkImpl.java:233)
      at com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
      at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
    Caused by: com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException:
CWTE_NORMAL_J2CA1009
      at com.ibm.ejs.j2c.FreePool.createOrWaitForConnection(FreePool.java:1783)
      at com.ibm.ejs.j2c.PoolManager.reserve(PoolManager.java:3896)
      at com.ibm.ejs.j2c.PoolManager.reserve(PoolManager.java:3116)
      at com.ibm.ejs.j2c.ConnectionManager.allocateMCWrapper(ConnectionManager.java:1548)
      at com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1031)
      at
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContextInternal(JMSConnectionFactoryHandle.java:4
43)
          ... 12 more
```

Lower-level pool example

```
***********************
[8/19/16 9:44:44:754 UTC] 000000ac SibMessage W   [:] CWSJY0003W: MQJCA4004: Message delivery to
an MDB
    'sib.test.svt.lite.mdb.xa.MDBProxyQueueReceiver_37ea5ce9@505d4b68
(BeanId(SibSVTLiteMDB#SibSVTLiteMDBXA_RecoveryEJB_undeployed.jar#QueueReceiver, null))' failed
with exception:
'nested exception is: javax.jms.JMSRuntimeException: Failed to set up new JMSContext'.
^C[root@username-instance-2 server1]# vi SystemOut.log
                  :com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
                  : com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
    Caused by [1] --> Message : javax.jms.JMSRuntimeException: Failed to set up new
JMSContext
               Class : class javax.jms.JMSRuntimeException
               Stack :
com.ibm.ejs.jms.JMSCMUtils.mapToJMSRuntimeException(JMSCMUtils.java:522)
                  :
com.ibm.ejs.jms.JMSContextHandle.setupInternalContext(JMSContextHandle.java:241)
                  :
com.ibm.ejs.jms.JMSManagedConnection.getConnection(JMSManagedConnection.java:783)
                  :
com.ibm.ejs.j2c.MCWrapper.getConnection(MCWrapper.java:2336)
                  :
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1064)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContextInternal(JMSConnectionFactoryHandle.java:4
43)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContext(JMSConnectionFactoryHandle.java:335)
                  :
sib.test.svt.lite.mdb.xa.SVTMDBBase.sendReplyMessage(SVTMDBBase.java:554)
                  :
sib.test.svt.lite.mdb.xa.QueueReceiverBean.onMessage(QueueReceiverBean.java:128)
                  :
sib.test.svt.lite.mdb.xa.MDBProxyQueueReceiver_37ea5ce9.onMessage(MDBProxyQueueReceiver_37ea5ce9.j
ava:-1)
                  :
com.ibm.mq.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:151)
                  :
com.ibm.mq.jms.MQSession$FacadeMessageListener.onMessage(MQSession.java:129)
                  :
com.ibm.msg.client.jms.internal.JmsSessionImpl.run(JmsSessionImpl.java:3236)
                  : com.ibm.mq.jms.MQSession.run(MQSession.java:937)
                  :
com.ibm.mq.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
                  :
com.ibm.mq.connector.inbound.AbstractWorkImpl.run(AbstractWorkImpl.java:233)
                  : com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
                  : com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
    Caused by [2] --> Message : com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException:
CWTE_NORMAL_J2CA1009
               Class : class
com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException
               Stack : com.ibm.ejs.j2c.FreePool.createOrWaitForConnection(FreePool.java:1783)
                  :
com.ibm.ejs.j2c.PoolManager.reserve(PoolManager.java:3840)
                  : com.ibm.ejs.j2c.PoolManager.reserve(PoolManager.java:3116)
                  :
com.ibm.ejs.j2c.ConnectionManager.allocateMCWrapper(ConnectionManager.java:1548)
                  :
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1031)
                  :
com.ibm.ejs.jms.JMSContextHandle.setupInternalContext(JMSContextHandle.java:222)
                  :
com.ibm.ejs.jms.JMSManagedConnection.getConnection(JMSManagedConnection.java:783)
                  :
com.ibm.ejs.j2c.MCWrapper.getConnection(MCWrapper.java:2336)
                  :
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1064)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContextInternal(JMSConnectionFactoryHandle.java:4
43)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContext(JMSConnectionFactoryHandle.java:335)
                  :
sib.test.svt.lite.mdb.xa.SVTMDBBase.sendReplyMessage(SVTMDBBase.java:554)
                  :
sib.test.svt.lite.mdb.xa.QueueReceiverBean.onMessage(QueueReceiverBean.java:128)
                  :
sib.test.svt.lite.mdb.xa.MDBProxyQueueReceiver_37ea5ce9.onMessage(MDBProxyQueueReceiver_37ea5ce9.j
ava:-1)
                  :
com.ibm.mq.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:151)
                  :
com.ibm.mq.jms.MQSession$FacadeMessageListener.onMessage(MQSession.java:129)
                  :
com.ibm.msg.client.jms.internal.JmsSessionImpl.run(JmsSessionImpl.java:3236)
                  : com.ibm.mq.jms.MQSession.run(MQSession.java:937)
                  :
com.ibm.mq.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
                  :
com.ibm.mq.connector.inbound.AbstractWorkImpl.run(AbstractWorkImpl.java:233)
                  : com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
                  : com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
```

Troubleshooting JMSCC0108 messages

There are a number of steps that you can take to prevent a `JMSCC0108` message from occurring when you are using activation specifications and WebSphere Application Server listener ports that are running in Application Server Facilities (ASF) mode.

When you are using activation specifications and WebSphere Application Server listener ports that are running in ASF mode, which is the default mode of operation, it is possible that the following message might appear in the application server log file:

```
JMSCC0108: The IBM MQ classes for JMS had detected a message, ready for asynchronous delivery to
an application.
When delivery was attempted, the message was no longer available.
```

Use the information in this topic to understand why this message appears, and the possible steps that you can take to prevent it from occurring.

How activation specifications and listener ports detect and process messages

An activation specification or WebSphere Application Server listener port performs the following steps when it starts up:

1. Create a connection to the queue manager that they have been set to use.
2. Open the JMS destination on that queue manager that they have been configured to monitor.
3. Browse that destination for messages.

When a message is detected, the activation specification or listener port performs the following steps:

1. Constructs an internal message reference that represents the message.
2. Gets a server session from its internal server session pool.
3. Loads the server session up with the message reference.
4. Schedules a piece of work with the application server Work Manager to run the server session and process the message.

The activation specification or listener port then goes back to monitoring the destination again, looking for another message to process.

The application server Work Manager runs the piece of work that the activation specification or listener port submitted on a new server session thread. When started, the thread completes the following actions:

- Starts either a local or global (XA) transaction, depending on whether the message-driven bean requires XA transactions or not, as specified in the message-driven bean's deployment descriptor.
- Gets the message from the destination by issuing a destructive MQGET API call.
- Runs the message-driven bean's onMessage() method.
- Completes the local or global transaction, once the onMessage() method has finished.
- Return the server session back to the server session pool.

Why the JMSCC0108 message occurs, and how to prevent it

The main activation specification or listener port thread browses messages on a destination. It then asks the Work Manager to start a new thread to destructively get the message and process it. This means that it is possible for a message to be found on a destination by the main activation specification or listener port thread, and no longer be available by the time the server session thread attempts to get it. If this happens, then the server session thread writes the following message to the application server's log file:

```
JMSCC0108: The IBM MQ classes for JMS had detected a message, ready for asynchronous delivery to
an application.
When delivery was attempted, the message was no longer available.
```

There are two reasons why the message is no longer on the destination when the server session thread tries to get it:

- Reason 1: The message has been consumed by another application
- Reason 2: The message has expired

Reason 1: The message has been consumed by another application

If two or more activation specifications and/or listener ports are monitoring the same destination, then it is possible that they could detect the same message and try to process it. When this happens:

- A server session thread started by one activation specification or listener port gets the message and delivers it to a message-driven bean for processing.
- The sever session thread started by the other activation specification or listener port tries to get the message, and finds that it is no longer on the destination.

If an activation specification or listener port is connecting to a queue manager in any of the following ways, the messages that the main activation specification or listener port thread detects are marked:

- A queue manager on any platform, using IBM MQ messaging provider normal mode.
- A queue manager on any platform, using IBM MQ messaging provider normal mode with restrictions
- A queue manager running on z/OS, using IBM MQ messaging provider migration mode.

Marking a message prevents any other activation specification or listener port from seeing that message, and trying to process it.

By default, messages are marked for five seconds. After the message has been detected and marked, the five second timer starts. During these five seconds, the following steps must be carried out:

- The activation specification or listener port must get a server session from the server session pool.
- The server session must be loaded with details of the message to process.
- The work must be scheduled.
- The Work Manager must process the work request and start the server session thread.
- The server session thread needs to start either a local or global transaction.
- The server session thread needs to destructively get the message.

On a busy system, it might take longer than five seconds for these steps to be carried out. If this happens, then the mark on the message is released. This means that other activation specifications or listener ports can now see the message, and can potentially try to process it, which can result in the `JMSCC0108` message being written to the application server's log file.

In this situation, you should consider the following options:

- Increase the value of the queue manager property Message mark browse interval (MARKINT), to give the activation specification or listener port that originally detected the message more time to get it. Ideally, the property should be set to a value greater than the time taken for your message-driven beans to process messages. This means that, if the main activation specification or listener port thread blocks waiting for a server session because all of the server sessions are busy processing messages, then the message should still be marked when a server session becomes available. Note that the MARKINT property is set on a queue manager, and so is applicable to all applications that browse messages on that queue manager.

- Increase the size of the server session pool used by the activation specification or listener port. This would mean that there are more server sessions available to process messages, which should ensure that messages can be processed within the specified mark interval. One thing to note with this approach is that the activation specification or listener port will now be able to process more messages concurrently, which could impact the overall performance of the application server.

If an activation specification or listener port is connecting to a queue manager running on a platform other than z/OS, using IBM MQ messaging provider migration mode, the marking functionality is not available. This means that it is not possible to prevent two or more activation specifications and/or listener ports from detecting the same message and trying to process it. In this situation, the `JMSCC0108` message is expected.

Reason 2: The message has expired

The other reason that a `JMSCC0108` message is generated is if the message has expired in between being detected by the activation specification or listener port and being consumed by the server session. If this happens, when the server session thread tries to get the message, it finds that it is no longer there and so reports the `JMSCC0108` message.

Increasing the size of the server session pool used by the activation specification or listener port can help here. Increasing the server session pool size means that there are more server sessions available to process messages, which can potentially mean that the message is processed before it expires. It is important to note that the activation specification or listener port is now able to process more messages concurrently, which could impact the overall performance of the application server.

CWSJY0003W warning messages in WebSphere Application Server SystemOut.log file

A CWSJY0003W warning message is logged in the WebSphere Application Server Version 7.0 SystemOut.log file when an MDB processes JMS messages from IBM WebSphere MQ.

Symptom

CWSJY0003W: IBM WebSphere MQ classes for JMS attempted to get a message for delivery to a message listener, that had previously been marked using browse-with-mark, however, the message is not available.

Cause

Activation specifications, and listener ports running in Application Server Facilities (ASF) mode, are used to monitor queues or topics hosted on IBM WebSphere MQ queue managers. Initially messages are browsed on either the queue or topic. When a message is found, a new thread is started which destructively gets the message and passes the message to an instance of a message-driven bean application for processing.

When the message is browsed, the queue manager marks the message for a period of time, and effectively hides the message from other application server instances. The time period that the message is marked for is determined by the queue manager attribute **`MARKINT`**, which by default is set to 5000 milliseconds (5 seconds). This means that, after an activation specification or listener port has browsed a message, the queue manager will wait for 5 seconds for the destructive get of the message to occur before allowing another application server instance to see that message and process it.

The following situation can occur:

- An activation specification running on Application Server 1 browses message A on a queue.
- The activation specification starts a new thread to process message A.
- An event occurs on Application Server 1, which means that message A is still on the queue after 5 seconds.

- An activation specification running on Application Server 2 now browses message A and starts a new thread to process message A.
- The new thread running on Application Server 2 destructively gets message A, and passes it to a message-driven bean instance.
- The thread running on Application Server 1 attempts to get message A, only to find that message A is no longer on the queue.
- At this point, Application Server 1 reports the CWSJY0003W message.

Resolving the problem

There are two ways that you can resolve this issue:

- Increase the value of queue manager attribute **`MARKINT`** to a higher value. The default value for **`MARKINT`** is 5000 milliseconds (5 seconds). Increasing this value gives an application server more time to destructively get a message after it is detected. Changing the **`MARKINT`** value affects all applications that connect to the queue manager, and browse messages before applications destructively get messages.
- Change the value to *true* for the **`com.ibm.msg.client.wmq.suppressBrowseMarkMessageWarning`** property in WebSphere Application Server to suppress the CWSJY0003W warning message. To set the variable in WebSphere Application Server, open the administrative console and navigate to **Servers -> Application Servers -> Java and Process Management -> Process Definition -> Java Virtual Machine -> Custom Properties -> New**

```
Name  =  com.ibm.msg.client.wmq.suppressBrowseMarkMessageWarning
Value = true
```

Note: If an activation specification or listener port is connecting to IBM WebSphere MQ using IBM WebSphere MQ messaging provider migration mode the messages can be ignored. The design of this mode of operation means that this message can occur during normal operation.

Related information

Avoiding repeated delivery of browsed messages
ALTER QMGR
Activation specifications
Listener ports running in Application Server Facilities (ASF) mode
Listener ports running in non Application Server Facilities (non-ASF) mode

J2CA0027E messages containing the error The `method 'xa_end' has failed with errorCode '100'`

J2CA0027E messages appear in the WebSphere Application Server SystemOut.log containing the error `The method 'xa_end' has failed with errorCode '100'`.

Introduction

The following errors appear in the WebSphere Application Server SystemOut.log file when applications using the WebSphere Application Server WebSphere MQ messaging provider try to commit a transaction:

```
J2CA0027E: An exception occurred while invoking end on an XA Resource Adapter from
DataSource <JMS Connection Factory>, within transaction ID <Transaction Identifier>:
javax.transaction.xa.XAException: The method 'xa_end' has failed with errorCode '100'.
```

```
J2CA0027E: An exception occurred while invoking rollback on an XA Resource Adapter
from DataSource <JMS Connection Factory>, within transaction ID <Transaction Identifier>:
javax.transaction.xa.XAException: The method 'xa_rollback' has failed with errorCode '-7'.
```

Cause

The cause of these errors can be the result of a WebSphere MQ messaging provider JMS Connection being closed off by WebSphere Application Server because the **Aged timeout** for the Connection has expired.

JMS Connections are created from a JMS Connection Factory. There is a Connection Pool associated with each Connection Factory, which is divided into two parts - the Active Pool and the Free Pool.

When an application closes off a JMS Connection that it has been using, that Connection is moved into the Free Pool of the Connection Pool for the Connection Factory unless the **Aged timeout** for that Connection has elapsed, in which case the Connection is destroyed. If the JMS Connection is still involved in an active transaction when it is destroyed, the application server flows an xa_end() to WebSphere MQ, indicating that all of the transactional work on that Connection has completed.

This causes issues if the JMS Connection had been created inside a transactional message-driven bean that was using either an Activation Specification or a Listener Port to monitor a JMS Destination on a WebSphere MQ queue manager.

In this situation, there is a single transaction that is using 2 connections to WebSphere MQ:

- A connection which is used to get a message from WebSphere MQ and deliver it to the message-driven bean instance for processing.
- A connection that is created within the message-driven bean's onMessage() method.

If the second connection is closed by the message-driven bean, and then destroyed as a result of the **Aged timeout** expiring, then an xa_end() is flown to WebSphere MQ indicating that all of the transactional work has completed.

When the message-driven bean application finishes processing the message it has been given, the application server needs to complete the transaction. It does this by flowing xa_end() to all of the resources that were involved in the transaction, including WebSphere MQ.

However, WebSphere MQ has already received an xa_end() for this particular transaction, and so returns an XA_RBROLLBACK (100) error back to WebSphere Application Server, indicating that the transaction has ended and all of the work WebSphere MQ has been rolled back. This causes the application server to report the following error:

```
J2CA0027E: An exception occurred while invoking end on an XA Resource Adapter from
DataSource <JMS Connection Factory>, within transaction ID <Transaction Identifier>:
javax.transaction.xa.XAException: The method 'xa_end' has failed with errorCode '100'.
```

and then roll back the entire transaction by flowing xa_rollback() to all of the resources enlisted in the transaction. When the application server flows xa_rollback() to WebSphere MQ, the following error occurs:

```
J2CA0027E: An exception occurred while invoking rollback on an XA Resource Adapter
from DataSource <JMS Connection Factory>, within transaction ID <Transaction Identifier>:
javax.transaction.xa.XAException: The method 'xa_rollback' has failed with errorCode '-7'.
```

Environment

Message-driven bean applications that use Activation Specifications or Listener Ports to monitor JMS Destinations hosted on a WebSphere MQ queue manager, and then create a new connection to WebSphere MQ using a JMS Connection Factory from within its onMessage() method, can be affected by this issue.

Resolving the problem

To resolve this issue, ensure that the JMS Connection Factory being used by the application has the Connection Pool property **Aged timeout** set to zero. This will prevent JMS Connections being closed when they are returned to the Free Pool, and so ensures that any outstanding transactional work can be completed.

2035 MQRC_NOT_AUTHORIZED when connecting to IBM MQ from WebSphere Application Server

The *2035 MQRC_NOT_AUTHORIZED* error can occur when an application connects to IBM MQ from WebSphere Application Server.

This topic covers the most common reasons why an application that is running in WebSphere Application Server receives a *2035 MQRC_NOT_AUTHORIZED* error when connecting to IBM MQ. Quick steps to work around the *2035 MQRC_NOT_AUTHORIZED* errors during development are provided in the Resolving the problem section, as well as considerations for implementing security in production environments. A summary is also provided of behavior for outbound scenarios with container-managed and component-managed security and inbound behavior for listener ports and activation specifications.

The cause of the problem

The two most common reasons for why the connection is refused by IBM MQ are described in the following list:

- The user identifier that is passed across the client connection from the application server to IBM MQ is either; not known on the server where the IBM MQ queue manager is running, is not authorized to connect to IBM MQ, or is longer than 12 characters and was truncated. There is more information about how this user identifier is obtained and passed over in *Diagnosing the problem*. For queue managers that are running on Windows, the following error might be seen in the IBM MQ error logs for this scenario: `AMQ8075: Authorization failed because the SID for entity 'wasuser' cannot be obtained.` For UNIX, no entry in the IBM MQ error logs would be seen.
- The user identifier that is passed across the client connection from the application server to IBM MQ is a member of the *mqm* group on the server that hosts the IBM MQ queue manager and a channel authentication record (CHLAUTH) exists that blocks administrative access to the queue manager. IBM MQ configures a CHLAUTH record by default in Version 7.1 and later that blocks all IBM MQ administrators from connecting as a client to the queue manager. The following error in the IBM MQ error logs would be seen for this scenario: `AMQ9777: Channel was blocked`.

For the location of the IBM MQ error logs, see Error log directories.

Diagnosing the problem

To understand the cause of the *2035 MQRC_NOT_AUTHORIZED* reason code, you must understand which user name and password is being used by IBM MQ to authorize the application server.

Note: The understanding that is provided in this topic is helpful for development environments, solving the security requirements of production environments usually requires one of the following approaches:

- Mutual SSL/TLS authentication

  IBM MQ provides features to authenticate a remotely connecting client using the digital certificate that is provided for the SSL/TLS connection.
- A custom, or third party supplied, IBM MQ security exit

  A security exit can be written for IBM MQ that performs user name and password authentication against a repository, such as the local operating system, an IBM MQ server, or an LDAP repository. When you use a security exit for authentication it is important that SSL/TLS transport security is still configured, to ensure that passwords are not sent in plain text.

MCA user ID configured on the server connection channel

If an MCA user ID configured on the server connection channel that the application server is using to connect, and no security exit or mapping channel authentication record is installed, then the MCA user ID overrides the user name that is provided by the application server. It is common practice for many customers to set an MCA user ID on every server connection channel and use mutual SSL/TLS authentication exclusively for authentication.

Default behavior when no credentials are supplied from the application server

If no credentials are supplied by the application on the **createConnection** call, and neither of the component managed or container managed security systems are configured, then WebSphere Application Server provides a blank user name to IBM MQ. This causes IBM MQ to authorize the client based on the user ID that the IBM MQ listener is running under. In most cases the user ID is *mqm* on UNIX or Linux systems and *MUSR_MQADMIN* on Windows. As these users are administrative IBM MQ users, they are blocked by default in Version 7.1 and later, with an *AMQ9777* error logged in the error logs of the queue manager.

Container-managed security for outbound connections

The recommended way to configure the user name and password that is passed to IBM MQ by the application server for outbound connections, is to use container-managed security. Outbound connections are those created by using a connection factory, rather than a listener port or activation specification.

User names of 12 characters or less are passed to IBM MQ by the application server. User names longer than 12 characters in length are truncated, either during authorization (on UNIX), or in the *MQMD* of messages that are sent. Container-managed security means that the deployment descriptor, or EJB 3.0 annotations, of the application declare a resource reference with authentication type set to Container. Then, when the application looks up the connection factory in JNDI, it does so indirectly through the resource reference. For example, an EJB 2.1 application would perform a JNDI lookup as follows, where `jms/MyResourceRef` is declared as a resource reference in the deployment descriptor:

```
ConnectionFactory myCF = (ConnectionFactory)ctx.lookup("java:comp/env/jms/MyResourceRef")
```

An EJB 3.0 application might declare an annotated object property on the bean as follows:

```
@Resource(name = "jms/MyResourceRef"
      authenticationType = AuthenticationType.CONTAINER)
  private javax.jms.ConnectionFactory myCF
```

When the application is deployed by an administrator, they bind this authentication alias to an actual connection factory that has been created in JNDI, and assign it a J2C authentication alias on deployment. It is the user name and password that is contained in this authentication alias that is then passed to IBM MQ or JMS by the application server when the application connects. This approach puts the administrator in control of which user name and password is used by each application, and prevents a different application from looking up the connection factory in JNDI directly to connect with the same user name and password. A default container-managed authentication alias can be supplied on the configuration panels in the administrative console for IBM MQ connection factories. This default is only used in the case that an application uses a resource reference that is configured for container-managed security, but the administrator has not bound it to an authentication alias during deployment.

Default component-managed authentication alias for outbound connection

For cases where it is impractical to change the application to use container-managed security, or to change it to supply a user name and password directly on the createConnection call, it is possible to supply a default. This default is called the component-managed authentication alias and cannot be configured in the administrative console (since WebSphere Application Server Version 7.0 when it was removed from the panels for IBM MQ connection factories). The following scripting samples show how to configure it using wsadmin:

- JACL

```
 wsadmin>set cell [ $AdminConfig getid "/Cell:mycell" ]
mycell(cells/mycell|cell.xml#Cell_1)
wsadmin>$AdminTask listWMQConnectionFactories $cell
MyCF(cells/mycell|resources.xml#MQConnectionFactory_1247500675104)
wsadmin>$AdminTask modifyWMQConnectionFactory MyCF(cells/mycell|
resources.xml#MQConnectionFactory_1247500675104) { -componentAuthAlias myalias }
MyCF(cells/mycell|resources.xml#MQConnectionFactory_1247500675104)
```

- Jython

```
wsadmin>cell = AdminConfig.getid("/Cell:mycell")
wsadmin>AdminTask.listWMQConnectionFactories(cell)
```

