10.0

Troubleshooting and Support for IBM MQ

Note

Before using this information and the product it supports, read the information in “Notices” on page 549.

This edition applies to version 9 release 4 of IBM® MQ and to all subsequent releases and modifications until otherwise indicated in new editions.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2007, 2026.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Troubleshooting and support..... 5
Making initial checks.....6
Making initial checks on AIX..... 7
Making initial checks on IBM i.....19
Making initial checks on Linux..... 28
Making initial checks on Windows..... 41
Making initial checks on z/OS..... 50
Detailed troubleshooting..... 63
Troubleshooting AMQP problems.....64
Troubleshooting AMS problems.....69
Troubleshooting command problems.....70
Troubleshooting distributed publish/subscribe problems.....74
Troubleshooting distributed queue management problems..... 78
Troubleshooting IBM MQ Console and REST API problems..... 89
Troubleshooting IBM MQ Internet Pass-Thru problems..... 92
Troubleshooting IBM MQ MQI client application problems.....95
Troubleshooting IBM MQ .NET problems..... 98
Troubleshooting Java and JMS problems..... 100
Troubleshooting Managed File Transfer problems..... 127
Troubleshooting message problems..... 177
Troubleshooting IBM MQ Telemetry problems..... 178
Troubleshooting multicast problems.....192
Troubleshooting queue manager problems..... 195
Troubleshooting queue manager cluster problems.....195
Troubleshooting RDQM configuration problems.....218
Troubleshooting Native HA and Native HA CRR configurations on Linux.....233
Troubleshooting security problems.....236
Troubleshooting WCF custom channel for IBM MQ problems.....250
Troubleshooting XMS .NET problems.....252
Troubleshooting IBM MQ for z/OS problems..... 255
Collecting troubleshooting information..... 310
Collecting troubleshooting information on Multiplatforms.....310
Collecting troubleshooting information on z/OS..... 361
Sending troubleshooting information to IBM.....404
Using error logs.....405
Error logs on AIX, Linux, and Windows.....407
Error logs on IBM i.....410
Error logs on z/OS.....413
Error logs in IBM MQ classes for JMS.....413
Suppressing channel error messages from error logs on Multiplatforms..... 413
First Failure Support Technology (FFST).....414
FFST: IBM MQ for AIX or Linux..... 415
FFST: IBM MQ for IBM i.....416
FFST: IBM MQ for Windows..... 418
FFST: IBM MQ classes for JMS..... 420
FFST: WCF XMS First Failure Support Technology.....425
FFDC configuration for XMS .NET applications..... 426
Tracing.....426
Tracing on AIX and Linux..... 427
Tracing on IBM i.....433
Tracing on Windows..... 443
Tracing on z/OS..... 447

Tracing the Advanced Message Queuing Protocol (AMQP) Service....464
Tracing the IBM MQ Console....466
Tracing errors in IBM MQ Internet Pass-Thru....469
Tracing IBM MQ .NET applications....471
Tracing JMS/Jakarta Messaging and Java applications....476
Tracing Managed File Transfer resources on Multiplatforms....490
Tracing Managed File Transfer for z/OS resources....496
Tracing the telemetry (MQXR) service....511
Tracing the REST API....513
Tracing **runmqakm**....514
Tracing the WCF custom channel for IBM MQ....514
Tracing XMS .NET applications....515
Enabling dynamic tracing of LDAP client library code....522
Recovering after failure....523
Disk drive failures....524
Damaged queue manager object....525
Damaged single object....525
Automatic media recovery failure....525
Example recovery procedures on z/OS....525

Notices....549
Programming interface information....550
Trademarks....550

IBM MQ troubleshooting and support

If you are having problems with your queue manager network or IBM MQ applications, you can use the information in this section to help you diagnose and either solve the problem yourself or obtain help to resolve it.

About this task

Troubleshooting is the process of finding and eliminating the cause of a problem. If you have a problem with your IBM software, the troubleshooting process for that problem 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
2. Re-creating the problem
3. Eliminating possible causes

If you need assistance with a problem that you are having with IBM MQ, you can obtain help through the IBM Support Site. You can also subscribe to notifications about IBM MQ fixes, troubleshooting, and other news. For more information, see “Collecting troubleshooting information” on page 310.

For more information about recovering after a problem, see “Recovering after failure” on page 523.

Procedure

1. Record 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 include:

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

   For more information, see the following topics:

   - “Using error logs” on page 405
   - “First Failure Support Technology (FFST)” on page 414
   - “Tracing” on page 426

   If an IBM MQ component or command has returned an error, and you want more information about a message that is written to the screen or the log, see Messages and reason codes.
2. Re-create the problem.

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

   - How did you first notice the problem?

© Copyright IBM Corp. 2007, 2026

- 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 that were occurring at around the same time?
- Does the same problem occur elsewhere? Does only one machine have the problem or do multiple machines have the same problem?
- What messages are being generated that might indicate what the problem is?

For more information about these types of question, see “Making initial checks” on page 6 and “Detailed troubleshooting” on page 63.

3. Eliminate 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 documentation and other available resources to help you with your elimination process. Has anyone else experienced this problem? Is there a fix that you can download? For more information, see “Collecting troubleshooting information” on page 310.

Making initial checks

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

About this task

Use the information and general advice given in the subtopics to help you to carry out the initial checks for your platform and rectify the problem.

Procedure

- Carry out the initial checks for your platform:
  - AIX “Making initial checks on AIX” on page 7
  - IBM i “Making initial checks on IBM i” on page 19
  - Linux “Making initial checks on Linux” on page 28
  - Windows “Making initial checks on Windows” on page 41
  - z/OS “Making initial checks on z/OS” on page 50

Tips for system administrators

- Check the error logs for messages for your operating system:
  - ALW “Error logs on AIX, Linux, and Windows” on page 407
  - IBM i “Error logs on IBM i” on page 410
  - z/OS “Diagnostic information produced on IBM MQ for z/OS” on page 261
- Check the contents of `qm.ini` for any configuration changes or errors.

  For more information on changing configuration information, see:
  - Multi Changing IBM MQ configuration information on Multiplatforms

  - 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 “Tracing” on page 426.

Tips for application developers

- Check the return codes from the MQI calls in your applications.

  For a list of reason codes, see API completion and reason codes. 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 “Tracing” on page 426. For more information about handling errors in MQI applications, see Handling program errors.

Related concepts

“Using error logs” on page 405
There are a variety of error logs that you can use to help with problem determination and troubleshooting.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Tracing” on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

AIX Making initial checks on AIX

Before you start problem determination in detail on AIX, 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.

About this task

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

As you go through the list of initial questions to consider and follow the links to more information, 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.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see “Collecting troubleshooting information” on page 310.

Procedure

1. Identify characteristics of the problem

   There are some initial questions that you can consider to help you to identify the cause of the problem:

   - Has IBM MQ run successfully before?

- Have any changes been made since the last successful run?
- Have you applied any maintenance updates?
- Has the application run successfully before?
- Are you receiving errors when you use special characters in descriptive text for some commands?
- Are there any error messages or return codes to help you to determine the location and cause of your problem?
- Can you reproduce the problem?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Is the problem intermittent?

2. Investigate potential problems with applications, commands and messages

   If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem:

   - Are messages failing to arrive on the queue?
   - Do messages contain unexpected or corrupted information?
   - Are unexpected messages received when using distributed queues?
   - Have you received no response from a PCF command?
   - Are only some of your queues failing?
   - Does the problem affect only remote queues?
   - Is your application or system running slowly?

3. Investigate any problems connected to IBM MQ resources, including resource usage by IBM MQ processes, problems related to insufficient resources, and your resource limit configurations.

   For more information, see "Additional checks for resource problems on AIX" on page 16.

4. If you need further information to help you with identifying the cause of the problem, see "Detailed troubleshooting" on page 63.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Making initial checks on IBM i" on page 19
Before you start problem determination in detail on IBM i, 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 Linux" on page 28
Before you start problem determination in detail on Linux, 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 Windows" on page 41
Before you start problem determination in detail on Windows, 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 50
Before you start problem determination in detail on z/OS, 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 reference

Messages and reason codes

AIX Identifying characteristics of the problem on AIX

Some initial questions to consider to help with identifying the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Has IBM MQ run successfully before?
- Have any changes been made since the last successful run?
- Have you applied any maintenance updates?
- Has the application run successfully before?
- Are you receiving errors when you use special characters in descriptive text for some commands?
- Are there any error messages or return codes to help you to determine the location and cause of your problem?
- Can you reproduce the problem?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Is the problem intermittent?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see “Collecting troubleshooting information” on page 310.

Procedure

1. 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. For more information, see IBM MQ installation overview and Installing and uninstalling IBM MQ on AIX.

   To run the verification procedure, see Verifying an IBM MQ installation on AIX. Also look at Configuring IBM MQ for information about post-installation configuration of IBM MQ.

2. Have any changes been made since the last successful run?

   Changes that have been made to your IBM MQ configuration, or changes to other applications 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 might affect the operation of IBM MQ?

3. 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 IBM Support for any maintenance package errors.
- If a maintenance package has been applied to any other application, consider the effect it might have on the way IBM MQ interfaces with it.

4. Has the application run successfully before?

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

   - 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 on 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. For information about building your application, see Developing applications.

   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? 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.

5. Are you receiving errors when you use special characters in descriptive text for some commands?

   Some characters, for example, backslash (\) and double quote (") characters have special meanings when used with commands.

   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.

6. Are there any error messages or return codes to 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. For information about the locations and contents of the error logs, see "Error logs on AIX, Linux, and Windows" on page 407.

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

7. 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.

8. Does the problem affect specific parts of the network?

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

   You might be able to identify specific parts of the network that are affected by the problem (for example, remote queues). 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.

9. 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.

10. 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.

Related concepts

“Additional checks for resource problems on AIX” on page 16
How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Determining problems with applications, commands and messages on AIX” on page 12
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

Related reference

Messages and reason codes

AIX Determining problems with applications, commands and messages on AIX

If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Are messages failing to arrive on the queue?
- Do messages contain unexpected or corrupted information?
- Are unexpected messages received when using distributed queues?
- Have you received no response from a PCF command?
- Are only some of your queues failing?
- Does the problem affect only remote queues?
- Is your application or system running slowly?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see “Collecting troubleshooting information” on page 310.

Procedure

1. Are messages failing to arrive on the queue?

   If messages do not arrive when you are expecting them, check if 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?

   Also check whether you are 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.

2. 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:

   - 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.

3. Are unexpected messages received 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. For more information, see Using the dead-letter (undelivered message) queue and MQDLH - Dead-letter header.
- 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. For more information, see Distributed queuing and clusters.
- 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. For more information, see Data conversion.

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

4. Have you received no response from a PCF command?

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

   - Is the command server running? Use 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. For more information, see MQDLH - Dead-letter header and Using the dead-letter (undelivered message) queue. 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? For more information, see “Error log directories on AIX, Linux, and Windows” on page 409.
   - 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 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 IBM Support for help.

5. Are only 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.

   Use the MQSC command **DISPLAY QUEUE** to display the information about each queue. If the **CURDEPTH** is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.

   If the **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, or the queue might have been open but is now closed.

   Check the status at the time you expect to put or get a message.

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

6. Does the problem affect 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.

7. 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.

   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.

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

Related concepts

“Additional checks for resource problems on AIX” on page 16
How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Identifying characteristics of the problem on AIX” on page 9
Some initial questions to consider to help with identifying the cause of the problem.

Related reference

Messages and reason codes

AIX Additional checks for resource problems on AIX

How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Useful commands and the configuration file for investigating resource issues

Useful commands that display current values on your system or make a temporary change to the system:

ulimit -a
Display user limits

ulimit -Ha
Display user hard limits

ulimit -Sa
Display user soft limits

ulimit -<paramflag> <value>
Where **paramflag** is the flag for the resource name, for example, **s** for stack.

To make permanent changes to the resource limits on your system use `/etc/security/limits.conf` or `/etc/security/limits`.

Basic checks before tuning IBM MQ or kernel parameters

You need to investigate the following:

- Whether the number of active connections is within the expected limit.

  For example, suppose that your system is tuned to allow 2000 connections when the number of user processes is no greater than 3000. If the number of connections increases to more than 2000, then either the number of user processes has increased to more than 3000 (because new applications have been added), or there is a connection leak.

  To check for these problems use the following commands:

  – AIX Number of IBM MQ processes:

    ```
    ps -elf|egrep "amq|run"|wc -l
    ```

  – Number of connections:

```
echo "dis conn(*) all" | runmqsc <qmgr name>|grep EXTCONN|wc -l
```

  - Shared memory usage:

```
ipcs -ma
```

- If the number of connections is higher than the expected limit, check the source of the connections.
- If the shared memory usage is very high, check the following number of:
  - Topics
  - Open queue handles
- From an IBM MQ perspective, the following resources need to be checked and tuned:
  - Data segment
  - Stack segment
  - File size
  - Open file handles
  - Shared memory limits
  - Thread limits
- Use the mqconfig command to check the current resource usage.

Notes:

1. Some of resources listed in the preceding text need to be tuned at user level and some at the operating system level.
2. The preceding list is not a complete list, but is sufficient for most common resource issues reported by IBM MQ.
3. Linux Tuning is required at thread level, as each thread is a light weight process (LWP).

Problem in creating threads or processes from IBM MQ or an application

Failure in `xcsExecProgram` and `xcsCreateThread`

Probe IDs, error messages, and components

XY348010 from **xtmStartTimerThread** from an IBM MQ process (for example **amqzlaa0**) or an application

XC037008 from **xcsExecProgram** with error code **xecP_E_PROC_LIMIT** from **amqzxma0**

XC035040 **xcsCreateThread**

XC037007 from **xcsExecProgram** with **xecP_E_NO_RESOURCE**

xcsCreateThread fails with **xecP_E_NO_RESOURCE** followed by failure data capture, for example ZL000066 from **zlaMain**

Probe IDs might be different. Check for the error codes **xecP_E_PROC_LIMIT** and **xecP_E_NO_RESOURCE**.

Error messages reporting errno 11 from pthead_create, for example: `AMQ6119S: An internal IBM MQ error has occurred ('11 - Resource temporarily unavailable'` from pthread_create.)

Resolving the problem on AIX

IBM MQ sets the error code **xecP_E_PROC_LIMIT** when `pthread_create` or fork fails with EAGAIN.

EAGAIN

Review and increase the max user processes and stack size user process resource limits.

ENOMEM

IBM MQ sets the error code **xecP_E_NO_RESOURCE** when `pthread_create` or fork fails with ENOMEM.

Check and increase the stack size and data resource limits.

Notes:

- You can increase the user process resource limits by using the **ulimit** command, or by changing the resource limit configuration file.
- The changes using the **ulimit** command are temporary. Modify `/etc/security/limits` or `/etc/security/limits.conf` to make the changes permanent. You must check the actual configuration on your operating system, as the configuration might be different.
- You should also review your OS manuals (for example, the man page for pthread_create) for more details on resource issues and tuning resource limits, and ensure that the resource limits are appropriately configured.
- You should also check if the system is running short of resources, both memory and CPU.

Problems in creating shared memory

Error : shmget fails with error number 28(ENOSPC)

```
| Probe Id          :- XY132002                                                |
| Component         :- xstCreateExtent                                         |
| ProjectID         :- 0                                                       |
| Probe Description :- AMQ6119: An internal IBM MQ error has occurred          |
|   (Failed to get memory segment: shmget(0x00000000, 2547712) [rc=-1          |
|   errno=28] No space left on device)                                         |
| FDCSequenceNumber :- 0                                                       |
| Arith1            :- 18446744073709551615 (0xffffffffffffffff)               |
| Arith2            :- 28 (0x1c)                                               |
| Comment1          :- Failed to get memory segment: shmget(0x00000000,        |
|   2547712) [rc=-1 errno=28] No space left on device                          |
| Comment2          :- No space left on device                                 |
+------------------------------------------------------------------------------+
MQM Function Stack
ExecCtrlrMain?
xcsAllocateMemBlock
xstExtendSet
xstCreateExtent
xcsFFST
```

shmget fails with error number 22(EINVAL)

```
| Operating System  :- SunOS 5.10                                              |
| Probe Id          :- XY132002                                                |
| Application Name  :- MQM                                                     |
| Component         :- xstCreateExtent                                         |
| Program Name      :- amqzxma0                                                |
| Major Errorcode   :- xecP_E_NO_RESOURCE                                      |
| Probe Description :- AMQ6024: Insufficient resources are available to        |
|   complete a system request.                                                 |
| FDCSequenceNumber :- 0                                                       |
| Arith1            :- 18446744073709551615 (0xffffffffffffffff)               |
| Arith2            :- 22 (0x16)                                               |
| Comment1          :- Failed to get memory segment: shmget(0x00000000,        |
|   9904128) [rc=-1 errno=22] Invalid argument                                 |
| Comment2          :- Invalid argument                                        |
| Comment3          :- Configure kernel (for example, shmmax) to allow a       |
|   shared memory segment of at least 9904128
bytes                                                  |
+------------------------------------------------------------------------------+
MQM Function Stack
ExecCtrlrMain
zxcCreateECResources
zutCreateConfig
xcsInitialize
xcsCreateSharedSubpool
xcsCreateSharedMemSet
xstCreateExtent
xcsFFST
```

Unexpected process termination and queue manager crash, or queue manager crash

Process ending unexpectedly followed by FDCs from `amqzxma0`

Example FDC:

```
Date/Time         :- Mon May 02 2016 01:00:58 CEST
Host Name         :- test.ibm.com
LVLS              :- 8.0.0.4
Product Long Name :- IBM MQ for Linux (x86-64 platform)
Probe Id          :- XC723010
Component         :- xprChildTermHandler
Build Date        :- Oct 17 2015
Build Level       :- p800-004-151017
Program Name      :- amqzxma0
Addressing mode   :- 64-bit
Major Errorcode   :- xecP_E_USER_TERM
Minor Errorcode   :- OK
Probe Description :- AMQ6125: An internal IBM MQ error has occurred.
```

Possible Causes and Solutions

- Check if the user has ended any process.
- Check if the IBM MQ process ended because of a memory exception:
  - Did the process end with an FDC of `Component :- xehExceptionHandler`?
  - Apply the fix for known issues corrected in this area.
- Check if the operating system ended the process because of high memory usage by the process:
  - Has the IBM MQ process consumed lot of memory?
  - Has the operating system ended the process?

    Review the operating system log. For example, the OOM-killer on Linux:

    ```
    Jan 2 01:00:57 ibmtest kernel:
    amqrmppa invoked oom-killer: gfp_mask=0x201da, order=0, oom_score_adj=0)
    ```

  - Apply the fix for known memory leak issues.

Difference in user limits used by a process against the configured limits

The user limits used by the process might be different from the configured limits. This is likely to happen if the process is started by a different user, or by user scripts, or a high availability script for example. It is important that you to check the user who is starting the queue manager, and set the appropriate resource limits for this user.

Related tasks

"Identifying characteristics of the problem on AIX" on page 9
Some initial questions to consider to help with identifying the cause of the problem.

"Determining problems with applications, commands and messages on AIX" on page 12
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

IBM i Making initial checks on IBM i

Before you start problem determination in detail on IBM i, 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.

About this task

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

Some preliminary questions for you to consider are listed in the following procedure. If you are able to find the cause of the problem by working through these preliminary checks, you can then, if needed, use the information in other sections of the IBM MQ product documentation, and in the libraries of other licensed programs, to help you resolve the problem.

If you are not able to identify the cause of the problem by carrying out the preliminary checks, and so need to carry out a more detailed investigation there are further questions for you to consider in the subtopics. As you work through the lists of questions, 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.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

- Consider the following questions.

  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 20.

     No
     It is likely that you have not installed or set up IBM MQ correctly. For more information, see IBM MQ installation overview and Installing and uninstalling IBM MQ on IBM i. For information on running the verification procedure, see Verifying an IBM MQ installation on IBM i.

  2. Has the IBM MQ application run successfully before?

     Yes
     Proceed to Step "3" on page 21.

     No
     Consider whether 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.

     Also 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. Has 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 21.

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

   Yes

   Proceed to "Identifying characteristics of the problem on IBM i" 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

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Manually applying required authority for commands and programs" on page 25
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.

"Determining problems with applications, commands and messages on IBM i" on page 26
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you determine the cause of the problem.

"Making initial checks on AIX" on page 7
Before you start problem determination in detail on AIX, 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 Linux" on page 28
Before you start problem determination in detail on Linux, 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 Windows" on page 41
Before you start problem determination in detail on Windows, 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 50
Before you start problem determination in detail on z/OS, 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 reference

Messages and reason codes

IBM i Identifying characteristics of the problem on IBM i

If you have not been able to identify the cause of the problem by using the preliminary checks, you should now start to look at the characteristics of the problem in greater detail.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Can you reproduce the problem?
- Is the problem intermittent?
- Are you receiving errors when you use special characters in commands?
- Does the problem affect specific parts of the network?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Does the problem occur at specific times of the day?
- Have you received no response from a command?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. 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?

2. 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).

3. Are you receiving errors when you use special characters in commands?

   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.

4. 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?

5. 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.

6. 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.

7. 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.)

8. Have you received no 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? For more information, see "Error logs on IBM i" on page 410.
   - 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 tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Manually applying required authority for commands and programs" on page 25
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.

"Determining problems with applications, commands and messages on IBM i" on page 26
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you determine the cause of the problem.

Related reference

Messages and reason codes

IBM i Manually applying required authority for commands and programs

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.

About this task

To enable this reliance, the commands and programs 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 installation process, and you do not need to apply it yourself. However, if you encounter problems, you can set the authorities manually as described in the following steps.

Procedure

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)
   ```

   You can set authorities for the following commands:

   - 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)
   ```

   You can set authorities for the following programs:

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

IBM i Determining problems with applications, commands and messages on IBM i

If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you determine the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Are some of your queues working?
- Does the problem affect only remote queues?
- Are messages failing to arrive on the queue?
- Do messages contain unexpected or corrupted information?
- Are unexpected messages received when using distributed queues?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see “Collecting troubleshooting information” on page 310.

Procedure

1. 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. Display the information about this queue, using **WRKMQMQSTS** or **DSPMQMQ** then 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, it might be 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 IBM Support for help.
2. Does the problem affect only remote queues?

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

   a. Check that the programs that should be putting messages to the remote queues have run successfully.

   b. 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.

c. If necessary, start the channel manually. See Distributed queuing and clusters.

d. Check the channel with a **PING** command.

3. 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 Step “5” on page 28.

4. 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.

5. 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. For information about the dead-letter header structure, see MQDLH - Dead-letter header. See also IBM i Application Programming Reference (ILE/RPG).
   - 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.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Identifying characteristics of the problem on IBM i” on page 22
If you have not been able to identify the cause of the problem by using the preliminary checks, you should now start to look at the characteristics of the problem in greater detail.

Related reference

Messages and reason codes

Linux Making initial checks on Linux

Before you start problem determination in detail on Linux, 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.

About this task

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

As you go through the list of initial questions to consider and follow the links to more information, 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.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. Identify characteristics of the problem

   There are some initial questions that you can consider to help you to identify the cause of the problem:

   - Has IBM MQ run successfully before?
   - Have any changes been made since the last successful run?
   - Have you applied any maintenance updates?
   - Has the application run successfully before?
   - Are you receiving errors when you use special characters in descriptive text for some commands?
   - Are there any error messages or return codes to help you to determine the location and cause of your problem?
   - Can you reproduce the problem?
   - Does the problem affect specific parts of the network?
   - Does the problem occur at specific times of the day?
   - Is the problem intermittent?
2. Investigate potential problems with applications, commands and messages

   If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem:

   - Are messages failing to arrive on the queue?
   - Do messages contain unexpected or corrupted information?
   - Are unexpected messages received when using distributed queues?
   - Have you received no response from a PCF command?
   - Are only some of your queues failing?
   - Does the problem affect only remote queues?
   - Is your application or system running slowly?
3. Investigate any problems connected to IBM MQ resources, including resource usage by IBM MQ processes, problems related to insufficient resources, and your resource limit configurations.