```
'MyCF(cells/mycell|resources.xml#MQConnectionFactory_1247500675104)'
wsadmin>AdminTask.modifyWMQConnectionFactory('MyCF(cells/mycell|resos
urces.xml#MQConnectionFactory_1247500675104)', "-componentAuthAlias myalias")
'MyCF(cells/mycell|resources.xml#MQConnectionFactory_1247500675104)'
```

Authentication alias for inbound MDB connections using an activation specification

For inbound connections that use an activation specification, an authentication alias can be specified by the administrator when the application is deployed, or a default authentication alias can be specified on the activation specification in the administrative console.

Authentication alias for inbound MDB connections using a listener port

For inbound connections that use a listener port, the value that is specified in the container-managed authentication alias setting of the connection factory is used. z/OS On z/OS, first the container-managed authentication alias is checked and used if set, then the component-managed authentication alias is checked and used it set.

Resolving the problem

The simplest steps to resolve the *2035 MQRC_NOT_AUTHORIZED* errors in a development environment, where full transport security is not required, are as follows:

- Choose a user that you want WebSphere Application Server to be authenticated as. Typically the user chosen should have authority relevant to the context of the operations required by the application running in WebSphere Application Server and no more. For example, *mqm* or other super user is not appropriate.
- If this user is an IBM MQ administrative user, then relax the channel authentication record (CHLAUTH) security in Version 7.1 or later so that administrative connections are not blocked on the server connection channel you want to use. An example MQSC command for a server connection channel called `WAS.CLIENTS` is, `SET CHLAUTH('WAS.CLIENTS') TYPE(BLOCKUSER) USERLIST(ALLOWANY)`.
- Configure the server connection channel to set the MCA user ID (MCAUSER) to the user you are using. An example MQSC command to configure a server connection channel to use `myuser` as the MCA user ID is, `ALTER CHL('WAS.CLIENTS') CHLTYPE(SVRCONN) MCAUSER('myuser')`.

Important extra considerations for production environments

For all production environments where transport security is required, SSL/TLS security must be configured between the application server and IBM MQ.

To configure SSL/TLS transport security, you must establish the appropriate trust between the IBM MQ queue manager and WebSphere Application Server. The application server initiates the SSL/TLS handshake and must always be configured to trust the certificate that is provided by the IBM MQ queue manager. If the application server is configured to send a certificate to the IBM MQ queue manager, then the queue manager must also be configured to trust it. If trust is not correctly configured on both sides, you will encounter *2393 MQRC_SSL_INITIALIZATION_ERROR* reason code after SSL/TLS is enabled on the connection.

If you do not have a security exit that performs username and password authentication, then you should configure mutual SSL/TLS authentication on your server connection channel to cause the queue manager to require a trusted certificate is provided by the application server. To do this you set *SSL Authentication* to `Required` in MQ Explorer or `SSLCAUTH(REQUIRED)` in MQSC.

If you do have a security exit that performs user name and password authentication that is installed in your IBM MQ server, then configure your application to supply a username and password for validation by that security exit. The details of how to configure the user name and password that is passed to IBM MQ by the application server are described previously in the *Diagnosing the problem* section.

All server connection channels that do not have SSL/TLS security should be disabled. Example MQSC commands to disable the *SYSTEM.DEF.SVRCONN* channel are provided as follows (assuming no user exists on the IBM MQ server called *('NOAUTH')*, `ALTER CHL(SYSTEM.DEF.SVRCONN) CHLTYPE(SVRCONN) MCAUSER('NOAUTH') STOP CHL(SYSTEM.DEF.SVRCONN)`.

For instructions to configure the private certificate and trust of a IBM MQ queue manager and to enable SSL security on a server connection channel, see Configuring SSL on queue managers and Configuring SSL channels.

For information about using SSL/TLS from WebSphere Application Server and whether the application server sends a certificate to IBM MQ for authentication, see the following information:

- To create or modify an SSL configuration to contain the appropriate SSL/TLS configuration for connection to IBM MQ, see SSL configurations in the WebSphere Application Server product documentation.
- It is required by IBM MQ that you must specify a matching CipherSpec on both ends of the connection. For more information about CipherSpecs and CipherSuites that can be used with IBM MQ, see CipherSuite and CipherSpec name mappings for connections to a WebSphere® MQ queue manager.
- For more information about enabling SSL/TLS on a client connect and choosing which SSL configuration to use, see WebSphere MQ messaging provider connection factory settings and WebSphere MQ messaging provider activation specification settings in the WebSphere Application Server product documentation.

Problem determination for the IBM MQ resource adapter

When using the IBM MQ resource adapter, most errors cause exceptions to be thrown, and these exceptions are reported to the user in a manner that depends on the application server. The resource adapter makes extensive use of linked exceptions to report problems. Typically, the first exception in a chain is a high-level description of the error, and subsequent exceptions in the chain provide the more detailed information that is required to diagnose the problem.

For example, if the IVT program fails to obtain a connection to a IBM MQ queue manager, the following exception might be thrown:

```
javax.jms.JMSException: MQJCA0001: An exception occurred in the JMS layer.
See the linked exception for details.
```

Linked to this exception is a second exception:

```
javax.jms.JMSException: MQJMS2005: failed to create an MQQueueManager for
'localhost:ExampleQM'
```

This exception is thrown by IBM MQ classes for JMS and has a further linked exception:

```
com.ibm.mq.MQException: MQJE001: An MQException occurred: Completion Code 2,
Reason 2059
```

This final exception indicates the source of the problem. Reason code 2059 is MQRC_Q_MGR_NOT_AVAILABLE, which indicates that the queue manager specified in the definition of the ConnectionFactory object might not have been started.

If the information provided by exceptions is not sufficient to diagnose a problem, you might need to request a diagnostic trace. For information about how to enable diagnostic tracing, see Configuration of the IBM MQ resource adapter.

Configuration problems commonly occur in the following areas:

- Deploying the resource adapter
- Deploying MDBs
- Creating connections for outbound communication

Related information

Using the IBM MQ resource adapter

Problems in deploying the resource adapter

If the resource adapter fails to deploy, check that Java EE Connector Architecture (JCA) resources are configured correctly. If IBM MQ is already installed, check that the correct versions of the JCA and IBM MQ classes for JMS are in the class path.

Failures in deploying the resource adapter are generally caused by not configuring JCA resources correctly. For example, a property of the ResourceAdapter object might not be specified correctly, or the deployment plan required by the application server might not be written correctly. Failures might also occur when the application server attempts to create objects from the definitions of JCA resources and bind the objects into the Java Naming Directory Interface (JNDI) namespace, but certain properties are not specified correctly or the format of a resource definition is incorrect.

The resource adapter can also fail to deploy because it loaded incorrect versions of JCA or IBM MQ classes for JMS classes from JAR files in the class path. This type of failure can commonly occur on a system where IBM MQ is already installed. On such a system, the application server might find existing copies of the IBM MQ classes for JMS JAR files and load classes from them in preference to the classes supplied in the IBM MQ resource adapter RAR file.

Related information

What is installed for IBM MQ classes for JMS

Configuring the application server to use the latest resource adapter maintenance level

Problems in deploying MDBs

Failures when the application server attempts to start message delivery to an MDB might be caused by an error in the definition of the associated ActivationSpec object, or by missing resources.

Failures might occur when the application server attempts to start message delivery to an MDB. This type of failure is typically caused by an error in the definition of the associated ActivationSpec object, or because the resources referenced in the definition are not available. For example, the queue manager might not be running, or a specified queue might not exist.

An ActivationSpec object attempts to validate its properties when the MDB is deployed. Deployment then fails if the ActivationSpec object has any properties that are mutually exclusive or does not have all the required properties. However, not all problems associated with the properties of the ActivationSpec object can be detected at this time.

Failures to start message delivery are reported to the user in a manner that depends on the application server. Typically, these failures are reported in the logs and diagnostic trace of the application server. If enabled, the diagnostic trace of the IBM MQ resource adapter also records these failures.

Problems in creating connections for outbound communication

A failure in outbound communication can occur if a ConnectionFactory object cannot be found, or if the ConnectionFactory object is found but a connection cannot be created. There are various reasons for either of these problems.

Failures in outbound communication typically occur when an application attempts to look up and use a ConnectionFactory object in a JNDI namespace. A JNDI exception is thrown if the ConnectionFactory object cannot be found in the namespace. A ConnectionFactory object might not be found for the following reasons:

- The application specified an incorrect name for the ConnectionFactory object.
- The application server was not able to create the ConnectionFactory object and bind it into the namespace. In this case, the startup logs of the application server typically contain information about the failure.

If the application successfully retrieves the ConnectionFactory object from the JNDI namespace, an exception might still be thrown when the application calls the ConnectionFactory.createConnection() method. An exception in this context indicates that it is not possible to create a connection to an IBM MQ queue manager. Here are some common reasons why an exception might be thrown:

- The queue manager is not available, or cannot be found using the properties of the ConnectionFactory object. For example, the queue manager is not running, or the specified host name, IP address, or port number of the queue manager is incorrect.
- The user is not authorized to connect to the queue manager. For a client connection, if the createConnection() call does not specify a user name, and the application server supplies no user identity information, the JVM process ID is passed to the queue manager as the user name. For the connection to succeed, this process ID must be a valid user name in the system on which the queue manager is running.
- The ConnectionFactory object has a property called ccdtURL and a property called channel. These properties are mutually exclusive.
- On an SSL connection, the SSL-related properties, or the SSL-related attributes in the server connection channel definition, have not been specified correctly.
- The sslFipsRequired property has different values for different JCA resources. For more information about this limitation, see Limitations of the IBM MQ resource adapter.

Related information

Specifying that only FIPS-certified CipherSpecs are used at run time on the MQI client

Federal Information Processing Standards (FIPS) for UNIX, Linux and Windows

V 8.0.0.4 Using IBM MQ connection property override

Connection property override allows you to change the details that are used by a client application to connect to a queue manager, without modifying the source code.

About this task

Sometimes, it is not possible to modify the source code for an application, for example, if the application is a legacy application and the source code is no longer available.

In this situation, if an application needs to specify different properties when it is connecting to a queue manager, or is required to connect to a different queue manager, then you can use the connection override functionality to specify the new connection details or queue manager name.

The connection property override is supported for two clients:

- IBM MQ classes for JMS
- IBM MQ classes for Java

You can override the properties that you want to change by defining them in a configuration file that is then read by the IBM MQ classes for JMS or IBM MQ classes for Java at startup.

When the connection override functionality is in use, all applications that are running inside the same Java runtime environment pick up and use the new property values. If multiple applications that are using either the IBM MQ classes for JMS or the IBM MQ classes for Java are running inside the same Java runtime environment, it is not possible to just override properties for individual applications.

Important: This functionality is only supported for situations where it is not possible to modify the source code for an application. It must not be used for applications where the source code is available and can be updated.

Related concepts

"Tracing IBM MQ classes for JMS applications" on page 373
The trace facility in IBM MQ classes for JMS is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

Related tasks

"Tracing IBM MQ classes for Java applications" on page 377

The trace facility in IBM MQ classes for Java is provided to help IBM Support to diagnose customer issues. Various properties control the behavior of this facility.

Related information

Using IBM MQ classes for JMS

Using IBM MQ classes for Java

V 8.0.0.4 Using connection property override in IBM MQ classes for JMS

If a connection factory is created programmatically, and it is not possible to modify the source code for the application that creates it, then the connection override functionality can be used to change the properties that the connection factory uses when a connection is created. However, the use of the connection override functionality with connection factories defined in JNDI is not supported.

About this task

In the IBM MQ classes for JMS, details about how to connect to a queue manager are stored in a connection factory. Connection factories can either be defined administratively and stored in a JNDI repository, or created programmatically by an application by using Java API calls.

If an application creates a connection factory programmatically, and it is not possible to modify the source code for that application, the connection override functionality allows you to override the connection factory properties in the short term. In the long term, though, you must put plans in place to allow the connection factory used by the application to be modified without using the connection override functionality.

If the connection factory that is created programmatically by an application is defined to use a Client Channel Definition Table (CCDT), then the information in the CCDT is used in preference to the overridden properties. If the connection details that the application uses need to be changed, then a new version of the CCDT must be created and made available to the application.

The use of the connection override functionality with connection factories defined in JNDI is not supported. If an application uses a connection factory that is defined in JNDI, and the properties of that connection factory need to be changed, then the definition of the connection factory must be updated in JNDI. Although the connection override functionality is applied to these connection factories (and the overridden properties take precedence over the properties in the connection factory definition that is looked up in JNDI), this use of the connection override functionality is not supported.

Important: The connection override functionality affects all of the applications that are running inside of a Java runtime environment, and applies to all of the connection factories used by those applications. It is not possible to just override properties for individual connection factories or applications.

When an application uses a connection factory to create a connection to a queue manager, the IBM MQ classes for JMS look at the properties that have been overridden and use those property values when creating the connection, rather than the values for the same properties in the connection factory.

For example, suppose a connection factory has been defined with the PORT property set to 1414. If the connection override functionality has been used to set the PORT property to 1420, then when the connection factory is used to create a connection, the IBM MQ classes for JMS use a value of 1420 for the PORT property, rather than 1414.

To modify any of the connection properties that are used when creating a JMS connection from a connection factory, the following steps need to be carried out:

1. Add the properties to be overridden to an IBM MQ classes for JMS configuration file.
2. Enable the connection override functionality.
3. Start the application, specifying the configuration file.

Procedure

1. Add the properties to be overridden to an IBM MQ classes for JMS configuration file.

a) Create a file containing the properties and values that need to be overridden in the standard Java properties format.

   For details about how you create a properties file, see The IBM MQ classes for JMS configuration file.

b) To override a property, add an entry to the properties file.

   Any IBM MQ classes for JMS connection factory property can be overridden. Add each required entry in the following format:

```
jmscf.<property name>=<value>
```

   where <*property name*> is the JMS administration property name or XMSC constant for the property that needs to be overridden. For a list of connection factory properties, see Properties of IBM MQ classes for JMS objects.

For example, to set the name of the channel that an application should use to connect to a queue manager, you can add the following entry to the properties file:

```
jmscf.channel=MY.NEW.SVRCONN
```

2. Enable the connection override functionality.

   To enable connection override, set the **com.ibm.msg.client.jms.overrideConnectionFactory** property to be true so that the properties that are specified in the properties file are used to override the values that are specified in the application. You can either set the extra property as another property in the configuration file itself, or pass the property as a Java system property by using:

```
-Dcom.ibm.msg.client.jms.overrideConnectionFactory=true
```

3. Start the application, specifying the configuration file.

   Pass the properties file that you created to the application at run time by setting the Java system property:

```
-Dcom.ibm.msg.client.config.location
```

   Note that the location of the configuration file must be specified as a URI, for example:

```
-Dcom.ibm.msg.client.config.location=file:///jms/jms.config
```

Results

When the connection override functionality is enabled, the IBM MQ classes for JMS write an entry to the jms log whenever a connection is made. The information in the log shows the connection factory properties that were overridden when the connection was created, as shown in the following example entry:

```
Overriding ConnectionFactory properties:
          Overriding property channel:
                Original value = MY.OLD.SVRCONN
                New value      = MY.NEW.SVRCONN
```

Related tasks

"Using connection property override in IBM MQ classes for Java" on page 466

In the IBM MQ classes for Java, connection details are set as properties using a combination of different values. The connection override functionality can be used to override the connection details that an application uses if it is not possible to modify the source code for the application.

"Overriding connection properties: example with IBM MQ classes for JMS " on page 468
This example shows how to override properties when you are using the IBM MQ classes for JMS.

Related information

Creating and configuring connection factories and destinations in an IBM MQ classes for JMS application

Configuring connection factories and destinations in a JNDI namespace

Using connection property override in IBM MQ classes for Java

In the IBM MQ classes for Java, connection details are set as properties using a combination of different values. The connection override functionality can be used to override the connection details that an application uses if it is not possible to modify the source code for the application.

About this task

The different values that are used to set the connection properties are a combination of:

- Assigning values to static fields on the **`MQEnvironment`** class.
- Setting property values in the properties `Hashtable` in the **`MQEnvironment`** class.
- Setting property values in a `Hashtable` passed into an **`MQQueueManager`** constructor.

These properties are then used when an application constructs an MQQueueManager object, which represents a connection to a queue manager.

If it is not possible to modify the source code for an application that uses the IBM MQ classes for Java to specify different properties that must be used when creating a connection to a queue manager, the connection override functionality allows you to override the connection details in the short term. In the long term, though, you must put plans in place to allow the connection details used by the application to be modified without using the connection override functionality.

When an application creates an MQQueueManager, the IBM MQ classes for Java look at the properties that have been overridden and use those property values when creating a connection to the queue manager, rather than the values in any of the following locations:

- The static fields on the MQEnvironment class
- The properties Hashtable stored in the MQEnvironment class
- The properties Hashtable that is passed into an MQQueueManager constructor

For example, suppose an application creates an MQQueueManager, passing in a properties Hashtable that has the CHANNEL property set to MY.OLD.CHANNEL. If the connection override functionality has been used to set the CHANNEL property to MY.NEW.CHANNEL, then when the MQQueueManager is constructed, the IBM MQ classes for Java attempt to create a connection to the queue manager by using the channel MY.NEW.CHANNEL rather than MY.OLD.CHANNEL.

Note: If an MQQueueManager is configured to use a Client Channel Definition Table (CCDT), then the information in the CCDT is used in preference to the overridden properties. If the connection details that the application creating the MQQueueManager uses need to be changed, then a new version of the CCDT must be created and made available to the application.

To modify any of the connection properties that are used when creating an MQQueueManager, the following steps need to be carried out:

1. Create a properties file called `mqclassesforjava.config`.
2. Enable the connection property override functionality by setting the **`OverrideConnectionDetails`** property to true.
3. Start the application, specifying the configuration file as part of the Java invocation.

Procedure

1. Create a properties file called `mqclassesforjava.config` containing the properties and values that need to be overridden.

   It is possible to override 13 properties that are used by the IBM MQ classes for Java when connecting to a queue manager as part of the MQQueueManager constructor. The names of these properties, and the keys that must be specified when you are overriding them, are shown in the following table:

   Table 20. Properties that can be overridden

Property	Property key
CCSID	$CCSID_PROPERTY
Channel	$CHANNEL_PROPERTY
Connect options	$CONNECT_OPTIONS_PROPERTY
Hostname	$HOST_NAME_PROPERTY
SSL key reset	$SSL_RESET_COUNT_PROPERTY
Local address	$LOCAL_ADDRESS_PROPERTY
Queue manager name	qmgr
Password	$PASSWORD_PROPERTY
Port	$PORT_PROPERTY
Cipher suite	$SSL_CIPHER_SUITE_PROPERTY
FIPS required	$SSL_FIPS_REQUIRED_PROPERTY
SSL peer name	$SSL_PEER_NAME_PROPERTY
User ID	$USER_ID_PROPERTY

   Note: All of the property keys start with the $ character, except for the queue manager name. The reason for this is because the queue manager name is passed in to the MQQueueManager constructor as an argument, rather than being set as either a static field on the MQEnvironment class, or a property in a Hashtable, and so internally this property needs to be treated slightly differently from the other properties.

   To override a property, add an entry in the following format to the properties file:

   ```
   mqj.<property key>=<value>
   ```

   For example, to set the name of the channel to be used when creating MQQueueManager objects, you can add the following entry to the properties file:

   ```
   mqj.$CHANNEL_PROPERTY=MY.NEW.CHANNEL
   ```

   To change the name of the queue manager that an MQQueueManager object connects to, you can add the following entry to the properties file:

   ```
   mqj.qmgr=MY.OTHER.QMGR
   ```

2. Enable the connection override functionality by setting the **`com.ibm.mq.overrideConnectionDetails`** property to be true.

   Setting the property **`com.ibm.mq.overrideConnectionDetails`** to be true means that the properties that are specified in the properties file are used to override the values specified in the application. You can either set the extra property as another property in the configuration file itself, or pass the property as a system property, by using:

```
-Dcom.ibm.mq.overrideConnectionDetails=true
```

3. Start the application.

   Pass the properties file you created to the client application at run time by setting the Java system property:

```
-Dcom.ibm.msg.client.config.location
```

   Note that the location of the configuration file must be specified as a URI, for example:

```
-Dcom.ibm.msg.client.config.location=file:///classesforjava/mqclassesforjava.config
```

V 8.0.0.4 Overriding connection properties: example with IBM MQ classes for JMS

This example shows how to override properties when you are using the IBM MQ classes for JMS.

About this task

The following code example shows how an application creates a ConnectionFactory programmatically:

```
JmsSampleApp.java
...
JmsFactoryFactory jmsff;
JmsConnectionFactory jmsConnFact;

jmsff = JmsFactoryFactory.getInstance(JmsConstants.WMQ_PROVIDER);
jmsConnFact = jmsff.createConnectionFactory();

jmsConnFact.setStringProperty(WMQConstants.WMQ_HOST_NAME,"127.0.0.1");
jmsConnFact.setIntProperty(WMQConstants.WMQ_PORT, 1414);
jmsConnFact.setStringProperty(WMQConstants.WMQ_QUEUE_MANAGER,"QM_V80");
jmsConnFact.setStringProperty(WMQConstants.WMQ_CHANNEL,"MY.CHANNEL");
jmsConnFact.setIntProperty(WMQConstants.WMQ_CONNECTION_MODE,
                           WMQConstants.WMQ_CM_CLIENT);
...
```

The ConnectionFactory is configured to connect to the queue manager QM_V80 using the CLIENT transport and channel MY.CHANNEL.

You can override the connection details by using a properties file, and force the application to connect to a different channel, by using the following procedure.

Procedure

1. Create an IBM MQ classes for JMS configuration file that is called `jms.config` in the /*<userHome>* directory (where *<userHome>* is your home directory).

   Create this file with the following contents:

```
jmscf.CHANNEL=MY.TLS.CHANNEL
jmscf.SSLCIPHERSUITE=TLS_RSA_WITH_AES_128_CBC_SHA256
```

2. Run the application, passing the following Java system properties into the Java runtime environment that the application is running in:

```
-Dcom.ibm.msg.client.config.location=file:///<userHome>/jms.config
-Dcom.ibm.msg.client.jms.overrideConnectionFactory=true
```

Results

Carrying out this procedure overrides the ConnectionFactory that was created programmatically by the application, so that when the application creates a connection, it tries to connect by using the channel MY.TLS.CHANNEL and the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA256.

Related tasks

"Using IBM MQ connection property override" on page 463
Connection property override allows you to change the details that are used by a client application to connect to a queue manager, without modifying the source code.

"Using connection property override in IBM MQ classes for JMS" on page 464
If a connection factory is created programmatically, and it is not possible to modify the source code for the application that creates it, then the connection override functionality can be used to change the properties that the connection factory uses when a connection is created. However, the use of the connection override functionality with connection factories defined in JNDI is not supported.

"Using connection property override in IBM MQ classes for Java" on page 466
In the IBM MQ classes for Java, connection details are set as properties using a combination of different values. The connection override functionality can be used to override the connection details that an application uses if it is not possible to modify the source code for the application.

Resolving problems with IBM MQ MQI clients

This collection of topics contains information about techniques for solving problems in IBM MQ MQI client applications.

An application running in the IBM MQ MQI client environment receives MQRC_* reason codes in the same way as IBM MQ server applications. However, there are additional reason codes for error conditions associated with IBM MQ MQI clients. For example:

- Remote machine not responding
- Communications line error
- Invalid machine address

The most common time for errors to occur is when an application issues an MQCONN or MQCONNX and receives the response MQRC_Q_MQR_NOT_AVAILABLE. Look in the client error log for a message explaining the failure. There might also be errors logged at the server, depending on the nature of the failure. Also, check that the application on the IBM MQ MQI client is linked with the correct library file.

IBM MQ MQI client fails to make a connection

An MQCONN or MQCONNX might fail because there is no listener program running on the server, or during protocol checking.

When the IBM MQ MQI client issues an MQCONN or MQCONNX call to a server, socket and port information is exchanged between the IBM MQ MQI client and the server. For any exchange of information to take place, there must be a program on the server with the role to 'listen' on the communications line for any activity. If there is no program doing this, or there is one but it is not configured correctly, the MQCONN or MQCONNX call fails, and the relevant reason code is returned to the IBM MQ MQI client application.

If the connection is successful, IBM MQ protocol messages are exchanged and further checking takes place. During the IBM MQ protocol checking phase, some aspects are negotiated while others cause the connection to fail. It is not until all these checks are successful that the MQCONN or MQCONNX call succeeds.

For information about the MQRC_* reason codes, see API reason codes.

Stopping IBM MQ MQI clients

Even though an IBM MQ MQI client has stopped, it is still possible for the associated process at the server to be holding its queues open. The queues are not closed until the communications layer detects that the partner has gone.

If sharing conversations is enabled, the server channel is always in the correct state for the communications layer to detect that the partner has gone.

Error messages with IBM MQ MQI clients

When an error occurs with an IBM MQ MQI client system, error messages are put into the IBM MQ system error files.

- On UNIX and Linux systems, these files are found in the `/var/mqm/errors` directory
- On Windows, these files are found in the errors subdirectory of the IBM MQ MQI client installation. Usually this directory is `C:\Program Files\IBM\WebSphere MQ\errors`.
- On IBM i, these files are found in the `/QIBM/UserData/mqm/errors` directory

Certain client errors can also be recorded in the IBM MQ error files associated with the server to which the client was connected.

Multicast troubleshooting

The following hints and tips are in no significant order, and might be added to when new versions of the documentation are released. They are subjects that, if relevant to the work that you are doing, might save you time.

Testing multicast applications on a non-multicast network

Use this information to learn how to test IBM MQ Multicast applications locally instead of over a multicast network.

When developing or testing multicast applications you might not yet have a multicast enabled network. To run the application locally, you must edit the `mqclient.ini` file as shown in the following example:

Edit the `Interface` parameter in the `Multicast` stanza of the *MQ_DATA_PATH* `/mqclient.ini`:

```
Multicast:
Interface       = 127.0.0.1
```

where *MQ_DATA_PATH* is the location of the IBM MQ data directory (`/var/mqm/mqclient.ini`).

The multicast transmissions now only use the local loopback adapter.

Setting the appropriate network for multicast traffic

When developing or testing multicast applications, after testing them locally, you might want to test them over a multicast enabled network. If the application only transmits locally, you might have to edit the `MQClient.ini` file as shown later in this section. If the machine setup is using multiple network adapters, or a virtual private network (VPN) for example, the **`Interface`** parameter in the `MQClient.ini` file must be set to the address of the network adapter you want to use.

If the `Multicast` stanza exists in the `MQClient.ini` file, edit the **`Interface`** parameter as shown in the following example:

Change:

```
Multicast:
Interface       = 127.0.0.1
```

To:

```
Multicast:
Interface       = IPAddress
```

where *IPAddress* is the IP address of the interface on which multicast traffic flows.

If there is no `Multicast` stanza in the `MQClient.ini` file, add the following example:

```
Multicast:
Interface       = IPAddress
```

where *IPAddress* is the IP address of the interface on which multicast traffic flows.

The multicast applications now run over the multicast network.

Multicast topic string is too long

If your IBM MQ Multicast topic string is rejected with reason code MQRC_TOPIC_STRING_ERROR, it might be because the string is too long.

WebSphereMQ Multicast has a 255 character limit for topic strings. This limitation means that care must be taken with the names of nodes and leaf-nodes within the tree; if the names of nodes and leaf-nodes are too long, the topic string might exceed 255 characters and return the "2425 (0979) (RC2425): MQRC_TOPIC_STRING_ERROR" on page 190 reason code. It is recommended to make topic strings as short as possible because longer topic strings might have a detrimental effect on performance.

Multicast topic topology issues

Use these examples to understand why certain IBM MQ Multicast topic topologies are not recommended.

As was mentioned in IBM MQ Multicast topic topology, IBM MQ Multicast support requires that each subtree has its own multicast group and data stream within the total hierarchy. Do not use a different multicast group address for a subtree and its parent.

The *classful network* IP addressing scheme has designated address space for multicast address. The full multicast range of IP address is `224.0.0.0` to `239.255.255.255`, but some of these addresses are reserved. For a list of reserved address either contact your system administrator or see https://www.iana.org/assignments/multicast-addresses for more information. It is recommended that you use the locally scoped multicast address in the range of `239.0.0.0` to `239.255.255.255`.

Recommended multicast topic topology

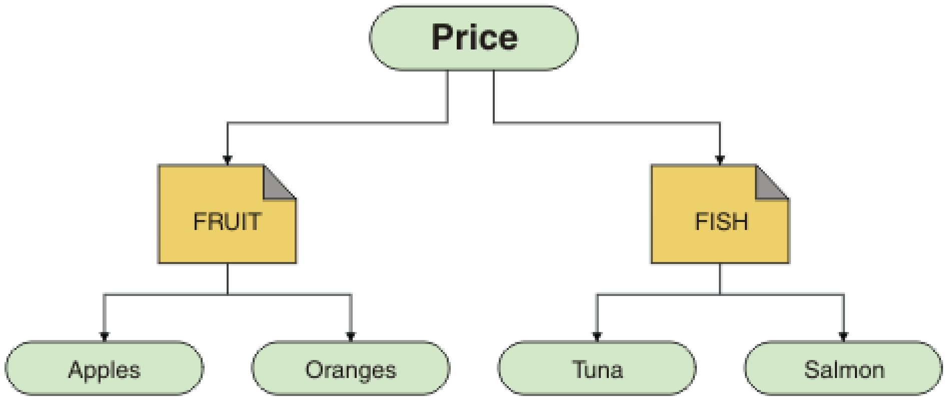
This example is the same as the one from IBM MQ Multicast topic topology, and shows 2 possible multicast data streams. Although it is a simple representation, it demonstrates the kind of situation that IBM MQ Multicast was designed for, and is shown here to contrast the second example:

```
DEF COMMINFO(MC1) GRPADDR(
227.20.133.1)

DEF COMMINFO(MC2) GRPADDR(227.20.133.2)
```

where *`227.20.133.1`* and *`227.20.133.2`* are valid multicast addresses.

These topic definitions are used to create a topic tree as shown in the following diagram:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)
```



Each multicast communication information (COMMINFO) object represents a different stream of data because their group addresses are different. In this example, the topic FRUIT is defined to use COMMINFO object MC1 , and the topic FISH is defined to use COMMINFO object MC2 .

IBM MQ Multicast has a 255 character limit for topic strings. This limitation means that care must be taken with the names of nodes and leaf-nodes within the tree; if the names of nodes and leaf-nodes are too long, the topic string might exceed 255 characters and return the MQRC_TOPIC_STRING_ERROR reason code.

Non-recommended multicast topic topology

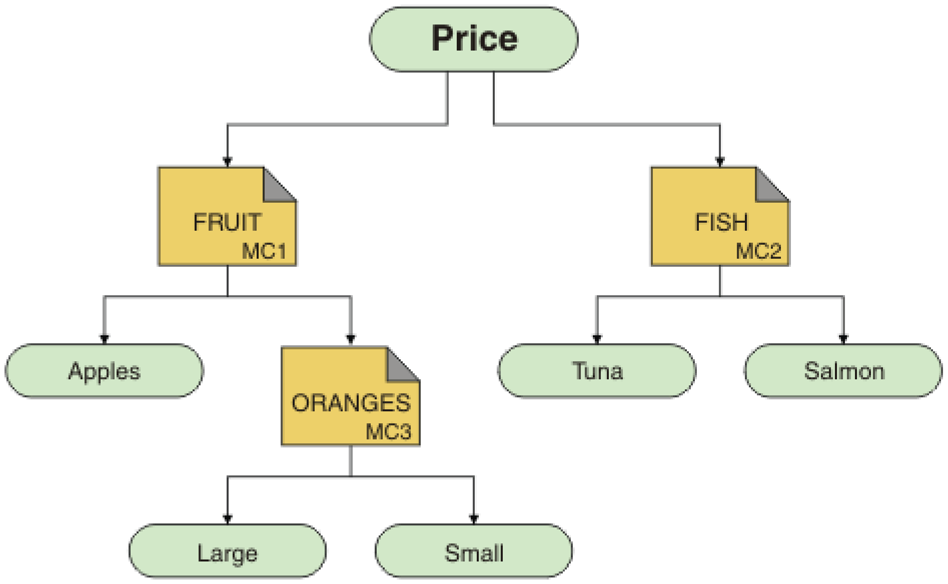
This example extends the previous example by adding another topic object called ORANGES which is defined to use another COMMINFO object definition (MC3):

```
DEF COMMINFO(MC1) GRPADDR(227.20.133.1
)

DEF COMMINFO(MC2) GRPADDR(227.20.133.2)

DEF COMMINFO(MC3) GRPADDR(227.20.133.3)
```

where *227.20.133.1*, *227.20.133.2*, and *227.20.133.3* are valid multicast addresses.

These topic definitions are used to create a topic tree as shown in the following diagram:

```
DEFINE TOPIC(FRUIT) TOPICSTRING('Price/FRUIT') MCAST(ENABLED) COMMINFO(MC1)

DEFINE TOPIC(FISH) TOPICSTRING('Price/FISH') MCAST(ENABLED) COMMINFO(MC2)

DEFINE TOPIC(ORANGES) TOPICSTRING('Price/FRUIT/ORANGES') MCAST(ENABLED) COMMINFO(MC3)
```



While this kind of multicast topology is possible to create, it is not recommended because applications might not receive the data that they were expecting.

An application subscribing on `'Price/FRUIT/#'` receives multicast transmission on the `COMMINFO MC1` group address. The application expects to receive publications on all topics at or below that point in the topic tree.

However, the messages created by an application publishing on `'Price/FRUIT/ORANGES/Small'` are not received by the subscriber because the messages are sent on the group address of `COMMINFO MC3`.

Queue manager clusters troubleshooting

Use the checklist given here, and the advice given in the subtopics, to help you to detect and deal with problems when you use queue manager clusters.

Before you begin

If your problems relate to publish/subscribe messaging using clusters, rather than to clustering in general, see "Routing for publish/subscribe clusters: Notes on behavior" on page 442.

Procedure

- Check that your cluster channels are all paired.

  Each cluster sender channel connects to a cluster receiver channel of the same name. If there is no local cluster receiver channel with the same name as the cluster sender channel on the remote queue manager, then it won't work.
- Check that your channels are running. No channels should be in RETRYING state permanently.

  Show which channels are running using the following command:

  ```
  runmqsc display chstatus(*)
  ```

  If you have channels in RETRYING state, there might be an error in the channel definition, or the remote queue manager might not be running. While channels are in this state, messages are likely to build up on transmit queues. If channels to full repositories are in this state, then the definitions of cluster objects (for example queues and queue managers) become out-of-date and inconsistent across the cluster.
- Check that no channels are in STOPPED state.

Channels go into STOPPED state when you stop them manually. Channels that are stopped can be restarted using the following command:

```
runmqsc start channel(xyz)
```

A clustered queue manager auto-defines cluster channels to other queue managers in a cluster, as required. These auto-defined cluster channels start automatically as needed by the queue manager, unless they were previously stopped manually. If an auto-defined cluster channel is stopped manually , the queue manager remembers that it was manually stopped and does not start it automatically in the future. If you need to stop a channel, either remember to restart it again at a convenient time, or else issue the following command:

```
stop channel(xyz) status(inactive)
```

The `status(inactive)` option allows the queue manager to restart the channel at a later date if it needs to do so.

- Check that all queue managers in the cluster are aware of all the full repositories.

  You can do this using the following command:

  ```
  runmqsc display clusqmgr(*) qmtype
  ```

  Partial repositories might not be aware of all other partial repositories. All full repositories should be aware of all queue managers in the cluster. If cluster queue managers are missing, this might mean that certain channels are not running correctly.
- Check that every queue manager (full repositories and partial repositories) in the cluster has a manually defined cluster receiver channel running and is defined in the correct cluster.

  To see which other queue managers are talking to a cluster receiver channel, use the following command:

  ```
  runmqsc display channel(*) rqmname
  ```

  Check that each manually defined cluster receiver has a **conname** parameter defined to be `ipaddress(port)`. Without a correct connection name, the other queue manager does not know the connection details to use when connecting back.
- Check that every partial repository has a manually defined cluster sender channel running to a full repository, and defined in the correct cluster.

  The cluster sender channel name must match the cluster receiver channel name on the other queue manager.
- Check that every full repository has a manually defined cluster sender channel running to every other full repository, and defined in the correct cluster.

  The cluster sender channel name must match the cluster receiver channel name on the other queue manager. Each full repository does not keep a record of what other full repositories are in the cluster. It assumes that any queue manager to which it has a manually defined cluster sender channel is a full repository.
- Check the dead letter queue.

  Messages that the queue manager cannot deliver are sent to the dead letter queue.
- Check that, for each partial repository queue manager, you have defined a single cluster-sender channel to one of the full repository queue managers.

  This channel acts as a "bootstrap" channel through which the partial repository queue manager initially joins the cluster.
- Check that the intended full repository queue managers are actual full repositories and are in the correct cluster.

You can do this using the following command:

```
runmqsc display qmgr repos reposnl
```

- Check that messages are not building up on transmit queues or system queues.

  You can check transmit queues using the following command:

  ```
  runmqsc display ql(*) curdepth where (usage eq xmitq)
  ```

  You can check system queues using the following command:

  ```
  display ql(system*) curdepth
  ```

Related concepts

"Making initial checks on Windows, UNIX and Linux systems" on page 9
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

"Making initial checks on z/OS" on page 28
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

"Making initial checks on IBM i" on page 18
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

"Reason codes and exceptions" on page 43
You can use the following messages and reason codes to help you solve problems with your IBM MQ components or applications.

Related information

Configuring a queue manager cluster

Application issues seen when running REFRESH CLUSTER

Issuing **REFRESH CLUSTER** is disruptive to the cluster. It might make cluster objects invisible for a short time until the **REFRESH CLUSTER** processing completes. This can affect running applications. These notes describe some of the application issues you might see.

Reason codes that you might see from MQOPEN, MQPUT, or MQPUT1 calls

During **REFRESH CLUSTER** the following reason codes might be seen. The reason why each of these codes appears is described in a later section of this topic.

- 2189 MQRC_CLUSTER_RESOLUTION_ERROR
- 2085 MQRC_UNKNOWN_OBJECT_NAME
- 2041 MQRC_OBJECT_CHANGED
- 2082 MQRC_UNKNOWN_ALIAS_BASE_Q
- 2270 MQRC_NO_DESTINATIONS_AVAILABLE

All these reason codes indicate name lookup failures at one level or another in the IBM MQ code, which is to be expected if apps are running throughout the time of the **REFRESH CLUSTER** operation.

The **REFRESH CLUSTER** operation might be happening locally, or remotely, or both, to cause these outcomes. The likelihood of them appearing is especially high if full repositories are very busy. This happens if **REFRESH CLUSTER** activities are running locally on the full repository, or remotely on other queue managers in the cluster or clusters that the full repository is responsible for.

In respect of cluster queues that are absent temporarily, and will shortly be reinstated, then all of these reason codes are temporary retry-able conditions (although for 2041 MQRC_OBJECT_CHANGED it can be a little complicated to decide whether the condition is retry-able). If consistent with application rules (for example maximum service times) you should probably retry for about a minute, to give time for the **REFRESH CLUSTER** activities to complete. For a modest sized cluster, completion is likely to be much quicker than that.

If any of these reason codes is returned from **MQOPEN**, then no object handle is created, but a later retry should be successful in creating one.

If any of these reason codes is returned from **MQPUT**, then the object handle is not automatically closed, and retrying should eventually succeed without a need first to close the object handle. However, if the application opened the handle using bind-on-open options, and so requires all messages to go to the same channel, then (contrary to the application's expectations) it is not guaranteed that the retried *put* would go to the same channel or queue manager as before. It is therefore wise to close the object handle and open a new one, in that case, to regain the bind-on-open semantics.

If any of these reason codes is returned from **MQPUT1**, then it is unknown whether the problem happened during the *open* or the *put* part of the operation. Whichever it is, the operation can be retried. There are no bind-on-open semantics to worry about in this case, because the **MQPUT1** operation is an *open-put-close* sequence that is performed in one continuous action.

Multi-hop scenarios

If the message flow incorporates a multi-hop, such as that shown in the following example, then a name lookup failure caused by **REFRESH CLUSTER** can occur on a queue manager that is remote from the application. In that case, the application receives a success (zero) return code, but the name lookup failure, if it occurs, prevents a **CLUSRCVR** channel program from routing the message to any proper destination queue. Instead, the **CLUSRCVR** channel program follows normal rules to write the message to a dead letter queue, based on the persistence of the message. The reason code associated with that operation is this:

- 2001 MQRC_ALIAS_BASE_Q_TYPE_ERROR

If there are persistent messages, and no dead letter queues have been defined to receive them, you will see channels ending.

Here is an example multi-hop scenario:

- **MQOPEN** on queue manager **QM1** specifies **Q2**.
- **Q2** is defined in the cluster on a remote queue manager **QM2**, as an alias.
- A message reaches **QM2**, and finds that **Q2** is an alias for **Q3**.
- **Q3** is defined in the cluster on a remote queue manager **QM3**, as a **qlocal**.
- The message reaches **QM3**, and is put to **Q3**.

When you test the multi-hop, you might see the following queue manager error log entries:

- On the sending and receiving sides, when dead letter queues are in place, and there are persistent messages:

  AMQ9544: Messages not put to destination queue
  During the processing of channel 'CHLNAME' one or more messages could not be put to the destination queue and attempts were made to put them to a dead letter queue. The location of the queue is $, where 1 is the local dead letter queue and 2 is the remote dead letter queue.

- On the receiving side, when a dead letter queue is not in place, and there are persistent messages:

  AMQ9565: No dead letter queue defined

  AMQ9599: Program could not open a queue manager object

AMQ9999: Channel program ended abnormally

- On the sending side, when a dead letter queue is not in place, and there are persistent messages:

  AMQ9506: Message receipt confirmation failed

  AMQ9780: Channel to remote machine 'a.b.c.d(1415)' is ending because of an error

  AMQ9999: Channel program ended abnormally

More details about why each of these reason codes might be displayed when running REFRESH CLUSTER

"2189 (088D) (RC2189): MQRC_CLUSTER_RESOLUTION_ERROR" on page 112
: The local queue manager asked its full repositories about the existence of a queue name. There was no response from the full repositories within a hard-coded timeout of 10 seconds. This is because the request message or the response message is on a queue for processing, and this condition will be cleared in due course. At the app, the condition is retry-able, and will succeed when those internal mechanisms have completed.

"2085 (0825) (RC2085): MQRC_UNKNOWN_OBJECT_NAME" on page 79
: The local queue manager asked (or has previously asked) its full repositories about the existence of a queue name. The full repositories have responded, saying that they did not know about the queue name. In the context of **REFRESH CLUSTER** taking place on full and partial repositories, the owner of the queue might not yet have told the full repositories about the queue. Or it might have done so, but the internal messages carrying this information are on a queue for processing, in which case this condition will be cleared in due course. At the app, the condition is retry-able, and will succeed when those internal mechanisms have completed.

"2041 (07F9) (RC2041): MQRC_OBJECT_CHANGED" on page 63
: Most likely to be seen from bind-on-open **MQPUT**. The local queue manager knows about the existence of a queue name, and about the remote queue manager where it resides. In the context of **REFRESH CLUSTER** taking place on full and partial repositories, the record of the queue manager has been deleted and is in the process of being queried from the full repositories. At the app, it is a little complicated to decide whether the condition is retry-able. In fact, if the **MQPUT** is retried, it will succeed when those internal mechanisms have completed the job of learning about the remote queue manager. However there is no guarantee that the same queue manager will be used. It is safer to follow the approach usually recommended when MQRC_OBJECT_CHANGED is received, which is to close the object handle and re-open a new one.

"2082 (0822) (RC2082): MQRC_UNKNOWN_ALIAS_BASE_Q" on page 78
: Similar in origin to the 2085 MQRC_UNKNOWN_OBJECT_NAME condition, this reason code is seen when a local alias is used, and its TARGET is a cluster queue that is inaccessible for the reasons previously described for reason code 2085.

"2001 (07D1) (RC2001): MQRC_ALIAS_BASE_Q_TYPE_ERROR" on page 45
: This reason code is not usually seen at applications. It is only likely to be seen in the queue manager error logs, in relation to attempts to send a message to a dead letter queue. A **CLUSRCVR** channel program has received a message from its partner **CLUSSDR** and is deciding where to put it. This scenario is just a variation of the same condition previously described for reason codes 2082 and 2085. In this case, the reason code is seen when an alias is being processed at a different point in the MQ product, compared to where it is processed during an application **MQPUT** or **MQOPEN**.

"2270 (08DE) (RC2270): MQRC_NO_DESTINATIONS_AVAILABLE" on page 140
: Seen when an application is using a queue that it opened with `MQOO_BIND_NOT_FIXED`, and the destination objects are unavailable for a short time until the **REFRESH CLUSTER** processing completes.

Further remarks

If there is any clustered publish/subscribe activity in this environment, then **REFRESH CLUSTER** can have additional unwanted effects. For example temporarily losing subscriptions for subscribers, that then find they missed a message. See REFRESH CLUSTER considerations for publish/subscribe clusters.

Related information

REFRESH CLUSTER considerations for publish/subscribe clusters

Clustering: Using REFRESH CLUSTER best practices

MQSC Commands reference: REFRESH CLUSTER

A cluster-sender channel is continually trying to start

Check the queue manager and listener are running, and the cluster-sender and cluster-receiver channel definitions are correct.

Symptom

```
1 : display chs(*)
AMQ8417: Display Channel Status details.
CHANNEL(DEMO.QM2)                        XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
CONNAME(computer.ibm.com(1414))
CURRENT                                  CHLTYPE(CLUSSDR)
STATUS(RETRYING)
```

Cause

1. The remote queue manager is not available.
2. An incorrect parameter is defined either for the local manual cluster-sender channel or the remote cluster-receiver channel.

Solution

Check whether the problem is the availability of the remote queue manager.

1. Are there any error messages?
2. Is the queue manager active?
3. Is the listener running?
4. Is the cluster-sender channel able to start?

If the remote queue manager is available, is there a problem with a channel definition? Check the definition type of the cluster queue manager to see if the channel is continually trying to start; for example:

```
1 : dis clusqmgr(*) deftype where(channel eq DEMO.QM2)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM2) CHANNEL(DEMO.QM2) CLUSTER(DEMO)
DEFTYPE(CLUSSDRA)
```

If the definition type is CLUSSDR the channel is using the local manual cluster-sender definition. Alter any incorrect parameters in the local manual cluster-sender definition and restart the channel.

If the definition type is either CLUSSDRA or CLUSSDRB the channel is using an auto-defined cluster-sender channel. The auto-defined cluster-sender channel is based on the definition of a remote cluster receiver channel. Alter any incorrect parameters in the remote cluster receiver definition. For example, the conname parameter might be incorrect:

```
1 : alter chl(demo.qm2) chltype(clusrcvr) conname('newhost(1414)')
AMQ8016: WebSphere MQ channel changed.
```

Changes to the remote cluster-receiver definition are propagated out to any cluster queue managers that are interested. The corresponding auto-defined channels are updated accordingly. You can check that the updates have been propagated correctly by checking the changed parameter. For example:

```
1 : dis clusqmgr(qm2) conname
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM2) CHANNEL(DEMO.QM2) CLUSTER(DEMO) CONNAME(newhost(1414))
```

If the auto-defined definition is now correct, restart the channel.

DISPLAY CLUSQMGR shows CLUSQMGR names starting SYSTEM.TEMP.

The queue manager has not received any information from the full repository queue manager that the manually defined CLUSSDR channel points to. Check that the cluster channels are defined correctly.

Symptom

```
1 : display clusqmgr(*)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)              CLUSTER(DEMO)
CHANNEL(DEMO.QM1)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(SYSTEM.TEMPUUID.computer.<yourdomain>(1414))
CLUSTER(DEMO)              CHANNEL(DEMO.QM2)
```

z/OS

```
CSQM201I +CSQ2 CSQMDRTC  DISPLAY CLUSQMGR DETAILS
CLUSQMGR(SYSTEM.TEMPQMGR.<HOSTNAME>(1716))
CLUSTER(DEMO)
CHANNEL(TO.CSQ1.DEMO)
END CLUSQMGR DETAILS
```

Cause

The queue manager has not received any information from the full repository queue manager that the manually defined CLUSSDR channel points to. The manually defined CLUSSDR channel must be in running state.

Solution

Check that the CLUSRCVR definition is also correct, especially its CONNAME and CLUSTER parameters. Alter the channel definition, if the definition is wrong.

You also need to give the correct authority to the SYSTEM.CLUSTER.TRANSMIT.QUEUE by issuing the following command:

```
setmqaut -m <QMGR Name> -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t q -g mqm +all
```

It might take some time for the remote queue managers to attempt a new restart, and start their channels with the corrected definition.

Return code= 2035 MQRC_NOT_AUTHORIZED

The RC2035 reason code is displayed for various reasons including an error on opening a queue or a channel, an error received when you attempt to use a user ID that has administrator authority, an error when using an IBM MQ JMS application, and opening a queue on a cluster. MQS_REPORT_NOAUTH and MQSAUTHERRORS can be used to further diagnose RC2035.

Specific problems

See "Specific problems generating RC2035" on page 61 for information on:

- JMSWMQ2013 invalid security authentication

- MQRC_NOT_AUTHORIZED on a queue or channel
- MQRC_NOT_AUTHORIZED (AMQ4036 on a client) as an administrator
- MQS_REPORT_NOAUTH and MQSAUTHERRORS environment variables

Opening a queue in a cluster

The solution for this error depends on whether the queue is on z/OS or not. On z/OS use your security manager. On other platforms create a local alias to the cluster queue, or authorize all users to have access to the transmission queue.

Symptom

Applications receive a return code of 2035 `MQRC_NOT_AUTHORIZED` when trying to open a queue in a cluster.

Cause

Your application receives the return code of `MQRC_NOT_AUTHORIZED` when trying to open a queue in a cluster. The authorization for that queue is correct. It is likely that the application is not authorized to put to the cluster transmission queue.

Solution

The solution depends on whether the queue is on z/OS or not. See the related information topic.

Return code= 2085 `MQRC_UNKNOWN_OBJECT_NAME` when trying to open a queue in the cluster

Symptom

Applications receive a return code of 2085 `MQRC_UNKNOWN_OBJECT_NAME` when trying to open a queue in the cluster.

Cause

The queue manager where the object exists or this queue manager might not have successfully entered the cluster.

Solution

Make sure that they can each display all the full repositories in the cluster. Also make sure that the `CLUSSDR` channels to the full repositories are trying to start.

If the queue is in the cluster, check that you have used appropriate open options. You cannot get messages from a remote cluster queue, so make sure that the open options are for output only.

```
1 : display clusqmgr(*) qmtype status
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)          CLUSTER(DEMO)
CHANNEL(DEMO.QM1)      QMTYPE(NORMAL)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM2)          CLUSTER(DEMO)
CHANNEL(DEMO.QM2)      QMTYPE(REPOS)
STATUS(RUNNING)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM3)          CLUSTER(DEMO)
CHANNEL(DEMO.QM3)      QMTYPE(REPOS)
STATUS(RUNNING)
```

Note: When using IBM MQ with WebSphere Application Server, you might also see this issue if you have a JMS application which connects to an IBM MQ queue manager belonging to an MQ cluster and your

JMS application tries to access a cluster queue which somewhere else in the cluster. For more information about this problem and how to deal with it, see the technote IBM MQ JMS application fails to open a cluster queue with reason code 2085 (MQRC_UNKNOWN_OBJECT_NAME).

Return code= 2189 `MQRC_CLUSTER_RESOLUTION_ERROR` when trying to open a queue in the cluster

Make sure that the `CLUSSDR` channels to the full repositories are not continually trying to start.

Symptom

Applications receive a return code of 2189 `MQRC_CLUSTER_RESOLUTION_ERROR` when trying to open a queue in the cluster.

Cause

The queue is being opened for the first time and the queue manager cannot contact any full repositories.

Solution

Make sure that the `CLUSSDR` channels to the full repositories are not continually trying to start.

```
1 : display clusqmgr(*) qmtype status
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)          CLUSTER(DEMO)
CHANNEL(DEMO.QM1)      QMTYPE(NORMAL)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM2)          CLUSTER(DEMO)
CHANNEL(DEMO.QM2)      QMTYPE(REPOS)
STATUS(RUNNING)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM3)          CLUSTER(DEMO)
CHANNEL(DEMO.QM3)      QMTYPE(REPOS)
STATUS(RUNNING)
```

Return code=2082 MQRC_UNKNOWN_ALIAS_BASE_Q opening a queue in the cluster

Applications get `rc=2082 MQRC_UNKNOWN_ALIAS_BASE_Q` when trying to open a queue in the cluster.

Problem

An MQOPEN or MQPUT1 call was issued specifying an alias queue as the target, but the *BaseQName* in the alias queue attributes is not recognized as a queue name.

This reason code can also occur when *BaseQName* is the name of a cluster queue that cannot be resolved successfully.

MQRC_UNKNOWN_ALIAS_BASE_Q might indicate that the application is specifying the **ObjectQmgrName** of the queue manager that it is connecting to, and the queue manager that is hosting the alias queue. This means that the queue manager looks for the alias target queue on the specified queue manager and fails because the alias target queue is not on the local queue manager.

Solution

Leave the **ObjectQmgrName** parameter blank, so that the clustering decides which queue manager to route to.

Messages are not arriving on the destination queues

Make sure that the corresponding cluster transmission queue is empty and also that the channel to the destination queue manager is running.

Symptom

Messages are not arriving on the destination queues.

Cause

The messages might be stuck at their origin queue manager.

Solution

1. Identify the transmission queue that is sending messages to the destination and the status of the channel.

```
1 : dis clusqmgr(QM1) CHANNEL(*) STATUS DEFTYPE QMTYPE XMITQ
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)     CLUSTER(DEMO)
CHANNEL(DEMO.QM1) DEFTYPE(CLUSSDRA)
QMTYPE(NORMAL)    STATUS(RUNNING)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM1)
```

2. Make sure that the cluster transmission queue is empty.

```
1 : display ql(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM1) curdepth
AMQ8409: Display Queue details.
QUEUE(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM1) CURDEPTH(0)
```

Messages put to a cluster alias queue go to SYSTEM.DEAD.LETTER.QUEUE

A cluster alias queue resolves to a local queue that does not exist.

Symptom

Messages put to an alias queue go to SYSTEM.DEAD.LETTER.QUEUE with reason MQRC_UNKNOWN_ALIAS_BASE_Q.

Cause

A message is routed to a queue manager where a clustered alias queue is defined. A local target queue is not defined on that queue manager. Because the message was put with the MQOO_BIND_ON_OPEN open option, the queue manager cannot requeue the message.

When MQOO_BIND_ON_OPEN is used, the cluster queue alias is firmly bound. The resolved name is the name of the target queue and any queue manager on which the cluster queue alias is defined. The queue manager name is placed in the transmission queue header. If the target queue does not exist on the queue manager to which the message is sent, the message is put on the dead letter queue. The destination is not recomputed, because the transmission header contains the name of the target queue manager resolved by MQOO_BIND_ON_OPEN. If the alias queue had been opened with MQOO_BIND_NOT_FIXED, then the transmission queue header would contain a blank queue manager name, and the destination would be recomputed. In which case, if the local queue is defined elsewhere in the cluster, the message would be sent there.

Solution

1. Change all alias queue definitions to specify DEFBIND(NOTFIXED).
2. Use MQOO_BIND_NOT_FIXED as an open option when the queue is opened.

3. If you specify MQOO_BIND_ON_OPEN, ensure that a cluster alias that resolves to a local queue defined on the same queue manager as the alias.

A queue manager has out of date information about queues and channels in the cluster

Symptom

DISPLAY QCLUSTER and DISPLAY CLUSQMGR show objects which are out of date.

Cause

Updates to the cluster only flow between the full repositories over manually defined CLUSSDR channels. After the cluster has formed CLUSSDR channels display as DEFTYPE (CLUSSDRB) channels because they are both manual and automatic channels. There must be enough CLUSSDR channels to form a complete network between all the full repositories.

Solution

- Check that the queue manager where the object exists and the local queue manager are still connected to the cluster.
- Check that each queue manager can display all the full repositories in the cluster.
- Check whether the CLUSSDR channels to the full repositories are continually trying to restart.
- Check that the full repositories have enough CLUSSDR channels defined to correctly connect them together.

```
1 : dis clusqmgr(QM1) CHANNEL(*) STATUS DEFTYPE QMTYPE
XMITQ
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)     CLUSTER(DEMO)
CHANNEL(DEMO.QM1) DEFTYPE(CLUSSDRA)
QMTYPE(NORMAL)    STATUS(RUNNING)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM1)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM2)     CLUSTER(DEMO)
CHANNEL(DEMO.QM2) DEFTYPE(CLUSRCVR)
QMTYPE(REPOS)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM2)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM3)     CLUSTER(DEMO)
CHANNEL(DEMO.QM3) DEFTYPE(CLUSSDRB)
QMTYPE(REPOS)     STATUS(RUNNING)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM3)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM4)     CLUSTER(DEMO)
CHANNEL(DEMO.QM4) DEFTYPE(CLUSSDRA)
QMTYPE(NORMAL)    STATUS(RUNNING)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.DEMO.QM4)
```

No changes in the cluster are being reflected in the local queue manager

The repository manager process is not processing repository commands, possibly because of a problem with receiving or processing messages in the command queue.

Symptom

No changes in the cluster are being reflected in the local queue manager.

Cause

The repository manager process is not processing repository commands.

Solution

1. Check that the SYSTEM.CLUSTER.COMMAND.QUEUE is empty.

```
1 : display ql(SYSTEM.CLUSTER.COMMAND.QUEUE) curdepth
AMQ8409: Display Queue details.
QUEUE(SYSTEM.CLUSTER.COMMAND.QUEUE) CURDEPTH(0)
```

2. z/OS Check that the channel initiator is running on z/OS.
3. Check that there are no error messages in the error logs indicating the queue manager has a temporary resource shortage.

DISPLAY CLUSQMGR displays a queue manager twice

Use the RESET CLUSTER command to remove all traces of an old instance of a queue manager.

```
1 : display clusqmgr(QM1) qmid
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)                           CLUSTER(DEMO)
CHANNEL(DEMO.QM1)                       QMID(QM1_2002-03-04_11.07.01)
AMQ8441: Display Cluster Queue Manager details.
CLUSQMGR(QM1)                           CLUSTER(DEMO)
CHANNEL(DEMO.QM1)                       QMID(QM1_2002-03-04_11.04.19)
```

The cluster functions correctly with the older version of the queue manager being ignored. After about 90 days, the cluster's knowledge of the older version of the queue manager expires, and is deleted automatically. However you might prefer to delete this information manually.

Cause

1. The queue manager might have been deleted and then re-created and redefined.
2. It might have been cold-started on z/OS, without first following the procedure to remove a queue manager from a cluster.

Solution

To remove all trace of the queue manager immediately use the RESET CLUSTER command from a full repository queue manager. The command removes the older unwanted queue manager and its queues from the cluster.

```
2 : reset cluster(DEMO) qmid('QM1_2002-03-04_11.04.19') action(FORCEREMOVE) queues(yes)
AMQ8559: RESET CLUSTER accepted.
```

Using the RESET CLUSTER command stops auto-defined cluster sender channels for the affected queue manager. You must manually restart any cluster sender channels that are stopped, after completing the RESET CLUSTER command.

A queue manager does not rejoin the cluster

After issuing a RESET or REFRESH cluster command the channel from the queue manager to the cluster might be stopped. Check the cluster channel status and restart the channel.

Symptom

A queue manager does not rejoin a cluster after issuing the RESET CLUSTER and REFRESH CLUSTER commands.

Cause

A side effect of the RESET and REFRESH commands might be that a channel is stopped. A channel is stopped in order that the correct version of the channel runs when RESET or REFRESH command is completed.

Solution

Check that the channels between the problem queue manager and the full repositories are running and use the `START CHANNEL` command if necessary.

Related information

Clustering: Using REFRESH CLUSTER best practices

Workload balancing set on a cluster-sender channel is not working

Any workload balancing you specify on a cluster-sender channel is likely to be ignored. Instead, specify the cluster workload channel attributes on the cluster-receiver channel at the target queue manager.

Symptom

You have specified one or more cluster workload channel attributes on a cluster-sender channel. The resulting workload balancing is not as you were expecting.

Cause

Any workload balancing you specify on a cluster-sender channel is likely to be ignored. For an explanation of this, see Cluster channels. Note that you still get some form of workload balancing, based either on cluster defaults or on properties set on the matching cluster-receiver channel at the target queue manager.

Solution

Specify the cluster workload channel attributes on the cluster-receiver channel at the target queue manager.

Related information

CLWLPRTY channel attribute

CLWLRANK channel attribute

CLWLWGHT channel attribute

NETPRTY channel attribute

Out of date information in a restored cluster

After restoring a queue manager, its cluster information is out of date. Refresh the cluster information with the **`REFRESH CLUSTER`** command.

Problem

After an image backup of `QM1`, a partial repository in cluster `DEMO` has been restored and the cluster information it contains is out of date.

Solution

On `QM1`, issue the command `REFRESH CLUSTER(DEMO)`.

Note: For large clusters, use of the **`REFRESH CLUSTER`** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

When you run REFRESH CLUSTER(DEMO) on QM1, you remove all the information QM1 has about the cluster DEMO, except for QM1's knowledge of itself and its own queues, and of how to access the full repositories in the cluster. QM1 then contacts the full repositories, and tells them about itself and its queues. QM1 is a partial repository, so the full repositories don't immediately tell QM1 about all the other partial repositories in the cluster. Instead, QM1 slowly builds up its knowledge of the other partial repositories through information it receives as and when each of the other queues and queue managers is next active in the cluster.

Cluster queue manager force removed from a full repository by mistake

Restore the queue manager to the full repository by issuing the command **REFRESH CLUSTER** on the queue manager that was removed from the repository.

Problem

The command, RESET CLUSTER(DEMO) QMNAME(QM1) ACTION(FORCEREMOVE) was issued on a full repository in cluster DEMO by mistake.

Solution

On QM1, issue the command REFRESH CLUSTER(DEMO).

Note: For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

Possible repository messages deleted

Messages destined for a queue manager were removed from the SYSTEM.CLUSTER.TRANSMIT.QUEUE in other queue managers. Restore the information by issuing the REFRESH CLUSTER command on the affected queue manager.

Problem

Messages destined for QM1 were removed from the SYSTEM.CLUSTER.TRANSMIT.QUEUE in other queue managers and they might have been repository messages.

Solution

On QM1, issue the command REFRESH CLUSTER(DEMO).

Note: For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

QM1 removes all information it has about the cluster DEMO, except that relating to the cluster queue managers which are the full repositories in the cluster. Assuming that this information is still correct, QM1 contacts the full repositories. QM1 informs the full repositories about itself and its queues. It recovers the information for queues and queue managers that exist elsewhere in the cluster as they are opened.

Two full repositories moved at the same time

If you move both full repositories to new network addresses at the same time, the cluster is not updated with the new addresses automatically. Follow the procedure to transfer the new network addresses. Move the repositories one at a time to avoid the problem.

Problem

Cluster `DEMO` contains two full repositories, `QM1` and `QM2`. They were both moved to a new location on the network at the same time.

Solution

1. Alter the `CONNAME` in the `CLUSRCVR` and `CLUSSDR` channels to specify the new network addresses.
2. Alter one of the queue managers (`QM1` or `QM2`) so it is no longer a full repository for any cluster.
3. On the altered queue manager, issue the command `REFRESH CLUSTER(*) REPOS(YES)`.

   Note: For large clusters, use of the **`REFRESH CLUSTER`** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.
4. Alter the queue manager so it is acting as a full repository.

Recommendation

You could avoid the problem as follows:

1. Move one of the queue managers, for example `QM2`, to its new network address.
2. Alter the network address in the `QM2` `CLUSRCVR` channel.
3. Start the `QM2` `CLUSRCVR` channel.
4. Wait for the other full repository queue manager, `QM1`, to learn the new address of `QM2`.
5. Move the other full repository queue manager, `QM1`, to its new network address.
6. Alter the network address in the `QM1` `CLUSRCVR` channel.
7. Start the `QM1` `CLUSRCVR` channel.
8. Alter the manually defined `CLUSSDR` channels for the sake of clarity, although at this stage they are not needed for the correct operation of the cluster.

The procedure forces `QM2` to reuse the information from the correct `CLUSSDR` channel to re-establish contact with `QM1` and then rebuild its knowledge of the cluster. Additionally, having once again contacted `QM1`, it is given its own correct network address based on the `CONNAME` in `QM2` `CLUSRCVR` definition.

Unknown state of a cluster

Restore the cluster information in all the full repositories to a known state by rebuilding the full repositories from all the partial repositories in the cluster.

Problem

Under normal conditions the full repositories exchange information about the queues and queue managers in the cluster. If one full repository is refreshed, the cluster information is recovered from the other.

The problem is how to completely reset all the systems in the cluster to restore a known state to the cluster.

Solution

To stop cluster information being updated from the unknown state of the full repositories, all the CLUSRCVR channels to full repositories are stopped. The CLUSSDR channels change to inactive.

When you refresh the full repository systems, none of them are able to communicate, so they start from the same cleared state.

When you refresh the partial repository systems, they rejoin the cluster and rebuild it to the complete set of queue managers and queues. The cluster information in the rebuilt full is restored to a known state.

Note: For large clusters, use of the **REFRESH CLUSTER** command can be disruptive to the cluster while it is in progress, and again at 27 day intervals thereafter when the cluster objects automatically send status updates to all interested queue managers. See Refreshing in a large cluster can affect performance and availability of the cluster.

1. On all the full repository queue managers, follow these steps:
   a. Alter queue managers that are full repositories so they are no longer full repositories.
   b. Resolve any in doubt CLUSSDR channels.
   c. Wait for the CLUSSDR channels to become inactive.
   d. Stop the CLUSRCVR channels.
   e. When all the CLUSRCVR channels on all the full repository systems are stopped, issue the command `REFRESH CLUSTER(DEMO) REPOS(YES)`.
   f. Alter the queue managers so they are full repositories.
   g. Start the CLUSRCVR channels to re-enable them for communication.
2. On all the partial repository queue managers, follow these steps:
   a. Resolve any in doubt CLUSSDR channels.
   b. Make sure all CLUSSDR channels on the queue manager are stopped or inactive.
   c. Issue the command `REFRESH CLUSTER(DEMO) REPOS(YES)`.

What happens when a cluster queue manager fails

When a cluster queue manager fails, some undelivered messages are sent to other queue managers in the cluster. Messages that are in-flight wait until the queue manager is restarted. Use a high-availability mechanism to restart a queue manager automatically.

Problem

If a message-batch is sent to a particular queue manager and that queue manager becomes unavailable, what happens at the sending queue manager?

Explanation

Except for non-persistent messages on an NPMSPEED(FAST) channel, the undelivered batch of messages is backed out to the cluster transmission queue on the sending queue manager. On an NPMSPEED(FAST) channel, non-persistent messages are not batched, and one might be lost.

- Indoubt messages, and messages that are bound to the unavailable queue manager, wait until the queue manager becomes available again.
- Other messages are delivered to alternative queue managers selected by the workload management routine.

Solution

The unavailable cluster queue manager can be restarted automatically, either by being configured as a multi-instance queue manager, or by a platform-specific high availability mechanism.

What happens when a repository fails

How you know a repository has failed and what to do to fix it?

Problem

1. Cluster information is sent to repositories (whether full or partial) on a local queue called SYSTEM.CLUSTER.COMMAND.QUEUE. If this queue fills up, perhaps because the queue manager has stopped working, the cluster-information messages are routed to the dead-letter queue.
2. The repository runs out of storage.

Solution

1. Monitor the messages on your queue manager log z/OS or z/OS system console to detect if SYSTEM.CLUSTER.COMMAND.QUEUE is filling up. If it is, you need to run an application to retrieve the messages from the dead-letter queue and reroute them to the correct destination.
2. If errors occur on a repository queue manager, messages tell you what error has occurred and how long the queue manager waits before trying to restart.
   - z/OS On IBM MQ for z/OS, the SYSTEM.CLUSTER.COMMAND.QUEUE is disabled for MQGET.
   - When you have identified and resolved the error, enable the SYSTEM.CLUSTER.COMMAND.QUEUE so that the queue manager can restart successfully.
3. In the unlikely event of the repository running out of storage, storage allocation errors are sent to the queue-manager log z/OS or z/OS system console. To fix the storage problem, stop and then restart the queue manager. When the queue manager is restarted, more storage is automatically allocated to hold all the repository information.

What happens if a cluster queue is disabled for MQPUT

All instances of a cluster queue that is being used for workload balancing might be disabled for MQPUT. Applications putting a message to the queue either receive a MQRC_CLUSTER_PUT_INHIBITED or a MQRC_PUT_INHIBITED return code. You might want to modify this behavior.

Problem

When a cluster queue is disabled for MQPUT, its status is reflected in the repository of each queue manager that is interested in that queue. The workload management algorithm tries to send messages to destinations that are enabled for MQPUT. If there are no destinations enabled for MQPUT and no local instance of a queue, an MQOPEN call that specified MQOO_BIND_ON_OPEN returns a return code of MQRC_CLUSTER_PUT_INHIBITED to the application. If MQOO_BIND_NOT_FIXED is specified, or there is a local instance of the queue, an MQOPEN call succeeds but subsequent MQPUT calls fail with return code MQRC_PUT_INHIBITED.

Solution

You can write a user exit program to modify the workload management routines so that messages can be routed to a destination that is disabled for MQPUT.

A message can arrive at a destination that is disabled for MQPUT. The message might have been in flight at the time the queue became disabled, or a workload exit might have chosen the destination explicitly. The workload management routine at the destination queue manager has a number of ways to deal with the message:

- Choose another appropriate destination, if there is one.
- Place the message on the dead-letter queue.
- Return the message to the originator, if there is no dead-letter queue

Potential issues when switching transmission queues

A list of some issues that might be encountered when switching transmission queue, their causes, and most likely solutions.

Insufficient access to transmission queues on z/OS

Symptom

A cluster-sender channel on z/OS might report it is not authorized to open its transmission queue.

Cause

The channel is switching, or has switched, transmission queue and the channel initiator has not been granted authority to access the new queue.

Solution

Grant the channel initiator the same access to the channel's transmission queue that is documented for the transmission queue SYSTEM.CLUSTER.TRANSMIT.QUEUE. When using DEFCLXQ a generic profile for SYSTEM.CLUSTER.TRANSMIT.** avoids this problem occurring whenever a new queue manager joins the cluster.

Moving of messages fails

Symptom

Messages stop being sent by a channel and they remain queued on the channel's old transmission queue.

Cause

The queue manager has stopped moving messages from the old transmission queue to the new transmission queue because an unrecoverable error occurred. For example, the new transmission queue might have become full or its backing storage exhausted.

Solution

Review the error messages written to the queue manager's error log (job log on z/OS) to determine the problem and resolve its root cause. Once resolved, restart the channel to resume the switching process, or stop the channel then use **`runswchl`** instead (CSQUTIL on z/OS).

A switch does not complete

Symptom

The queue manager repeatedly issues messages that indicate it is moving messages. The switch never completes because there are always messages remaining on the old transmission queue.

Cause 1

Messages for the channel are being put to the old transmission queue faster than the queue manager can move them to the new transmission queue. This is likely to be a transient issue during peak workload because if were commonplace then it is unlikely the channel would be able to transmit the messages over the network fast enough.

Cause 2

There are uncommitted messages for the channel on the old transmission queue.

Solution

Resolve the units of work for any uncommitted messages, and/or reduce or suspend the application workload, to allow the moving message phase to complete.

Accidental deletion of a transmission queue

Symptom 1

Channels unexpectedly switch due to the removal of a matching CLCHNAME value.

Symptom 2

A put to a cluster queue fails with MQRC_UNKNOWN_XMIT_Q.

Symptom 3

A channel abnormally ends because its transmission queue does not exist.

Symptom 4

The queue manager is unable to move messages to complete a switch operation because it cannot open either the old or the new transmission queue.

Cause

The transmission queue currently used by a channel, or its previous transmission queue if a switch has not completed, has been deleted.

Solution

Redefine the transmission queue. If it is the old transmission queue that has been deleted then an administrator may alternatively complete the switch operation using **`runswchl`** with the **`-n`** parameter (or CSQUTIL with MOVEMSGS(NO) on z/OS).