   For more information, see "Additional checks for resource problems on Linux" on page 37.
4. If you need further information to help you with identifying the cause of the problem, see "Detailed troubleshooting" on page 63.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Making initial checks on AIX" on page 7
Before you start problem determination in detail on AIX, 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 19

Before you start problem determination in detail on IBM i, 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 Windows" on page 41
Before you start problem determination in detail on Windows, 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 50
Before you start problem determination in detail on z/OS, 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 reference

Messages and reason codes

Linux Identifying characteristics of the problem on Linux

Some initial questions to consider to help with identifying the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Has IBM MQ run successfully before?
- Have any changes been made since the last successful run?
- Have you applied any maintenance updates?
- Has the application run successfully before?
- Are you receiving errors when you use special characters in descriptive text for some commands?
- Are there any error messages or return codes to help you to determine the location and cause of your problem?
- Can you reproduce the problem?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Is the problem intermittent?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. 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. For more information, see IBM MQ installation overview and Installing and uninstalling IBM MQ on Linux.

   To run the verification procedure, see Verifying an IBM MQ installation on Linux. Also look at Configuring IBM MQ for information about post-installation configuration of IBM MQ.
2. Have any changes been made since the last successful run?

   Changes that have been made to your IBM MQ configuration, or changes to other applications 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 might affect the operation of IBM MQ?

3. 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 IBM Support for any maintenance package errors.
   - If a maintenance package has been applied to any other application, consider the effect it might have on the way IBM MQ interfaces with it.

4. Has the application run successfully before?

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

   - 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 on 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. For information about building your application, see Developing applications.

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? 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.

5. Are you receiving errors when you use special characters in descriptive text for some commands?

   Some characters, for example, backslash (\) and double quote (") characters have special meanings when used with commands.

   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.

6. Are there any error messages or return codes to 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. For information about the locations and contents of the error logs, see “Error logs on AIX, Linux, and Windows” on page 407.

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

7. 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.

8. Does the problem affect specific parts of the network?

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

   You might be able to identify specific parts of the network that are affected by the problem (for example, remote queues). 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.

9. 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.

10. 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.

Related concepts

"Additional checks for resource problems on Linux" on page 37
How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Determining problems with applications, commands and messages on Linux" on page 33
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

Related reference

Messages and reason codes

Linux Determining problems with applications, commands and messages on Linux

If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Are messages failing to arrive on the queue?
- Do messages contain unexpected or corrupted information?
- Are unexpected messages received when using distributed queues?
- Have you received no response from a PCF command?
- Are only some of your queues failing?
- Does the problem affect only remote queues?
- Is your application or system running slowly?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. Are messages failing to arrive on the queue?

   If messages do not arrive when you are expecting them, check if 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?

   Also check whether you are 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.

2. 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:

- 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.

3. Are unexpected messages received 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. For more information, see Using the dead-letter (undelivered message) queue and MQDLH - Dead-letter header.
   - 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. For more information, see Distributed queuing and clusters.
   - 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. For more information, see Data conversion.

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

4. Have you received no response from a PCF command?

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

   - Is the command server running? Use 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. For more information, see MQDLH - Dead-letter header and Using the dead-letter (undelivered message) queue. 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? For more information, see "Error log directories on AIX, Linux, and Windows" on page 409.
- 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 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 IBM Support for help.

5. Are only 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.

   Use the MQSC command **DISPLAY QUEUE** to display the information about each queue. If the **CURDEPTH** is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.

   If the **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, or the queue might have been open but is now closed.

   Check the status at the time you expect to put or get a message.

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

6. Does the problem affect 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.

7. 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.

   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.

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

Related concepts

“Additional checks for resource problems on Linux” on page 37
How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Related tasks

“Identifying characteristics of the problem on Linux” on page 30
Some initial questions to consider to help with identifying the cause of the problem.

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Related reference

Messages and reason codes

Linux Additional checks for resource problems on Linux

How you determine and resolve problems connected to IBM MQ resources, including resource usage by IBM MQ processes, determining and resolving problems related to insufficient resources, and your resource limit configurations.

Useful commands and the configuration file for investigating resource issues

Useful commands that display current values on your system or make a temporary change to the system:

ulimit -a
Display user limits

ulimit -Ha
Display user hard limits

ulimit -Sa
Display user soft limits

ulimit -<paramflag> <value>
Where **paramflag** is the flag for the resource name, for example, **s** for stack.

To make permanent changes to the resource limits on your system use `/etc/security/limits.conf` or `/etc/security/limits`.

You can obtain the current resource limit set for a process from the `proc` file system on Linux. For example, `cat /proc/<pid of MQ process>/limits`.

Basic checks before tuning IBM MQ or kernel parameters

You need to investigate the following:

- Whether the number of active connections is within the expected limit.

  For example, suppose that your system is tuned to allow 2000 connections when the number of user processes is no greater than 3000. If the number of connections increases to more than 2000, then either the number of user processes has increased to more than 3000 (because new applications have been added), or there is a connection leak.

  To check for these problems use the following commands:

  – Number of IBM MQ processes:

  ```
  ps -eLf|egrep "amq|run"|wc -l
  ```

  – Number of connections:

  ```
  echo "dis conn(*) all" | runmqsc <qmgr name>|grep EXTCONN|wc -l
  ```

  – Shared memory usage:

  ```
  ipcs -ma
  ```

- If the number of connections is higher than the expected limit, check the source of the connections.
- If the shared memory usage is very high, check the following number of:
  – Topics
  – Open queue handles
- From an IBM MQ perspective, the following resources need to be checked and tuned:
  – Maximum number of threads allowed for a given number of user processes.
  – Data segment
  – Stack segment
  – File size
  – Open file handles
  – Shared memory limits
  – Thread limits, for example, `threads-max` on Linux
- Use the mqconfig command to check the current resource usage.

Notes:

1. Some of resources listed in the preceding text need to be tuned at user level and some at the operating system level.
2. The preceding list is not a complete list, but is sufficient for most common resource issues reported by IBM MQ.
3. Tuning is required at thread level, as each thread is a light weight process (LWP).

Problem in creating threads or processes from IBM MQ or an application

Failure in `xcsExecProgram` and `xcsCreateThread`

Probe IDs, error messages, and components

XY348010 from **xtmStartTimerThread** from an IBM MQ process (for example **amqzlaa0**) or an application

XC037008 from **xcsExecProgram** with error code **xecP_E_PROC_LIMIT** from **amqzxma0**

XC035040 **xcsCreateThread**

XC037007 from **xcsExecProgram** with **xecP_E_NO_RESOURCE**

xcsCreateThread fails with **xecP_E_NO_RESOURCE** followed by failure data capture, for example ZL000066 from **zlaMain**

Probe IDs might be different. Check for the error codes **xecP_E_PROC_LIMIT** and **xecP_E_NO_RESOURCE**.

Error messages reporting errno 11 from pthead_create, for example: `AMQ6119S: An internal IBM MQ error has occurred ('11 - Resource temporarily unavailable'` from pthread_create.)

Resolving the problem on Linux

IBM MQ sets the error code **xecP_E_PROC_LIMIT** when `pthread_create` or fork fails with EAGAIN.

EAGAIN

Review and increase the max user processes and stack size user process resource limits.

Additional configuration required on Linux

Review and increase the limits for kernel.pid_max (`/proc/sys/kernel/kernel.pid_max`) and `kernel.threads-max (/proc/sys/kernel/threads-max)` kernel parameters.

You need to increase the maximum user processes (**nproc**) and stack size resources limits for the `mqm` user and any other any other user that is used to start the queue manager and the IBM MQ applications.

ENOMEM

IBM MQ sets the error code **xecP_E_NO_RESOURCE** when `pthread_create` or fork fails with ENOMEM.

Check and increase the stack size and data resource limits.

Notes:

- You can increase the user process resource limits by using the **ulimit** command, or by changing the resource limit configuration file.
- The changes using the **ulimit** command are temporary. Modify `/etc/security/limits` or `/etc/security/limits.conf` to make the changes permanent. You must check the actual configuration on your operating system, as the configuration might be different.
- You should also review your OS manuals (for example, the man page for pthread_create) for more details on resource issues and tuning resource limits, and ensure that the resource limits are appropriately configured.
- You should also check if the system is running short of resources, both memory and CPU.

Problems in creating shared memory

Error : shmget fails with error number 28(ENOSPC)

```
| Probe Id          :- XY132002                                                |
| Component         :- xstCreateExtent                                         |
| ProjectID         :- 0                                                       |
| Probe Description :- AMQ6119: An internal IBM MQ error has occurred          |
|   (Failed to get memory segment: shmget(0x00000000, 2547712) [rc=-1          |
|   errno=28] No space left on device)                                         |
| FDCSequenceNumber :- 0                                                       |
| Arith1            :- 18446744073709551615 (0xffffffffffffffff)               |
| Arith2            :- 28 (0x1c)                                               |
| Comment1          :- Failed to get memory segment: shmget(0x00000000,        |
|   2547712) [rc=-1 errno=28] No space left on device                          |
| Comment2          :- No space left on device                                 |
+------------------------------------------------------------------------------+
MQM Function Stack
ExecCtrlrMain?
xcsAllocateMemBlock
xstExtendSet
```

```
xstCreateExtent
xcsFFST
```

shmget fails with error number 22(EINVAL)

```
| Operating System  :- SunOS 5.10                                              |
| Probe Id          :- XY132002                                                |
| Application Name  :- MQM                                                     |
| Component         :- xstCreateExtent                                         |
| Program Name      :- amqzxma0                                                |
| Major Errorcode   :- xecP_E_NO_RESOURCE                                      |
| Probe Description :- AMQ6024: Insufficient resources are available to       |
|   complete a system request.                                                 |
| FDCSequenceNumber :- 0                                                       |
| Arith1            :- 18446744073709551615 (0xffffffffffffffff)               |
| Arith2            :- 22 (0x16)                                               |
| Comment1          :- Failed to get memory segment: shmget(0x00000000,        |
|   9904128) [rc=-1 errno=22] Invalid argument                                 |
| Comment2          :- Invalid argument                                        |
| Comment3          :- Configure kernel (for example, shmmax) to allow a       |
|   shared memory segment of at least 9904128
bytes                                                  |
+------------------------------------------------------------------------------+
MQM Function Stack
ExecCtrlrMain
zxcCreateECResources
zutCreateConfig
xcsInitialize
xcsCreateSharedSubpool
xcsCreateSharedMemSet
xstCreateExtent
xcsFFST
```

Unexpected process termination and queue manager crash, or queue manager crash

Process ending unexpectedly followed by FDCs from `amqzxma0`

Example FDC:

```
Date/Time         :- Mon May 02 2016 01:00:58 CEST
Host Name         :- test.ibm.com
LVLS              :- 8.0.0.4
Product Long Name :- IBM MQ for Linux (x86-64 platform)
Probe Id          :- XC723010
Component         :- xprChildTermHandler
Build Date        :- Oct 17 2015
Build Level       :- p800-004-151017
Program Name      :- amqzxma0
Addressing mode   :- 64-bit
Major Errorcode   :- xecP_E_USER_TERM
Minor Errorcode   :- OK
Probe Description :- AMQ6125: An internal IBM MQ error has occurred.
```

Possible Causes and Solutions

- Check if the user has ended any process.
- Check if the IBM MQ process ended because of a memory exception:
  - Did the process end with an FDC of `Component :- xehExceptionHandler`?
  - Apply the fix for known issues corrected in this area.
- Check if the operating system ended the process because of high memory usage by the process:
  - Has the IBM MQ process consumed lot of memory?
  - Has the operating system ended the process?

    Review the operating system log. For example, the OOM-killer on Linux:

    ```
    Jan 2 01:00:57 ibmtest kernel:
    amqrmppa invoked oom-killer: gfp_mask=0x201da, order=0, oom_score_adj=0)
    ```

– Apply the fix for known memory leak issues.

Difference in user limits used by a process against the configured limits

The user limits used by the process might be different from the configured limits. This is likely to happen if the process is started by a different user, or by user scripts, or a high availability script for example. It is important that you to check the user who is starting the queue manager, and set the appropriate resource limits for this user.

Related tasks

"Identifying characteristics of the problem on Linux" on page 30
Some initial questions to consider to help with identifying the cause of the problem.

"Determining problems with applications, commands and messages on Linux" on page 33
If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Windows Making initial checks on Windows

Before you start problem determination in detail on Windows, 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.

About this task

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

As you go through the list of initial questions to consider and follow the links to more information, 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.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. Identify characteristics of the problem

   There are some initial questions that you can consider to help you to identify the cause of the problem:

   - Has IBM MQ run successfully before?
   - Have any changes been made since the last successful run?
   - Have you applied any maintenance updates?
   - Has the application run successfully before?
   - Are you receiving errors when you use special characters in descriptive text for some commands?
   - Are there any error messages or return codes to help you to determine the location and cause of your problem?
   - Can you reproduce the problem?

- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Is the problem intermittent?

2. Investigate potential problems with applications, commands and messages

   If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem:

   - Are messages failing to arrive on the queue?
   - Do messages contain unexpected or corrupted information?
   - Are unexpected messages received when using distributed queues?
   - Have you received no response from a PCF command?
   - Are only some of your queues failing?
   - Does the problem affect only remote queues?
   - Are you receiving an error code when creating or starting a queue manager on Windows?
   - Is your application or system running slowly?

3. If you need further information to help you with identifying the cause of the problem, see "Detailed troubleshooting" on page 63.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Making initial checks on AIX" on page 7
Before you start problem determination in detail on AIX, 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 19
Before you start problem determination in detail on IBM i, 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 Linux" on page 28
Before you start problem determination in detail on Linux, 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 50
Before you start problem determination in detail on z/OS, 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 reference

Messages and reason codes

Windows Identifying characteristics of the problem on Windows

Some initial questions to consider to help with identifying the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Has IBM MQ run successfully before?
- Have any changes been made since the last successful run?
- Have you applied any maintenance updates?

- Has the application run successfully before?
- Are you receiving errors when you use special characters in descriptive text for some commands?
- Are there any error messages or return codes to help you to determine the location and cause of your problem?
- Can you reproduce the problem?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day?
- Is the problem intermittent?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. 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. For more information, see IBM MQ installation overview and Installing and uninstalling IBM MQ on Windows.

   To run the verification procedure, see Verifying an IBM MQ installation on Windows. Also look at Configuring IBM MQ for information about post-installation configuration of IBM MQ.

2. Have any changes been made since the last successful run?

   Changes that have been made to your IBM MQ configuration, or changes to other applications 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 might affect the operation of IBM MQ? For example, have you modified the Windows Registry.

3. 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 IBM Support for any maintenance package errors.
   - If a maintenance package has been applied to any other application, consider the effect it might have on the way IBM MQ interfaces with it.

4. Has the application run successfully before?

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

- 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 on 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. For information about building your application, see Developing applications.

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? 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.

5. Are you receiving errors when you use special characters in descriptive text for some commands?

   Some characters, for example, backslash (\) and double quote (") characters have special meanings when used with commands.

   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.

6. Are there any error messages or return codes to 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. For information about the locations and contents of the error logs, see "Error logs on AIX, Linux, and Windows" on page 407.

   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**.

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

7. 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.

8. Does the problem affect specific parts of the network?

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

   You might be able to identify specific parts of the network that are affected by the problem (for example, remote queues). 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.

9. 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.

10. 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.

Related tasks

"Determining problems with applications, commands and messages on Windows" on page 46

If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Related reference

Messages and reason codes

Windows Determining problems with applications, commands and messages on Windows

If you encounter problems with IBM MQ applications, commands, and messages, there are a number of questions that you can consider to help you to determine the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Are messages failing to arrive on the queue?
- Do messages contain unexpected or corrupted information?
- Are unexpected messages received when using distributed queues?
- Have you received no response from a PCF command?
- Are only some of your queues failing?
- Does the problem affect only remote queues?
- Are you receiving an error code when creating or starting a queue manager on Windows?
- Is your application or system running slowly?

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 need to carry out a systematic problem determination exercise.

When you open a case with IBM, you can include additional IBM MQ troubleshooting information (MustGather data) that you have collected to help with investigating the problem. For more information, see "Collecting troubleshooting information" on page 310.

Procedure

1. Are messages failing to arrive on the queue?

   If messages do not arrive when you are expecting them, check if 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?

   Also check whether you are 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.

2. 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:

   - 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.

3. Are unexpected messages received 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. For more information, see Using the dead-letter (undelivered message) queue and MQDLH - Dead-letter header.
- 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. For more information, see Distributed queuing and clusters.
- 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. For more information, see Data conversion.

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

4. Have you received no response from a PCF command?

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

   - Is the command server running? Use 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. For more information, see MQDLH - Dead-letter header and Using the dead-letter (undelivered message) queue. 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? For more information, see “Error log directories on AIX, Linux, and Windows” on page 409.
   - 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 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 IBM Support for help.

5. Are only 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.

Use the MQSC command **DISPLAY QUEUE** to display the information about each queue. If the **CURDEPTH** is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.

If the **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, or the queue might have been open but is now closed.

Check the status at the time you expect to put or get a message.

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

6. Does the problem affect 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.

7. Are you receiving an error code when creating or starting a queue manager on Windows?

   If 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.

8. 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.

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.

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

Related tasks

"Identifying characteristics of the problem on Windows" on page 42
Some initial questions to consider to help with identifying the cause of the problem.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Related reference

Messages and reason codes

z/OS Making initial checks on z/OS

Before you start problem determination in detail on z/OS, 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.

About this task

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

As you go through the list of initial questions to consider and follow the links to more information, 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.

Procedure

1. Identify characteristics of the problem

   There are some initial questions that you can consider to help you to identify the cause of the problem:

   - Has IBM MQ for z/OS run successfully before?
   - Are there any error messages, return codes or other error conditions?
   - Can you reproduce the problem?
   - Have you applied any APARs or PTFs?
   - Have any changes been made since the last successful run?
   - Has the application run successfully before?
   - Does the problem affect specific parts of the network?
   - Does the problem occur at specific times of the day or affect specific users?
   - Is the problem intermittent or does the problem occur with all z/OS, CICS®, or IMS systems?
   - Do you have a program error?
2. Look at the problem in more detail

There are some further questions to look at when you have established that no changes have been made to your system, and that there are no problems with your application programs, but the preliminary checks have not enabled you to solve your problem.

- Have you received some incorrect output?
- Have you received an unexpected error message or return code?
- Has there been an abend?
- Have you received no response from an MQSC command?
- Is there a problem with the IBM MQ queues?
- Are some of your queues working?
- Are the correct queues defined?
- Does the problem affect only remote or cluster queues?
- Does the problem affect only shared queues?
- Is your application or IBM MQ for z/OS running slowly?
- Has your application or IBM MQ for z/OS stopped processing work?

3. If you need further information to help you with identifying the cause of the problem, see "Detailed troubleshooting" on page 63.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Making initial checks on AIX" on page 7
Before you start problem determination in detail on AIX, 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 19
Before you start problem determination in detail on IBM i, 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 Linux" on page 28
Before you start problem determination in detail on Linux, 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 Windows" on page 41
Before you start problem determination in detail on Windows, 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 reference

Messages and reason codes

z/OS Identifying characteristics of the problem on z/OS

Some initial questions to consider to help with identifying the cause of the problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Has IBM MQ for z/OS run successfully before?
- Are there any error messages, return codes or other error conditions?
- Can you reproduce the problem?

- Have you applied any APARs or PTFs?
- Have any changes been made since the last successful run?
- Has the application run successfully before?
- Does the problem affect specific parts of the network?
- Does the problem occur at specific times of the day or affect specific users?
- Is the problem intermittent or does the problem occur with all z/OS, CICS, or IMS systems?
- Do you have a program error?

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 need to carry out a systematic problem determination exercise.

Procedure

1. Has IBM MQ for z/OS run successfully before?

   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.
   - Use the command **DISPLAY DQM** to check that the channel initiator address space is running, and that the appropriate listeners are started.
2. Are there any error messages, return codes or other error conditions?

   Investigate any 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.
3. Can you reproduce the problem?

   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.

4. 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 IBM Support 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.)

5. Have any changes been made since the last successful run?

   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 do 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.

6. Has the application run successfully before?

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

   - 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 Step “10” on page 55 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.

7. 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 (for example, remote queues). 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.

8. Does the problem occur at specific times of the day or affect specific 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 296.

   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.

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

   A problem might be caused by application interaction or be 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.

10. Do you have a program error?

    The following examples show 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.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Examining the problem in greater depth on z/OS" on page 56
Further checks to carry out when you have established that no changes have been made to your system, and that there are no problems with your application programs, but the preliminary checks have not enabled you to solve your problem.

Related reference

Messages and reason codes

z/OS Examining the problem in greater depth on z/OS

Further checks to carry out when you have established that no changes have been made to your system, and that there are no problems with your application programs, but the preliminary checks have not enabled you to solve your problem.

About this task

Use the following questions as pointers to help you to identify the cause of the problem:

- Have you received some incorrect output?
- Have you received an unexpected error message or return code?
- Has there been an abend?

- Have you received no response from an MQSC command?
- Is there a problem with the IBM MQ queues?
- Are some of your queues working?
- Are the correct queues defined?
- Does the problem affect only remote or cluster queues?
- Does the problem affect only shared queues?
- Is your application or IBM MQ for z/OS running slowly?
- Has your application or IBM MQ for z/OS stopped processing work?

Procedure

1. Have you received some incorrect output?

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

   - When to classify output as incorrect

     "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, see Are there any error messages, return codes or other error conditions? in “Identifying characteristics of the problem on z/OS” on page 51.

   - Unexpected messages

     Your application might not have received a message that it was expecting, or 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). For more information, see “Dealing with incorrect output on z/OS” on page 302.

2. 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, which might give you enough information to resolve the problem quickly, or it might redirect you to further information. If you cannot deal with the message, you might have to contact the IBM Support 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 appropriate messages and codes documentation 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.

3. Has there been an abend?

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

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

   - For Batch applications, your listing shows the abend.
   - For CICS applications, 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.
   - For IMS applications, 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.
   - For TSO applications, 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.)

   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 z/OS" on page 293. 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 258 for information about the abend codes, and see Return codes for an explanation of the reason code.

   If your program has terminated abnormally, see "Dealing with abends on IBM MQ for z/OS" on page 260.

   If your system has terminated abnormally, and you want to analyze the dump produced, see "IBM MQ for z/OS dumps" on page 276. This section tells you how to format the dump, and how to interpret the data contained in it.
4. Have you received no 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 following questions:

   - Is the command server running?

     Check that the command server is running, as follows:

     a. Use the **DISPLAY CMDSERV** command at the z/OS console to display the status of the command server.
     b. If the command server is not running, start it using the **START CMDSERV** command.
     c. If the command server is running, use the **DISPLAY QUEUE** command with 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.
     d. Try stopping the command server and then restarting it, responding to any error messages that are produced.
     e. Issue the display command again to see if it is working now.
   - Has a reply been sent to the dead-letter queue?

If you do not know the name of the system dead-letter queue, use the **DISPLAY QMGR DEADQ** command to find the name. 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. For information about the dead-letter header structure, see Reason (MQLONG).

- 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 276.

- Were any error messages issued?

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

For information about the different methods you can use to enter MQSC commands, see Sources from which you can issue MQSC and PCF commands on IBM MQ for z/OS.

5. Is there a problem with the 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.

   - Has the system responded? If the system responds, then at least one queue is working. In this case, continue with Step "6" on page 59.
   - Has the system not responded? 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.
   - Does the problem still occur after restart? Contact IBM Support for help (see "Collecting troubleshooting information" on page 310).

6. 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 and 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).
  - Use the command `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?

      Use the command `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, either use the command `DIS QSTATUS(xx) TYPE(HANDLE)` to show you information about applications that have the queue open, or use the command `DIS CONN(xx)` to give you more information about an application that is connected to the queue.
- How many tasks are accessing the queues?

  Use the command `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, which might, for example, be because the applications need to be started, or 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 the command `D XCF, STRUCTURE, STRNAME=ALL` to check that the Coupling Facility structures are accessible.

  Use the command `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 IBM Support for help (see "Collecting troubleshooting information" on page 310).

7. 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.

8. Does the problem affect only 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 z/OS" on page 302).
   - 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.