Use the -n parameter with caution because, if it is used inappropriately, messages for the channel can complete and finish processing but not be updated on the old transmission queue. In this scenario it is safe because as the queue does not exist there cannot be any messages to complete and finish processing.

Queue managers troubleshooting

Use the advice given here to help you to resolve common problems that can arise when you use queue managers.

Queue manager unavailable error

- **Scenario:** You receive a *queue manager unavailable* error.
- **Explanation:** Configuration file errors typically prevent queue managers from being found, and result in *queue manager unavailable* errors. On Windows, problems in the qm.ini file can cause *queue manager unavailable* errors when a queue manager is started.

- **Solution:** Ensure that the configuration files exist, and that the IBM MQ configuration file references the correct queue manager and log directories. On Windows, check for problems in the qm.ini file.

Undelivered messages troubleshooting

Use the advice given here to help you to resolve problems when messages do are not delivered successfully.

- **Scenario:** Messages do not arrive on a queue when you are expecting them.
- **Explanation:** Messages that cannot be delivered for some reason are placed on the dead-letter queue.
- **Solution:** You can check whether the queue contains any messages by issuing an MQSC DISPLAY QUEUE command.

  If the queue contains messages, you can use the provided browse sample application (amqsbcg) to browse messages on the queue using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.

  You must decide how to dispose of any messages found on the dead-letter queue, depending on the reasons for the messages being put on the queue. Problems might occur if you do not associate a dead-letter queue with each queue manager.

For more information about dead-letter queues and handling undelivered messages, see Working with dead-letter queues.

TLS/SSL troubleshooting information

Use the information listed here to help you solve problems with your TLS/SSL system.

Overview

For the error caused by *Using non-FIPS cipher with FIPS enabled on client*, you receive the following error message:

JMSCMQ001

IBM MQ call failed with completion code *2 ('MQCC_FAILED')* reason *2397 ('MQRC_JSSE_ERROR')*

For every other problem documented within this topic you receive either the previous error message, or the following error message, or both:

JMSWMQ0018

Failed to connect to queue manager *'queue-manager-name'* with connection mode *'connection-mode'* and host name *'host-name'*

For each problem documented within this topic, the following information is provided:

- Output from the sample `SystemOut.log` or `Console`, detailing the cause of the exception..
- Queue manager error log information.
- Solution to the problem.

Note:

- You should always list out the stacks and the cause of the first exception.
- Whether or not the error information is written to the `stdout` log file depends on how the application is written, and on which framework you are using.
- The sample code includes stacks and line numbers. This information is useful guidance, but the stacks and line numbers are likely to change from one fix pack to another. You should use the stacks and line numbers as a guide to locating the correct section, and not use the information specifically for diagnostic purposes.

Cipher suite not set on client

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9641: Remote CipherSpec error for channel
'SYSTEM.DEF.SVRCONN' to host ''. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9639: Remote channel *'SYSTEM.DEF.SVRCONN'* did not specify a CipherSpec.

Solution

Set a CipherSuite on the client so that both ends of the channel have a matching CipherSuite or CipherSpec pair.

Cipher suite not set on server

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9641: Remote CipherSpec error
for channel 'SYSTEM.DEF.SVRCONN' to host ''. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9639: Remote channel *'SYSTEM.DEF.SVRCONN'* did not specify a CipherSpec.

Solution

Change channel *'SYSTEM.DEF.SVRCONN'* to specify a valid CipherSpec.

Cipher Mismatch

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9641: Remote CipherSpec error
for channel 'SYSTEM.DEF.SVRCONN' to host ''. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9631: The CipherSpec negotiated during the SSL handshake does not match the required CipherSpec for channel *'SYSTEM.DEF.SVRCONN'*.

Solution

Change either the SSLCIPH definition of the server-connection channel or the Cipher suite of the client so that the two ends have a matching CipherSuite or CipherSpec pair.

Missing client personal certificate

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2059;AMQ9503: Channel negotiation failed. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9637: Channel is lacking a certificate.

Solution

Ensure that the key database of the queue manager contains a signed personal certificate from the truststore of the client.

Missing server personal certificate

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9771: SSL handshake failed.
[1=javax.net.ssl.SSLHandshakeException[Remote host closed connection during handshake],
3=localhost/127.0.0.1:1418 (localhost),4=SSLSocket.startHandshake,5=default]
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1173)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:835)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
... 12 more
```

Caused by:

```
javax.net.ssl.SSLHandshakeException: Remote host closed connection during handshake
at com.ibm.jsse2.qc.a(qc.java:158)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
... 17 more
```

Caused by:

```
java.io.EOFException: SSL peer shut down incorrectly
at com.ibm.jsse2.a.a(a.java:19)
at com.ibm.jsse2.qc.a(qc.java:207)
```

Queue manager error logs

AMQ9637: Channel is lacking a certificate.

Solution

Ensure that the key database of the queue manager contains a signed personal certificate from the truststore of the client.

Missing server signer on client

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9771: SSL handshake failed.
[1=javax.net.ssl.SSLHandshakeException[com.ibm.jsse2.util.j:
PKIX path validation failed: java.security.cert.CertPathValidatorException:
```

```
The certificate issued by CN=JohnDoe, O=COMPANY, L=YOURSITE, C=XX is not trusted; internal cause is:
java.security.cert.CertPathValidatorException: Signature does not match.],3=localhost/127.0.0.1:1418
(localhost),4=SSLSocket.startHandshake,5=default]
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1173)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:835)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
...
```

Caused by:

```
javax.net.ssl.SSLHandshakeException: com.ibm.jsse2.util.j: PKIX path validation failed:
java.security.cert.CertPathValidatorException:
The certificate issued by CN=JohnDoe, O=COMPANY, L=YOURSITE, C=XX is not trusted;
internal cause is: java.security.cert.CertPathValidatorException: Signature does not match.
...
```

Caused by:

```
com.ibm.jsse2.util.j: PKIX path validation failed: java.security.cert.CertPathValidatorException:
The certificate issued by CN=JohnDoe, O=COMPANY, L=YOURSITE, C=XX is not trusted;
internal cause is:    java.security.cert.CertPathValidatorException: Signature does not match.
at com.ibm.jsse2.util.h.a(h.java:99)
at com.ibm.jsse2.util.h.b(h.java:27)
at com.ibm.jsse2.util.g.a(g.java:14)
at com.ibm.jsse2.yc.a(yc.java:68)
at com.ibm.jsse2.yc.a(yc.java:17)
at com.ibm.jsse2.yc.checkServerTrusted(yc.java:154)
at com.ibm.jsse2.bb.a(bb.java:246)
... 28 more
```

Caused by:

```
java.security.cert.CertPathValidatorException:
The certificate issued by CN=JohnDoe, O=COMPANY, L=YOURSITE, C=XX is not trusted;
internal cause is:    java.security.cert.CertPathValidatorException: Signature does not match.
at com.ibm.security.cert.BasicChecker.(BasicChecker.java:111)
at com.ibm.security.cert.PKIXCertPathValidatorImpl.engineValidate(PKIXCertPathValidatorImpl.java:174)
at java.security.cert.CertPathValidator.validate(CertPathValidator.java:265)
at com.ibm.jsse2.util.h.a(h.java:13)
... 34 more
```

Caused by:

```
java.security.cert.CertPathValidatorException: Signature does not match.
at com.ibm.security.cert.CertPathUtil.findIssuer(CertPathUtil.java:297)
at com.ibm.security.cert.BasicChecker.(BasicChecker.java:108)
```

Queue manager error logs

AMQ9665: SSL connection closed by remote end of channel *'????'*.

Solution

Add the certificate used to sign the personal certificate of the queue manager to the truststore of the client.

Missing client signer on server

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9771: SSL handshake failed.
[1=java.net.SocketException[Software caused connection abort: socket write error],
3=localhost/127.0.0.1:1418 (localhost),4=SSLSocket.startHandshake,5=default]
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1173)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:835)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
... 12 more
```

Caused by:

```
java.net.SocketException: Software caused connection abort: socket write error
at java.net.SocketOutputStream.socketWrite(SocketOutputStream.java:120)
at java.net.SocketOutputStream.write(SocketOutputStream.java:164)
at com.ibm.jsse2.c.a(c.java:57)
at com.ibm.jsse2.c.a(c.java:34)
at com.ibm.jsse2.qc.b(qc.java:527)
at com.ibm.jsse2.qc.a(qc.java:635)
at com.ibm.jsse2.qc.a(qc.java:743)
at com.ibm.jsse2.ab.a(ab.java:550)
at com.ibm.jsse2.bb.b(bb.java:194)
at com.ibm.jsse2.bb.a(bb.java:162)
at com.ibm.jsse2.bb.a(bb.java:7)
at com.ibm.jsse2.ab.r(ab.java:529)
at com.ibm.jsse2.ab.a(ab.java:332)
at com.ibm.jsse2.qc.a(qc.java:435)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
```

Queue manager error logs

AMQ9633: Bad SSL certificate for channel *'????'*.

Solution

Add the certificate used to sign the personal certificate of the client to the key database of the queue manager.

SSLPEER set on server does not match certificate

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9643: Remote SSL peer name error for channel
'SYSTEM.DEF.SVRCONN' on host ''. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9636: SSL distinguished name does not match peer name, channel *'SYSTEM.DEF.SVRCONN'*.

Solution

Ensure the value of SSLPEER set on the server-connection channel matches the distinguished name of the certificate.

SSLPEER set on client does not match certificate

Output

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2398;AMQ9636: SSL distinguished name does not match peer name,
channel '?'. [CN=JohnDoe, O=COMPANY, L=YOURSITE, C=XX]
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1215)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:835)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs

AMQ9208: Error on receive from host *host-name (address)*.

Solution

Ensure the value of SSLPEER set in the client matches the distinguished name of the certificate.

Using a non-FIPS cipher with FIPS enabled on client

Output

```
Check the queue manager is started and if running in client mode, check there is a listener running.
Please see the linked exception for more information.
at com.ibm.msg.client.wmq.common.internal.Reason.reasonToException(Reason.java:578)
at com.ibm.msg.client.wmq.common.internal.Reason.createException(Reason.java:214)
at com.ibm.msg.client.wmq.internal.WMQConnection.getConnectOptions(WMQConnection.java:1423)
at com.ibm.msg.client.wmq.internal.WMQConnection.(WMQConnection.java:339)
at com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createV7ProviderConnection
(WMQConnectionFactory.java:6865)
at com.ibm.msg.client.wmq.factories.WMQConnectionFactory.createProviderConnection
(WMQConnectionFactory.java:6221)
at com.ibm.msg.client.jms.admin.JmsConnectionFactoryImpl._createConnection
(JmsConnectionFactoryImpl.java:285)
at com.ibm.msg.client.jms.admin.JmsConnectionFactoryImpl.createConnection
(JmsConnectionFactoryImpl.java:233)
at com.ibm.mq.jms.MQConnectionFactory.createCommonConnection(MQConnectionFactory.java:6016)
at com.ibm.mq.jms.MQConnectionFactory.createConnection(MQConnectionFactory.java:6041)
at tests.SimpleSSLConn.runTest(SimpleSSLConn.java:46)
at tests.SimpleSSLConn.main(SimpleSSLConn.java:26)
```

Caused by:

```
com.ibm.mq.MQException: JMSCMQ0001: IBM MQ call failed with compcode '2' ('MQCC_FAILED')
reason '2400' ('MQRC_UNSUPPORTED_CIPHER_SUITE').
at com.ibm.msg.client.wmq.common.internal.Reason.createException(Reason.java:202)
```

Queue manager error logs
Not applicable.

Solution
Use a FIPS-enabled cipher, or disable FIPS on the client.

Using a non-FIPS cipher with FIPS enabled on the queue manager

Output
Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2397;AMQ9771: SSL handshake failed.
[1=javax.net.ssl.SSLHandshakeException[Received fatal alert: handshake_failure],
3=localhost/127.0.0.1:1418 (localhost),4=SSLSocket.startHandshake,5=default]
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1173)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:835)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
... 12 more
```

Caused by:

```
javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure
at com.ibm.jsse2.j.a(j.java:13)
at com.ibm.jsse2.j.a(j.java:18)
at com.ibm.jsse2.qc.b(qc.java:601)
at com.ibm.jsse2.qc.a(qc.java:100)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
```

Queue manager error logs
AMQ9616: The CipherSpec proposed is not enabled on the server.

Solution
Use a FIPS-enabled cipher, or disable FIPS on the queue manager.

Can not find client keystore using IBM JRE

Output
Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2059;AMQ9204: Connection to host 'localhost(1418)' rejected.
[1=com.ibm.mq.jmqi.JmqiException[CC=2;RC=2059;AMQ9503: Channel negotiation failed.
[3=SYSTEM.DEF.SVRCONN]],3=localhost(1418),5=RemoteConnection.analyseErrorSegment]
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:2450)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1396)
at com.ibm.mq.ese.jmqi.InterceptedJmqiImpl.jmqiConnect(InterceptedJmqiImpl.java:376)
at com.ibm.mq.ese.jmqi.ESEJMQI.jmqiConnect(ESEJMQI.java:561)
at com.ibm.msg.client.wmq.internal.WMQConnection.(WMQConnection.java:342)
... 8 more
```

Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2059;AMQ9503: Channel negotiation failed. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs
AMQ9637: Channel is lacking a certificate.

Solution
Ensure the JVM property `javax.net.ssl.keyStore` specifies the location of a valid keystore.

Can not find client keystore using Oracle JRE

Output
Caused by:

```
java.security.PrivilegedActionException: java.io.FileNotFoundException:
C:\<filepath>\wrongkey.jks (The system cannot find the file specified)
at java.security.AccessController.doPrivileged(Native Method)
at sun.security.ssl.SSLContextImpl$DefaultSSLContext.getDefaultKeyManager(Unknown Source)
at sun.security.ssl.SSLContextImpl$DefaultSSLContext.(Unknown Source)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at sun.reflect.NativeConstructorAccessorImpl.newInstance(Unknown Source)
at sun.reflect.DelegatingConstructorAccessorImpl.newInstance(Unknown Source)
at java.lang.reflect.Constructor.newInstance(Unknown Source)
at java.lang.Class.newInstance0(Unknown Source)
at java.lang.Class.newInstance(Unknown Source)
... 28 more
```

Caused by:

```
java.io.FileNotFoundException: C:\<filepath>\wrongkey.jks (The system cannot find the file specified)
at java.io.FileInputStream.open(Native Method)
at java.io.FileInputStream.(Unknown Source)
at java.io.FileInputStream.(Unknown Source)
at sun.security.ssl.SSLContextImpl$DefaultSSLContext$2.run(Unknown Source)
at sun.security.ssl.SSLContextImpl$DefaultSSLContext$2.run(Unknown Source)
```

Queue manager error logs
AMQ9637: Channel is lacking a certificate.

Solution
Ensure the JVM property `javax.net.ssl.keyStore` specifies the location of a valid keystore.

Keystore password error - IBM JRE

Output
Caused by:

```
com.ibm.mq.jmqi.JmqiException: CC=2;RC=2059;AMQ9503: Channel negotiation failed. [3=SYSTEM.DEF.SVRCONN]
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.analyseErrorSegment(RemoteConnection.java:4176)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.receiveTSH(RemoteConnection.java:2969)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.initSess(RemoteConnection.java:1180)
at com.ibm.mq.jmqi.remote.impl.RemoteConnection.connect(RemoteConnection.java:838)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSessionFromNewConnection
(RemoteConnectionSpecification.java:409)
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionSpecification.getSession
(RemoteConnectionSpecification.java:305)
```

```
at com.ibm.mq.jmqi.remote.impl.RemoteConnectionPool.getSession(RemoteConnectionPool.java:146)
at com.ibm.mq.jmqi.remote.api.RemoteFAP.jmqiConnect(RemoteFAP.java:1868)
```

Queue manager error logs
AMQ9637: Channel is lacking a certificate.

Solution
Ensure that the value of the JVM property `javax.net.ssl.keyStorePassword` specifies the password for the keystore specified by `javax.net.ssl.keyStore`.

Truststore password error - IBM JRE

Output
Caused by:

```
javax.net.ssl.SSLHandshakeException: java.security.cert.CertificateException:
No X509TrustManager implementation available
at com.ibm.jsse2.j.a(j.java:13)
at com.ibm.jsse2.qc.a(qc.java:204)
at com.ibm.jsse2.ab.a(ab.java:342)
at com.ibm.jsse2.ab.a(ab.java:222)
at com.ibm.jsse2.bb.a(bb.java:157)
at com.ibm.jsse2.bb.a(bb.java:492)
at com.ibm.jsse2.ab.r(ab.java:529)
at com.ibm.jsse2.ab.a(ab.java:332)
at com.ibm.jsse2.qc.a(qc.java:435)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
... 17 more
```

Caused by:

```
java.security.cert.CertificateException: No X509TrustManager implementation available
at com.ibm.jsse2.xc.checkServerTrusted(xc.java:2)
at com.ibm.jsse2.bb.a(bb.java:246)
```

Queue manager error logs
AMQ9665: SSL connection closed by remote end of channel `'????'`.

Solution
Ensure that the value of the JVM property `javax.net.ssl.trustStorePassword` specifies the password for the keystore specified by `javax.net.ssl.trustStore`.

Can not find or open queue manager key database

Output
Caused by:

```
javax.net.ssl.SSLHandshakeException: Remote host closed connection during handshake
at com.ibm.jsse2.qc.a(qc.java:158)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
... 17 more
```

Caused by:

```
java.io.EOFException: SSL peer shut down incorrectly
at com.ibm.jsse2.a.a(a.java:19)
at com.ibm.jsse2.qc.a(qc.java:207)
```

Queue manager error logs
AMQ9657: The key repository could not be opened (channel `'????'`).

Solution

Ensure that the key repository you specify exists and that its permissions are such that the IBM MQ process involved can read from it.

Can not find or use queue manager key database password stash file

Output

Caused by:

```
javax.net.ssl.SSLHandshakeException: Remote host closed connection during handshake
at com.ibm.jsse2.qc.a(qc.java:158)
at com.ibm.jsse2.qc.h(qc.java:185)
at com.ibm.jsse2.qc.a(qc.java:566)
at com.ibm.jsse2.qc.startHandshake(qc.java:120)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1142)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection$6.run(RemoteTCPConnection.java:1134)
at java.security.AccessController.doPrivileged(AccessController.java:229)
at com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.protocolConnect(RemoteTCPConnection.java:1134)
... 17 more
```

Caused by:

```
ava.io.EOFException: SSL peer shut down incorrectly
at com.ibm.jsse2.a.a(a.java:19)
at com.ibm.jsse2.qc.a(qc.java:207)
```

Queue manager error logs

AMQ9660: SSL key repository: password stash file absent or unusable.

Solution

Ensure that a password stash file has been associated with the key database file in the same directory, and that the user ID, under which IBM MQ is running, has read access to both files.

IBM MQ Telemetry troubleshooting

Look for a troubleshooting task to help you solve a problem with running IBM MQ Telemetry applications.
Related information

IBM MQ Telemetry

Location of telemetry logs, error logs, and configuration files

Find the logs, error logs, and configuration files used by IBM MQ Telemetry.

Note: The examples are coded for Windows systems. Change the syntax to run the examples on AIX or Linux systems.

Server-side logs

The telemetry (MQXR) service writes FDC files to the IBM MQ error directory:

```
WMQ data directory\errors\AMQ nnn.n.FDC
```

The format of the FDC files is `MQXRn.FDC`.

It also writes a log for the telemetry (MQXR) service. The log path is:

```
WMQ data directory\Qmgrs\qMgrName\errors\mqxr.log
```

The format of the log file is `mqxr_n.log`.

The IBM MQ telemetry sample configuration created by MQ Explorer starts the telemetry (MQXR) service using the command **`runMQXRService`**, which is in *`WMQ Telemetry installation directory`*`\bin`. This command writes to:

```
WMQ data directory\Qmgrs\qMgrName\mqxr.stdout
WMQ data directory\Qmgrs\qMgrName\mqxr.stderr
```

Server-side configuration files

Telemetry channels and telemetry (MQXR) service

Restriction: The format, location, content, and interpretation of the telemetry channel configuration file might change in future releases. You must use IBM MQ Explorer, or MQSC commands, to configure telemetry channels.

MQ Explorer saves telemetry configurations in the `mqxr_win.properties` file on Windows systems, and the `mqxr_unix.properties` file on AIX or Linux systems. The properties files are saved in the telemetry configuration directory:

```
WMQ data directory\Qmgrs\qMgrName\mqxr
```

Figure 23. Telemetry configuration directory on Windows

```
 /var/mqm/qmgrs/qMgrName/mqxr
```

Figure 24. Telemetry configuration directory on AIX or Linux

JVM

Set Java properties that are passed as arguments to the telemetry (MQXR) service in the file, `java.properties`. The properties in the file are passed directly to the JVM running the telemetry (MQXR) service. They are passed as additional JVM properties on the Java command line. Properties set on the command line take precedence over properties added to the command line from the `java.properties` file.

Find the `java.properties` file in the same folder as the telemetry configurations. See Figure 23 on page 501 and Figure 24 on page 501.

Modify `java.properties` by specifying each property as a separate line. Format each property exactly as you would to pass the property to the JVM as an argument. For example:

```
-Xmx1024m
-Xms1024m
```

JAAS

The JAAS configuration file is described in Telemetry channel JAAS configuration, which includes the sample JAAS configuration file, JAAS.config, shipped with IBM MQ Telemetry.

If you configure JAAS, you are almost certainly going to write a class to authenticate users to replace the standard JAAS authentication procedures.

To include your `Login` class in the class path used by the telemetry (MQXR) service class path, provide an IBM MQ `service.env` configuration file.

Set the class path for your JAAS `LoginModule` in `service.env`. You cannot use the variable, `%classpath%` in `service.env`. The class path in `service.env` is added to the class path already set in the telemetry (MQXR) service definition.

Display the class paths that are being used by the telemetry (MQXR) service by adding `echo set classpath` to `runMQXRService.bat`. The output is sent to `mqxr.stdout`.

The default location for the `service.env` file is:

```
  WMQ data directory\service.env
```

Override these settings with a `service.env` file for each queue manager in:

```
  WMQ data directory\Qmgrs\qMgrName\service.env
```

```
CLASSPATH= WMQ Installation Directory\mqxr\samples\samples
```

Note: `service.env` must not contain any variables. Substitute the actual value of *`WMQ Install Directory`*.

Figure 25. Sample `service.env` for Windows

Trace

See "Tracing the telemetry (MQXR) service" on page 502. The parameters to configure trace are stored in these files:

```
WMQ data directory\Qmgrs\qMgrName\mqxr\trace.config
WMQ data directory\Qmgrs\qMgrName\mqxr\mqxrtraceOn.properties
WMQ data directory\Qmgrs\qMgrName\mqxr\mqxrtraceOff.properties
```

Client-side log files and client-side configuration files

For the latest information and downloads, see the following resources:

- The Eclipse Paho project, and MQTT.org, have free downloads of the latest telemetry clients and samples for a range of programming languages. Use these sites to help you develop sample programs for publishing and subscribing IBM MQ Telemetry Transport, and for adding security features.
- The MA9C: IBM Messaging Telemetry Clients SupportPac is no longer available for download. If you have a previously downloaded copy, it has the following contents:
  - The MA9B version of the MA9C: IBM Messaging Telemetry Clients SupportPac included a compiled sample application (`mqttv3app.jar`) and associated client library (`mqttv3.jar`). They were provided in the following directories:
    - `ma9b/SDK/clients/java/org.eclipse.paho.sample.mqttv3app.jar`
    - `ma9b/SDK/clients/java/org.eclipse.paho.client.mqttv3.jar`
  - In the MA9C version of this SupportPac, the `/SDK/` directory and contents was removed:
    - Only the source for the sample application (`mqttv3app.jar`) was provided. It was in this directory:

      ```
      ma9c/clients/java/samples/org/eclipse/paho/sample/mqttv3app/*.java
      ```

    - The compiled client library was still provided. It was in this directory:

      ```
      ma9c/clients/java/org.eclipse.paho.client.mqttv3-1.0.2.jar
      ```

Tracing the telemetry (MQXR) service

The trace facility provided by the IBM MQ telemetry (MQXR) service is provided to help IBM Support diagnose customer issues related to the service.

About this task

There are two ways to control trace for the IBM MQ telemetry service:

- By using the **strmqtrc** and **endmqtrc** commands to start and stop trace. Enabling trace, using the **strmqtrc** command, generates trace information for the entire queue manager where the IBM MQ telemetry service is running. This includes the IBM MQ telemetry service itself, and the underlying Java Message Queuing Interface (JMQI) that the service uses to communicate with other queue manager components.
- By running the **controlMQXRChannel** command. Note, that turning trace on using the **controlMQXRChannel** command traces only the IBM MQ telemetry service.

If you are unsure which option to use, contact your IBM Support representative and they will be able to advise you on the best way to collect trace for the issue that you are seeing.

Procedure

1. Method one
   a) Bring up a command prompt and navigate to the directory:
      MQ_INSTALLATION_PATH\bin
   b) Run the **strmqtrc** command to enable trace.

      ```
      strmqtrc -m qmgr_name
      ```

      where *qmgr_name* is the name of the queue manager where the IBM MQ MQXR service is running.
   c) Reproduce the issue.
   d) Stop trace, by running the command:

      ```
      endmqtrc -m qmgr_name
      ```

2. Method two.
   a) Bring up a command prompt and navigate to the directory:
      MQ_INSTALLATION_PATH\mqxr\bin
   b) Run the following command to enable trace:
      - Windows

        ```
        controlMQXRChannel -qmgr=qmgr_name -mode=starttrace [clientid=ClientIdentifier]
        ```

      - Linux UNIX

        ```
        ./controlMQXRChannel.sh -qmgr=qmgr_name -mode=starttrace [clientid=ClientIdentifier]
        ```

      where *qmgr_name* is the name of the queue manager where the MQXR Service is running.
      Set *ClientIdentifier* to the client identifier of an MQTT client. If you specify the **clientid** parameter, the IBM MQ telemetry service trace captures activity for only the MQTT client with that client identifier.

      If you want to trace the IBM MQ telemetry service activity for more than one specific MQTT client, you can run the command multiple times, specifying a different client identifier each time.
   c) Reproduce the issue.
   d) When the issue occurs, stop trace by running the following command:
      - Windows

        ```
        controlMQXRChannel -qmgr=qmgr_name -mode=stoptrace
        ```

      - Linux UNIX

        ```
        ./controlMQXRChannel.sh -qmgr=qmgr_name -mode=stoptrace [clientid=ClientIdentifier]
        ```

where *qmgr_name* is the name of the queue manager where the MQXR Service is running.

Results

To view the trace output, go to the following directory:

- Windows *MQ_DATA_PATH*`\trace`.
- Linux UNIX `/var/mqm/trace`.

The trace files containing the information from the MQXR service are called `mqxr_`*N*`.trc`, where *N* is a number.

Trace information generated by the JMQI is written to a trace file called `mqxr_`*PPPPP*`.trc`, where *PPPPP* is the process identifier for the MQXR Service.

Related information

strmqtrc

Resolving problem: MQTT client does not connect

Resolve the problem of an MQTT client program failing to connect to the telemetry (MQXR) service.

Before you begin

Is the problem at the server, at the client, or with the connection? Have you have written your own MQTT v3 protocol handling client, or an MQTT client application using the C or Java MQTT clients?

See Verifying the installation of IBM MQ Telemetry for further information, and check that the telemetry channel and telemetry (MQXR) service are running correctly.

About this task

There are a number of reasons why an MQTT client might not connect, or you might conclude it has not connected, to the telemetry server.

Procedure

1. Consider what inferences can be drawn from the reason code that the telemetry (MQXR) service returned to `MqttClient.Connect`. What type of connection failure is it?

Option	Description
`REASON_CODE_INVALID_PROTOCOL_VERSION`	Make sure that the socket address corresponds to a telemetry channel, and you have not used the same socket address for another broker.
`REASON_CODE_INVALID_CLIENT_ID`	Check that the client identifier is no longer than 23 bytes, and contains only characters from the range: `A-Z, a-z, 0-9, './_%`
`REASON_CODE_SERVER_CONNECT_ERROR`	Check that the telemetry (MQXR) service and the queue manager are running normally. Use **`netstat`** to check that the socket address is not allocated to another application.

   If you have written an MQTT client library rather than use one of the libraries provided by IBM MQ Telemetry, look at the `CONNACK` return code.

   From these three errors you can infer that the client has connected to the telemetry (MQXR) service, but the service has found an error.

2. Consider what inferences can be drawn from the reason codes that the client produces when the telemetry (MQXR) service does not respond:

Option	Description
`REASON_CODE_CLIENT_EXCEPTION` **`REASON_CODE_CLIENT_TIMEOUT`**	Look for an FDC file at the server; see "Server-side logs" on page 500. When the telemetry (MQXR) service detects the client has timed out, it writes a first-failure data capture (FDC) file. It writes an FDC file whenever the connection is unexpectedly broken.

   The telemetry (MQXR) service might not have responded to the client, and the timeout at the client expires. The IBM MQ Telemetry Java client only hangs if the application has set an indefinite timeout. The client throws one of these exceptions after the timeout set for `MqttClient.Connect` expires with an undiagnosed connection problem.

   Unless you find an FDC file that correlates with the connection failure you cannot infer that the client tried to connect to the server:

   a) Confirm that the client sent a connection request.

      Check the TCPIP request with a tool such as **`tcpmon`**, available from (for example) https://code.google.com/p/tcpmon/

   b) Does the remote socket address used by the client match the socket address defined for the telemetry channel?

      The default file persistence class in the Java SE MQTT client supplied with IBM MQ Telemetry creates a folder with the name: *clientIdentifier*-tcp*hostNameport* or *clientIdentifier*-ssl*hostNameport* in the client working directory. The folder name tells you the `hostName` and `port` used in the connection attempt ; see "Client-side log files and client-side configuration files" on page 502.

   c) Can you ping the remote server address?

   d) Does **`netstat`** on the server show the telemetry channel is running on the port the client is connecting too?

3. Check whether the telemetry (MQXR) service found a problem in the client request.

   The telemetry (MQXR) service writes errors it detects into `mqxr_n.log`, and the queue manager writes errors into `AMQERR01.LOG`.

4. Attempt to isolate the problem by running another client.

   See Verifying the installation of IBM MQ Telemetry for further information.

   Run the sample programs on the server platform to eliminate uncertainties about the network connection, then run the samples on the client platform.

5. Other things to check:

   a) Are tens of thousands of MQTT clients trying to connect at the same time?

      Telemetry channels have a queue to buffer a backlog of incoming connections. Connections are processed in excess of 10,000 a second. The size of the backlog buffer is configurable using the telemetry channel wizard in IBM MQ Explorer. Its default size is 4096. Check that the backlog has not been configured to a low value.

   b) Are the telemetry (MQXR) service and queue manager still running?

   c) Has the client connected to a high availability queue manager that has switched its TCPIP address?

   d) Is a firewall selectively filtering outbound or return data packets?

Resolving problem: MQTT client connection dropped

Find out what is causing a client to throw unexpected `ConnectionLost` exceptions after successfully connecting and running for either a short or long while.

Before you begin

The MQTT client has connected successfully. The client might be up for a long while. If clients are starting with only a short interval between them, the time between connecting successfully and the connection being dropped might be short.

It is not hard to distinguish a dropped connection from a connection that was successfully made, and then later dropped. A dropped connection is defined by the MQTT client calling the `MqttCallback.ConnectionLost` method. The method is only called after the connection has been successfully established. The symptom is different to `MqttClient.Connect` throwing an exception after receiving a negative acknowledgment or timing out.

If the MQTT client application is not using the MQTT client libraries supplied by IBM MQ, the symptom depends on the client. In the MQTT v3 protocol, the symptom is a lack of timely response to a request to the server, or the failure of the TCP/IP connection.

About this task

The MQTT client calls `MqttCallback.ConnectionLost` with a throwable exception in response to any server-side problems encountered after receiving a positive connection acknowledgment. When an MQTT client returns from `MqttTopic.publish` and `MqttClient.subscribe` the request is transferred to an MQTT client thread that is responsible for sending and receiving messages. Server-side errors are reported asynchronously by passing a throwable exception to the `ConnectionLost` callback method.

Procedure

1. Has another client started that used the same `ClientIdentifier` ?

   If a second client is started, or the same client is restarted, using the same `ClientIdentifier`, the first connection to the first client is dropped.
2. Has the client accessed a topic that it is not authorized to publish or subscribe to?

   Any actions the telemetry service takes on behalf of a client that return `MQCC_FAIL` result in the service dropping the client connection.

   The reason code is not returned to the client.

   - Look for log messages in the `mqxr.log` and `AMQERR01.LOG` files for the queue manager the client is connected to; see "Server-side logs" on page 500.
3. Has the TCP/IP connection dropped?

   A firewall might have a low timeout setting for marking a TCPIP connection as inactive, and dropped the connection.

   - Shorten the inactive TCPIP connection time using `MqttConnectOptions.setKeepAliveInterval`.

Resolving problem: Lost messages in an MQTT application

Resolve the problem of losing a message. Is the message non-persistent, sent to the wrong place, or never sent? A wrongly coded client program might lose messages.

Before you begin

How certain are you that the message you sent, was lost? Can you infer that a message is lost because the message was not received? If message is a publication, which message is lost: the message sent by the publisher, or the message sent to the subscriber? Or did the subscription get lost, and the broker is not sending publications for that subscription to the subscriber?

If the solution involves distributed publish/subscribe, using clusters or publish/subscribe hierarchies, there are numerous configuration issues that might result in the appearance of a lost message.

If you sent a message with "At least once" or "At most once" quality of service, it is likely that the message you think is lost was not delivered in the way you expected. It is unlikely that the message has been wrongly deleted from the system. It might have failed to create the publication or the subscription you expected.

The most important step you take in doing problem determination of lost messages is to confirm the message is lost. Recreate the scenario and lose more messages. Use the "At least once" or "At most once" quality of service to eliminate all cases of the system discarding messages.

About this task

There are four legs to diagnosing a lost message.

1. "Fire and forget" messages working as-designed. "Fire and forget" messages are sometimes discarded by the system.
2. Configuration: setting up publish/subscribe with the correct authorities in a distributed environment is not straightforward.
3. Client programming errors: the responsibility for message delivery is not solely the responsibility of code written by IBM.
4. If you have exhausted all these possibilities, you might decide to involve IBM service.

Procedure

1. If the lost message had the "Fire and forget" quality of service, set the "At least once" or "At most once" quality of service. Attempt to lose the message again.
   - Messages sent with "Fire and forget" quality of service are thrown away by IBM MQ in a number of circumstances:
     - Communications loss and channel stopped.
     - Queue manager shut down.
     - Excessive number of messages.
   - The delivery of "Fire and forget" messages depends upon the reliability of TCP/IP. TCP/IP continues to send data packets again until their delivery is acknowledged. If the TCP/IP session is broken, messages with the "Fire and forget" quality of service are lost. The session might be broken by the client or server closing down, a communications problem, or a firewall disconnecting the session.
2. Check that client is restarting the previous session, in order to send undelivered messages with "At least once" or "At most once" quality of service again.

   a) If the client application is using the Java SE MQTT client, check that it sets `MqttClient.CleanSession` to `false`

   b) If you are using different client libraries, check that a session is being restarted correctly.
3. Check that the client application is restarting the same session, and not starting a different session by mistake.

   To start the same session again, `cleanSession = false`, and the `Mqttclient.clientIdentifier` and the `MqttClient.serverURI` must be the same as the previous session.
4. If a session closes prematurely, check that the message is available in the persistence store at the client to send again.

   a) If the client application is using the Java SE MQTT client, check that the message is being saved in the persistence folder; see “Client-side log files and client-side configuration files” on page 502

   b) If you are using different client libraries, or you have implemented your own persistence mechanism, check that it is working correctly.
5. Check that no one has deleted the message before it was delivered.

   Undelivered messages awaiting delivery to MQTT clients are stored in `SYSTEM.MQTT.TRANSMIT.QUEUE`. Messages awaiting delivery to the telemetry server are stored by the client persistence mechanism; see Message persistence in MQTT clients.
6. Check that the client has a subscription for the publication it expects to receive.

   List subscriptions using MQ Explorer, or by using **runmqsc** or PCF commands. All MQTT client subscriptions are named. They are given a name of the form: *ClientIdentifier*:*Topic name*
7. Check that the publisher has authority to publish, and the subscriber to subscribe to the publication topic.

   ```
   dspmqaut -m qMgr -n topicName -t topic -p user ID
   ```

   In a clustered publish/subscribe system, the subscriber must be authorized to the topic on the queue manager to which the subscriber is connected. It is not necessary for the subscriber to be authorized to subscribe to the topic on the queue manager where the publication is published. The channels between the queue managers must be correctly authorized to pass on the proxy subscription and forward the publication.

   Create the same subscription and publish to it using IBM MQ Explorer. Simulate your application client publishing and subscribing by using the client utility. Start the utility from IBM MQ Explorer and change its user ID to match the one adopted by your client application.
8. Check that the subscriber has permission to put the publication on the `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   ```
   dspmqaut -m qMgr -n queueName -t queue -p user ID
   ```
9. Check that the IBM MQ point-to-point application has authority to put its message on the `SYSTEM.MQTT.TRANSMIT.QUEUE`.

   ```
   dspmqaut -m qMgr -n queueName -t queue -p user ID
   ```

   See Sending a message to a client directly.

Resolving problem: Telemetry (MQXR) service does not start

Resolve the problem of the telemetry (MQXR) service failing to start. Check the IBM MQ Telemetry installation and no files are missing, moved, or have the wrong permissions. Check the paths used by the telemetry (MQXR) service locate the telemetry (MQXR) service programs.

Before you begin

The WebSphere MQ Telemetry feature is installed. The IBM MQ Explorer has a Telemetry folder in **IBM MQ > Queue Managers > *qMgrName* > Telemetry**. If the folder does not exist, the installation has failed.

The Telemetry (MQXR) service must have been created for it to start. If the telemetry (MQXR) service has not been created, then run the **Define sample configuration...** wizard in the `Telemetry` folder.

If the telemetry (MQXR) service has been started before, then additional **Channels** and **Channel Status** folders are created under the `Telemetry` folder. The Telemetry service, `SYSTEM.MQXR.SERVICE`, is in the **Services** folder. It is visible if the Explorer radio button to show System Objects is clicked.

Right click `SYSTEM.MQXR.SERVICE` to start and stop the service, show its status, and display whether your user ID has authority to start the service.

About this task

The SYSTEM.MQXR.SERVICE telemetry (MQXR) service fails to start. A failure to start manifests itself in two different ways:

1. The start command fails immediately.
2. The start command succeeds, and is immediately followed by the service stopping.

Procedure

1. Start the service

   Result
   The service stops immediately. A window displays an error message; for example:

   ```
   IBM MQ cannot process the request because the
   executable specified cannot be started. (AMQ4160)
   ```

   Reason

   Files are missing from the installation, or the permissions on installed files are set wrongly.

   The IBM MQ Telemetry feature is installed only on one of a pair of highly available queue managers. If the queue manager instance switches over to a standby, it tries to start SYSTEM.MQXR.SERVICE. The command to start the service fails because the telemetry (MQXR) service is not installed on the standby.

   Investigation
   Look in error logs; see "Server-side logs" on page 500.

   Actions
   Install, or uninstall and reinstall the IBM MQ Telemetry feature.
2. Start the service; wait for 30 seconds; refresh the Explorer and check the service status.

   Result
   The service starts and then stops.

   Reason
   SYSTEM.MQXR.SERVICE started the **runMQXRService** command, but the command failed.

   Investigation

   Look in error logs; see "Server-side logs" on page 500.

   See if the problem occurs with only the sample channel defined. Backup and the clear the contents of the *WMQ data directory*\Qmgrs*qMgrName*\mqxr\ directory. Run the sample configuration wizard and try to start the service.

   Actions
   Look for permission and path problems.

Resolving problem: JAAS login module not called by the telemetry service

Find out if your JAAS login module is not being called by the telemetry (MQXR) service, and configure JAAS to correct the problem.

Before you begin

You have modified *WMQ installation directory*\mqxr\samples\samples\LoginModule.java to create your own authentication class *WMQ installation directory*\mqxr\samples\samples\LoginModule.class. Alternatively, you have written your own JAAS authentication classes and placed them in a directory of your choosing. After some initial testing with the telemetry (MQXR) service, you suspect that your authentication class is not being called by the telemetry (MQXR) service.

Note: Guard against the possibility that your authentication classes might be overwritten by maintenance being applied to IBM MQ. Use your own path for authentication classes, rather than a path within the IBM MQ directory tree.

About this task

The task uses a scenario to illustrate how to resolve the problem. In the scenario, a package called `security.jaas` contains a JAAS authentication class called `JAASLogin.class`. It is stored in the path `C:\WMQTelemetryApps\security\jaas`. Refer to Telemetry channel JAAS configuration and `AuthCallback` MQXR class for help in configuring JAAS for IBM MQ Telemetry. The example, "Example JAAS configuration" on page 510 is a sample configuration.

Procedure

1. Look in `mqxr.log` for an exception thrown by `javax.security.auth.login.LoginException`.

   See "Server-side logs" on page 500 for the path to `mqxr.log`, and Figure 31 on page 512 for an example of the exception listed in the log.
2. Correct your JAAS configuration by comparing it with the worked example in "Example JAAS configuration" on page 510.
3. Replace your login class by the sample `JAASLoginModule`, after refactoring it into your authentication package and deploy it using the same path. Switch the value of `loggedIn` between `true` and `false`.

   If the problem goes away when `loggedIn` is `true`, and appears the same when `loggedIn` is `false`, the problem lies in your login class.
4. Check whether the problem is with authorization rather than authentication.
   a) Change the telemetry channel definition to perform authorization checking using a fixed user ID. Select a user ID that is a member of the `mqm` group.
   b) Rerun the client application.

      If the problem disappears, the solution lies with the user ID being passed for authorization. What is the user name being passed? Print it to file from your login module. Check its access permissions using IBM MQ Explorer, or **`dspmqauth`**.

Example JAAS configuration

Use the **New telemetry channel** wizard, in IBM MQ Explorer, to configure a telemetry channel.

The JAAS configuration file has a stanza named `JAASConfig` that names the Java class `security.jaas.JAASLogin`, which JAAS is to use to authenticate clients.

```
JAASConfig {
  security.jaas.JAASLogin required debug=true;
};
```

Figure 26. `WMQ Installation directory\data\qmgrs\qMgrName\mqxr\jaas.config`

When `SYSTEM.MQTT.SERVICE` starts, it adds the path in Figure 27 on page 510 to its classpath.

```
CLASSPATH=C:\WMQTelemtryApps;
```

Figure 27. `WMQ Installation directory\data\qmgrs\qMgrName\service.env`

Figure 28 on page 511 shows the additional path in Figure 27 on page 510 added to the classpath that is set up for the telemetry (MQXR) service.

```
CLASSPATH=;C:\IBM\MQ\Program\mqxr\bin\\..\lib\MQXRListener.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\WMQCommonServices.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\objectManager.utils.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\lib\com.ibm.micro.xr.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mq.jmqi.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mqjms.jar;
C:\IBM\MQ\Program\mqxr\bin\\..\..\java\lib\com.ibm.mq.jar;
C:\WMQTelemtryApps;
```

Figure 28. Classpath output from runMQXRService.bat

The output in Figure 29 on page 511 shows that the telemetry (MQXR) service has started.

```
21/05/2010 15:32:12 [main] com.ibm.mq.MQXRService.MQXRPropertiesFile
AMQXR2011I: Property com.ibm.mq.MQXR.channel/JAASMCAUser value
com.ibm.mq.MQXR.Port=1884;
com.ibm.mq.MQXR.JAASConfig=JAASConfig;
com.ibm.mq.MQXR.UserName=Admin;
com.ibm.mq.MQXR.StartWithMQXRService=true
```

Figure 29. `WMQ Installation directory\data\qmgrs\qMgrName\errors\`

When the client application connects to the JAAS channel, if `com.ibm.mq.MQXR.JAASConfig=JAASWrongConfig` does not match the name of a JAAS stanza in the `jaas.config` file, the connection fails, and the client throws an exception with a return code of `0`; see Figure 30 on page 511. The second exception, `Client is not connected (32104)`, was thrown because the client attempted to disconnect when it was not connected.

```
Connecting to tcp://localhost:1883 with client ID SampleJavaV3_publish
reason 5
msg Not authorized to connect
loc Not authorized to connect
cause null
excep Not authorized to connect (5)
Not authorized to connect (5)
        at
org.eclipse.paho.client.mqttv3.internal.ExceptionHelper.createMqttException(ExceptionHelper.java
:28)
        at
org.eclipse.paho.client.mqttv3.internal.ClientState.notifyReceivedAck(ClientState.java:885)
        at org.eclipse.paho.client.mqttv3.internal.CommsReceiver.run(CommsReceiver.java:118)
        at java.lang.Thread.run(Thread.java:809)
```

Figure 30. Exception thrown when connecting to the Eclipse Paho sample

`mqxr.log` contains additional output shown in Figure 30 on page 511.

The error is detected by JAAS which throws `javax.security.auth.login.LoginException` with the cause `No LoginModules configured for JAAS`. It could be caused, as in Figure 31 on page 512, by a bad configuration name. It might also be the result of other problems JAAS has encountered loading the JAAS configuration.

If no exception is reported by JAAS, JAAS has successfully loaded the `security.jaas.JAASLogin` class named in the `JAASConfig` stanza.

```
15/06/15 13:49:28.337
AMQXR2050E: Unable to load JAAS config:MQXRWrongConfig.
The following exception occurred javax.security.auth.login.LoginException:
No LoginModules configured for MQXRWrongConfig
```

Figure 31. Error loading JAAS configuration

Recovering after failure

Follow a set of procedures to recover after a serious problem.

About this task

Use the recovery methods described here if you cannot resolve the underlying problem by using the diagnostic techniques described throughout the Troubleshooting and support section. If your problem cannot be resolved by using these recovery techniques, contact your IBM Support Center.

Procedure

See the following links for instructions on how to recover from different types of failures:

- "Disk drive failures" on page 513
- "Damaged queue manager object" on page 514
- "Damaged single object" on page 514
- "Automatic media recovery failure" on page 514

z/OS

See the following links for instructions on how to recover from different types of failures on IBM MQ for z/OS:

- z/OS
  "Shared queue problems" on page 515
- z/OS
  "Active log problems" on page 516
- z/OS
  "Archive log problems" on page 521
- z/OS
  "BSDS problems" on page 524
- z/OS
  "Page set problems" on page 530
- z/OS
  "Coupling facility and Db2 problems" on page 532
- z/OS
  "Problems with long-running units of work" on page 535
- z/OS
  "IMS-related problems" on page 535
- z/OS
  "Hardware problems" on page 537

Related concepts

"Troubleshooting and support" on page 7
If you are having problems with your queue manager network or IBM MQ applications, use the techniques described to help you diagnose and solve the problems.

"Troubleshooting overview" on page 7
Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself "what happened?"

"Making initial checks on Windows, UNIX and Linux systems" on page 9
Before you start problem determination in detail, consider whether there is an obvious cause of the problem, or an area of investigation that is likely to give useful results. This approach to diagnosis can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

Related tasks

"Contacting IBM Software Support" on page 328
You can contact IBM Support through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting and other news.

Related information

Backing up and restoring IBM MQ

z/OS Planning for backup and recovery on z/OS

Disk drive failures

You might have problems with a disk drive containing either the queue manager data, the log, or both. Problems can include data loss or corruption. The three cases differ only in the part of the data that survives, if any.

In ***all*** cases first check the directory structure for any damage and, if necessary, repair such damage. If you lose queue manager data, the queue manager directory structure might have been damaged. If so, re-create the directory tree manually before you restart the queue manager.

If damage has occurred to the queue manager data files, but not to the queue manager log files, then the queue manager will normally be able to restart. If any damage has occurred to the queue manager log files, then it is likely that the queue manager will not be able to restart.

Having checked for structural damage, there are a number of things you can do, depending on the type of logging that you use.

- **Where there is major damage to the directory structure or any damage to the log**, remove all the old files back to the `QMgrName` level, including the configuration files, the log, and the queue manager directory, restore the last backup, and restart the queue manager.
- **For linear logging with media recovery**, ensure that the directory structure is intact and restart the queue manager. If the queue manager restarts, check, using MQSC commands such as DISPLAY QUEUE, whether any other objects have been damaged. Recover those you find, using the `rcrmqobj` command. For example:

  ```
  rcrmqobj -m QMgrName -t all *
  ```

  where `QMgrName` is the queue manager being recovered. `-t all *` indicates that all damaged objects of any type are to be recovered. If only one or two objects have been reported as damaged, you can specify those objects by name and type here.
- **For linear logging with media recovery and with an undamaged log**, you might be able to restore a backup of the queue manager data leaving the existing log files and log control file unchanged. Starting the queue manager applies the changes from the log to bring the queue manager back to its state when the failure occurred.

  This method relies on two things:

1. You must restore the checkpoint file as part of the queue manager data. This file contains the information determining how much of the data in the log must be applied to give a consistent queue manager.
2. You must have the oldest log file required to start the queue manager at the time of the backup, and all subsequent log files, available in the log file directory.

   If this is not possible, restore a backup of both the queue manager data and the log, both of which were taken at the same time. This causes message integrity to be lost.

- **For circular logging**, if the queue manager log files are damaged, restore the queue manager from the latest backup that you have. Once you have restored the backup, restart the queue manager and check for damaged objects. However, because you do not have media recovery, you must find other ways of re-creating the damaged objects.

  If the queue manager log files are not damaged, the queue manager will normally be able to restart. Following the restart you must identify all damaged objects, then delete and redefine them.

Damaged queue manager object

What to do if the queue manager reports a damaged object during normal operation.

There are two ways of recovering in these circumstances, depending on the type of logging you use:

- **For linear logging**, manually delete the file containing the damaged object and restart the queue manager. (You can use the `dspmqfls` command to determine the real, file-system name of the damaged object.) Media recovery of the damaged object is automatic.
- **For circular logging**, restore the last backup of the queue manager data and log, and restart the queue manager.

  There is a further option if you are using circular logging. For a damaged queue, or other object, delete the object and define the object again. In the case of a queue, this option does not allow you to recover any data on the queue.

  Note: Restoring from backup is likely to be out of date, due to the fact that you must have your queue manager shutdown in order to get a clean backup of the queue files.

Damaged single object

If a single object is reported as damaged during normal operation, for linear logging you can re-create the object from its media image. However, for circular logging you cannot re-create a single object.

Automatic media recovery failure

If a local queue required for queue manager startup with a linear log is damaged, and the automatic media recovery fails, restore the last backup of the queue manager data and log and restart the queue manager.

Example recovery procedures on z/OS

Use this topic as a reference for various recovery procedures.

This topic describes procedures for recovering IBM MQ after various error conditions. These error conditions are grouped in the following categories:

Table 21. Example recovery procedures

Problem category	Problem	Where to look next
Shared queue problems	Conflicting definitions for both private and shared queues.	"Shared queue problems" on page 515

Table 21. Example recovery procedures (continued)

Problem category	**Problem**	**Where to look next**
Active log problems	• Dual logging is lost. • Active log has stopped. • One or both copies of the active log data set are damaged. • Write errors on active log data set. • Active log is becoming full or is full. • Read errors on active log data set.	"Active log problems" on page 516
Archive log problems	• Insufficient DASD space to complete offloading active log data sets. • Offload task has terminated abnormally. • Archive data set allocation problem. 1 • Read I/O errors on the archive data set during restart.	"Archive log problems" on page 521
BSDS problems	• Error opening BSDS. • Log content does not correspond with BSDS information. • Both copies of the BSDS are damaged. • Unequal time stamps. • Dual BSDS data sets are out of synchronization. • I/O error on BSDS.	"BSDS problems" on page 524
Page set problems	• Page set full. • A page set has an I/O error.	"Page set problems" on page 530
coupling facility and Db2 problems	• Storage medium full. • Db2 system fails. • Db2 data-sharing group fails. • Db2 and the coupling facility fail.	"Coupling facility and Db2 problems" on page 532
Unit of work problems	A long-running unit of work is encountered.	"Problems with long-running units of work" on page 535
IMS problems	• An IMS application terminates abnormally. • The IMS adapter cannot connect to IBM MQ. • IMS not operational.	"IMS-related problems" on page 535
Hardware problems	Media recovery procedures	"Hardware problems" on page 537

Shared queue problems

Problems occur if IBM MQ discovers that a page set based queue, and a shared queue of the same name are defined.

Symptoms
IBM MQ issues the following message:

```
CSQI063E +CSQ1 QUEUE queue-name IS BOTH PRIVATE AND SHARED
```

During queue manager restart, IBM MQ discovered that a page set based queue and a shared queue of the same name coexist.

System action
Once restart processing has completed, any MQOPEN request to that queue name fails, indicating the coexistence problem.

System programmer action
None.

Operator action
Delete one version of the queue to allow processing of that queue name. If there are messages on the queue that must be kept, you can use the MOVE QLOCAL command to move them to the other queue.

Active log problems

Use this topic to resolve different problems with the active logs.

This topic covers the following active log problems:

- “Dual logging is lost” on page 516
- “Active log stopped” on page 516
- “One or both copies of the active log data set are damaged” on page 517
- “Write I/O errors on an active log data set” on page 518
- “I/O errors occur while reading the active log” on page 518
- “Active log is becoming full” on page 520
- Active log is full

Dual logging is lost

Symptoms
IBM MQ issues the following message:

```
CSQJ004I +CSQ1 ACTIVE LOG COPY n INACTIVE, LOG IN SINGLE MODE,
           ENDRBA=...
```

Having completed one active log data set, IBM MQ found that the subsequent (COPY n) data sets were not offloaded or were marked stopped.

System action
IBM MQ continues in single mode until offloading has been completed, then returns to dual mode.

System programmer action
None.

Operator action
Check that the offload process is proceeding and is not waiting for a tape mount. You might need to run the print log map utility to determine the state of all data sets. You might also need to define additional data sets.

Active log stopped

Symptoms
IBM MQ issues the following message:

```
CSQJ030E +CSQ1 RBA RANGE startrba TO endrba NOT AVAILABLE IN ACTIVE
            LOG DATA SETS
```

System action
The active log data sets that contain the RBA range reported in message CSQJ030E are unavailable to IBM MQ. The status of these logs is STOPPED in the BSDS. The queue manager terminates with a dump.

System programmer action
You must resolve this problem before restarting the queue manager. The log RBA range must be available for IBM MQ to be recoverable. An active log that is marked as STOPPED in the BSDS will never be reused or archived and this creates a hole in the log.

Look for messages that indicate why the log data set has stopped, and follow the instructions for those messages.

Modify the BSDS active log inventory to reset the STOPPED status. To do this, follow this procedure after the queue manager has terminated:

1. Use the print log utility (CSQJU004) to obtain a copy of the BSDS log inventory. This shows the status of the log data sets.
2. Use the DELETE function of the change log inventory utility (CSQJU003) to delete the active log data sets that are marked as STOPPED.
3. Use the NEWLOG function of CSQJU003 to add the active logs back into the BSDS inventory. The starting and ending RBA for each active log data set must be specified on the NEWLOG statement. (The correct values to use can be found from the print log utility report obtained in Step 1.)
4. Rerun CSQJU004. The active log data sets that were marked as STOPPED are now shown as NEW and NOT REUSABLE. These active logs will be archived in due course.
5. Restart the queue manager.

Note: If your queue manager is running in dual BSDS mode, you must update both BSDS inventories.

One or both copies of the active log data set are damaged

Symptoms
IBM MQ issues the following messages:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,
           STARTRBA=..., ENDRBA=...,
           DOES NOT AGREE WITH BSDS INFORMATION
CSQJ232E +CSQ1 OUTPUT DATA SET CONTROL INITIALIZATION PROCESS FAILED
```

System action
Queue manager startup processing is terminated.

System programmer action
If one copy of the data set is damaged, carry out these steps:

1. Rename the damaged active log data set and define a replacement data set.
2. Copy the undamaged data set to the replacement data set.
3. Use the change log inventory utility to:
   - Remove information relating to the damaged data set from the BSDS.

- Add information relating to the replacement data set to the BSDS.

4. Restart the queue manager.

If both copies of the active log data sets are damaged, the current page sets are available, **and the queue manager shut down cleanly**, carry out these steps:

1. Rename the damaged active log data sets and define replacement data sets.
2. Use the change log records utility to:
   - Remove information relating to the damaged data set from the BSDS.
   - Add information relating to the replacement data set to the BSDS.
3. Rename the current page sets and define replacement page sets.
4. Use CSQUTIL (FORMAT and RESETPAGE) to format the replacement page sets and copy the renamed page sets to them. The RESETPAGE function also resets the log information in the replacement page sets.

If the queue manager did not shut down cleanly, you must either restore your system from a previous known point of consistency, or perform a cold start (described in Reinitializing a queue manager).

Operator action
None.

Write I/O errors on an active log data set

Symptoms
IBM MQ issues the following message:

```
CSQJ105E +CSQ1 csect-name LOG WRITE ERROR DSNAME=...,
           LOGRBA=..., ERROR STATUS=ccccffss
```

System action
IBM MQ carries out these steps:

1. Marks the log data set that has the error as TRUNCATED in the BSDS.
2. Goes on to the next available data set.
3. If dual active logging is used, truncates the other copy at the same point.

The data in the truncated data set is offloaded later, as usual.

The data set will be reused on the next cycle.

System programmer action
None.

Operator action
If errors on this data set still exist, shut down the queue manager after the next offload process. Then use Access Method Services (AMS) and the change log inventory utility to add a replacement. (For instructions, see Changing the BSDS.)

I/O errors occur while reading the active log

Symptoms
IBM MQ issues the following message:

```
CSQJ106E +CSQ1 LOG READ ERROR DSNAME=..., LOGRBA=...,
          ERROR STATUS=ccccffss
```

System action

This depends on when the error occurred:

- If the error occurs during the offload process, the process tries to read the RBA range from a second copy.
  - If no second copy exists, the active log data set is stopped.
  - If the second copy also has an error, only the original data set that triggered the offload process is stopped. The archive log data set is then terminated, leaving a gap in the archived log RBA range.
  - This message is issued:

    ```
    CSQJ124E +CSQ1 OFFLOAD OF ACTIVE LOG SUSPENDED FROM
              RBA xxxxxx TO RBA xxxxxx DUE TO I/O ERROR
    ```

  - If the second copy is satisfactory, the first copy is not stopped.
- If the error occurs during recovery, IBM MQ provides data from specific log RBAs requested from another copy or archive. If this is unsuccessful, recovery does not succeed, and the queue manager terminates abnormally.
- If the error occurs during restart, if dual logging is used, IBM MQ continues with the alternative log data set, otherwise the queue manager ends abnormally.

System programmer action

Look for system messages, such as IEC prefixed messages, and try to resolve the problem using the recommended actions for these messages.

If the active log data set has been stopped, it is not used for logging. The data set is not deallocated; it is still used for reading. Even if the data set is not stopped, an active log data set that gives persistent errors should be replaced.

Operator action

None.

Replacing the data set

How you replace the data set depends on whether you are using single or dual active logging.

If you are using dual active logging:

1. Ensure that the data has been saved.

   The data is saved on the other active log and this can be copied to a replacement active log.
2. Stop the queue manager and delete the data set with the error using Access Method Services.
3. Redefine a new log data set using Access Method Services DEFINE so that you can write to it. Use DFDSS or Access Method Services REPRO to copy the good log in to the redefined data set so that you have two consistent, correct logs again.
4. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupted data set as follows:
   a. Use the DELETE function to remove information about the corrupted data set.
   b. Use the NEWLOG function to name the new data set as the new active log data set and give it the RBA range that was successfully copied.

      You can run the DELETE and NEWLOG functions in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.

5. Restart the queue manager.

If you are using single active logging:

1. Ensure that the data has been saved.
2. Stop the queue manager.
3. Determine whether the data set with the error has been offloaded:
   a. Use the CSQJU003 utility to list information about the archive log data sets from the BSDS.
   b. Search the list for a data set with an RBA range that includes the RBA of the corrupted data set.
4. If the corrupted data set has been offloaded, copy its backup in the archive log to a new data set. Then, skip to step 6.
5. If an active log data set is stopped, an RBA is not offloaded. Use DFDSS or Access Method Services REPRO to copy the data from the corrupted data set to a new data set.

   If further I/O errors prevent you from copying the entire data set, a gap occurs in the log.

   Note: Queue manager restart will not be successful if a gap in the log is detected.
6. Use the change log inventory utility, CSQJU003, to update the information in the BSDS about the corrupted data set as follows:
   a. Use the DELETE function to remove information about the corrupted data set.
   b. Use the NEWLOG function to name the new data set as the new active log data set and to give it the RBA range that was successfully copied.

      The DELETE and NEWLOG functions can be run in the same job step. Put the DELETE statement before NEWLOG statement in the SYSIN input data set.
7. Restart the queue manager.

Active log is becoming full

The active log can fill up for several reasons, for example, delays in offloading and excessive logging. If an active log runs out of space, this has serious consequences. When the active log becomes full, the queue manager halts processing until an offload process has been completed. If the offload processing stops when the active log is full, the queue manager can end abnormally. Corrective action is required before the queue manager can be restarted.

Symptoms

Because of the serious implications of an active log becoming full, the queue manager issues the following warning message when the last available active log data set is 5% full:

```
CSQJ110E +CSQ1 LAST COPYn ACTIVE LOG DATA SET IS nnn PERCENT FULL
```

and reissues the message after each additional 5% of the data set space is filled. Each time the message is issued, the offload process is started.

System action

Messages are issued and offload processing started. If the active log becomes full, further actions are taken. See “Active log is full” on page 521

System programmer action

Use the DEFINE LOG command to dynamically add further active log data sets. This permits IBM MQ to continue its normal operation while the error causing the offload problems is corrected. For more information about the DEFINE LOG command, see DEFINE LOG.

Active log is full

Symptoms
When the active log becomes full, the queue manager halts processing until an offload process has been completed. If the offload processing stops when the active log is full, the queue manager can end abnormally. Corrective action is required before the queue manager can be restarted.

IBM MQ issues the following CSQJ111A message:

```
CSQJ111A +CSQ1 OUT OF SPACE IN ACTIVE LOG DATA SETS
```

and an offload process is started. The queue manager then halts processing until the offload process has been completed.

System action
IBM MQ waits for an available active log data set before resuming normal IBM MQ processing. Normal shut down, with either QUIESCE or FORCE, is not possible because the shutdown sequence requires log space to record system events related to shut down (for example, checkpoint records). If the offload processing stops when the active log is full, the queue manager stops with an X'6C6' abend; restart in this case requires special attention. For more details, see "Problem determination on z/OS" on page 389.

System programmer action
You can provide additional active log data sets before restarting the queue manager. This permits IBM MQ to continue its normal operation while the error causing the offload process problems is corrected. To add new active log data sets, use the change log inventory utility (CSQJU003) when the queue manager is not active. For more details about adding new active log data sets, see Changing the BSDS.

Consider increasing the number of logs by:

1. Making sure that the queue manager is stopped, then using the Access Method Services DEFINE command to define a new active log data set.
2. Defining the new active log data set in the BSDS using the change log inventory utility (CSQJU003).
3. Adding additional log data sets dynamically, using the DEFINE LOG command.

When you restart the queue manager, offloading starts automatically during startup, and any work that was in progress when IBM MQ was forced to stop is recovered.

Operator action
Check whether the offload process is waiting for a tape drive. If it is, mount the tape. If you cannot mount the tape, force IBM MQ to stop by using the z/OS CANCEL command.

Archive log problems

Use this topic to investigate, and resolve problems with the archive logs.

This topic covers the following archive log problems:

- "Allocation problems" on page 521
- "Offload task terminated abnormally" on page 522
- "Insufficient DASD space to complete offload processing" on page 523
- "Read I/O errors on the archive data set while IBM MQ is restarting" on page 524

Allocation problems

Symptoms
IBM MQ issues message: CSQJ103E

```
CSQJ103E +CSQ1 LOG ALLOCATION ERROR DSNAME=dsname,
           ERROR STATUS=eeeeiiii, SMS REASON CODE=sss
```

z/OS dynamic allocation provides the ERROR STATUS. If the allocation was for offload processing, the following message is also displayed: CSQJ115E:

```
CSQJ115E +CSQ1 OFFLOAD FAILED, COULD NOT ALLOCATE AN ARCHIVE
           DATA SET
```

System action
The following actions take place:

- If the input is needed for recovery, and recovery is not successful, and the queue manager ends abnormally.
- If the active log had become full and an offload task was scheduled but not completed, the offload task tries again the next time it is triggered. The active log does not reuse a data set that has not yet been archived.

System programmer action
None.

Operator action
Check the allocation error code for the cause of the problem, and correct it. Ensure that drives are available, and either restart or wait for the offload task to be retried. Be careful if a DFP/DFSMS ACS user-exit filter has been written for an archive log data set, because this can cause a device allocation error when the queue manager tries to read the archive log data set.

Offload task terminated abnormally

Symptoms
No specific IBM MQ message is issued for write I/O errors.

Only a z/OS error recovery program message appears. If you get IBM MQ message CSQJ128E, the offload task has ended abnormally.

System action
The following actions take place:

- The offload task abandons the output data set; no entry is made in the BSDS.
- The offload task dynamically allocates a new archive and restarts offloading from the point at which it was previously triggered.
- If an error occurs on the new data set:
  - In dual archive mode, message CSQJ114I is generated and the offload processing changes to single mode:

    ```
    CSQJ114I +CSQ1 ERROR ON ARCHIVE DATA SET, OFFLOAD
               CONTINUING WITH ONLY ONE ARCHIVE DATA SET BEING
               GENERATED
    ```
  - In single archive mode, the output data set is abandoned. Another attempt to process this RBA range is made the next time offload processing is triggered.
  - The active log does not wrap around; if there are no more active logs, data is not lost.

System programmer action
None.

Operator action
Ensure that offload task is allocated on a reliable drive and control unit.

Insufficient DASD space to complete offload processing

Symptoms
While offloading the active log data sets to DASD, the process terminates unexpectedly. IBM MQ issues message CSQJ128E:

```
CSQJ128E +CSQ1 LOG OFF-LOAD TASK FAILED FOR ACTIVE LOG nnnnn
```

The error is preceded by z/OS messages IEC030I, IEC031I, or IEC032I.

System action
IBM MQ de-allocates the data set on which the error occurred. If IBM MQ is running in dual archive mode, IBM MQ changes to single archive mode and continues the offload task. If the offload task cannot be completed in single archive mode, the active log data sets cannot be offloaded, and the state of the active log data sets remains NOT REUSABLE. Another attempt to process the RBA range of the abandoned active log data sets is made the next time the offload task is triggered.

System programmer action
The most likely causes of these symptoms are:

- The size of the archive log data set is too small to contain the data from the active log data sets during offload processing. All the secondary space allocations have been used. This condition is normally accompanied by z/OS message IEC030I. The return code in this message might provide further explanations for the cause of these symptoms.

  To solve the problem

  1. Issue the command `CANCEL <queue-manager name>` to cancel the queue manager job
  2. Increase the primary or secondary allocations (or both) for the archive log data set (in the CSQ6ARVP system parameters), or reduce the size of the active log data set.

     If the data to be offloaded is large, you can mount another online storage volume or make one available to IBM MQ.
  3. Restart the queue manager.
- All available space on the DASD volumes to which the archive data set is being written has been exhausted. This condition is normally accompanied by z/OS message IEC032I.

  To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for IBM MQ.
- The primary space allocation for the archive log data set (as specified in the CSQ6ARVP system parameters) is too large to allocate to any available online DASD device. This condition is normally accompanied by z/OS message IEC032I.

  To solve the problem, make more space available on the DASD volumes, or make another online storage volume available for IBM MQ. If this is not possible, you must adjust the value of PRIQTY in the CSQ6ARVP system parameters to reduce the primary allocation. (For details, see Using CSQ6ARVP.)

  Note: If you reduce the primary allocation, you might have to increase the size of the secondary space allocation to avoid future abends.

Operator action
None.

Read I/O errors on the archive data set while IBM MQ is restarting

Symptoms
No specific IBM MQ message is issued; only the z/OS error recovery program message appears.

System action
This depends on whether a second copy exists:

- If a second copy exists, it is allocated and used.
- If a second copy does not exist, restart is not successful.

System programmer action
None.

Operator action
Try to restart, using a different drive.

BSDS problems

Use this topic to investigate, and resolve problems with BSDS.

For background information about the bootstrap data set (BSDS), see the Planning your IBM MQ environment on z/OS .

This topic describes the following BSDS problems:

- "Error occurs while opening the BSDS" on page 524
- "Log content does not agree with the BSDS information" on page 525
- "Both copies of the BSDS are damaged" on page 525
- "Unequal time stamps" on page 526
- "Out of synchronization" on page 527
- "I/O error" on page 527
- "Log range problems" on page 528

Normally, there are two copies of the BSDS, but if one is damaged, IBM MQ immediately changes to single BSDS mode. However, the damaged copy of the BSDS must be recovered before restart. If you are in single mode and damage the only copy of the BSDS, or if you are in dual mode and damage both copies, use the procedure described in Recovering the BSDS.

This section covers some of the BSDS problems that can occur at startup. Problems *not* covered here include:

- RECOVER BSDS command errors (messages CSQJ301E - CSQJ307I)
- Change log inventory utility errors (message CSQJ123E)
- Errors in the BSDS backup being dumped by offload processing (message CSQJ125E)

Error occurs while opening the BSDS

Symptoms

IBM MQ issues the following message:

```
CSQJ100E +CSQ1 ERROR OPENING BSDSn DSNAME=..., ERROR STATUS=eeii
```

where *eeii* is the VSAM return code. For information about VSAM codes, see the *DFSMS/MVS™ Macro Instructions for Data Sets* documentation.

System action

During system initialization, the startup is terminated.

During a RECOVER BSDS command, the system continues in single BSDS mode.

System programmer action

None.

Operator action

Carry out these steps:

1. Run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.
2. Rename the data set that had the problem, and define a replacement for it.
3. Copy the accurate data set to the replacement data set, using Access Method Services.
4. Restart the queue manager.

Log content does not agree with the BSDS information

Symptoms

IBM MQ issues the following message:

```
CSQJ102E +CSQ1 LOG RBA CONTENT OF LOG DATA SET DSNAME=...,
           STARTRBA=..., ENDRBA=...,
           DOES NOT AGREE WITH BSDS INFORMATION
```

This message indicates that the change log inventory utility was used incorrectly or that a down-level data set is being used.

System action

Queue manager startup processing is terminated.

System programmer action

None.

Operator action

Run the print log map utility and the change log inventory utility to print and correct the contents of the BSDS.

Both copies of the BSDS are damaged

Symptoms

IBM MQ issues the following messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
           DSNAME=... ERROR STATUS=0874
CSQJ117E +CSQ1 REG8 INITIALIZATION ERROR READING BSDS
           DSNAME=... ERROR STATUS=0874
CSQJ119E +CSQ1 BOOTSTRAP ACCESS INITIALIZATION PROCESSING FAILED
```

System action

Queue manager startup processing is terminated.

System programmer action

Carry out these steps:

1. Rename the data set, and define a replacement for it.

2. Locate the BSDS associated with the most recent archive log data set, and copy it to the replacement data set.
3. Use the print log map utility to print the contents of the replacement BSDS.
4. Use the print log records utility to print a summary report of the active log data sets missing from the replacement BSDS, and to establish the RBA range.
5. Use the change log inventory utility to update the missing active log data set inventory in the replacement BSDS.
6. If dual BSDS data sets had been in use, copy the updated BSDS to the second copy of the BSDS.
7. Restart the queue manager.

Operator action
None.