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

9. Does the problem affect only 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
```

10. 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.

    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.

For guidance on dealing with waits and loops, see "Dealing with applications that are running slowly or have stopped on z/OS" on page 297, and on dealing with performance problems, see "Dealing with performance problems on z/OS" on page 296.

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

    There are several reasons why your system might unexpectedly stop processing work. The problem areas to check for include:

    - Are there any queue manager problems? The queue manager might be shutting down.
    - Are there any 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.
    - Are there any problems with IBM MQ? 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.
    - Are there any z/OS or 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.
    - Are there any Db2 or RRS problems? Check that Db2 and RRS are active.

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

    a) Check for error messages.

       Use the **DISPLAY THREAD(*)** command to check if the queue manager is running. If the queue manager has stopped running, look for any messages 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.

    b) If no there are no error messages, issue the following 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.

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.

c) Check whether the queue manager or channel initiator terminated has 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. For more information, see IBM MQ dumps.

d) Check whether 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 Step “10” on page 61. For advice about what to do next, see Dealing with performance problems.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Identifying characteristics of the problem on z/OS” on page 51
Some initial questions to consider to help with identifying the cause of the problem.

Related reference

Messages and reason codes

Detailed troubleshooting

Troubleshooting information to help you solve problems with your queue manager network or IBM MQ applications.

Related concepts

“Using error logs” on page 405
There are a variety of error logs that you can use to help with problem determination and troubleshooting.

“First Failure Support Technology (FFST)” on page 414
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

“Making initial checks” on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Tracing" on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

ALW Troubleshooting AMQP problems

Troubleshooting information to help you solve a problem with running AMQP applications.

Related tasks

"Tracing the Advanced Message Queuing Protocol (AMQP) Service" on page 464
The trace facility provided by the Advanced Message Queuing Protocol (AMQP) Service is provided to help IBM Support to diagnose customer issues that are related to the service.

"Additional diagnostics using the controlAMQPChannel command" on page 466
Using the **controlAMQPChannel** command to provide additional diagnostic information about the AMQP service.

ALW AMQP logs, error logs, and configuration files

Enable and find the logs, error logs, and configuration files used by AMQP.

Note: The examples are coded for Windows systems. Change the syntax to run the examples on AIX or Linux systems.

Server-side logs

The Advanced Message Queuing Protocol (AMQP) 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 `AMQPn.FDC`.

It also writes a log for the AMQP service. The log path is:

```
WMQ data directory\Qmgrs\qMgrName\errors\amqp.log
```

The format of the log file is `amqp_n.log`.

Optionally, if JSON format logging for AMQP is enabled, a JSON log file is written. The log path is:

```
WMQ data directory\Qmgrs\qMgrName\errors\amqp.json
```

The format of the log file is `amqp_n.json`.

When the AMQP Service starts, output is written to:

```
WMQ data directory\Qmgrs\qMgrName\amqp.stdout
WMQ data directory\Qmgrs\qMgrName\amqp.stderr
```

Server-side configuration files

Advanced Message Queuing protocol (AMQP) service

From IBM MQ 9.3.0, the JAAS.Login module files have been moved from the `&MQ_INSTALL_DIRECTORY&/ mqxr/samples/samples` directory to the directory `&MQ_INSTALL_DIRECTORY&/ mqxr/samples/jaas`

The AMQP configurations file name is `amqp_win.properties` file on Windows systems, and the `amqp_unix.properties` file on AIX or Linux systems. The properties files are saved in the AMQP configuration directory:

```
WMQ data directory\Qmgrs\qMgrName\amqp
```

Figure 1. AMQP configuration directory on Windows

```
 /var/mqm/qmgrs/qMgrName/amqp
```

Figure 2. AMQP configuration directory on AIX or Linux

JVM

Set Java properties that are passed as arguments to the AMQP service in the `amqp_java.properties` file. The properties in the file are passed directly to the JVM running the AMQP 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 `amqp_java.properties` file.

Find the `amqp_java.properties` file in the same folder as the AMQP configurations. See Figure 1 on page 65 and Figure 2 on page 65.

Modify `amqp_java.properties` by using the **setamqp** command . For example:

```
V 10.0.0 setamqp properties -m <QueueManager> -k -Xmx1024m -v
setamqp properties -m <QueueManager> -k -Xms1024m -v
```

JSSE

Enable a JSSE debug trace for the AMQP service, so you can see the TLS handshake.

Do this by adding the following property to the `amqp_java.properties` file, as previously described in the "JVM" on page 65 section.

```
Djavax.net.debug=<value>
```

where *<value>* is one or more of `all`, `true`, or `ssl`, optionally followed by debug specifiers. For more information about these options, see Debugging Utilities in the IBMJSSE2 provider documentation.

The output is sent to the `amqp.stdout` file.

JAAS

The JAAS configuration file is described in Configuring JAAS for AMQP channels, which includes the sample JAAS configuration file, JAAS.config, shipped with IBM MQ AMQP.

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 AMQP 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 AMQP service definition.

Display the class paths that are being used by the AMQP service by adding `echo set classpath` to `runAMQPService.bat`. The output is sent to `amqp.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\amqp\samples\samples
```

Note: `service.env` must not contain any variables. Substitute the actual value of *WMQ Installation Directory*.

Figure 3. Sample `service.env` for Windows

Trace
See “Tracing the Advanced Message Queuing Protocol (AMQP) Service” on page 464. The parameters to configure trace are stored in two files:

```
WMQ data directory\Qmgrs\qMgrName\amqp\trace.config
WMQ data directory\Qmgrs\qMgrName\amqp\amqptraceOn.properties
```

and there is a corresponding file:

```
WMQ data directory\Qmgrs\qMgrName\amqp\amqptraceOff.properties
```

Client-side log files and client-side configuration files

See Developing AMQP client applications for information on client side applications.

ALW Enabling JSON formatted logs for AMQP

To enable JSON format error logging, you need to modify the AMQP configuration files `amqptraceOn.properties` and `amqptraceOff.properties`.

About this task

You can configure JSON format logging to be used exclusively, or simultaneously, alongside text-based error logging.

Note: Once you have modified the configuration files, you need to restart the AMQP service for any changes to come into effect.

Procedure

1. To enable JSON format logging exclusively, modify the `amqptraceOn.properties` and `amqptraceOff.properties` configuration files by using the **setamqp** command and update the **handlers** property as follows:

   V 10.0.0

   ```
   setamqp properties -m <QueueManager> -k handlers -v
   com.ibm.mq.util.logging.MQJSONErrorLogFileHandler
   ```

2. To enable JSON format logging simultaneously alongside text-based error logging, modify the `amqptraceOn.properties` and `amqptraceOff.properties` configuration files by using the **setamqp** command and update the **handlers** property as follows:

   V 10.0.0

   ```
   setamqp properties -m <QueueManager> -k
   handlers -v "com.ibm.mq.util.logging.MQErrorLogFileHandler,
   com.ibm.mq.util.logging.MQJSONErrorLogFileHandler"
   ```

Results

In either case, this enables JSON format error logging, and configures the logger with its following default properties:

- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.level`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.filter`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.limit`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.count`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.append`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.permissions`

- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.formatter`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.pattern`

Functionally, these properties work identically to the text-based error logger configured by the `com.ibm.mq.util.logging.MQErrorLogFileHandler`.

Related reference

AMQP channel log files

Related information

V 10.0.0 setamqp

V 10.0.0 ALW Enabling Java heap dump generation to troubleshoot low memory exceptions for AMQP

To enable automatic generation of a Java heap dump when the AMQP service reports a low memory exception, you need to modify the AMQP Java Virtual Machine (JVM) configuration file `amqp_java.properties`.

About this task

Note: After you have modified the configuration file, you need to restart the AMQP service for any changes to come into effect.

Procedure

1. V 10.0.0
   Modify the `amqp_java.properties` file by using the following command:

   ```
   setamqp properties -m <QueueManager> -k -Dcom.ibm.mq.MQXR.GenerateHeapDump -v true
   ```

2. V 10.0.0
   Restart the AMQP service.

Results

A Java heap dump is generated automatically when the AMQP service reports a low memory exception. Only one Java heap dump file is generated. To enable the creation of another file, simply restart the service.

The newly generated Java heap dump file can be found in the IBM MQ errors directory at `WMQ data directory\errors`.

Related information

V 10.0.0 setamqp

ALW Restarting the IBM MQ service for AMQP

From IBM MQ 9.4.0, by default, if you have installed the AMQP file sets, when a new queue manager is created, it has the definition for a SERVICE as shown in the following command inside **runmqsc**.

About this task

```
display service(*) all
AMQ8629I: Display service information details.
   SERVICE(SYSTEM.AMQP.SERVICE)            CONTROL(MANUAL)
   SERVTYPE(SERVER)
   STARTCMD(+MQ_INSTALL_PATH+/bin/amqp.sh)
   STARTARG(start -m +QMNAME+ -d "+MQ_Q_MGR_DATA_PATH+/." -g "+MQ_DATA_PATH+/.")
   STOPCMD(+MQ_INSTALL_PATH+/bin/endmqsde)
   STOPARG(-m +QMNAME+ -n "+MQ_SERVICE_NAME+" -t AMQP)
   STDOUT(+MQ_Q_MGR_DATA_PATH+/amqp.stdout)
```

```
   STDERR(+MQ_Q_MGR_DATA_PATH+/amqp.stderr)
   DESCR(Manages clients that use the AMQP protocol)
```

Note that this service has the following attribute:

```
CONTROL(MANUAL)
```

This means that Service is not started as part of queue manager startup. Change the **CONTROL** attribute to QMGR if you want the Service to start with queue manager startup.

From IBM MQ 9.1.0 to IBM MQ 9.3.0, the SYSTEM.AMQP.SERVICE has its **CONTROL** attribute set to QMGR.

To display the service, use the DISPLAY SVSTATUS command:

```
display svstatus(*)
     1 : display svstatus(*)
AMQ8632I: Display service status details.
   SERVICE(SYSTEM.AMQP.SERVICE)            STATUS(RUNNING)
   PID(18510)
```

and then issue the following command to list the service status:

```
display svstatus(SYSTEM.AMQP.SERVICE) all
     1 : display svstatus(*) all
AMQ8632I: Display service status details.
   SERVICE(SYSTEM.AMQP.SERVICE)            STATUS(RUNNING)
   PID(26263)                              SERVTYPE(SERVER)
   STARTDA(2023-08-22)                     STARTTI(06.25.31)
   CONTROL(QMGR)                           STARTCMD(/opt/mqm93//bin/amqp.sh)
   STARTARG(start -m AMQP -d "/var/mqm/qmgrs/AMQP//." -g "/var/mqm//.")
   STOPCMD(/opt/mqm93//bin/endmqsde)
   STOPARG(-m AMQP -n "SYSTEM.AMQP.SERVICE" -t AMQP)
   DESCR(Manages clients that use the AMQP protocol)
   STDOUT(/var/mqm/qmgrs/AMQP//amqp.stdout)
   STDERR(/var/mqm/qmgrs/AMQP//amqp.stderr)
```

Procedure

1. On Linux, use the following command to identify the process id (where QMGRNAME is the name of the actual queue manager):

   ```
   ps -ef | grep QMGRNAME
   ```

   Note the entry for `com.ibm.mq.MQXRService.RunMQXRService`.
2. Use the following command within **runmqsc**, to stop the service:

```
stop SERVICE(SYSTEM.AMQP.SERVICE)
AMQ8732I: Request to stop Service accepted.
```

3. Wait a few seconds for the actual program to end, and then repeat:

   ```
   ps -ef | grep QMGRNAME
   ```

   Note that this time the entry for `RunMQXRService` is not found.
4. Use the following command within **runmqsc**, to start the service:

```
start SERVICE(SYSTEM.AMQP.SERVICE)
AMQ8733I: Request to start Service accepted.
```

5. Issue the `ps -ef | grep QMGRNAME` again .

   Check that the entry for `RunMQXRService` is there again, which indicates that the AMQP service has restarted.

   Note: The `ampq` service runs as a Java process, and that if attempts to stop the service seem to not complete, that is, the process stays in a stopping state, such as shown in the following example:

```
dis svstatus (SYSTEM.AMQP.SERVICE)
    27 : dis svstatus (SYSTEM.AMQP.SERVICE)
```

```
AMQ8632I: Display service status details.
   SERVICE(SYSTEM.AMQP.SERVICE)            STATUS(STOPPING)
```

you must end the Java process manually.

Troubleshooting AMS problems

Troubleshooting information to help you identify and resolve problems relating to Advanced Message Security (AMS).

About this task

For problems relating to Advanced Message Security, check the queue manager error log first. For more information about error logs, see “Error logs on AIX, Linux, and Windows” on page 407.

Related concepts

“Enabling internal trace for the AMSM address space” on page 458
Trace for the AMSM address space can be enabled using the _AMS_MSG_LEVEL variable, which is passed into the AMSM address space through the ENVARS DD card.

Related tasks

“Collecting information for AMS problems on Multiplatforms” on page 324
If you need assistance to resolve a problem with AMS on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

“Collecting information for AMS problems on z/OS” on page 364
If you need assistance to resolve a problem with Advanced Message Security (AMS) on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

com.ibm.security.pkcsutil.PKCSException: Error encrypting contents for AMS

Error `com.ibm.security.pkcsutil.PKCSException: Error encrypting contents` suggests that Advanced Message Security has problems with accessing cryptographic algorithms.

If the following error is returned by Advanced Message Security:

```
DRQJP0103E  The Advanced Message Security Java interceptor failed to protect message.
com.ibm.security.pkcsutil.PKCSException: Error encrypting contents
(java.security.InvalidKeyException: Illegal key size or default parameters)
```

verify if the JCE security policy in `JAVA_HOME/lib/security/local_policy.jar/*.policy` grants access to the signature algorithms used in MQ AMS policy.

If the signature algorithm you want to use is not specified in your current security policy, download the correct Java policy file, for your version of the product, from the following location: IBM Developer Kits.

OSGi support for AMS

To use OSGi bundle with Advanced Message Security additional parameters are required.

Run the following parameter during the OSGi bundle startup:

```
-Dorg.osgi.framework.system.packages.extra=com.ibm.security.pkcs7
```

When using encrypted password in your keystore.conf, the following statement must be added when OSGi bundle is running:

```
-Dorg.osgi.framework.system.packages.extra=com.ibm.security.pkcs7,com.ibm.misc
```

Restriction: AMS supports communication using only MQ Base Java Classes for queues protected from within the OSGi bundle.

Problems opening protected queues when using AMS with JMS

Various problems can arise when you open protected queues when using Advanced Message Security.

You are running JMS and you receive error 2085 (MQRC_UNKNOWN_OBJECT_NAME) together with error JMSMQ2008.

You have verified that you have set up your AMS as described in Quick Start Guide for AMS with Java clients.

There are a number of IBM MQ options that are either not supported, or have limitations for Advanced Message Security; details are described in Known limitations of AMS.

You have not set the AMQ_DISABLE_CLIENT_AMS environment variable.

Resolving the problem

There are four options for working around this problem:

1. Start your JMS application under a supported IBM Java runtime environment (JRE).
2. Move your application to the same machine where your queue manager is running and have it connect using a bindings mode connection.

   A bindings mode connection uses platform native libraries to perform the IBM MQ API calls. Accordingly, the native AMS interceptor is used to perform the AMS operations and there is no reliance on the capabilities of the JRE.
3. Use an MCA interceptor, because this allows signing and encryption of messages as soon as they arrive at the queue manager, without the need for the client to perform any AMS processing.

   Given that the protection is applied at the queue manager, an alternate mechanism must be used to protect the messages in transit from the client to the queue manager. Most commonly this is achieved by configuring TLS encryption on the server connection channel used by the application.
4. Set the AMQ_DISABLE_CLIENT_AMS environment variable if you do not want to use AMS.

For more information, see Message Channel Agent (MCA) interception and AMS.

Note: A security policy must be in place for each queue that the MCA Interceptor will deliver messages onto. In other words, the target queue needs to have an AMS security policy in place with the distinguished name (DN) of the signer and recipient matching that of the certificate assigned to the MCA Interceptor. That is, the DN of the certificate designated by `cms.certificate.channel.SYSTEM.DEF.SVRCONN` property in the `keystore.conf` used by the queue manager.

Troubleshooting command problems

There are a number of common problems that might occur when you are trying to run a command.

About this task

Some characters, for example, backslash (\) and double quote (") characters have special meanings when used with commands. If you receive an error when you use special characters in descriptive text of a command, check whether you have entered the special characters correctly. For more information about characters with special meanings and how to use them, see Characters with special meanings.

Resolving problems with MQSC commands

If you cannot get MQSC commands to run, use the information in this topic to see if any of these common problems apply to you. It is not always obvious what the problem is when you read the error that a command generates.

Procedure

- If you are redirecting input from a file, ensure that you use the < redirection operator.

  If you omit this operator, the queue manager interprets the file name as a queue manager name, and issues the following error message:

  ```
  AMQ8118E: IBM MQ queue manager does not exist.
  ```

- If you are redirecting output to a file, ensure that you use the > redirection operator.

  By default, the file is put in the current working directory at the time **runmqsc** is invoked. Specify a fully-qualified file name to send your output to a specific file and directory.

- Check that you have created the queue manager that is going to run the commands, and that the queue manager is running, by using the following command to display all queue managers:

  ```
  dspmq
  ```

  If the queue manager is not running, start it. For more information, see Starting a queue manager.

- Ensure that you specified a queue manager on the **runmqsc** command, or that you have a default queue manager defined.

  If you have not defined a default queue manager, or you get this error:

  ```
  AMQ8146E: IBM MQ queue manager not available.
  ```

- Check that you are not attempting to specify an MQSC command as a parameter of the **runmqsc** command. You cannot specify an MQSC command as a parameter of the **runmqsc** command. For example, this is not valid:

  ```
  runmqsc DEFINE QLOCAL(FRED)
  ```

- Check that you have issued the **runmqsc** command. You cannot enter MQSC commands before you issue the **runmqsc** command.

- Check that you are not attempting to run a control command from **runmqsc**. You cannot run control commands from **runmqsc**. For example, you cannot issue the **strmqm** command to start a queue manager while you are running MQSC commands interactively. If you do this, you receive error messages similar to the following message:

  ```
  runmqsc
  .
  .
  Starting MQSC for queue manager jupiter.queue.manager.
  1 : strmqm saturn.queue.manager
  AMQ8405: Syntax error detected at or near end of cmd segment below:-s
  AMQ8426: Valid MQSC commands are:
  ALTER
  CLEAR
  DEFINE
  DELETE
  DISPLAY
  END
  PING
  REFRESH
  RESET
  RESOLVE
  RESUME
  START
  STOP
  SUSPEND
  2 : end
  ```

Related tasks

Administering IBM MQ using MQSC commands

Initializing the current command environment with setmqenv

If you receive an error message indicating that an IBM MQ command is unknown or unavailable, use the **setmqenv** command to initialize the current command environment so that the operating system can find the appropriate command.

About this task

Each operating system uses the directories in the **PATH** environment variable to determine which program to run. If none of the directories in the current **PATH** environment variable include an executable program of the specified name, this causes the operating system to respond with a message indicating that it could not locate the command, as shown in the following examples:

- Linux UNIX On UNIX and Linux:

```
% dspmqver
ksh: dspmqver:  not found.
%
```

- Windows On Windows:

```
C:\> dspmqver
'dspmqver' is not recognized as an internal or external command,
operable program or batch file.
```

To resolve this issue, you should use the **setmqenv** command to initialize the current environment so that the appropriate IBM MQ commands can be located and executed by the operating system.

Procedure

1. Locate where the **setmqenv** command is on the available file systems.
   - Linux UNIX On UNIX and Linux:

     ```
     % find / -name setmqenv 2>/dev/null
     ...
     %
     ```

   - Windows On Windows:

     ```
     C:\> dir /b /s C:\setmqenv.cmd
     ...

     C:\>
     ```

     Note: If multiple drive letters exist, they may be included, for example: `C:\> dir /b /s C:\setmqenv.cmd D:\setmqenv.cmd .`

     Directories with `source`, or `Maint` can be ignored since they almost always refer to a backup copy of the command created during the installation of a fix pack.
2. Use the **setmqenv** command to initialize the current environment.
   - Linux UNIX On UNIX and Linux:

     ```
     % . <pathtoexecutable>/setmqenv -s
     %
     ```

     On UNIX and Linux, the **setmqenv** command must be sourced so the current environment can be modified.

- Windows On Windows:

```
C:\> <pathtoexecutable>\setmqenv.cmd -s
...

C:\>
```

3. Verify that the environment is suitable for the appropriate IBM MQ installation by using the **dspmq** command with the **-o installation** option.
   - Linux UNIX On UNIX and Linux:

```
% dspmq -o installation
...
%
```

   - Windows On Windows:

```
C:\> dspmq -o installation
...

C:\>
```

4. If necessary, run the **setmqenv** command again, with adjusted parameters, to initialize the environment for a more appropriate installation of IBM MQ.

   For example:

   - Linux UNIX On UNIX and Linux:

```
% dspmqver | grep Version
Version:     8.0.0.15

% dspmq -o Inst | sed -e "s/          //g"
QMNAME(V8QM1)     INSTNAME(Installation1) INSTPATH(/opt/mqm) INSTVER(8.0.0.15)
QMNAME(SPIDEY)    INSTNAME(Installation4) INSTPATH(/opt/MQ91CD) INSTVER(9.1.5.0)
QMNAME(BOB)       INSTNAME(Installation3) INSTPATH(/opt/MQ92) INSTVER(9.2.2.0)

% . setmqenv -n Installation3

% dspmqver | grep Version
Version:     9.2.2.0
```

     Note: The **setmqenv** command must still must be sourced so the current environment can be modified.

   - Windows On Windows:

```
C:\> dspmqver | findstr Version
Version:     9.1.0.5

C:\> dspmq -o inst | sed -e "s/          //g"
QMNAME(A)         INSTNAME(Installation2) INSTPATH(C:\IBM\MQ91LTS) INSTVER(9.1.0.5)
QMNAME(B)         INSTNAME(MQ91CD) INSTPATH(C:\IBM\MQ91CD) INSTVER(9.2.2.0)
QMNAME(C)         INSTNAME(MQ92) INSTPATH(C:\IBM\MQ92) INSTVER(9.2.2.0)

C:\> setmqenv -n MQ92

C:\> dspmqver | findstr Version
Version:     9.2.2.0
```

Troubleshooting distributed publish/subscribe problems

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 "Troubleshooting queue manager cluster problems" on page 195.

There are also some helpful troubleshooting tips in Design considerations for retained publications in publish/subscribe clusters.

Related concepts

Distributed publish/subscribe system queue errors

Related tasks

Configuring a publish/subscribe cluster

Designing publish/subscribe clusters

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 "Troubleshooting queue manager cluster problems" on page 195.

- 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.

Attention: Attempting to define a clustered topic with **CLROUTE**(`TOPICHOST`) underneath an existing administration topic, also defined with **CLROUTE**(`TOPICHOST`), results in error message AMQ8849 being reported.

Although the configuration described in the preceding text seemingly results in a consistent routing behavior, IBM MQ prevents conflicting (and potentially conflicting definitions on remote members of the cluster) in order to eliminate unexpected and potentially random routing behavior, resulting from inconsistent definitions elsewhere in the topology. For example:

- Publications being discarded, when wildcard subscribers are attached in the topic tree above a new administration topic.
- Definitions not being created equally in number.

Related tasks

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 concepts

REFRESH CLUSTER considerations for publish/subscribe clusters

Related tasks

Checking that async commands for distributed networks have finished

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. IBM MQ 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 IBM MQ 8.0 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.

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.

Troubleshooting distributed queue management problems

Troubleshooting information to help you solve problems relating to distributed queue management (DQM).

About this task

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.

IBM Support 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:

- “Using Ping to test communications” on page 81
- “Considerations for retrying a link” on page 87
- “Troubleshooting a problem where a channel refuses to run” on page 83
- “Resolving problems where a channel stops running” on page 88
- “Monitoring messages with dspmqrte” on page 81

- “Disaster recovery” on page 89

Related tasks

“Making initial checks” on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

“Collecting information for channel problems on Multiplatforms” on page 326
If you need assistance to resolve a problem when an IBM MQ channel is reporting a problem or failing to run on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Configuring distributed queuing

Related reference

Messages and reason codes

Communications protocol return codes for z/OS

Where to find information to help with troubleshooting

Depending on the type of problem you are experiencing, there are a number of possible sources of information that you can use to help you with troubleshooting.

Command validation problems

Commands and panel data must be free from errors before they are accepted for processing. Any errors that are found by the validation checks are immediately notified to the user by error messages.

A number of validation checks are made when creating, altering, and deleting channels and, where appropriate, an error message is returned. Errors might 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

Problem diagnosis begins with the interpretation of the error messages and taking corrective action.

Processing problems during normal channel operation

Problems that are found during normal operation of the channels are notified to the system console or the system log. On Windows, they are reported to the channel log. Problem diagnosis begins with the collection of all relevant information from the log, and continues with analysis to identify the problem. Confirmation and error messages are returned to the terminal that initiated the commands, when possible.

Problem diagnosis might be difficult in a network where a problem might 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 that you receive in your error log will indicate a problem originating from the remote site, but might not be able to tell you any details about the error at that site. You must therefore contact your counterpart at the remote site to obtain details of the problem, and to receive notification of that channel becoming available again.

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 might 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.

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.

Client application problems

A client application might 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 might also be errors logged at the server, depending on the nature of the failure.

Note: Even though a client application 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.

Diagnostic messages and reason codes

For messages and codes to help with the primary diagnosis of the problem, see Messages and reason codes.

Accounting and statistical data

IBM MQ produces accounting and statistical data, which you can use to identify trends in utilization and performance:

- Multi On Multiplatforms, this information is produced as PCF records, see Structure data types.
- z/OS On z/OS, this information is produced as SMF records, see Monitoring performance and resource usage.

Data structures

Data structures are needed for reference when checking logs and trace entries during problem diagnosis.

For more information, see Channel-exit calls and data structures and Developing applications reference.

Related concepts
Channel control function

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 four ways that the message channel agent (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

Monitoring messages with dspmqrte

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.

You can use the IBM MQ display route application (**dspmqrte**) command to work with trace-route messages and activity information related to a trace-route message, using a command-line interface.

The IBM MQ display route application (**dspmqrte**) command can be run on all platforms except z/OS. You can run the IBM MQ display route application as a client to an IBM MQ for z/OS queue manager by specifying the **-c** parameter when issuing the **dspmqrte** command.

For more information, see IBM MQ display route application and dspmqrte (display route information).

Using Ping to test communications

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.

About this task

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.

Procedure

- Use the MQSC command PING CHANNEL to test a channel by sending data as a special message to the remote queue manager, and checking that the data is returned.

  The data is generated by the local queue manager.

  IBM i z/OS On z/OS and IBM i, you can also use the panel interface to select this option.
- Multi

  On Multiplatforms, use the MQSC command PING QMGR to test whether the queue manager is responsive to commands.

Related concepts

Checking links using Ping

z/OS Using SUBSTATE to troubleshoot channels

If your channel is running slowly or appears to have no messages to send, use the SUBSTATE field to help resolve the problem.

Environment

Diagnosing the problem

Sender channel status

```
CHSTATUS(MQ23.TO.MQ24)  XMITQ(TO.MQ24)    CONNAME(127.0.0.1)
CURRENT     CHLTYPE(SDR)    STATUS(RUNNING)  SUBSTATE(MQGET)
RQMNAME(MQ24)
```

Receiver channel status

```
CHSTATUS(MQ23.TO.MQ24)      CONNAME(127.0.0.1)      CURRENT
CHLTYPE(RCVR)           STATUS(RUNNING)     SUBSTATE(RECEIVE)
RQMNAME(MQ23)
```

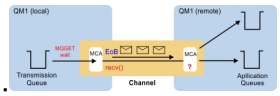
If you see the channel in this state, the channel behaves as it has no messages to send. As an aside, this can be a useful check to make at the same time as checking for uncommitted messages on the transmission queue, when you suggest that the channel is not moving the messages just put.

It is worth noting at this point that the resting state of a server-connection channel is similar to a receiver channel as it spends its time sitting in a network receive waiting for the client to send it an API call to issue.

Slow channels

The sub status of channels is also useful, when other evidence suggests that your channel is running really slowly. Having described the resting state of channels in the previous section, if you see the sender, rather than the receiver channel, is sitting in a network receive, this can indicate that the line turn around on the network is slow.

This is because the sender channel is waiting on the receiver to process all the messages that have been sent in the batch and then respond to the End of Batch (EoB in the following diagram) notification to say the batch has been successfully completed.



This state happens at the end of every batch, but it is usually so fleeting, that you are unlikely to see it. If you see it for a prolonged period of time it can be indicative of major network issues, such as:

- Router problems
- Retransmission of dropped packets
- Other issues that cause the network to run slow
- Something else that causes the receiving end to slow down, such as message retry, or issues in a exit

At this point looking at how the receiver channel is doing is the next step.

Hung channels

When a channel is hung, either stuck in BINDING state for a long time, or even hung in RUNNING state where fields such as Number of Bytes Sent (NUMBYTES) is not increasing, then sub status can help.

When a channel, whether a queue manager channel or a server-connection channel, is starting up there are a number of different operations that it has to do before it can be considered to be RUNNING. All these different operations happen while the channel is in BINDING state.

To help break that down into a more granular view of what the channel is doing, there are sub status values for all these major operations. Some of the more common ones you see are listed below:

Connecting to the network
When there are problems reaching the host, perhaps because the IP address is not longer in service, this can take a very noticeable number of seconds to timeout and return an error. Until it does, the channel shows SUBSTATE(NETCONNECT).

Making a Domain Name Server (DNS) request
If the local DNS doesn't know how to resolve your CONNAME, it may be able to ask other related DNS (if your system is configured this way). This can all take considerable time. While this is happening, the channel shows SUBSTATE(NAMESERVER).

SSL/TLS Handshake
When using SSL or TLS, by specifying a cipher spec on a channel definition in the SSLCIPH field, this means that the channel must complete an SSL handshake before any other data can flow across the channel. While this is happening, the channel shows SUBSTATE(SSLHANDSHAKE).

In an exit
There are various different exits than can be used on a channel. While the exit is processing, the channel will show a sub state reflecting which exit it is in, for example, SUBSTATE(SENDEXIT). If the exit is taking a long time then you may see this for more than a fleeting moment.

Resolving the problem

See DISPLAY CHSTATUS for a summary of all the sub states and when they can be seen.

Troubleshooting a problem where a channel refuses to run

If a channel refuses to run, there are a number of potential reasons such as the DMQ and channels not being set up correctly, or the channel being in-doubt.

About this task

If the channel has never run, a likely cause of the problem is that DQM and the channels might not have been set up correctly.

If the automatic synchronization on startup has failed for some reason, the channel might be in-doubt. In-doubt channel problems are typically resolved automatically but in some cases you might need to resynchronize the channel manually.

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.

Procedure

1. Check that DQM and the channels have been set up correctly.
   This is a likely source of the problem 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?).

For more information about setting up channels, see Configuring distributed queueing.

2. Check whether the channel is in-doubt.

   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 might be used to show channels that are in doubt. If a channel is in doubt, it is usually resolved automatically on restart, so you do not need to resolve a channel manually in normal circumstances. However, you can, when necessary, resynchronize the channel manually. For more information, see Handling in-doubt channels.

   The possible responses to a situation where you need to resynchronize the channel manually are:

   - Issue a **RESOLVE CHANNEL** command to either back out or commit the in-doubt messages.

     To determine whether a backout or commit is needed, check with your remote link supervisor to establish the number of the last-committed unit of work ID (LUWID) committed then check this number 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, use the **RESOLVE CHANNEL** command to commit the messages. In all other cases, use the **RESOLVE CHANNEL** command to back out the messages. For more information, see Handling in-doubt channels.

     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 might 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 must use it only after using the **RESOLVE CHANNEL** command to resolve any in-doubt situations.

   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.

3. If the status of a receiver end of the channel is STOPPED, reset it 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.

Related reference

RESOLVE CHANNEL (ask a channel to resolve in-doubt messages)

RESET CHANNEL (reset message sequence number for a channel)

Troubleshooting triggered channels

If a triggered channel refuses to run, it might be in-doubt. Another possibility is that the channel has set the trigger control parameter on the transmission queue to NOTRIGGER.

About this task

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.

z/OS On 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.

Procedure

1. Check whether the channel is in-doubt.

   If a triggered channel refuses to run, investigate the possibility of in-doubt messages as described in Step "2" on page 84 of "Troubleshooting a problem where a channel refuses to run" on page 83.
2. Check whether 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.
3. After diagnosing and fixing the problem, start the channel manually.

Troubleshooting network problems

There are a number of things to check if you are experiencing network problems.

Procedure

- 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, try setting 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 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.

For more information, see Checking that the other end of the channel is still available, and Keepalive Interval (KAINT).

Note: 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.

- If the receiver channel has been left in a 'communications receive' state after the channel lost contact, check whether user intervention is needed to address the problem.

  If a channel loses contact, the receiver channel can be left in a 'communications receive' state. When communications are re-established the sender channel attempts to reconnect. If the remote queue manager finds that the receiver channel is already running it does not allow another version of the same receiver channel to start. This problem requires user intervention to rectify the problem or the use of system keepalive.

  The Adopt MCA function solves the problem automatically. It enables IBM MQ to cancel a receiver channel and to start a new one in its place.

Related concepts

Monitoring your IBM MQ network

Related reference

Channels stanza of the qm.ini file

z/OS ALW Channel failure with return code ECONNRESET for TCP/IP

There is a channel failure, and on z/OS you receive the following: `CSQX208E TRPTYPE=TCP RC=00000461`, or `CSQX208E TRPTYPE=TCP RC=00000461 reason=76650446`.

Cause

Depending upon the platform or platforms your enterprise uses, you receive the following return code when the connection is reset by peer (ECONNRESET):

AIX **AIX**

ECONNRESET 73 (hexadecimalm49)

Linux **Linux**

ECONNRESET 104 (hexadecimal 68)

Windows **Windows**

WSAECONNRESET 10054 (hexadecimal 2746)

z/OS **z/OS**

10054 or RC461

This return code is often the results of a problem in the TCP/IP network. There are various reasons for TCP/IP sending a reset:

- An unorderly connection termination, such as a rebooting the client box, can cause a reset.
- An application requests a connect to a port and IP address on which no server is listening.
- An application closes a socket with data still in the application receive buffer. The connection is reset to allow the remote partner to know that the data was not delivered.
- Any data that arrives for a connection that has been closed can cause a reset.
- An application closes a socket and sets the linger socket option to zero. This notifies TCP/IP that the connection should not linger.

  Note: IBM MQ does not code the linger time = `0`, therefore IBM MQ itself does not cause a reset.

- A TCP segment that is not valid arrives for a connection. For example, a bad acknowledge or sequence number can cause a reset.
- The connect request times out. TCP stops trying to connect to a particular port and IP address and reset the connection.
- A firewall can reset connections if the packet does not adhere to the firewall rules and policies. For example, a source or destination port, or IP address does not match the firewall rule or policy.
- The retransmit timer expires. TCP stops trying to retransmit a packet and reset the connection.
- A bad hardware device can cause resets.

You need to be aware that the effect of your configuration at higher levels, for example, the channel initiator dispatching priority being too low, could exhibit itself as a reset. Therefore, you should also consider the effect of your configuration when trying to determine why a reset is happening.

Diagnosing the problem

Use TCP/IP packet traces to determine why the reset occurred.

See z/OS UNIX reason codes for the last two bytes of the reason code found in the CSQX208E error message.

Considerations for retrying a link

If a link failure occurs during normal operation, a sender or server channel program will itself start another instance, subject to certain conditions being met. Other error scenarios might be more difficult to troubleshoot and require further manual investigation.

Link failure during normal operation

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 Multiplatforms, to attempt a retry a channel initiator must be running. For IBM MQ for z/OS, this channel initiator must be monitoring the initiation queue specified in the transmission queue that the channel is using.

Difficult to recognize error scenarios

An error scenario might occur that is difficult to recognize. For example, the link and channel might 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.

z/OS *Shared channel recovery on z/OS*

Shared channel recovery is one of the benefits of using queue sharing groups on IBM MQ for z/OS.

The following table shows the types of shared channel failure and how each type is handled:

Table 1. Handling of shared channel failures	
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.

Related concepts

Preparing IBM MQ for z/OS for DQM with queue sharing groups

Resolving problems where a channel stops running

Two possible solutions to the problem of a channel ceasing to run are channel switching and connection switching.

About this task

Two possible solutions to the problem of a channel ceasing to run are:

Channel switching

For channel switching, two message channels are defined for the same transmission queue, but with different communication links. One message channel is preferred, the other is a replacement for use when the preferred channel is unavailable.

Note: If triggering is required for these message channels, the associated process definitions must exist for each sender channel end.

Connection switching

Another solution is to switch communication connections from the transmission queues.

Procedure

- To switch message channels:
  - If the channel is triggered, set the transmission queue attribute **NOTRIGGER**.
  - Ensure that 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**.
- To switch communication connections from the transmission queues:

- If the sender channel is triggered, set the transmission queue attribute **NOTRIGGER**.
- Ensure that 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**.

Disaster recovery

Disaster recovery planning is the responsibility of individual installations, and the functions that are performed might 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 might 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 might 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 might 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 might be recovered to an earlier state, and the messages held on local queues at the sending and receiving end of channels might be inconsistent.

Messages might 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 might need to restart from an earlier message sequence number.

Troubleshooting IBM MQ Console and REST API problems

Diagnose problems with the IBM MQ Console and REST API by looking at the available logs. When asked by IBM Support, you might also need to configure trace.

About this task

If you are experiencing problems with the IBM MQ Console or REST API, check the following things.

Procedure

- The status of the mqweb server.

  If the mqweb server is stopped, you cannot use the IBM MQ Console or REST API. You can check the status of the server by using the following command:

  ```
  dspmqweb status
  ```

  Attention:

  Before you issue either the **setmqweb** or **dspmqweb** commands on z/OS, you must set the WLP_USER_DIR environment variable so that the variable points to your mqweb server configuration.

  To set the WLP_USER_DIR environment variable, issue the following command:

  ```
  export WLP_USER_DIR=WLP_user_directory
  ```

where *WLP_user_directory* is the name of the directory that is passed to `crtmqweb`. For example:

```
export WLP_USER_DIR=/var/mqm/web/installation1
```

For more information, see Create the mqweb server.

Attention: Linux

Before you issue either the **setmqweb** or **dspmqweb** commands in a stand-alone IBM MQ Web Server installation, you must set the **MQ_OVERRIDE_DATA_PATH** environment variable to the IBM MQ Web Server data directory.

If the mqweb server is stopped, start the server with the following command:

```
strmqweb
```

z/OS On z/OS, check that the mqweb server started task is running. If necessary, start the procedure that you created in Create a procedure for the mqweb server.

- Ensure that the following mqweb configuration files exist:

  ```
  jvm.options
  mqwebuser.xml
  server.xml
  ```

  Look for the files in the `MQ_DATA_PATH/web/installations/installationName/servers/mqweb/` directory by using the crtmqdir command.

  To check the installation, which includes searching for these files, use the following command:

  ```
  crtmqdir -a
  ```

  If the files are missing, you can re-create them by using the command:

  ```
  crtmqdir -s -f
  ```

- Examine the mqweb server log files, `console.log`, and `messages.log`.

  These log files can be found in the following location:

  – ALW *MQ_DATA_PATH*`/web/installations/`*installationName*`/servers/mqweb/logs`

  – z/OS The directory that was specified when the **crtmqweb** script ran to create the mqweb server definition. By default, this directory is `/var/mqm/web/installation1/servers/mqweb/logs`.

    These files are in UTF-8. To view the files, you can use one of the following methods:

    - Use the **oedit** command from a z/OS UNIX System Services command line.
    - Enter ISPF option 3.17, and use the **va** (view ASCII) line command.

- z/OS

  On z/OS, check STDERR and STDOUT in the mqweb server started task output. STDERR contains no messages, unless an error occurred.

- If you cannot access the IBM MQ Console or REST API from a host other than the system where the mqweb server is running, check that remote connections are enabled.

  Issue the following command to display the mqweb server configuration:

  ```
  dspmqweb properties -a
  ```

If the value of the *httpHost* property is `localhost`, the IBM MQ Console and REST API are available only from the same host as the mqweb server. Enable remote connections to the mqweb server by entering the following command where *hostname* specifies the IP address or domain name server (DNS) hostname where IBM MQ is installed:

```
setmqweb properties -k httpHost -v hostname
```

Use an asterisk, *, in double quotation marks, to specify all available network interfaces, as shown in the following example:

```
setmqweb properties -k httpHost -v "*"
```

- If no queue managers are displayed in the local queue manager widget in the IBM MQ Console, check that you have queue managers on the same host as the mqweb server that you can manage with the IBM MQ Console.
  - ALW Only queue managers in the same installation as the mqweb server are listed in the IBM MQ Console.
  - z/OS On z/OS, only queue managers that are started at the same version as the mqweb server since the last IPL are listed in the IBM MQ Console.
- If messages are truncated when you browse queues in the IBM MQ Console, adjust the `mqConsoleMaxMsgCharsToDisplay` and `mqConsoleMaxMsgRequestSize` properties by using the setmqweb properties command.
- If you are using the messaging REST API with remote queue managers and you cannot connect to the queue manager, check the following things:
  - You are specifying the unique name for the queue manager, not the queue manager name, within the resource URL.
  - You are trying to connect to a queue manager that is visible to the messaging REST API. Some queue managers might be available only to the IBM MQ Console. You can verify whether the queue manager is configured to be visible to the messaging REST API by using the **dspmqweb remote** command. View the **visibility** parameter to check whether the queue manager is visible to the messaging REST API.
  - You are not prefixing an asterisk to the queue manager name to specify that you want to connect to a queue manager group. To specify you want to connect to a queue manager group, you must set the **group** attribute in the remote queue manager configuration. For more information, see Setting up a remote queue manager group to use with the messaging REST API.
  - Ensure that the CCDT file is in the location that is specified by the **ccdtUrl** parameter in the **dspmqweb remote** command.
- z/OS

  If you are still experiencing problems, the mqweb server started task might not be configured correctly, or there might be a problem with the IBM MQ for z/OS UNIX System Services Web Components installation files.

  You might see the following message in the IBM MQ Console:

  ```
  Lost communication with the server Could not establish communication with the server.
  ```

  In the procedure that is used to start the mqweb server check the following things:

  1. Check that the STEPLIB libraries are at the correct level, and are APF authorized.
  2. Check that INSTDIR, USERDIR, PATH, and LIBPATH point to the correct path.

  In z/OS UNIX System Services, enter the following command where *PathPrefix* is the IBM MQ for z/OS UNIX System Services Components installation path:

  ```
  ls -Eltr PathPrefix/web/bin/dspmq
  ```

  This command displays an output similar to the following output:

```
-rwxr-xr-t a-s- ... /mqm/V10R0M0/web/bin/dspmq
```

Check the **t** and **a** flags are set. If necessary, use the commands:

- `chmod +t` *PathPrefix*`/web/bin/dspmq` to set the sticky bit (t)
- `extattr +a` *PathPrefix*`/web/bin/dspmq` to set the APF-authorized attribute (a)

What to do next

For more information about gathering trace for the IBM MQ Console and REST API, see "Tracing the REST API" on page 513, and "Tracing the IBM MQ Console" on page 466.

Troubleshooting IBM MQ Internet Pass-Thru problems

There are a number of steps you can follow to help determine the nature of any problems you might encounter when using IBM MQ Internet Pass-Thru (MQIPT).

Procedure

1. Check for the following common errors:
   - The **HTTP** property is set to `true` on a route directly connected to a queue manager.
   - The **SSLClient** property is set to `true` on a route directly connected to a queue manager that is not configured to use SSL/TLS.
   - The passwords stored for the key ring files are case-sensitive.
2. Check for FFST reports in the errors subdirectory.

   If you find any FFST reports in the errors subdirectory, this means that MQIPT was correctly installed but there might have been a problem with the configuration.

   Each FFST reports a problem that causes MQIPT or a route to terminate its startup process. Fix the problem that caused each FFST. Then delete the old FFST and restart or refresh MQIPT.
3. If there are no FFST reports, and there is no trace output, this means that MQIPT has not been installed correctly. Check that all the files have been put in the correct place. To check this, try to start MQIPT manually:
   a. Open a command prompt. Go to the `bin` subdirectory and type:

      ```
      mqipt xxx
      ```

      where *xxx* is the MQIPT home directory.
   b. When MQIPT starts, look for the configuration in the home directory. Look for any error messages and FFST instances in the `errors` subdirectory.
   c. Look at the text output from MQIPT for any error messages. Check for instances of FFST. Correct any errors.

      Note: MQIPT will not start if there is a problem in the `[global]` section of the configuration file. A route will not start if there is a problem in the `[route]` section of the configuration file.
4. If there are no FFST reports, but you do have trace output, configure the MQIPT connections (`ConnectionLog=true`) and make the sender attempt a connection. Then check that a connection from the host has been logged.
   - If a connection from the host has been logged, the sender has not been configured correctly.
   - If a connection has not been logged, check that MQIPT is configured to forward the message to the correct host and port. Then treat as a normal channel problem.

Related tasks

"Tracing errors in IBM MQ Internet Pass-Thru" on page 469

IBM MQ Internet Pass-Thru (MQIPT) provides a detailed execution trace facility, which is controlled by the **Trace** property.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Collecting information for MQIPT problems on Multiplatforms" on page 354
If you need to report a problem with MQIPT to IBM, send relevant information that will help to resolve the problem more quickly.

Checking for end-to-end connectivity

If you cannot make a connection, check the connection log to see if the routes are set up correctly.

Create the connection log: In the `mqipt.conf` configuration file, set the **ConnectionLog** property to `true`. Start or refresh MQIPT, and attempt a connection. See Connection logs for details.

1. If the connection log is not created in the logs directory below the home directory, MQIPT has not been installed correctly.
2. If no connection attempts are recorded, the sender has not been set up correctly.
3. If attempts are recorded, check that MQIPT is forwarding the messages to the correct address.

Automatically starting MQIPT

If you install MQIPT as a Windows service, or as a Linux or AIX System V init service, it starts when the system is started. If the service does not start correctly, follow the steps in this topic.

Always try to start MQIPT manually before it is installed as a service, to confirm that MQIPT is correctly installed and configured.

For information about using the **mqiptService** command to start MQIPT automatically, see Starting and stopping MQIPT.

If the MQIPT service does not start correctly, complete the following steps for your platform.

On Windows systems

Windows

Note: On Windows, the MQIPT service runs under the LocalSystem account. The service cannot be configured to run under a different user ID.

If the MQIPT service does not start correctly, complete the following steps:

1. Open the Windows Registry Editor and navigate to the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\MQInternetPassThru` key. Check that the **ConfigFilePath** setting contains the correct path to the `mqipt.conf` configuration file. Also, check that the **ImagePath** setting contains the correct path to `mqiptService.exe`.
2. Run the **mqiptService -debugevent** command from an Administrator Command Prompt to write service startup information in the Windows application event log. Additional information is also displayed in the Command Prompt console window. Examine the diagnostic information to determine the cause of the failure.
3. If the cause of the failure is still not clear, use Windows file explorer to navigate to the directory specified in **ConfigFilePath** where `mqipt.conf` is located. Examine the contents of the errors subdirectory to look for FDC files containing FFST records.
4. If the cause of the failure is still not clear, enable trace by setting the **Trace** property to 5 in the `[global]` section of `mqipt.conf`. Restart the MQIPT service. A trace file is be written in the MQIPT errors directory. If necessary, contact IBM Support and supply the trace file along with any FDC files and the diagnostic output from the **mqiptService -debugevent** command.

On AIX and Linux systems

Linux AIX

Note: Linux

1. Deprecated The MQIPT service on Linux is deprecated. The service on Linux uses System V init, which is not supported on some recent versions of Linux.
2. The MQIPT service is not supported on the following versions of Linux which do not support System V init:
   - Red Hat® Enterprise Linux 9.
   - SUSE Linux Enterprise Server 15.

   On these versions of Linux, use another method, such as `systemd`, to manage MQIPT as a service.

If the MQIPT service does not start correctly, complete the following steps as the root user:

1. Check that the MQIPT service is installed. You might need to uninstall and reinstall the service. To check that the service is installed:
   - AIX On AIX, run the command **`lsitab mqipt`** and check that the output shows the correct installation directory. Here is an example of the output for an MQIPT service running from the `/usr/opt/mqipt` installation:

     ```
     mqipt:2:once:/usr/opt/mqipt/bin/mqipt /usr/opt/mqipt > /dev/console 2>&1
     ```

     Check that the MQIPT executable named exists and is executable by the root user.
   - Linux On Linux, check for the existence of the MQIPT **`init.d`** script named `/etc/init.d/mqipt`. The script must exist and must be executable by the root user.
2. Ensure that the installation directory contains the `mqipt.conf` file. The file must be readable by the root user.
3. Check the output from the MQIPT startup.
   - AIX On AIX, the MQIPT output is sent to `/dev/console`.
   - Linux On Linux, the output is sent to a file named `console.log` in the `logs` directory of the MQIPT installation.

   Look for any MQIPT errors and address the cause. If no console output is present then MQIPT was not started by the operating system. Consult your operating system documentation for details of how to diagnose service startup failures.
4. If the cause of the failure is still not clear, navigate to the MQIPT installation directory where `mqipt.conf` is located. Examine the contents of the errors subdirectory to look for FDC files containing FFST records.
5. If the cause of the failure is still not clear, enable trace by setting the Trace property to 5 in the `[global]` section of `mqipt.conf`. Restart the MQIPT service. A trace file is written in the MQIPT errors directory. If necessary, contact IBM Support and supply the trace file along with any FDC files and the diagnostic output from `/dev/console` (on AIX) or `console.log` (on Linux).

Windows MQIPT fails to start on a Windows server

IBM MQ Internet Pass-Thru (MQIPT) connects client applications to a queue manager, or a queue manager to a queue manager, through the internet by reading the configuration data in the `mqipt.conf` file.

The `mqipt.conf` file is a text file that can be modified in any editor, including Notepad.

Important: If you use Notepad, you must take care when saving the file.

If you receive the following error:

```
MQCPI001 IBM MQ Internet Pass-Thru V2.1.0.1 starting
  MQCPI004 Reading configuration information from mqipt.conf
  MQCPE017 The exception java.io.IOException: MQCPE019 The stanza
           [global] was not found before the following :
           ?[Uxbb][Uxbf][Ux23][Ux23][Ux23][Ux23]
```

MQIPT has not found the required configuration details.

MQIPT found some unknown characters before the `[global]` stanza and reopening the `mqipt.conf` file in Notepad again does not show up these unknown characters.

The problem lies with the way the configuration file was saved.

Solution

Notepad saves the configuration file, by default, using UTF-8 encoding which adds the "Byte order mark" before the actual text in the file. This makes MQIPT fail with the error specified previously.

To solve the problem, choose the `File/ Save as` menu, and in the **Encoding** field select `ANSI` in the drop down list. Save the file again

Using JRE diagnostic options

In some cases you might need to use diagnostic functions that are built into the Java runtime environment (JRE). You should usually do this only under the direction of an IBM Support representative, as some diagnostic settings might impair normal MQIPT operation.

The **`MQIPT_JVM_OPTIONS`** environment variable can be used to pass diagnostic options to the underlying MQIPT JRE by using the command line. All command parameters that are valid for the IBM JRE supplied with MQIPT can be used.

You can use the common diagnostic option `-Djavax.net.debug=all` to enable diagnostics for SSL/TLS and network throughput. Setting this option causes a detailed log of internal network operations to be written to the console where MQIPT was started. This is useful for debugging SSL/TLS handshake errors on routes with **`SSLClient`** or **`SSLServer`** set to `true`.

Here is an example of enabling this setting on AIX and Linux platforms:

```
MQIPT_JVM_OPTIONS="-Djavax.net.debug=all"
export MQIPT_JVM_OPTIONS
```

Here is an example of enabling this setting on Windows platforms:

```
set MQIPT_JVM_OPTIONS=-Djavax.net.debug=all
```

For this setting to take effect, you must restart MQIPT from the command prompt where the environment variable is set.

Troubleshooting IBM MQ MQI client application problems

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 completion and 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.

- Linux AIX On AIX and Linux systems, these files are found in the `/var/mqm/errors` directory
- Windows 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\MQ\errors`.
- IBM i 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.

ECONNRESET error over an IBM MQ MQI client channel connection

You are receiving message AMQ9208 or AMQ9209 intermittently from TCP/IP on an IBM MQ MQI client channel connection to a local server, which is an ECONNRESET error.

An ECONNRESET TCP/IP error is caused by a connection reset by peer. This occurs when an established connection is shut down for some reason by the remote computer.

Symptom

The most common scenario shows the error:

AMQ9208I:
Error on receive from host <*hostname*>.

Explanation
An error occurred receiving data from <*hostname*> over TCP/IP. This might be due to a communications failure.

Action
The return code from the TCP/IP recv() call was <xxxxx>. Record these values and tell the systems administrator.

You might receive a different message, for example:

AMQ9209I
Connection to host <*hostname*> for channel <*channelname*> closed.

The following table shows the return codes for different operating systems for the error *ECONNRESET Connection Reset by Peer:*

Operating system	Decimal	Hexadecimal
AIX	73	x49
IBM i	3426	xD62
Linux	104	x68
Windows	10054	x2746
z/OS	1121	x461

Diagnosing the problem

An ECONNRESET error usually indicates a problem in the TCP/IP network.

There are numerous reasons TCP/IP sends a reset:

- A connection termination that is not orderly, such as a rebooting the client box, can cause a reset.
- An application request a connect to a port and IP address for which no server is listening.
- An application closes a socket with data still in the application receive buffer. The connection is reset to allow the remote partner to know that the data was not delivered.
- Any data that arrives for a connection that has been closed can cause a reset.
- An application closes a socket and sets the linger socket option to zero. This notifies TCP/IP that the connection should not linger.

  Note: IBM MQ does not code the linger socket option, therefore IBM MQ does not cause a reset.
- A TCP segment that is not valid arrives for a connection, for example, a bad acknowledge or sequence number can cause a reset.
- The connect request times out. TCP gives up trying to connect to an particular port and IP address and resets the connection.
- A firewall can reset connections if the packet does not adhere to the firewall rules and policies. For example, a source or destination port, or IP address does not match the firewall rule or policy.
- The retransmit timer expires. TCP gives up trying to retransmit a packet and resets the connection.
- A bad hardware device can cause resets

Diagnostic hints and tips:

Consult with your network administrator, who can use TCP/IP packet and a sniffer traces to determine why the reset occurred.

z/OS For z/OS, see :

- Instructions for setting a SLIP on an IBM MQ error message and including TCP/IP CTRACE and TCP/IP PACKET trace
- z/OS UNIX reason codes for the last two bytes of the reason code found in the CSQX208E error message.

Resolving the problem

These types of errors are not generated by IBM MQ. IBM MQis simply informing you that the network is having a problem.

Important: This problem is beyond the scope of IBM Support and there is nothing that can be done from the IBM MQ perspective to resolve this network problem. You need to work with your network support team.

See Automatic client reconnection for more information, which could be useful for your development team to use in IBM MQ MQI client applications.

Troubleshooting IBM MQ .NET problems

Troubleshooting information to help you solve a problem with running IBM MQ .NET applications.

Related tasks

"Tracing IBM MQ .NET applications" on page 471
There are several different ways to enable trace for IBM MQ .NET applications. You normally need to use the trace facility only at the request of IBM Support.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Troubleshooting XMS .NET problems" on page 252
Troubleshooting information to help you solve a problem with running XMS .NET applications.

Developing .NET applications

Troubleshooting with .NET sample applications and error messages

You can use the .NET sample applications and error messages to help with troubleshooting problems.

Procedure

- If an IBM MQ .NET application does not complete successfully, run one of the .NET sample applications, and follow the advice given in the diagnostic messages.

  These sample applications are described in Sample applications for .NET.

  If the problems continue and you need to contact IBM Support, you might be asked to turn on the trace facility. For information on using the trace facility, see "Tracing IBM MQ .NET applications" on page 471.
- If the `An unhandled exception of type System.IO.FileNotFoundException occurred in unknown module` error occurs for either `amqmdnet.dll` or `amqmdxcs.dll`, take one of the following actions:
  - Ensure that both are registered in the global assembly cache.
  - Create a configuration file that points to the `amqmdnet.dll` and `amqmdxcs.dll` assemblies.

  You can examine and change the contents of the assembly cache using mscorcfg.msc, which is supplied as part of .NET Framework.

  If .NET Framework was unavailable when IBM MQ was installed, the classes might not be registered in the global assembly cache. You can manually rerun the registration process using the command

  ```
  amqidnet -c MQ_INSTALLATION_PATH\bin\amqidotn.txt -l logfile.txt
  ```

MQ_INSTALLATION_PATH represents the high-level directory in which IBM MQ is installed.

Information about this installation is written to the specified log file, which is `logfile.txt` in this example.

Related tasks

Installing IBM MQ classes for .NET

Common SSL error codes thrown by IBM MQ .NET client libraries

From IBM MQ 9.4.0, improvements to the IBM MQ .NET client (`amqmdnetstd.dll`) diagnostic messages provide a more specific exception mechanism for SSL-related issues.

From IBM MQ 9.4.0, the IBM MQ .NET client libraries provide meaningful and specific errors for SSL-related errors. The MQRC reason codes are in line with the other .NET client libraries, such as C.

The following table shows some of the many scenarios where, from IBM MQ 9.4.0, IBM MQ .NET client libraries throw errors that are aligned with other client .NET libraries:

Table 2. Comparison of error messages

Scenario	Error message before IBM MQ 9.4.0	Error message from IBM MQ 9.4.0
When the parameter provided for the SSL key repository, MQC.MQCA_SSL_KEY_REPOSITORY or MQEnvironment.SSLKeyRepository, is given incorrectly.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2381 – MQRC_KEY_REPOSITORY_ERROR
When an invalid cipher suite is set on the client while connecting to the IBM MQ server using SSL.	2538 – MQRC_HOST_NOT_AVAILABLE	2393 – MQRC_SSL_INITIALIZATION_ERROR
When an invalid peer name is set on the client while connecting to the IBM MQ server using SSL.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2399 – MQRC_SSL_PEER_NAME_ERROR
When peer names do not match while connecting to the IBM MQ server using SSL.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2398 – MQRC_SSL_PEER_NAME_MISMATCH
When an invalid certificate is set on either the client or server during the SSL handshake.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_SSL_INITIALIZATION_ERROR
When the channel that is used to provide an SSL connectivity between the client and server does not provide a cipher suite.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_SSL_INITIALIZATION_ERROR
When there is a mismatch between the cipher suite algorithms set from the client and the IBM MQ server.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_SSL_INITIALIZATION_ERROR
When there is no permission to the folder where IBM MQ key repository is created.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_SSL_INITIALIZATION_ERROR

For a more detailed explanation of the errors, refer to the queue manager logs found in *MQ_DATA_DIRECTORY*`/qmgrs/errors/AMQERR*.log`.

Related concepts

Installing IBM MQ classes for .NET

V 10.0.0 Troubleshooting issues with .NET 10 runtime for IBM MQ classes for .NET

From IBM MQ 9.4.5, supports .NET 10 at runtime for applications using IBM MQ classes for .NET.

About this task

When IBM MQ classes for .NET applications run against the new runtime or component, existing binaries might encounter a breaking change in behavior, such as failure to load or execute, and they might require recompilation.

- If you are using a .NET 6 or .NET 8 application, you can run this application without any recompilation being required by making a small edit in the `runtimeconfig` file to set the `targetframeworkversion` to `"net10.0"`. For more information, see Breaking changes in .NET 10.

Troubleshooting Java and JMS problems

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.

Related concepts

Using IBM MQ classes for JMS

Using the IBM MQ resource adapter

Using IBM MQ classes for Java

Related tasks

"Tracing JMS/Jakarta Messaging and Java applications" on page 476

The trace facilities for JMS/Jakarta Messaging and Java applications are provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

"Collecting troubleshooting information" on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Troubleshooting IBM MQ classes for JMS problems

You can investigate problems by running the installation verification programs, and by using the trace and log facilities.

Procedure

- If an application does not complete successfully, run one of the installation verification programs and follow the advice given in the diagnostic messages.

  For more information, see The point-to-point IVT for IBM MQ classes for JMS and The publish/subscribe IVT for IBM MQ classes for JMS.

Related tasks

"Tracing IBM MQ classes for JMS applications" on page 476

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.

Logging errors for IBM MQ classes for JMS

By default, log output is sent to the `mqjms.log` file. You can redirect it to a specific file or directory.

About this task

The IBM MQ classes for JMS log facility is provided to report serious problems, particularly problems that might indicate configuration errors rather than programming errors. By default, log output is sent to the `mqjms.log` file in the JVM working directory.

You can redirect log output to another file by setting the property com.ibm.msg.client.commonservices.log.outputName. The value for this property can be:

- A single path name.
- A comma-separated list of path names (all data is logged to all files).

Each path name can be:

- Absolute or relative.
- `stderr` or `System.err` to represent the standard error stream.
- `sttdout` or `System.out` to represent the standard output stream.

If the value of the property identifies a directory, log output is written to `mqjms.log` in that directory. If the value of the property identifies a specific file, log output is written to that file.

Procedure

- Set the property com.ibm.msg.client.commonservices.log.outputName in the IBM MQ classes for JMS configuration file or as a system property on the **java** command.

  In the following example, the property is set as a system property and identifies a specific file:

  ```
  java -Djava.library.path= library_path
  -Dcom.ibm.msg.client.commonservices.log.outputName=/mydir/mylog.txt
  MyAppClass
  ```

  In the command, *library_path* is the path to the directory containing the IBM MQ classes for JMS libraries (see Configuring the Java Native Interface (JNI) libraries).

  The values `System.err` and `System.out` can be set to send log output to the `System.err` and `System.out` streams.
- To disable log output, set the property com.ibm.msg.client.commonservices.log.status to **OFF**. The default value of this property is **ON**.

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 a 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 a 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 a queue manager that is using IBM MQ messaging provider Normal or Version 8 mode. You might, for example, receive this error if you are connecting by using migration mode.
- **Solution:** Change the application to not use the JMS 2.0 API, or ensure that the application connects to a 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 attempted 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 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 that are 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, but 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 must deal with issues of numeric encoding - the headers contain length fields that 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 must 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 that tells 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 How MDB listener ports use 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 (Jakarta Messaging 3.0)

JM 3.0

```
**********[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: jakarta.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.jakarta.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:
151)
      at com.ibm.mq.jms.MQSession$FacadeMessageListener.onMessage(MQSession.java:129)
      at com.ibm.msg.client.jms.internal.JmsSessionImpl.run(JmsSessionImpl.java:3236)
      at com.ibm.mq.jms.MQSession.run(MQSession.java:937)
      at com.ibm.mq.jakarta.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
      at
com.ibm.mq.jakarta.connector.inbound.AbstractWorkImpl.run(AbstractWorkImpl.java:233)
      at com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
      at com.ibm.ws.util.ThreadPool$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
```

Top-level pool example (JMS 2.0)

JMS 2.0

```
**********[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 (Jakarta Messaging 3.0)

JM 3.0

```
**********************
[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: jakarta.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$Worker.run(ThreadPool.java:1892)
    Caused by [1] --> Message : jakarta.jms.JMSRuntimeException: Failed to set up new
JMSContext
               Class : class jakarta.jms.JMSRuntimeException
               Stack :
com.ibm.ejs.jms.JMSCMUtils.mapToJMSRuntimeException(JMSCMUtils.java:522)
                  :
com.ibm.ejs.jms.JMSContextHandle.setupInternalContext(JMSContextHandle.java:241)
                  :
com.ibm.ejs.jms.JMSManagedConnection.getConnection(JMSManagedConnection.java:783)
                  :
com.ibm.ejs.j2c.MCWrapper.getConnection(MCWrapper.java:2336)
                  :
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:1064)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContextInternal(JMSConnectionFactoryHandle.java:4
43)
                  :
com.ibm.ejs.jms.JMSConnectionFactoryHandle.createContext(JMSConnectionFactoryHandle.java:335)
                  :
sib.test.svt.lite.mdb.xa.SVTMDBBase.sendReplyMessage(SVTMDBBase.java:554)
                  :
sib.test.svt.lite.mdb.xa.QueueReceiverBean.onMessage(QueueReceiverBean.java:128)
                  :
sib.test.svt.lite.mdb.xa.MDBProxyQueueReceiver_37ea5ce9.onMessage(MDBProxyQueueReceiver_37ea5ce9.j
ava:-1)
                  :
com.ibm.mq.jakarta.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:
151)
                  :
com.ibm.mq.jms.MQSession$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.jakarta.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
                  :
com.ibm.mq.jakarta.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.jakarta.connector.inbound.MessageEndpointWrapper.onMessage(MessageEndpointWrapper.java:
151)
                     :
com.ibm.mq.jms.MQSession$FacadeMessageListener.onMessage(MQSession.java:129)
                     :
com.ibm.msg.client.jms.internal.JmsSessionImpl.run(JmsSessionImpl.java:3236)
                     : com.ibm.mq.jms.MQSession.run(MQSession.java:937)
                     :
com.ibm.mq.jakarta.connector.inbound.ASFWorkImpl.doDelivery(ASFWorkImpl.java:104)
                     :
com.ibm.mq.jakarta.connector.inbound.AbstractWorkImpl.run(AbstractWorkImpl.java:233)
                     : com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:668)
                     : com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1892)
```

Lower-level pool example (JMS 2.0)

JMS 2.0

```
**********************
[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 three 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 3: ReadAhead is enabled for the destination from which the activation specification or listener port is consuming messages

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 IBM MQ for Multiplatforms, 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.

Reason 3: ReadAhead is enabled for the destination from which the activation specification or listener port is consuming messages

ReadAhead operates:

- On Non-persistent messages outside of syncpoint
- When browsing messages

Browsing messages is what an IBM MQ activation specification or WebSphere Application Server listener port does, before destructively consuming the message.

As a consequence, rather than browsing just a single message when the activation specification browsing thread starts, it might browse (for example) 100 messages in one go, and then slowly feed them to threads to destructively be consumed from the queue, and given to the application message driven bean class to process.

As the default value of the queue manager property Message mark browse interval (MARKINT), for browsed messages to be consumed before being browsed again, is five seconds, this typically results in the same message being browsed a second time by the activation specification, or by another browsing thread if there are multiple Java Extended Edition servers running an activation specification against the same queue, before the message driven bean has had time to process all of the messages which were initially browsed.

Only the first destructively consuming thread gets the message. All subsequent destructively consuming threads are given an MQRC 2033 return code from the queue manager, which is turned into the JMSCC0108 message. Also, before returning the 2033 return code, the queue manager searches for the message across the entire queue which, if the queue is deep - some thousands of messages, can take a significant amount of time.

CWSJY0003W warning messages in WebSphere Application Server SystemOut.log file

A CWSJY0003W warning message is logged in the WebSphere Application Server SystemOut.log file when an MDB processes JMS messages from IBM MQ.

Symptom

CWSJY0003W: IBM 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 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 MQ using IBM 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 reference

Activation specifications

Listener ports running in Application Server Facilities (ASF) mode

Listener ports running in non Application Server Facilities (non-ASF) mode

Related information

Avoiding repeated delivery of browsed messages

ALTER QMGR

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 IBM 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 an IBM 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 IBM 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 an IBM MQ queue manager.

In this situation, there is a single transaction that is using two connections to IBM MQ:

- A connection which is used to get a message from IBM 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 IBM 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 IBM MQ.

However, IBM 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 IBM 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 IBM 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 an IBM MQ queue manager, and then create a new connection to IBM 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 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" on page 114.

  Windows 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.`

  Linux AIX For AIX and Linux, 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 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`.
- The presence of an Advanced Message Security security policy.

For the location of the IBM MQ error logs, see Error log directories.

The default authentication method that is used by IBM MQ classes for JMS applications that use client transport changed from compatibility mode to MQCSP authentication in IBM MQ 9.3.0. Also, the methods that can be used to select a specific authentication mode, and which methods take precedence, has been clarified in IBM MQ 9.3.0. This might cause applications that previously connected successfully to a queue manager to fail to connect with reason code *2035 MQRC_NOT_AUTHORIZED*, as the authentication method that is used by the application might change when the IBM MQ classes for Java or IBM MQ classes for JMS are upgraded to IBM MQ 9.3.0 or later. For more information about the authentication mode used by Java clients, see Connection authentication with the Java client.

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:

- Linux AIX *mqm* on AIX and Linux systems.
- Windows *MUSR_MQADMIN* on Windows.

As these users are administrative IBM MQ users, they are blocked by default 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 AIX and Linux), 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:

JM 3.0
```
@Resource(name = "jms/MyResourceRef"
    authenticationType = AuthenticationType.CONTAINER)
private jakarta.jms.ConnectionFactory myCF
```

JMS 2.0
```
@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 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 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 IBM 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 an 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.
- Compatible CipherSpecs must be set for 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 an IBM 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.

Related reference

“Return code= 2035 MQRC_NOT_AUTHORIZED” on page 206
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.

2035 (07F3) (RC2035): MQRC_NOT_AUTHORIZED

Troubleshooting IBM MQ resource adapter problems

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:

```
jakarta.jms.JMSException: MQJCA0001: An exception occurred in the JMS layer. See the
linked exception for details.
```

```
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:

```
jakarta.jms.JMSException: MQJMS2005: failed to create an MQQueueManager for
'localhost:ExampleQM'
```

```
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 Configuring 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 tasks

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 concepts

What is installed for IBM MQ classes for JMS

Related tasks

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 a TLS connection, the TLS-related properties, or the TLS-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 tasks

Specifying that only FIPS certified CipherSpecs are used at run time on the MQI client

Related reference

Federal Information Processing Standards (FIPS) for AIX, Linux, and Windows

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 tasks

"Tracing IBM MQ classes for JMS applications" on page 476
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 IBM MQ classes for Java applications" on page 481
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.

Using IBM MQ classes for JMS

Using IBM MQ classes for Java

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 122
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 126
This example shows how to override properties when you are using the IBM MQ classes for JMS.

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.

Each property has an identifier - the property name - which is a character string literal. For example, the property which specifies the host name to IBM MQ is identified by the literal value `"hostname"`.

To define the application name for your application, in your Java code you can use code similar to this:

```
Hashtable properties = new Hashtable();
properties.Add("hostname", "localhost" );
MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
```

However, the literal value is part of the IBM MQ classes for Java internal implementation. In case the literal part ever changes (although this is unlikely) rather than using the literal value you should use the corresponding constant value, defined in the `MQConstants` class.

The constant is part of the documented external interfaces for IBM MQ classes for Java and will not change.

For host name this constant is `HOST_NAME_PROPERTY`, so the preferred code is:

```
Hashtable properties = new Hashtable();
properties.Add( MQConstants.HOST_NAME_PROPERTY, "ExampleApplName" );
MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
```

The full set of properties that can be set in a program is shown in the following table:

Property	**Constant name in `MQConstants`**
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
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
Application name	APPNAME_PROPERTY

Note: The table does not list the literal values because, as already noted, they are part of the IBM MQ classes for Java implementation and could change.

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**, you need to carry out the following steps:

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.

Table 3. 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
Application name	$APPNAME_PROPERTY

   Notes:

   a. 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.

   b. Property keys starting with the $ character are processed by reference to the constant values defined in `MQConstants.java`, as discussed in the preceding text.

      You can, but should not, use the literal values of these constants, in which case the $ character is omitted

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
   ```

   Applications that need to set a specific application name with IBM MQ can do so in one of three ways:

   - Using the override mechanism described in the preceding text, define the **mqj.$APPNAME_PROPERTY** property.

     The value of the **mqj.$APPNAME_PROPERTY** property specifies the name used to identify the connection to the queue manager, with only the first 28 characters being used. For example:

     ```
     mqj.$APPNAME_PROPERTY=ExampleApplName
     ```

     Note: You might see examples using the literal value of the property name, for example in older documentation. For example, `mqj.APPNAME=ExampleApplName`.

   - You can pass this value to the **MQQueueManager** constructor in the **properties** HashTable, with only the first 28 characters being used. For example:

```
Hashtable properties = new Hashtable();
properties.Add( MQConstants.APPNAME_PROPERTY, "ExampleApplName" );
MQQueueManager qMgr = new MQQueueManager("qmgrname", properties);
```

   - You can set the *AppName* property in the `MQEnvironment` class, with only the first 28 characters being used. For example:

     ```
     MQEnvironment.AppName = "ExampleApplName";
     ```

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
   ```

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 120
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 120
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 122

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.

Troubleshooting Managed File Transfer problems

Use this information to help you to diagnose errors in Managed File Transfer (MFT).

Related tasks

"Tracing Managed File Transfer resources on Multiplatforms" on page 490
The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

"Tracing Managed File Transfer for z/OS resources" on page 496
The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

"Collecting information for Managed File Transfer problems on Multiplatforms" on page 345
If you need assistance to resolve a problem when a Managed File Transfer (MFT) agent, logger or command is reporting a problem or failing to work properly on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem. The information that is needed depends on the problem that you are seeing.

"Collecting information for Managed File Transfer for z/OS problems" on page 382
If you need assistance to resolve a problem when a Managed File Transfer (MFT) for z/OS agent, logger or command is reporting a problem or failing to work properly you might need to collect troubleshooting information to include with your support case to help find a solution to the problem. The information that is needed depends on the problem that you are seeing.

Related reference

Return codes for MFT

MFT commands

Hints and tips for using MFT

Here are some suggestions to help you to make best use of Managed File Transfer

- If you change the `agent.properties` file, stop and restart the agent to pick up the changes.
- If you start a file transfer and there is no sign of transfer progress and no errors are reported, check that the source agent is running. If the transfer is shown but does not progress, check that the destination agent is also running. You can check the current state of agents in the agent log or verify that the agent is active with an **ftePingAgent** command.
- When you cancel an individual transfer using the **fteCancelTransfer** command, you can use either the source or destination agent in the **-agentName** parameter. However, when you delete a transfer schedule using the **fteDeleteScheduledTransfer** command, you must use the source agent name in the **-agentName** parameter.
- When you create a file transfer the source and destination file paths, either absolute or relative, are significant only on the source and destination agents. The system and directory that the **fteCreateAgent** command is issued from has no relevance to the file being transferred.
- Your default environment setup might not be able to fully support Managed File Transfer, particularly if you are running multiple concurrent transfers. If an agent has an error indicating it has run out of memory, check and update the following parameters as required:
  - Linux AIX For AIX and Linux platforms: run the command: `ulimit -m 1048576` (or approximately 1 GB). This maximum resident set size is enough to allow a maximum of 25 concurrent transfers (25 concurrent transfers is the default for the maximum number of transfers for an agent).
  - For all platforms: set the **BFG_JVM_PROPERTIES** environment variable as follows: `BFG_JVM_PROPERTIES="-Xmx1024M"`

If you want to allow numbers of concurrent transfers greater than the maximum default of 25, use larger sizes for **ulimit** and **BFG_JVM_PROPERTIES** than those suggested.

Note: For Connect:Direct® bridge agents the default for the maximum number of concurrent transfers is 5.

- You can set an operating system environment variable called **BFG_JVM_PROPERTIES** to manipulate your agent JVM. You can use this environment variable to directly pass properties to the JVM. However, if you set this environment variable to an incorrect value, it causes a JVM initialization failure for your agent.

  AIX and Linux, set **BFG_JVM_PROPERTIES** by using the following command :

  ```
  export BFG_JVM_PROPERTIES="......"
  ```


  On Windows, set **BFG_JVM_PROPERTIES** by using the following command :

  ```
  set BFG_JVM_PROPERTIES=......
  ```

  Note: Every MFT command checks if BFG_JVM_PROPERTIES variable has been set, then the value of the variable is passed on as parameters while launching agent, logger, or MFT commands
- When you use Managed File Transfer to transfer files in text mode between different platforms, the default file encoding of the source platform might not be supported by the destination platform. This causes a transfer to fail with the following error:

  ```
  BFGIO0058E: The transfer source encoding xxx is illegal or for an unsupported character set.
  ```

  You can resolve this error by setting the source encoding to one that is supported by the destination platform using an environment variable. Set the **BFG_JVM_PROPERTIES** system environment variable on the source system as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=`*xxx*`"`, where *xxx* is an encoding supported by the destination platform. For example, if you are transferring files in text mode to a different platform and the source locale is set to "ja", set **BFG_JVM_PROPERTIES** as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=EUC-JP"`. If the source locale is set to "ja_JP.PCK", set **BFG_JVM_PROPERTIES** as follows: `BFG_JVM_PROPERTIES="-Dfile.encoding=Shift_JIS"`.

  You can also resolve this error for an individual transfer by using the **-sce** parameter when you start a new transfer. For more information, see **fteCreateTransfer**: start a new file transfer.
- Where possible, do not use a single agent as both the source agent and destination agent for the same managed transfer. This puts extra load on the agent, which can impact other managed transfers that it is participating in and cause those transfers to go into recovery.

Related reference

Java system properties for MFT

Troubleshooting MFT agent status problems

Use these tasks, logs and links to help you to resolve issues with the status of Managed File Transfer (MFT) agents.

About this task

Steps for troubleshooting MFT agents are described in the child tasks and the related links.

Each MFT agent also writes to a `stdout.log` and a `stderr.log` file. These files are located in the agent's logs directory.

- The `stdout.log` file is used to record general information about MFT agent activity, notably messages related to agent startup and execution status. For example, for an agent that started successfully a message similar to the following is logged:

  ```
  BFGCL0030I: The request to start agent '86' on this machine has been submitted.
  ```

This log is not a complete audit trail. For example, no message is written to this log when the **fteStopAgent** command is issued, or when the agent is stopped and restarted.

- The `stderr.log` file is used to log error messages generated by the MFT agent, including those related to system or application-level issues.

When an MFT agent is started, the launch scripts or service often redirect the `stdout` and `stderr` streams to these log files. Messages such as `BFGCL0030I` are emitted by the command shell wrapper or the controller process, depending on how the agent is started (either by using the CLI, or as a managed service). Native errors and uncaught Java exceptions are sent to the `stderr.log` file.

Related tasks

“Collecting information for MFT agent problems” on page 346
The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) agent problem.

“Troubleshooting common MFT problems” on page 166
A checklist of common MFT problems that are usually straightforward to define and fix.

Configuring MFT Agent queue managers

Related reference

Return codes for MFT

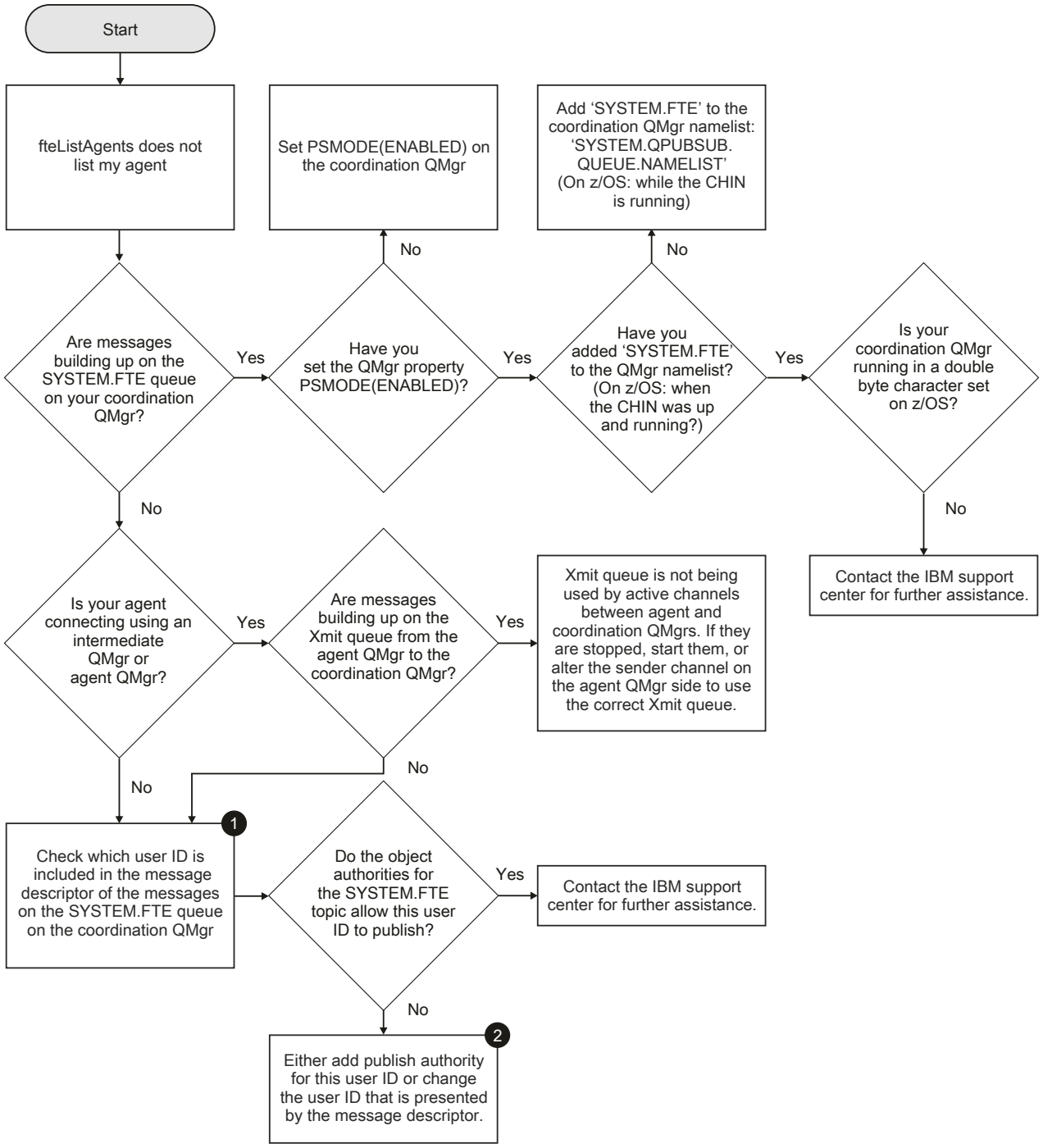
Troubleshooting an MFT agent that is not listed by the `fteListAgents` command

If your agent is not listed by the **fteListAgents** command or is not displayed in IBM MQ Explorer, or your file transfers are not displayed in the **Transfer Log** of IBM MQ Explorer, use a flowchart to investigate the cause.

About this task

Use the following flowchart to help you to diagnose problems and decide what action to take next.

Flowchart:



Flowchart key:

1. For more information about how to check the user ID that is presented, see "Examining messages before publication" on page 171. User IDs must conform to the MQ user name 12 character limit. If a user name is longer than 12 characters (`Administrator`, for example) the user name will be truncated before being checked for authorisation. In an example using `Administrator`, the following error message is added to the queue manager error log:

   ```
   AMQ8075: Authorization failed because the SID for entity 'administrato' cannot be obtained.
   ```

2. For more information about the authority needed for the SYSTEM.FTE queue, see Authority to publish MFT Agents log and status messages.

Troubleshooting an MFT agent that is in an UNKNOWN state

Your agent is running and responds successfully to the **ftePingAgent** command, and items are being transferred normally. However, the **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer Managed File Transfer plug-in, report the agent as being in an UNKNOWN state.

About this task

Periodically, each agent publishes its status to the SYSTEM.FTE topic on the coordination queue manager. How frequently an agent publishes its status is controlled by the following agent properties:

agentStatusPublishRateLimit
The maximum rate, in seconds, that the agent republishes its status because of a change in file transfer status. The default value of this property is 30 seconds.

agentStatusPublishRateMin
The minimum rate, in seconds, that the agent publishes its status. This value must be greater than or equal to the value of the **agentStatusPublishRateLimit** property. The default value for the **agentStatusPublishRateMin** property is 300 seconds (or 5 minutes).

The **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer Managed File Transfer (MFT) plug-in, use these publications to determine the status of an agent. To do this, the commands and the plug-in perform the following steps:

1. Connect to the coordination queue manager.
2. Subscribe to the SYSTEM.FTE topic.
3. Receive agent status publications.
4. Create a temporary queue on the coordination queue manager.
5. Put a message to the temporary queue, and save the put time in order to get the current time on the coordination queue manager system.
6. Close the temporary queue.
7. Use the information contained within the publications, and the current time, to determine the status of an agent.
8. Disconnect from the coordination queue manager.

The status message of an agent is considered stale if the difference between the time that it was published, and the current time, is greater than: *The value of the agent property **agentStatusPublishRateMin** (included in the status message) plus the value of the advanced coordination queue manager property **agentStatusJitterTolerance***.

By default, the **agentStatusJitterTolerance** property has a value of 3000 milliseconds (3 seconds).

If the **agentStatusPublishRateMin** and **agentStatusJitterTolerance** properties are set to their default values, then the status of an agent is considered stale if the difference between the time that it was published, and the current time, is greater than 303 seconds (or 5 minutes 3 seconds).

Any agent with a stale status message is reported by the **fteListAgents** and **fteShowAgentDetails** commands, and the IBM MQ Explorer MFT plug-in, as being in an UNKNOWN state.

The status publication of an agent can be stale for one of the following reasons:

1. There is a significant difference in the system time between the system where the agent queue manager is running, and the system where the coordination queue manager is located.
2. The channels between the agent queue manager and the coordination queue manager are stopped (which prevents new status messages from reaching the coordination queue manager).
3. An authorization issue is preventing the agent from publishing its status to the SYSTEM.FTE topic on the coordination queue manager.
4. An agent failure has occurred.

To determine why the status of an agent is being reported as UNKNOWN, complete the following steps:

Procedure

1. Check whether the agent is running, by logging on to the agent system. If the agent is stopped, then investigate why it is no longer running. When it is running again, check whether its status is now being reported correctly.
2. Check that the coordination queue manager is running. If it is not, restart it and then use the **fteListAgents** or **fteShowAgentDetails** command, or the IBM MQ Explorer MFT plug-in, to see if the agent status is now being reported correctly.
3. If the agent and the coordination queue managers are running, check the value of the *Status Age* value for the agent in the **fteListAgents** output or the IBM MQ Explorer MFT plug-in.

   This value shows the difference between the time that the status message of the agent was published, and the time that the status message was processed.

   If the difference is:

   - Always slightly higher than *the value of the agent property* ***agentStatusPublishRateMin*** *(included in the status message) plus the value of the advanced coordination queue manager property* ***agentStatusJitterTolerance*** , consider increasing the value of the **agentStatusJitterTolerance** property. This introduces a slight tolerance to allow for a delay in between the status publications being received and processed, as well as allowing for a difference in the system clocks between the agent queue manager and the coordination queue manager systems.
   - More than 10 minutes higher than *the value of the agent property* ***agentStatusPublishRateMin*** *(included in the status message) plus the value of the advanced coordination queue manager property* ***agentStatusJitterTolerance***, and continues to increase each time the status of the agent is checked, then the status messages from the agent are not reaching the coordination queue manager.

     In this situation, the first thing to do is check the error logs for the agent queue manager and the coordination queue manager to see if there are any authorization issues that are preventing the agent from publishing its status messages. If the logs show that authorization issues are occurring, then ensure that the user running the agent process has the correct authority to publish messages to the SYSTEM.FTE topic on the coordination queue manager.

     If the error logs of the queue manager do not report any authorization issues, check that the status messages have not got stuck in the IBM MQ network. Verify that all of the sender and receiver channels that are used to route the messages from the agent queue manager to the coordination queue manager are running.

     If the channels are running, then check the transmission queues associated with the channels, to make sure that the status messages are not stuck on them. Also, you should check any dead letter queues for the queue managers to make sure that the status messages have not been placed there for some reason.
4. If the channels are running, and the status messages are flowing through the IBM MQ network, then check that the queue manager's queued publish/subscribe engine is picking up the messages.

   The **fteSetupCoordination** command, which is used to define the coordination queue manager, provides you with some MQSC commands that must be run on the coordination queue manager to configure the queued publish/subscribe engine to receive publications. These commands perform the following steps:

   - Create the SYSTEM.FTE topic and its associated topic string.
   - Define a local queue called SYSTEM.FTE that will be used to receive incoming status messages.
   - Enable the queued publish/subscribe engine, by setting the **PSMODE** attribute on the queue manager to ENABLED.
   - Modify the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist, which is used by the queued publish/subscribe engine, so that it includes an entry for the new SYSTEM.FTE queue.

   For more information on this, including the MQSC commands that need to be run, see fteSetupCoordination (set up properties files and directories for coordination queue manager).

If there are messages on the SYSTEM.FTE queue, then you should check that the SYSTEM.QPUBSUB.QUEUE.NAMELIST namelist has been set up correctly and contains an entry for that queue. If the entry is missing, then the queued publish/subscribe engine will not detect any incoming status messages from the agent and will not process them.

You should also ensure that the **PSMODE** attribute on the queue manager is set to ENABLED, which turns on the queued publish/subscribe engine.

5. If the channels are running, and the status messages are flowing through the IBM MQ network and are being picked up from the SYSTEM.FTE queue by the queue manager's queued publish/subscribe engine, then collect the following traces:

   - An IBM MQ MFT trace from the agent, covering a time period equal to three times the value of the agent property **agentStatusPublishRateMin**. This ensures that the trace covers the time when the agent is publishing at least three messages containing its status. The trace should be collected dynamically, using the trace specification:

     ```
     com.ibm.wmqfte.statestore.impl.FTEAgentStatusPublisher,
     com.ibm.wmqfte.utils.AgentStatusDetails,
     com.ibm.wmqfte.wmqiface.AgentPublicationUtils,
     com.ibm.wmqfte.wmqiface.RFHMessageFactory=all
     ```

     Note: A reduced amount of trace is output using these strings.

     Multi For information on how to enable the trace for agents running on IBM MQ for Multiplatforms, see “Collecting a Managed File Transfer agent trace dynamically” on page 491.

     z/OS For information on how to enable the trace for agents running on IBM MQ for z/OS, see “Collecting a Managed File Transfer for z/OS agent trace dynamically” on page 499.

   - A concurrent trace of the queue managers used to route the status messages from the agent queue manager to the coordination queue manager.
   - A trace of the **fteListAgents** command, covering the time when the agent is shown as being in an UNKNOWN state. The trace should be collected using the trace specification:

     ```
     com.ibm.wmqfte=all
     ```

     Multi For information on how to enable the trace for commands running on IBM MQ for Multiplatforms, see “Tracing Managed File Transfer commands on Multiplatforms” on page 492.

     z/OS For information on how to enable the trace for commands running on IBM MQ for z/OS, see “Tracing Managed File Transfer for z/OS commands” on page 503.

   Once the traces have been collected, they should be made available to IBM Support for analysis. For more information, see “Collecting troubleshooting information” on page 310.

What to do next

View the status age from the command line.

The **Status Age** information of a publication is displayed as part of the output from the **fteListAgents** and **fteShowAgentDetails** commands.

For more information, see **fteListAgents** and **fteShowAgentDetails**.

View the status age in IBM MQ Explorer.

The **Status Age** information is available in the IBM MQ Explorer MFT plug-in when you view the list of agents and display individual agent properties.

Related reference

fteListAgents

fteShowAgentDetails

MFT agent status values
The MFT `agent.properties` file
The MFT coordination.properties file

Troubleshooting an `ftePingAgent` that times out and reports a BFGCL0214I message

`ftePingAgent` is a useful command-line utility that you can use to check whether an agent is reachable, and whether it is able to process requests.

About this task

The command is described in detail here: **`ftePingAgent`**. When it is run, it performs the following steps:

- Connects to the command queue manager for the Managed File Transfer (MFT) topology.
- Creates a temporary reply queue on the command queue manager.

  By default, the temporary queue has a name that starts with the prefix `WMQFTE`. However, you can change this by setting the **`dynamicQueuePrefix`** property in The MFT command.properties file for the installation.
- Sends a Ping MFT agent request message to the queue SYSTEM.FTE.COMMAND.*agent_name* on the agent queue manager, through the command queue manager. The request message contains the name of the temporary reply queue.
- Waits for an MFT agent reply message reply message to arrive on the temporary reply queue.

One of the threads within an agent is the `CommandHandler`. This thread gets messages from the SYSTEM.FTE.COMMAND.*agent_name* queue of the agent, and processes them.

If this thread receives a message containing a Ping MFT agent request, it builds an MFT agent reply message, and sends it to the temporary queue on the command queue manager. This message goes through the agent's queue manager.

After the message arrives on the temporary queue, it is picked up by the **`ftePingAgent`** command. The command then writes a message similar to the one shown below to the console before exiting:

```
BFGCL0793I: Agent <agent_name> running on host <hostname> responded to ping in <time> seconds.
```

The following two diagrams show the flow:

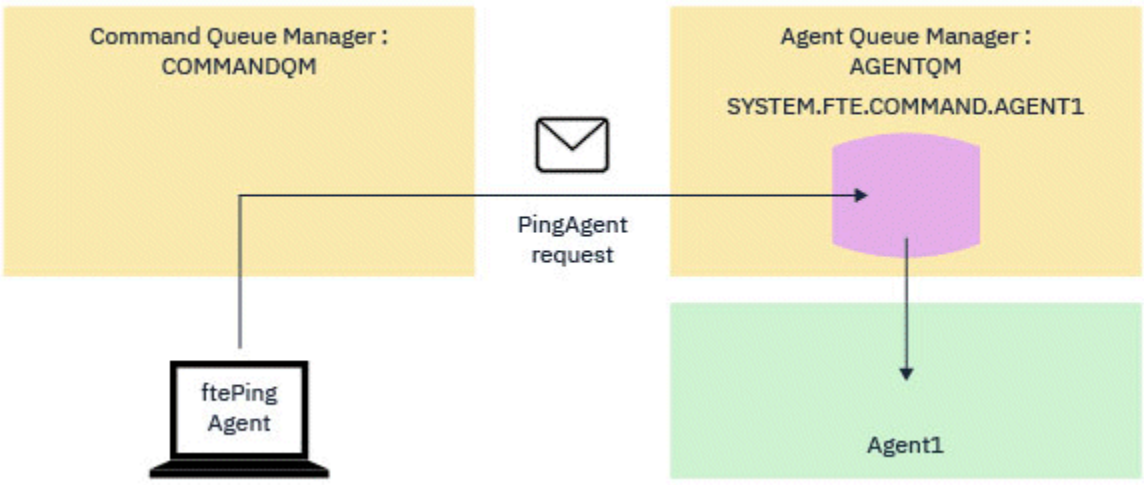


Figure 4. The pingAgent request goes to the SYSTEM.FTE.COMMAND.agent_name queue on the agent queue manger, through the command queue manager

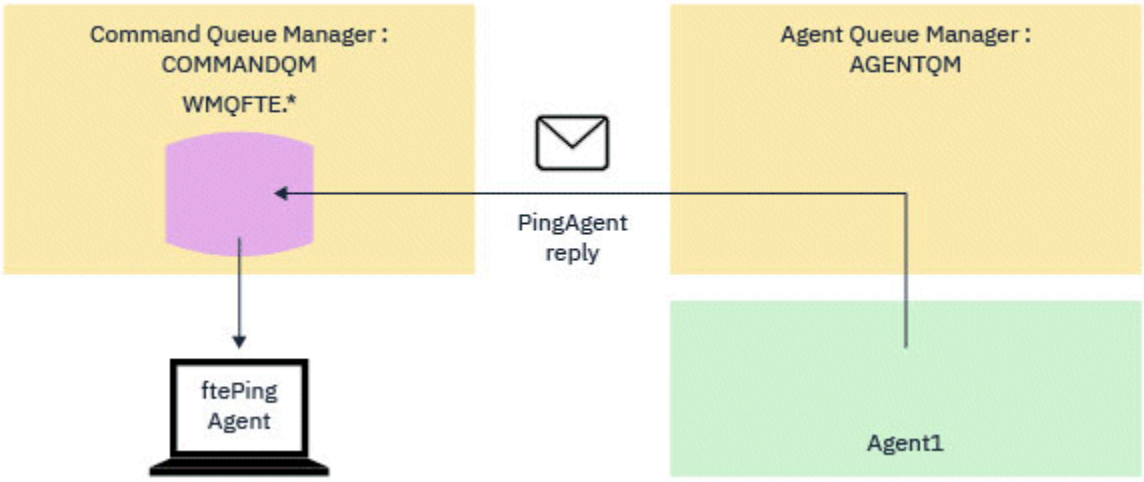


Figure 5. The pingAgent reply comes back through the agent queue manager to the command queue manager.

By default, the **ftePingAgent** command waits for five seconds for the MFT agent reply message to arrive on the temporary queue. If the reply message does not arrive within five seconds, the command writes a BFGCL0214I message to the console. The following message is an example:

```
BFGCL0214I: agent AGENT1 didn't respond to ping after 5 seconds.
```

Procedure

1. Investigate why the reply message did not arrive.
   a) Check that the agent is running. If it is not, it can not respond to the Ping MFT agent request sent by the command.
   b) If the agent is running and busy processing requests, it might take longer than five seconds to get the Ping MFT agent request and send back the reply. To see if this is the case, rerun the **ftePingAgent** command, using the **-w** parameter to specify a longer wait interval.

      For example, to specify a 60 second wait interval, issue the following command:

      ```
      ftePingAgent -w 60 AGENT1
      ```

   c) If the command still times out, check the path through the IBM MQ network between the command queue manager and the agent queue manager. If one or more channels in the path have failed, the Ping MFT agent request message and/or the MFT agent reply message will be stuck on a transmission queue somewhere. In this situation, you should restart the channels and re-run the **ftePingAgent** command.
2. If the command still reports a BFGCL0214I message, track the Ping MFT agent request and MFT agent reply messages as they flow through the IBM MQ network.

   You do this to see if the Ping MFT agent message ever reaches the SYSTEM.FTE.COMMAND.*agent_name* queue, and to see if the agent picks up the message from the queue and sends back an MFT agent reply message.

   a) Enable queue manager traces on both the command and agent queue managers.
   b) Enable trace on the agent dynamically using the trace specification `com.ibm.wmqfte=all`.

      The way to do this depends upon the platform the agent is running on:

      - Multi For agents running on IBM MQ for Multiplatforms, see “Collecting a Managed File Transfer agent trace dynamically” on page 491.

- z/OS For agents running on IBM MQ for z/OS, see “Collecting a Managed File Transfer for z/OS agent trace dynamically” on page 499.

c) Run the **ftePingAgent** command with trace enabled, using the trace specification `com.ibm.wmqfte=all`.

- Multi For information about tracing the command on IBM MQ for Multiplatforms, see “Tracing Managed File Transfer commands on Multiplatforms” on page 492.
- z/OS For information about tracing the command on IBM MQ for z/OS, see “Tracing Managed File Transfer for z/OS commands” on page 503.

3. When the command times out, stop the agent trace and the queue manager trace. Make the agent and queue manager traces, along with the trace from the command, available to IBM Support for analysis.

Troubleshooting managed transfer problems

Use the following tasks to help you to resolve issues with managed transfers.

Related tasks

“Troubleshooting common MFT problems” on page 166
A checklist of common MFT problems that are usually straightforward to define and fix.

Related reference

Return codes for MFT

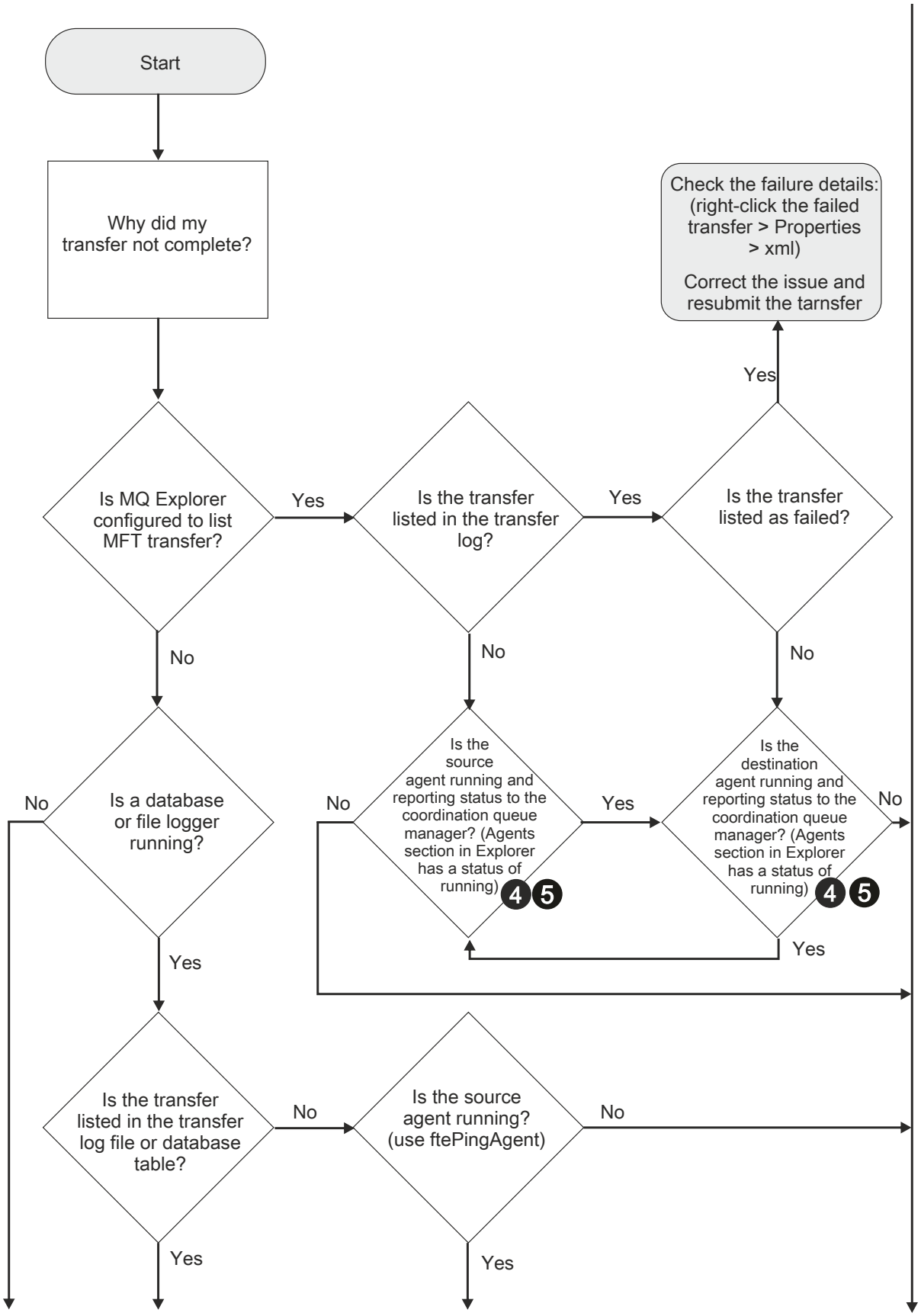
Troubleshooting an MFT transfer that does not complete

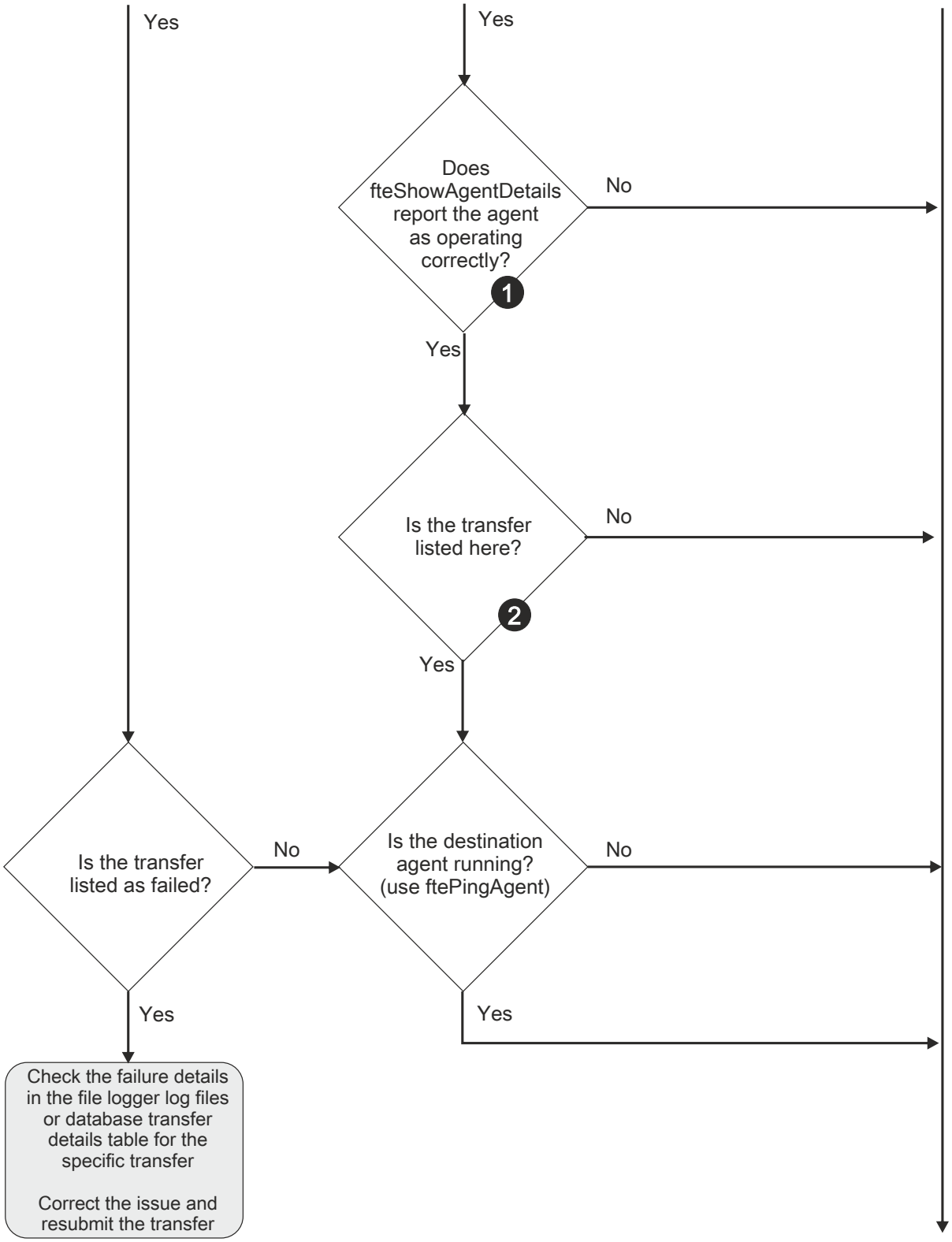
If your transfer does not complete, use a flowchart to investigate the cause..

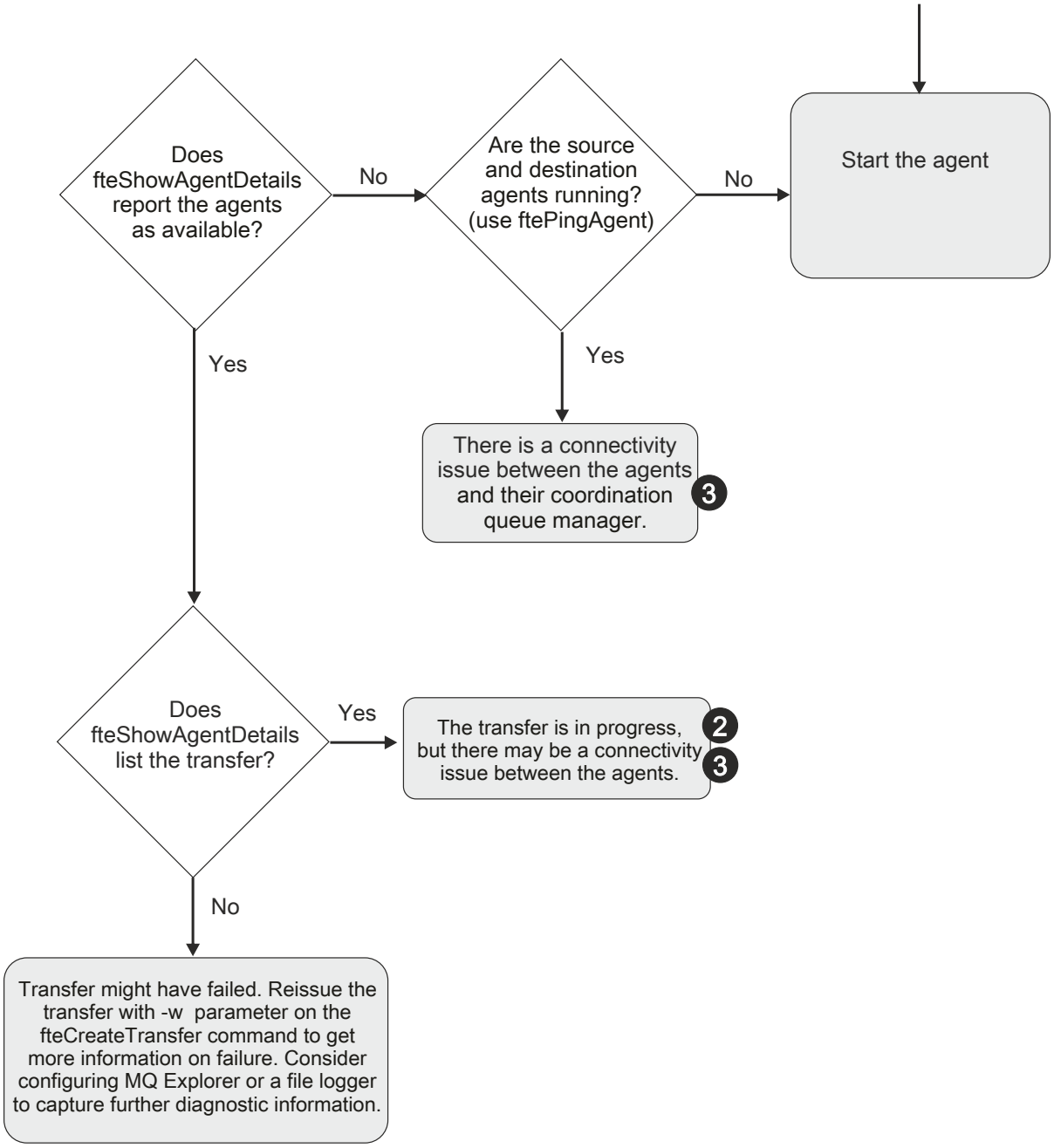
About this task

Use the following flowchart to help you to diagnose problems and decide what action to take next.

Flowchart:


Start
Why did my transfer not complete?
Is MQ Explorer configured to list MFT transfer?
Yes
Is the transfer listed in the transfer log?
Yes
Is the transfer listed as failed?
Yes
Check the failure details: (right-click the failed transfer > Properties > xml)
Correct the issue and resubmit the tarnsfer
No
No
No
Is a database or file logger running?
No
Is the source agent running and reporting status to the coordination queue manager? (Agents section in Explorer has a status of running)
4 5
No
Yes
Is the destination agent running and reporting status to the coordination queue manager? (Agents section in Explorer has a status of running)
4 5
No
Yes
Yes
Is the transfer listed in the transfer log file or database table?
No
Is the source agent running? (use ftePingAgent)
No
Yes
Yes


Yes
Yes
Does fteShowAgentDetails report the agent as operating correctly?
No
1
Yes
Is the transfer listed here?
No
2
Yes
Is the transfer listed as failed?
No
Is the destination agent running? (use ftePingAgent)
No
Yes
Yes
Check the failure details in the file logger log files or database transfer details table for the specific transfer
Correct the issue and resubmit the transfer



Flowchart key:

1. Check the agent `output0.log` for errors. If the agent reports it has successfully started, but neither IBM MQ Explorer nor **fteShowAgentDetails** report the agent as running, then check the connectivity between the agent queue manager and the coordination queue manager. It may be that a queue manager to queue manager channel is unavailable.
2. If the source agent lists the transfer ID as an `In progress` transfer but the destination agent does not, there might be a connectivity issue between the source and destination queue managers. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.
3. If both the source and destination agents list the transfer ID as `In progress`, this suggests there has been a connectivity issue between the source and destination queue managers since the transfer was initiated. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the

`command.properties` file. You can also run this command the other way round, from source to destination.

If the **ftePingAgent** command is successful, check that the sender/receiver channels between the agent queue managers are running to allow messages to flow between them.

4. If you have been round this loop already, check whether either of statements are relevant to your situation:
   - Both source and destination agents report as `Running`, but no transfer is listed. Either the transfer request did not reach the agent command queue, or the agent although reporting as `Running`, is no longer monitoring the command queue. Check for errors in the source agent `output0.log`. Use the **ftePingAgent** command from the same machine the transfer was sent from, to the source agent, to verify the connectivity between the command queue manager and the agent queue manager, and that the agent is servicing the command queue.
   - Both source and destination agents report as `Running`, and the transfer is listed as `In progress, recovering`. Use the **ftePingAgent** command from the destination agent machine to the source agent using the destination agent queue manager as the command queue manager, in the `command.properties` file. You can also run this command the other way round, from source to destination.
5. Check that there is a sender channel between the agent queue manager and the coordination queue manager, and that it is running. If there is no sender channel, or it is not running, then the status publications sent by the agent are not able to flow from the agent queue manager to the coordination queue manager, resulting in the status information stored on the coordination queue manager being out of date.

Troubleshooting a file transfer that might be stuck

On a heavily loaded system, or when there are network problems between the source and destination agents, transfers can occasionally appear to be stuck in a queued or recovering state.

About this task

Transfers are processed by an agent in priority order. Therefore in a loaded system, a low-priority transfer can remain in the queued state for some time while the agent is loaded with higher priority transfers. Eventually a low-priority transfer is started if that transfer has been queued for a while, even though there are newer higher priority transfers.

To determine the cause of the problem, complete the following checks.

Procedure

1. Check whether the source and destination agents are active and responding to new requests.
   a) Run the **ftePingAgent** command, or in the IBM MQ Explorer **Agents** panel right-click on the agent name and select **Ping**.
   b) Look at the agent logs to see if there is a current network connection problem.
2. Check whether the destination agent is running at capacity.

   It might be that there are numerous source agents all requesting file transfers to the same destination agent.

   a) Run the **fteShowAgentDetails** command with the **-v** (verbose) parameter, or in the IBM MQ Explorer **Agents** panel right-click on the agent name and select **Properties**.
   b) Look at the current transfer activity for an agent.

      If the number of running destination transfers is at or close to the agent's maximum number of destination transfers, this might explain why some transfers for source agents appear to be stuck.
3. Look at the agent logs to see if there is a current connection problem.

   Transfers to and from protocol bridge agents enter a recovering state if there is a problem contacting the protocol file server.

Troubleshooting a failed file transfer to a clustered queue

When using Managed File Transfer to transfer a file into a queue, if you use a destination that is a clustered queue, or an alias to a clustered queue, you might get reason code 2085, or 2082. This issue is resolved if you set up a remote queue definition that points to the clustered queue.

About this task

The queue manager name of the destination agent is being appended to the queue name of the **-dq** parameter, when there is no explicit queue manager name on the **-dq**. The reason code 2085, or 2082, occurs because the `queueManager` object cannot be specified on an MQOPEN call when connecting to a clustered MQ `queueManager` that does not have that local clustered queue.

To avoid this problem, complete the following steps:

Procedure

1. Create a clustered queue on the queue manager.
2. Set up a remote queue definition that points to the clustered queue.

Example

This example uses a remote queue definition.

Configuration:

- Source Agent: *SAGENT*
- Source Agent Queue Manager: *SQM*
- Destination Agent: *DAGENT*
- Destination Agent Queue Manager: *DQM*
- The destination queue of the transfer is *CQ6* on queue manager *SQM*

To define remote queue definition Q6_SQM on DQM to clustered queue CQ6 in SQM (assuming that the clustered queue CQ6 is already defined in SQM), issue the MQSC command on the DQM queue manager:

```
define qremote(Q6_SQM) rname(CQ6) rqmname(SQM) xmitq(SQM)
```

Note: `rname` points to the clustered queue.

You can now transfer to the queue. For example:

```
 fteCreateTransfer -sa SAGENT -sm SQM -da DAGENT -dm DQM -dq Q6_SQM  /tmp/single_record.txt
```

Troubleshooting a scheduled file transfer that does not run or is delayed

If you have a scheduled transfer that does not run when it is due or is delayed, it might be because the agent is processing commands on its command queue. Because the agent is busy, scheduled transfers are not checked and are therefore not run.

About this task

To work around this problem, use one of the following steps:

Procedure

- Configure the **maxSchedulerRunDelay** property in the `agent.properties` file to set the maximum interval in minutes that the agent waits to check for scheduled transfers.

  Setting this property ensures that the agent keeps checking for scheduled transfers even when the agent is busy.

  For more information about the property, see The MFT `agent.properties` file.

- Use a resource monitor instead of a scheduled transfer.

  Resource monitors work differently from scheduled transfers and are not affected by the agent being busy. For example, if you want an up-to-date file on the destination system, resource monitors reduce network traffic. This is because the file is transferred only when a new version becomes available, rather than the file being transferred automatically. However, resource monitoring is not supported on protocol bridge agents or Connect:Direct bridge agents.

  For more information, see Monitoring MFT resources.

IBM i *Troubleshooting errors when transferring IBM i save files*

If you use Managed File Transfer to transfer the same IBM i save file several times, the transfer might fail.

About this task

If you issue several concurrent requests for an MFT agent to transfer the same IBM i save file, Managed File Transfer might generate one or both of the following errors:

```
BFGII0003E: Unable to open file "/qsys.lib/library.lib/SAVF.FILE"
for reading
```

```
BFGII0082E: A file open for read failed due to a Java IOException
with message text "Sharing violation occurred"
```

Procedure

- To concurrently transfer the same save file several times, use several source agents. Use a different source agent for each concurrent transfer.
- To transfer the same save file several times with a single source agent, wait until the previous transfer request is complete before submitting each new transfer request.

Related tasks

Transferring files to or from IBM i systems

Configuring MFT on IBM i

Transferring save files located in QSYS.LIB on IBM i

Troubleshooting transfers that fail with error BFGIO0341E

If a managed transfer is transferring a file into a location that is being monitored by an external process, the transfer might fail with the error: `BFGIO0341E: The rename of temporary file` *`destination_filename`*`.part to` *`destination_filename`* `failed because the temporary file does not exist.` This is because of the way that the destination agent for managed transfers uses temporary files when writing a destination file.

About this task

How a destination agent uses temporary files

By default, when a managed file transfer takes place, the destination agent performs the following steps:

- Create a temporary file, called *`destination_ filename`*`.part`.
- Lock the temporary file.
- Write file data into the temporary file, when it is received from the source agent.
- Unlock the temporary file after all of the file data has been received and written out.
- Rename the temporary file, from *`destination_ filename`*`.part` to *`destination_ filename`*.

If a managed transfer goes into recovery, then the destination agent might create temporary files called *`destination_filename`*`.part`*`number`*. The destination agent then writes the file data to this file, instead of the one called *`destination_filename`*`.part`.

If the temporary filename *destination_filename*.part*number* already exists, the destination agent tries to create a new temporary file with the name *destination_ filename*.part(*number + 1*). If that file already exists, the destination agent attempts to create a temporary file with the name *destination_ filename*.part(*number + 2*), and so on until it is successfully able to create the file. In the situation that the agent tries, and fails, to create the temporary file *destination_ filename*.part1000, it writes directly to the destination file and does not use a temporary file.

When a managed transfer completes, the destination agent deletes all of the temporary files that are called *destination_ filename*.part*number*, because the assumption is that these were created by the agent during the managed transfer.

Why this problem occurs

A BFGIO0341E error is generated if the destination agent attempts to rename the temporary file, only to find that file is no longer there. A typical scenario that can cause this problem is as follows:

- A *staging directory* has been set up on the target file system.
- An external process is configured to monitor the *staging directory*, and move any files that it finds to a new location.
- The destination agent creates and locks the temporary file *destination_ filename*.part in the *staging directory*.
- The destination agent writes file data into the temporary file.
- After all of the file data has been written to the temporary file, the destination agent unlocks the file.
- The external process finds the temporary file, and moves it to the new location.
- The destination agent attempts to rename the temporary file, and finds that it is no longer there. As a result, the transfer item is marked as **Failed** with a BFGIO0341E error.

To avoid this problem, complete either of the following steps:

Procedure

- Configure the external process to ignore temporary files.

  Temporary files written by a destination agent always end with the .part or .part*number* suffix. If you can configure the external process to ignore those files rather than moving them, the files will still exist in the target directory when the destination agent performs the rename operation.
- Configure the destination agent so that it does not use temporary files, and writes directly to the destination file.

  The destination file is unlocked only when all of the file data has been written to it, at which point it can be picked up by the external process.

  To configure the destination agent to write directly to the destination file, set the agent property **doNotUseTempOutputFile=true**. For more information about this property, see The MFT agent.properties file.

Troubleshooting a protocol bridge agent that reports a file not found

A protocol bridge agent reports that the SFTP or FTP server that the protocol bridge connects to returns a File not found error message. This can mean that one of a number of different error cases has occurred.

About this task

The following possible error cases can result in a File not found error being returned by the SFTP or FTP server. For each case, additional information and steps to resolve the issue are described.

Procedure

- **The file does not exist.** Check that the file you are attempting to transfer exists on the system hosting the SFTP or FTP server.
- **The file path does not exist.** Check that the file path exists on the system hosting the SFTP or FTP server. Check that you have entered the file path correctly into the transfer request. If necessary, correct the file path and submit the transfer request again.
- **The file is locked by another application.** Check whether the file is locked by another application. Wait until the file is no longer locked then submit the transfer request again.
- **The file permissions do not allow the file to be read.** Check whether the file has the correct file permissions. If necessary, change the file permissions and submit the transfer request again.
- **The SFTP or FTP server uses a virtualized root path.** If a relative file path is specified in a transfer request, the protocol bridge agent attempts to convert the relative path into an absolute file path based on the home directory used to login to the protocol server. The Managed File Transfer protocol bridge agent supports only SFTP or FTP servers that allow files to be accessed by their absolute file path. Protocol servers that allow access to files based only on the current directory are not supported by the protocol bridge agent.

Related reference

The protocol bridge

Troubleshooting resource monitor problems

Use the following tasks to help you to diagnose issues with resource monitors.

Related tasks

“Troubleshooting common MFT problems” on page 166

A checklist of common MFT problems that are usually straightforward to define and fix.

Related reference

Return codes for MFT

Troubleshooting when an MFT directory resource monitor triggers files

A directory resource monitor polls a directory for files that match a trigger specification. For each file that matches the trigger specification, a transfer request is generated to the agent. When the request is submitted, the triggering file is ignored until the file is changed. In some situations, files are not triggered or are triggered twice.

About this task

Possible reasons why the files are not triggering

1. The directory resource monitor found a file that matched the trigger specification, but the generated transfer request was invalid and the agent was unable to process the request. The reasons can include the following:
   - Invalid destination agent
   - Missing destination agent
   - Transfer canceled by program invocation

   In all these cases, the directory resource monitor marks the triggering file as processed and ignores the file even though the transfer failed.
2. The file is outside the scope of the resource monitor trigger specification. The reasons can include the following:
   - Incorrect trigger pattern
   - Monitoring the incorrect directory
   - Insufficient file permissions

- Failure to connect to remote file system

Why a file can trigger a second transfer

A trigger file can generate a Managed File Transfer transfer request for the following reasons:

- If the presence of the trigger file is detected, when it was not there before.
- If the trigger file has been updated, causing the last modified date to change.

Potential scenarios for a second trigger are:

- The file is removed, then replaced.
- The file is locked by one application, then unlocked by another application.
- The monitor file system fails. For example, if the network connection fails, this can give the appearance of the file being removed, then replaced.
- The file directory is updated by another application, causing the last modified date to change.

Example

You can run a command to set info level output for all resource monitors of an agent.

In this example, all resource monitors are being monitored because you have not specified a name, or names, of specific resource monitors. The name of the agent is AGENT1.

```
fteSetAgentLogLevel -logMonitor=info AGENT1
```

For details of the **logMonitor** parameter, and examples of how you use the different options, see fteSetAgentLogLevel.

Here is an example of info level output for all resource monitors of an agent.

```
================================================================================================================================================
================================
[11/01/2022 11:08:49:367 IST] BFGUT0036I: Resource monitor event log level has changed to "info" for all resource monitors of
this agent.
================================================================================================================================================
================================

================================================================================================================================================
================================
Date        Time              Thread ID   Monitor Name                    Event
Description
================================================================================================================================================
================================
[11/01/2022 11:08:51:842 IST] 00000023    QMON                            Monitor Started     Resource Monitor Started
[11/01/2022 11:08:51:844 IST] 00000025    QMON                            Start Poll          New poll cycle started
[11/01/2022 11:08:51:924 IST] 00000023    MON1                            Monitor Started     Resource Monitor Started
[11/01/2022 11:08:51:925 IST] 00000026    MON1                            Start Poll          New poll cycle started
[11/01/2022 11:08:52:029 IST] 00000026    MON1                            End Poll            Poll cycle completed in 105
milli seconds. Total items found in this poll [ 50 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:08:52:055 IST] 00000025    QMON                            End Poll            Poll cycle completed in 212
milli seconds. Total items found in this poll [ 72 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:09:51:840 IST] 00000025    QMON                            Start Poll          New poll cycle started
[11/01/2022 11:09:51:875 IST] 00000025    QMON                            End Poll            Poll cycle completed in 34
milli seconds. Total items found in this poll [ 72 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:09:51:924 IST] 00000026    MON1                            Start Poll          New poll cycle started
[11/01/2022 11:09:51:969 IST] 00000026    MON1                            End Poll            Poll cycle completed in 45
milli seconds. Total items found in this poll [ 50 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:10:51:840 IST] 00000025    QMON                            Start Poll          New poll cycle started
[11/01/2022 11:10:51:924 IST] 00000026    MON1                            Start Poll          New poll cycle started
[11/01/2022 11:10:51:962 IST] 00000025    QMON                            End Poll            Poll cycle completed in 121
milli seconds. Total items found in this poll [ 72 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:10:51:963 IST] 00000026    MON1                            End Poll            Poll cycle completed in 39
milli seconds. Total items found in this poll [ 50 ]. Total items that matched the trigger condition [ 0 ]
[11/01/2022 11:10:55:063 IST] 00000041    MON1                            Monitor Stopped     Resource Monitor Stopped
[11/01/2022 11:10:55:079 IST] 00000041    QMON                            Monitor Stopped     Resource Monitor Stopped
```

Related reference

fteSetAgentLogLevel

Configuring an MFT resource monitor to avoid overloading an agent

You can configure the property and parameter values of a Managed File Transfer resource monitor to reduce the load on an agent. Reducing the load on the agent improves the performance of that agent.

There are several settings you can use, and you may need to use trial and error to find the best settings for your system configuration.

About this task

When a resource monitor polls a directory or a queue, the agent completes the following stages:

- Find all the files that match a trigger pattern (for example, all the `*.txt` files in the directory). Or find all complete groups of messages on the queue.
- Determine which files are new or changed, or determine which groups are new on the queue.
- Initiate transfers for the files or groups that match the criteria in the two previous stages.
- Add to the list of files and groups already transferred so they are not transferred again until they change.

For a directory monitor, the more files in the source directory and the broader the triggering pattern, the bigger the list of files the agent has to parse and compare against the list of files already transferred.

For a queue monitor, the more groups on the queue the bigger the list of groups the agent has to compare against the list of groups already transferred.

Procedure

- Use agent property **`monitorMaxResourcesInPoll`** to set the maximum number of files or groups the agent includes on each poll.

  Using this parameter limits the number of transfers in a polling interval. It also means that the agent has less parsing to do before initiating a transfer for that number of files or groups. The next time the directory monitor or queue monitor polls, the agent includes the next set of files or groups.
- When creating a directory monitor, ensure that the transfer definition you configure has a source disposition of `delete`.

  Setting this disposition means that when the file transfer completes it is removed from the monitored directory and the agent no longer keeps it on its internal list.
- When creating a directory monitor, use the **`-rl`** parameter in the **`fteCreateMonitor`** command to limit the number of levels of the directory the agent has to recurse through.

  Using this parameter means that lower-level directories are not scanned unnecessarily.

What to do next

The process of resource monitor polling consumes agent resources. Increasing the polling interval of a monitor reduces the load placed on the agent. However, the setting of the polling interval must be balanced against generating too many transfers per polling interval. Consider the following when you set the polling interval for a resource monitor:

- How quickly you need a transfer to be initiated after a file is placed in a directory, or a group on a queue.
- The rate which files are placed into a directory, or groups onto a queue.
- The maximum transfer rate of the agent. The agent must be able to handle all the transfers that a monitor generates.

The polling interval is specified when the resource monitor is created with the **`fteCreateMonitor`** command by specifying the **`-pi`** (polling interval) and **`-pu`** (polling interval units) parameters. You may need to experiment to determine the best settings for your configuration.

An option to improve the stability of highly loaded agents that run resource monitors, is to reduce the agent property value of `maxSourceTransfers`. With this option the agent splits its processing time between the resource monitor and transferring files. The higher the value of agent property `maxSourceTransfers`, the more processing time is consumed by transferring files and less is available for the resource monitor. If you reduce the value of agent property `maxSourceTransfers`, the agent does fewer transfers in parallel, but it should have enough processing time to poll its resource monitors.

If you lower the value of this agent property you should consider increasing the value of agent property `maxQueuedTransfers` because the number of queued transfers may increase.

If after optimizing your monitor you find that some transfers enter recovery, consider increasing an agent timeout value. The heavy load placed on the agent, may mean that the transfers timeout when negotiating the start of the transfer with the destination agent. This timeout causes the transfer to go into recovery and delays the completion of the transfer. The agent property `maxTransferNegotiationTime` specifies the time the source agent waits for a response from the destination agent. If this time is exceeded the transfer goes into recovery. The default value of this property is 30000 milliseconds (30 seconds). Increasing the value of the property, for example to 300000 Milliseconds (5 minutes), may allow the transfers to continue without timing out and avoid going into recovery.

Related tasks

Monitoring MFT resources

Using transfer definition files

Related reference

fteCreateMonitor (create an MFT resource monitor)

Troubleshooting destination files created by a queue resource monitor that contain the wrong data

You can create a resource monitor to monitor a queue and transfer a message or a group of messages on a queue to a file. The file name can be specified by using the MQMD message descriptors on the message or the first message in a group. If a message-to-file transfer fails and the message or group is left on the queue, the next time the monitor is triggered it might result in files being created that contain the wrong data.

About this task

This problem results from the following sequence of events:

1. A message-to-file transfer fails and the message or group is left on the queue.
2. A new message or group arrives on the queue.
3. The new message or group triggers the resource monitor.
4. The resource monitor creates a new transfer that uses the MQMD message descriptors from the new message or group and the data from the first message or group on the queue.
5. Files are created that contain the wrong data.

Example

To avoid experiencing this problem, manually create a transfer definition file by using the **`fteCreateTransfer`** command and edit the <queue> element of the file to include the attribute `groupId="${GROUPID}"`. Then submit the transfer definition file by using the **`fteCreateMonitor`** command.

In this example:

1. The source agent, which is also the monitoring agent, is called AGENT_MON.
2. The destination agent is called AGENT_DEST.
3. The destination file name is `/out/files/${WMQFTEFileName}`.

This example requires that the message has the MQMD message descriptor `WMQFTEFileName` set. The queue being monitored is LIVE_QUEUE.

1. Create a transfer definition file by running the following command:

```
fteCreateTransfer -sa AGENT_MON -da AGENT_DEST -df "/out/files/${WMQFTEFileName}"
                -de error -gt /tmp/TransferDefinition1.xml -sqgi -sq LIVE_QUEUE
```

   The transfer definition file `/tmp/TransferDefinition1.xml` is generated.

2. Edit the <queue> element to include the attribute groupId="${GROUPID}". Change the line

```
<queue useGroups="true">LIVE_QUEUE</queue>
```

to

```
<queue useGroups="true" groupId="${GROUPID}">LIVE_QUEUE</queue>
```

This attribute is required so that the transfer reads the group or message that triggered the transfer from the queue instead of the first group or message on the queue.

3. Create the monitor by running the following command:

```
fteCreateMonitor -ma AGENT_MON -mq LIVE_QUEUE -mn QueueMon1 -mt /tmp/TransferDefinition1.xml
              -tr completeGroups -dv WMQFTEFileName=UNKNOWN
```

This monitor polls the queue every 60 seconds to see if a new group or message has arrived on the queue.

Troubleshooting variable substitution that causes multiple files to go to a single file name

For Managed File Transfer, if you are monitoring a directory and transferring multiple files from a source to a destination location and you are using ${FileName} variable substitution, you must test the variable substitution results. The results need to be tested because the use of variable substitution might cause unexpected combinations of file transfer commands to be invoked.

About this task

When multiple files are being processed by an MFT directory monitor, the Task XML runs for every file that the monitor finds in the directory being monitored. If the ${FileName} is only specified in the destination of the XML task file and not the source, the transfer is invoked for each file multiple times, once for each file name combination.

For example:

```
<source disposition="delete" recursive="false">
    <file>e:\temp</file>
</source>
<destination exist="overwrite" type="file">
   <file>s:\outdir\${FileName}</file>
</destination>
```

To determine whether the problem is occurring, look for cases of multiple files appearing to transfer but only one file arriving at the destination. You might see errors in the file transfer log showing multiple files attempting to transfer to the same destination file name and failing transfers to the same file name.

Example

If you are using ${FileName} variable substitution in the source or destination and are expecting a variation of the same file name to arrive at the destination, be sure to specify ${FileName} in BOTH the source and destination of your task XML definition.

The following example takes a file from e:\temp\<filename> and transfers it to s:\outdir\<filename>.out:

```
<source disposition="delete" recursive="false">
      <file>e:\temp\${FileName}</file>
  </source>
  <destination exist="overwrite" type="file">
     <file>s:\outdir\${FileName}.out</file>
  </destination>
```

Related tasks

Customizing MFT tasks with variable substitution

Related reference

Examples: Variable substitution

Troubleshooting a resource monitor that reports a BFGDM0107W message

A resource monitor configured to poll either a directory or a queue looks for items that match a specified trigger condition and submits managed transfers to its associated agent to process them. Periodically, the monitor writes a BFGDM0107W message to the agent's event log (`output0.log`).

About this task

The following text shows a typical BFGDM0107W message:

```
BFGDM0107W: The number of tasks generated by monitor MONITOR1 during a
polling interval has exceeded twice the value of the maxSourceTransfers agent property
and the agent property monitorMaxResourcesInPoll is set to its default value of -1.
```

Every agent has a number of transfer slots that it uses to hold details about the managed transfers and managed calls that are currently in progress, as well as the managed transfer and managed call requests that are currently on its backlog. For more information about how these slots are used, see How MFT agents allocate source transfer slots to new requests.

By default, a monitor submits a task (which is either a single managed transfer or managed call request) for every item that it triggers on during a poll. For example, if a resource monitor has been configured to poll a source queue looking for complete message groups or individual messages not in a group then, if the monitor finds:

- 10 messages or complete message groups on the queue during a poll, it submits 10 tasks (or managed transfer requests) to the agent.
- 200 messages or complete message groups on the queue during a poll, it submits 200 tasks (or managed transfer requests) to the agent.

Monitors contain some logic to compare the number of tasks they have submitted to the agent during a poll against the number of source transfer slots that the agent has (as specified by the agent property **`maxSourceTransfers`**). If the tasks are greater than twice the number of source transfer slots, the monitor writes the BFGDM0107W message to the agent's event log. This lets you know that it has submitted a large number of tasks to the agent, more than half of which are going on to the agent's backlog.

Going back to our earlier example, where a monitor finds 200 messages during a single poll, and assuming that the agent in question has its **`maxSourceTransfers`** property set to the default value of 25, when the monitor submits the 200 tasks to the agent:

- 25 are assigned source transfer slots, and the agent starts to process those straight away.
- The remaining 175 are assigned queued transfer slots; these go onto the agent's backlog to be processed at some point in the future.

Having a large number of managed transfers on an agent's backlog takes up resources such as memory, and so can potentially affect an agent's performance. Because of this, it is good practise to try and keep the number of managed transfers or managed calls occupying queued transfer slots down to a low number where possible.

Example

One thing that can help prevent the warning from occurring is the **`monitorMaxResourcesInPoll`** property mentioned in the BFGMD0107W message. This is an agent property which applies to all resource monitors running within the agent and limits the number of items that monitors trigger on during a single poll. The default value of the property is `-1`, which means that monitors trigger on every item that they find in a poll and submit a task for each one.

When the property is set to something other than `-1`, the monitor stops scanning the resource once it triggers on that many items. This means that the monitor is sending work to the agent in small chunks, rather than giving it lots of work to do all in one go.

For example, if **monitorMaxResourcesInPoll** is set to 25, once the monitor finds 25 new items that match its trigger condition, it stops its current poll and submits 25 tasks to the agent.

When changing **monitorMaxResourcesInPoll**, another thing to consider is increasing the polling interval of the monitor. Ideally, if a resource monitor submits some tasks to an agent, it should allow most (if not all) of them to complete before starting a new poll and potentially giving some more work to the agent to do. This also helps to reduce the overall load on the agent, and can improve its throughput.

Suppose you have a resource monitor that has been configured to monitor a source queue every minute, looking for either complete message groups or individual messages not in a group. For each message group or individual message that the monitor finds, it submits a task (in the form of a managed transfers request) to move the contents of that message or message group to a file.

The agent where the monitor is running has the following agent properties set:

```
maxQueuedTransfers=1000
maxSourceTransfers=25
monitorMaxResourcesInPoll=25
```

This means that during every poll, the monitor has the potential to submit 25 tasks to the agent. Assuming that it takes the agent approximately two minutes to process all 25 tasks, then with a polling interval of one minute the following behavior takes place:

Minute 0

- The monitor starts a poll, scans the source queue, and finds 25 messages (the value of **monitorMaxResourcesInPoll**).
- The monitor now submits 25 tasks (or managed transfer requests) to the agent, and then stops its poll.
- The agent picks up the 25 managed transfer requests, assigns each of them a source transfer slot and begins processing them.

  At this point in time, the agent's transfer slots look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |   0  | 1000
```

Minute 1

- The monitor now starts its second poll.
- The monitor, once again, scans the source queue, finds 25 messages and submits 25 managed transfer requests to the agent.
- The poll ends.
- The agent receives these new managed transfer requests. As all of its source transfer slots are occupied, it assigns each of the managed transfer requests a queued transfer slot and puts them on its backlog.

  The agent's transfer slots now look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |  25  |  975
```

Minute 2

- By this time, all of the 25 managed transfers have finished processing and their associated source transfer slots are released. As a result, the agent moves the 25 managed transfers from the queued transfer slots to the source transfer slots.

This leaves the agent's transfer slots looking like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |   0  | 1000
```

- The monitor performs another poll, finds another batch of 25 messages and submits 25 managed transfer requests to the agent.
- The agent picks up these requests and puts them onto its backlog

  This means that the transfer slots now look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |  25  |  975
```

Minute 3

- During the next poll, the monitor finds 25 more messages, and so submits 25 more managed transfer requests to the agent.
- The agent receives these managed transfer requests and assigns them each a queued transfer slot.

  As a result, the agent's transfer slots are now like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |  50  |  950
```

and so on.

Suppose you increase the polling interval to two minutes. This means that the 25 managed transfers submitted during one poll are completed by the time the next one starts, and so the agent is able to assign these managed transfers a source transfer slot and not have to put them onto its backlog. This is shown in the following example:

Minute 0

- The monitor starts a poll, scans the source queue, and finds 25 messages (the value of **monitorMaxResourcesInPoll**).
- The monitor now submits 25 managed transfer requests to the agent, and then stops its poll.
- The agent picks up the 25 managed transfer requests, assigns each of them a source transfer slot and begins processing them.

  At this point in time, the agent's transfer slots look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |   0  | 1000
```

Minute 2

- By this time, all of the 25 managed transfers have finished processing and their associated source transfer slots are released.

  This means that the agent's transfer slots look like this:

```
                      | Used | Free
----------------------|------|------
```

```
Source transfer slots |   0  |   25
Queued transfer slots |   0  | 1000
```

- The monitor performs another poll, finds another batch of 25 messages and submits 25 managed transfer requests to the agent.
- The agent picks up these requests and assigns each of them a source transfer slot.

  This means that the transfer slots now look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |   0  | 1000
```

Minute 4

- Two minutes later, the 25 managed transfer requests submitted by the monitor in minute 2 have completed, and their associated "source transfer slots" have been freed up and released.

  The agent's source transfer slots are now this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |   0  |   25
Queued transfer slots |   0  | 1000
```

- The monitor now performs a new poll and finds 25 more messages on the queue. As a result, it submits 25 managed transfer requests to the agent.
- The agent picks up the managed transfer requests. As it is not currently acting as the source agent for any managed transfers, it assigns a "source transfer slot" to each of the new requests.

  This makes its transfer slots look like this:

```
                      | Used | Free
----------------------|------|------
Source transfer slots |  25  |    0
Queued transfer slots |   0  | 1000
```

The advantage of this approach is that managed transfers never go onto an agent's backlog, which reduces the overall resource usage of the agent and in turn can help with performance.

Troubleshooting java.lang.OutOfMemoryError problems

Use the following tasks to help you to resolve issues with agents stopping due to `java.lang.OutOfMemoryErrors`.

Related tasks

“Troubleshooting common MFT problems” on page 166
A checklist of common MFT problems that are usually straightforward to define and fix.

Related reference

Return codes for MFT

Troubleshooting an MFT agent with a Java heap exhaustion error

While processing a number of managed transfer requests, such as file-to-file, message-to-file or file-to-message transfers, the agent abnormally ends (ABENDS) reporting a `java.lang.OutOfMemoryError`, and at the time your total RAM memory was not fully utilized. This exception has been caused by Java heap exhaustion.

About this task

When this issue occurs, the affected agent ABENDs and generates three files that provide details on the root cause:

- An ABEND file. The name of this file conforms to the naming convention ABEND.FTE.*date_timestamp*.*identifier*.log.

  Multi On Multiplatforms, the file is written to the MQ_DATA_PATH/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/ffdc directory.

  z/OS On z/OS, the file is written to the z/OS UNIX System Services (z/OS UNIX) location $BFG_CONFIG/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/ffdc
- A Javacore file. The name of this file has the following format: javacore.*datestamp*.*timestamp*.*pid*.*identifier*.txt

  Multi On Multiplatforms, the file is written to the MQ_DATA_PATH/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.

  z/OS On z/OS, the file is written to the z/OS UNIX location $BFG_CONFIG/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.
- A Java snap dump. The name of this file has the following format: snap.*datestamp*.*timestamp*.*pid*.*identifier*.txt

  Multi On Multiplatforms, the file is written to the MQ_DATA_PATH/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.

  z/OS On z/OS, the file is written to the z/OS UNIX location $BFG_CONFIG/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.

The ABEND and Javacore pair contain information similar to the following examples:

Abend file

```
Filename:
C:\ProgramData\IBM\MQ\mqft\logs\QM1\agents\AGENT1\logs\ffdc\ABEND.FTE.20220810102649225.18938124211177445
3.log
Level:      p920-005-220208
Time:       10/08/2022 10:26:49:225 BST
Thread:     45 (FileIOWorker-0:0)
Class:      com.ibm.wmqfte.thread.FTEThread
Instance:   a393304f
Method:     uncaughtException
Probe:      ABEND_001
Cause:      java.lang.OutOfMemoryError: Java heap space

java.lang.OutOfMemoryError: Java heap space
    at java.nio.HeapByteBuffer.<init>(HeapByteBuffer.java:57)
    at java.nio.ByteBuffer.allocate(ByteBuffer.java:335)
    at com.ibm.wmqfte.util.impl.ByteBufferPoolImpl.getBuffer(ByteBufferPoolImpl.java:44)
    at com.ibm.wmqfte.transfer.frame.impl.TransferChunkImpl.getByteBuffer(TransferChunkImpl.java:181)
    at com.ibm.wmqfte.transfer.frame.impl.TransferChunkImpl.<init>(TransferChunkImpl.java:143)
    at
com.ibm.wmqfte.transfer.frame.impl.TransferFrameSenderImpl.requestChunk(TransferFrameSenderImpl.java:636)
    at
com.ibm.wmqfte.transfer.frame.impl.