Unequal time stamps

Symptoms
IBM MQ issues the following message:

```
CSQJ120E +CSQ1 DUAL BSDS DATA SETS HAVE UNEQUAL TIME STAMPS,
           SYSTEM BSDS1=...,BSDS2=...,
           UTILITY BSDS1=...,BSDS2=...
```

The possible causes are:

- One copy of the BSDS has been restored. All information about the restored BSDS is down-level. The down-level BSDS has the lower time stamp.
- One of the volumes containing the BSDS has been restored. All information about the restored volume is down-level. If the volume contains any active log data sets or IBM MQ data, they are also down-level. The down-level volume has the lower time stamp.
- Dual logging has degraded to single logging, and you are trying to start without recovering the damaged log.
- The queue manager terminated abnormally after updating one copy of the BSDS but before updating the second copy.

System action
IBM MQ attempts to resynchronize the BSDS data sets using the more recent copy. If this fails, queue manager startup is terminated.

System programmer action
None.

Operator action
If automatic resynchronization fails, carry out these steps:

1. Run the print log map utility on both copies of the BSDS, compare the lists to determine which copy is accurate or current.
2. Rename the down-level data set and define a replacement for it.
3. Copy the good data set to the replacement data set, using Access Method Services.
4. If applicable, determine whether the volume containing the down-level BSDS has been restored. If it has been restored, all data on that volume, such as the active log data, is also down-level.

   If the restored volume contains active log data and you were using dual active logs on separate volumes, you need to copy the current version of the active log to the down-level log data set. See Recovering logs for details of how to do this.

Out of synchronization

Symptoms
IBM MQ issues the following message during queue manager initialization:

```
CSQJ122E +CSQ1 DUAL BSDS DATA SETS ARE OUT OF SYNCHRONIZATION
```

The system time stamps of the two data sets are identical. Differences can exist if operator errors occurred while the change log inventory utility was being used. (For example, the change log inventory utility was only run on one copy.) The change log inventory utility sets a private time stamp in the BSDS control record when it starts, and a close flag when it ends. IBM MQ checks the change log inventory utility time stamps and, if they are different, or they are the same but one close flag is not set, IBM MQ compares the copies of the BSDSs. If the copies are different, CSQJ122E is issued.

This message is also issued by the BSDS conversion utility if two input BSDS are specified and a record is found that differs between the two BSDS copies. This situation can arise if the queue manager terminated abnormally prior to the BSDS conversion utility being run.

System action
Queue manager startup or the utility is terminated.

System programmer action
None.

Operator action

If the error occurred during queue manager initialization, carry out these steps:

1. Run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.
2. Rename the data set that had the problem, and define a replacement for it.
3. Copy the accurate data set to the replacement data set, using access method services.
4. Restart the queue manager.

If the error occurred when running the BSDS conversion utility, carry out these steps:

1. Attempt to restart the queue manager and shut it down cleanly before attempting to run the BSDS conversion utility again.
2. If this does not solve the problem, run the print log map utility on both copies of the BSDS, and compare the lists to determine which copy is accurate or current.
3. Change the JCL used to invoke the BSDS conversion utility to specify the current BSDS in the SYSUT1 DD statement, and remove the SYSUT2 DD statement, before submitting the job again.

I/O error

Symptoms
IBM MQ changes to single BSDS mode and issues the user message:

```
CSQJ126E +CSQ1 BSDS ERROR FORCED SINGLE BSDS MODE
```

This is followed by one of the following messages:

```
CSQJ107E +CSQ1 READ ERROR ON BSDS
           DSNAME=... ERROR STATUS=...

CSQJ108E +CSQ1 WRITE ERROR ON BSDS
           DSNAME=... ERROR STATUS=...
```

System action
The BSDS mode changes from dual to single.

System programmer action
None.

Operator action
Carry out these steps:

1. Use Access Method Services to rename or delete the damaged BSDS and to define a new BSDS with the same name as the BSDS that had the error. Example control statements can be found in job CSQ4BREC in thlqual.SCSQPROC.
2. Issue the IBM MQ command RECOVER BSDS to make a copy of the good BSDS in the newly allocated data set and reinstate dual BSDS mode. See also Recovering the BSDS.

Log range problems

Symptoms

IBM MQ has issued message CSQJ113E when reading its own log, or message CSQJ133E or CSQJ134E when reading the log of a queue manager in the queue-sharing group. This can happen when you do not have the archive logs needed to restart the queue manager or recover a CF structure.

System action

Depending upon what log record is being read and why, the requestor might end abnormally with a reason code of `X'00D1032A'`.

System programmer action

Run the print log map utility (CSQJU004) to determine the cause of the error. When message CSQJ133E or CSQJ134E has been issued, run the utility against the BSDS of the queue manager indicated in the message.

If you have:

- Deleted the entry with the log range (containing the log RBA or LRSN indicated in the message) from the BSDS, and
- Not deleted or reused the data set

you can add the entry back into the BSDS using the following procedure:

1. Identify the data set containing the required RBA or LRSN, by looking at an old copy of the contents of BSDS, or by running CSQJU004 against a backup of the BSDS.
2. Add the data set back into the BSDS using the change log inventory utility (CSQJU003).
3. Restart the queue manager.

If an archive log data set has been deleted, you will not be able to recover the page set or CF structure that needs the archive logs. Identify the reason that the queue manager needs to read the log record, then take one of the following actions depending on the page set or CF structure affected.

Page sets

Message CSQJ113E during the recovery phase of queue manager restart indicates that the log is needed to perform media recovery to bring a page set up to date.

Identify the page sets that need the deleted log data set for media recovery, by looking at the media recovery RBA in the CSQI1049I message issued for each page set during queue manager restart, then perform the following actions.

- **Page set zero**

  You can recover the objects on page set zero, by using the following procedure.

  Attention: All data in all other page sets will be lost when you carry out the procedure.

  1. Use function SDEFS of the CSQUTIL utility to produce a file of IBM MQ DEFINE commands.
  2. Format page set zero using CSQUTIL, then redefine the other page sets as described in the next section.
  3. Restart the queue manager.
  4. Use CSQUTIL to redefine the objects using the DEFINE commands produced by the utility in step 1.
- **Page sets 1-99**

  Use the following procedure to redefine the page sets.

  Attention: Any data on the page set is lost when you carry out this operation.

  1. If you can access the page set without any I/O errors, reformat the page set using the CSQUTIL utility with the command FORMAT TYPE(NEW).
  2. If I/O errors occurred when accessing the page set, delete the page set and recreate it.

     If you want the page set to be the same size as before, use the command LISTCAT ENT(*dsname*) ALLOC to obtain the existing space allocations, and use these in the z/OS DEFINE CLUSTER command.

     Format the new page set using the CSQUTIL utility with the command FORMAT TYPE(NEW).
  3. Restart the queue manager. You might have to take certain actions, such as resetting channels or resolving indoubt channels.

CF structures

Messages CSQJ113E, CSQJ133E, or CSQJ134E, during the recovery of a CF structure, indicate that the logs needed to recover the structure are not available on at least one member of the queue-sharing group.

Take one of the following actions depending on the structure affected:

Application CF structure
: Issue the command RECOVER CFSTRUCT(*structure-name*) TYPE(PURGE).

  This process empties the structure, so any messages on the structure are lost.

CSQSYSAPPL structure
: Contact your IBM support center.

Administration structure
: This structure is rebuilt using log data since the last checkpoint on each queue manager, which should be in active logs.

  If you get this error during administration structure recovery, contact your IBM support center as this indicates that the active log is not available.

Once you have recovered the page set or CF structure, perform a backup of the logs, BSDS, page sets, and CF structures.

To prevent this problem from occurring again, increase the:

- Archive log retention (ARCRETN) value to be longer, and
- Increase the frequency of the CF structure backups.

Recovering a CF structure

Conceptually, the data from the previously backed up CF structure is read from the IBM MQ log; the log is read forwards from the backup and any changes are reapplied to the restored structure.

About this task

The log range to use is found from the latest backup of each structure to be recovered, to the current time. The log range is identified by log range sequence number (LRSN) values.

A LRSN uses the 6 most significant digits of a 'store clock value'.

Note that the whole log (back to the time the structure was created) is read, if you have not done a backup of the structure.

Procedure

1. Check that the logs from each queue manager in the queue-sharing group (QSG) are read for records in this LSRN range.

   Note that the logs are read backwards.
2. Check that a list of changes for each structure to be recovered is built.
3. Data from the coupling facility (CF) structure backup is read and the data is restored.

   For example, if the backup was done on queue manager A, and the recovery is running on queue manager B, queue manager B reads the logs from queue manager A to restore the structure.

   When the start of the backup of the CF structure is read, an internal task is started to take the restored data for the structure and merge it with the changes read from the log.
4. Check that processing continues for each structure being restored.

Example

In the following example, the command RECOVER CFSTRUCT(APP3) has been issued, and the following messages produced:

```
04:00:00 CSQE132I CDL2 CSQERRPB Structure recovery started, using log range from
LRSN=CC56D01026CC
to LRSN=CC56DC368924
This is the start of reading the logs backwards from each qmgr in the QSG from the time
of failure to the to the structure backup. The LRSN values give the ranges being used.
Log records for all structures (just one structure in this example) being recovered are
processed at the same time.

04:02:00 CSQE133I CDL2 CSQERPLS Structure recovery reading log backwards, LRSN=CC56D0414372
This message is produced periodically to show the process

04:02:22 CSQE134I CDL2 CSQERRPB Structure recovery reading log completed
The above process of replaying the logs backwards has finished,

04:02:22 CSQE130I CDL2 CSQERCF2 Recovery of structure APP3 started, using CDL1 log range
from RBA=000EE86D902E to RBA=000EF5E8E4DC
The task to process the data for APP3 has been started. The last backup of CF structure
APP3 was done on CDL1 within the given RBA range, so this log range has to be read.

04:02:29 CSQE131I CDL2 CSQERCF2 Recovery of structure APP3 completed
The data merge has completed. The structure is recovered.
```

Page set problems

Use this topic to investigate, and resolve problems with the page sets.

This topic covers the problems that you might encounter with page sets:

• "Page set I/O errors" on page 531 describes what happens if a page set is damaged.

- “Page set full” on page 531 describes what happens if there is not enough space on the page set for any more MQI operations.

Page set I/O errors

Problem
A page set has an I/O error.

Symptoms
This message is issued:

```
CSQP004E +CSQ1 csect-name I/O ERROR STATUS ret-code
PSID psid RBA rba
```

System action
The queue manager terminates abnormally.

System programmer action
None.

Operator action
Repair the I/O error cause.

If none of the page sets are damaged, restart the queue manager. IBM MQ automatically restores the page set to a consistent state from the logs.

If one or more page sets are damaged:

1. Rename the damaged page sets and define replacement page sets.
2. Copy the most recent backup page sets to the replacement page sets.
3. Restart the queue manager. IBM MQ automatically applies any updates that are necessary from the logs.

You cannot restart the queue manager if page set zero is not available. If one of the other page sets is not available, you can comment out the page set DD statement in the queue manager start-up JCL procedure. This lets you defer recovery of the defective page set, enabling other users to continue accessing IBM MQ.

When you add the page set back to the JCL procedure, system restart reads the log from the point where the page set was removed from the JCL to the end of the log. This procedure might take a long time if a large amount of data has been logged.

A reason code of MQRC_PAGESET_ERROR is returned to any application that tries to access a queue defined on a page set that is not available.

When you have restored the defective page set, restore its associated DD statement and restart the queue manager.

The operator actions described here are only possible if all log data sets are available. If your log data sets are lost or damaged, see Restarting if you have lost your log data sets.

Page set full

Problem
There is not enough space on a page set for one of the following:

- `MQPUT` or `MQPUT1` calls to be completed
- Object manipulation commands to be completed (for example, DEFINE QLOCAL)
- `MQOPEN` calls for dynamic queues to be completed

Symptoms

The request fails with reason code MQRC_STORAGE_MEDIUM_FULL. The queue manager cannot complete the request because there is not enough space remaining on the page set.

Reason code MQRC_STORAGE_MEDIUM_FULL can occur even when the page set expand attribute is set to EXPAND(USER). Before the reason code MQRC_STORAGE_MEDIUM_FULL is returned to the application code, the queue manager will attempt to expand the page set and retry the API request. On a heavily loaded system it is possible that the expanded storage can be used by other IO operations before the retry of the API. See Managing page sets.

The cause of this problem could be messages accumulating on a transmission queue because they cannot be sent to another system.

System action

Further requests that use this page set are blocked until enough messages are removed or objects deleted to make room for the new incoming requests.

Operator action

Use the IBM MQ command DISPLAY USAGE PSID(*) to identify which page set is full.

System programmer action

You can either enlarge the page set involved or reduce the loading on that page set by moving queues to another page set. See Managing page sets for more information about these tasks. If the cause of the problem is messages accumulating on the transmission queue, consider starting distributed queuing to transmit the messages.

Coupling facility and Db2 problems

Use this topic to investigate, and resolve problems with the coupling facility, and Db2.

This section covers the problems that you might encounter with the coupling facility and Db2:

- "Storage medium full" on page 532
- "A Db2 system fails" on page 533
- "A Db2 data-sharing group fails" on page 533
- "Db2 and the coupling facility fail" on page 534

Storage medium full

Problem

A coupling facility structure is full.

Symptoms

If a queue structure becomes full, return code MQRC_STORAGE_MEDIUM_FULL is returned to the application.

If the administration structure becomes full, the exact symptoms depend on which processes experience the error, they might range from no responses to CMDSCOPE(GROUP) commands, to queue manager failure as a result of problems during commit processing.

System programmer action

You can use IBM MQ to inhibit MQPUT operations to some of the queues in the structure to prevent applications from writing more messages, start more applications to get messages from the queues, or quiesce some of the applications that are putting messages to the queue.

Alternatively you can use XES facilities to alter the structure size in place. The following z/OS command alters the size of the structure:

```
SETXCF START,ALTER,STRNAME= structure-name,SIZE= newsize
```

where *newsize* is a value that is less than the value of MAXSIZE specified on the CFRM policy for the structure, but greater than the current coupling facility size.

You can monitor the utilization of a coupling facility structure with the DISPLAY CFSTATUS command.

A Db2 system fails

If a Db2 subsystem that IBM MQ is connected to fails, IBM MQ attempts to reconnect to the subsystem, and continue working. If you specified a Db2 group attach name in the QSGDATA parameter of the CSQ6SYSP system parameter module, IBM MQ reconnects to another active Db2 that is a member of the same data-sharing group as the failed Db2, if one is available on the same z/OS image.

There are some queue manager operations that do not work while IBM MQ is not connected to Db2. These are:

- Deleting a shared queue or group object definition.
- Altering, or issuing `MQSET` on, a shared queue or group object definition. The restriction of `MQSET` on shared queues means that operations such as triggering or the generation of performance events do not work correctly.
- Defining new shared queues or group objects.
- Displaying shared queues or group objects.
- Starting, stopping, or other actions for shared channels.
- Reading the shared queue definition from Db2 the first time that the shared queue is open by issuing an MQOPEN.

Other IBM MQ API operations continue to function as normal for shared queues, and all IBM MQ operations can be performed against the queue manager private versions (COPY objects) built from GROUP objects. Similarly, any shared channels that are running continue normally until they end or have an error, when they go into retry state.

When IBM MQ reconnects to Db2, resynchronization is performed between the queue manager and Db2. This involves notifying the queue manager of new objects that have been defined in Db2 while it was disconnected (other queue managers might have been able to continue working as normal on other z/OS images through other Db2 subsystems), and updating object attributes of shared queues that have changed in Db2. Any shared channels in retry state are recovered.

If a Db2 fails, it might have owned locks on Db2 resources at the time of failure. In some cases, this might make certain IBM MQ objects unavailable to other queue managers that are not otherwise affected. To resolve this, restart the failed Db2 so that it can perform recovery processing and release the locks.

A Db2 data-sharing group fails

If an entire Db2 data-sharing group fails, recovery might be to the time of failure, or to a previous point in time.

In the case of recovery to the point of failure, IBM MQ reconnects when Db2 has been recovered, the resynchronization process takes places, and normal queue manager function is resumed.

However, if Db2 is recovered to a previous point in time, there might be inconsistencies between the actual queues in the coupling facility structures and the Db2 view of those queues. For example, at the point in time Db2 is recovered to, a queue existed that has since been deleted and its location in the coupling facility structure reused by the definition of a new queue that now contains messages.

If you find yourself in this situation, you must stop all the queue managers in the queue-sharing group, clear out the coupling facility structures, and restart the queue managers. You must then use IBM MQ commands to define any missing objects. To do this, use the following procedure:

1. Prevent IBM MQ from reconnecting to Db2 by starting Db2 in utility mode, or by altering security profiles.

2. If you have any important messages on shared queues, you might be able to offload them using the COPY function of the CSQUTIL utility program, but this might not work.
3. Terminate all queue managers.
4. Use the following z/OS command to clear all structures:

```
SETXCF FORCE,STRUCTURE,STRNAME=
```

5. Restore Db2 to a historical point in time.
6. Reestablish queue manager access to Db2.
7. Restart the queue managers.
8. Recover the IBM MQ definitions from backup copies.
9. Reload any offloaded messages to the shared queues.

When the queue managers restart, they attempt to resynchronize local COPY objects with the Db2 GROUP objects. This might cause IBM MQ to attempt to do the following:

- Create COPY objects for old GROUP objects that existed at the point in time Db2 has recovered to.
- Delete COPY objects for GROUP objects that were created since the point in time Db2 has recovered to and so do not exist in the database.

The DELETE of COPY objects is attempted with the NOPURGE option, so it fails for queue managers that still have messages on these COPY queues.

Db2 and the coupling facility fail

If the coupling facility fails, the queue manager might fail, and Db2 will also fail if it is using this coupling facility.

Recover Db2 using Db2 recovery procedures. When Db2 has been restarted, you can restart the queue managers. The CF administration structure will also have failed, but this is rebuilt by restarting all the queue managers within the queue-sharing group.

If a single application structure within the coupling facility suffers a failure, the effect on the queue manager depends on the level of the queue manager and the CFLEVEL of the failed CF structure:

- If the CF application structure is CFLEVEL(3) or higher and RECOVER is set to YES, it will not be usable until you recover the CF structure by issuing an MQSC RECOVER CFSTRUCT command to the queue manager that will do the recovery. You can specify a single CF structure to be recovered, or you can recover several CF structures simultaneously. The queue manager performing the recovery locates the relevant backups on all the other queue managers' logs using the data in Db2 and the bootstrap data sets. The queue manager replays these backups in the correct time sequence across the queue sharing group, from just before the last backup through to the point of failure. If a recoverable application structure has failed, any further application activity is prevented until the structure has been recovered. If the administration structure has also failed, all the queue managers in the queue-sharing group must be started before the RECOVER CFSTRUCT command can be issued. All queue managers can continue working with local queues and queues in other CF structures during recovery of a failed CF structure.
- If the CF application structure is CFLEVEL(3) or higher and RECOVER is set to NO, the structure is automatically reallocated by the next `MQOPEN` request performed on a queue defined in the structure. All messages are lost, as the structure can only contain non-persistent messages.
- If the CF application structure has a CFLEVEL less than 3, the queue manager fails. On queue manager restart, peer recovery attempts to connect to the structure, detect that the structure has failed and allocate a new version of the structure. All messages on shared queues that were in CF structures affected by the coupling facility failure are lost.

Since IBM WebSphere MQ 7.1, queue managers in queue-sharing-groups have been able to tolerate loss of connectivity to coupling facility structures without failing. If the structure has experienced a connection failure, attempts are made to rebuild the structure in another coupling facility with better connectivity in order to regain access to shared queues as soon as possible.

Problems with long-running units of work

Use this topic to investigate, and resolve problems with long-running units of work.

This topic explains what to do if you encounter a long-running unit of work during restart. In this context, this means a unit of work that has been active for a long time (possibly days or even weeks) so that the origin RBA of the unit of work is outside the scope of the current active logs. This means that restart could take a long time, because all the log records relating to the unit of work have to be read, which might involve reading archive logs.

Old unit of work found during restart

Problem
A unit of work with an origin RBA that predates the oldest active log has been detected during restart.

Symptoms
IBM MQ issues the following message:

```
CSQR020I +CSQ1 OLD UOW FOUND
```

System action
Information about the unit of work is displayed, and message CSQR021D is issued, requesting a response from the operator.

System programmer action
None.

Operator action
Decide whether to commit the unit of work or not. If you choose not to commit the unit of work, it is handled by normal restart recovery processing. Because the unit of work is old, this is likely to involve using the archive log, and so takes longer to complete.

IMS-related problems

Use this topic to investigate, and resolve problems with IMS and IBM MQ..

This topic includes plans for the following problems that you might encounter in the IMS environment:

- “IMS cannot connect to IBM MQ” on page 535
- “IMS application problem” on page 536
- “IMS is not operational” on page 536

IMS cannot connect to IBM MQ

Problem
The IMS adapter cannot connect to IBM MQ.

Symptoms
IMS remains operative. The IMS adapter issues these messages for control region connect:

- CSQQ001I
- CSQQ002E
- CSQQ003E

- CSQQ004E
- CSQQ005E
- CSQQ007E

For details, see the IBM MQ for z/OS messages, completion, and reason codes documentation.

If an IMS application program tries to access IBM MQ while the IMS adapter cannot connect, it can either receive a completion code and reason code, or terminate abnormally. This depends on the value of the REO option in the SSM member of IMS PROCLIB.

System action
All connection errors are also reported in the IMS message DFS3611.

System programmer action
None.

Operator action
Analyze and correct the problem, then restart the connection with the IMS command:

```
/START SUBSYS subsysname
```

IMS requests the adapter to resolve in-doubt units of recovery.

IMS application problem

Problem
An IMS application terminates abnormally.

Symptoms
The following message is sent to the user's terminal:

```
DFS555I  TRANSACTION tran-id ABEND abcode
MSG IN PROCESS: message data:
```

where *tran-id* represents any IMS transaction that is terminating abnormally and *abcode* is the abend code.

System action
IMS requests the adapter to resolve the unit of recovery. IMS remains connected to IBM MQ.

System programmer action
None.

Operator action
As indicated in message DFS554A on the IMS master terminal.

IMS is not operational

Problem
IMS is not operational.

Symptoms
More than one symptom is possible:

- IMS waits or loops

IBM MQ cannot detect a wait or loop in IMS, so you must find the origin of the wait or loop. This can be IMS, IMS applications, or the IMS adapter.

- IMS terminates abnormally.
  - See the manuals *IMS/ESA Messages and Codes* and *IMS/ESA Failure Analysis Structure Tables* for more information.
  - If threads are connected to IBM MQ when IMS terminates, IBM MQ issues message CSQ3201E. This message indicates that IBM MQ end-of-task (EOT) routines have been run to clean up and disconnect any connected threads.

System action

IBM MQ detects the IMS error and:

- Backs out in-flight work.
- Saves in-doubt units of recovery to be resolved when IMS is reconnected.

System programmer action

None.

Operator action

Resolve and correct the problem that caused IMS to terminate abnormally, then carry out an emergency restart of IMS. The emergency restart:

- Backs out in-flight transactions that changed IMS resources.
- Remembers the transactions with access to IBM MQ that might be in doubt.

You might need to restart the connection to IBM MQ with the IMS command:

```
/START SUBSYS subsysname
```

During startup, IMS requests the adapter to resolve in-doubt units of recovery.

Hardware problems

Use this topic as a starting point to investigate hardware problems.

If a hardware error causes data to be unreadable, IBM MQ can still be recovered by using the *media recovery* technique:

1. To recover the data, you need a backup copy of the data. Use DFDSS or Access Method Services REPRO regularly to make a copy of your data.
2. Reinstate the most recent backup copy.
3. Restart the queue manager.

The more recent your backup copy, the more quickly your subsystem can be made available again.

When the queue manager restarts, it uses the archive logs to reinstate changes made since the backup copy was taken. You must keep sufficient archive logs to enable IBM MQ to reinstate the changes fully. Do not delete archive logs until there is a backup copy that includes all the changes in the log.

Notices

Programming interface information

Programming interface information, if provided, is intended to help you create application software for use with this program.

This book contains information on intended programming interfaces that allow the customer to write programs to obtain the services of WebSphere MQ.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Important: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks

Part Number:

(1P) P/N: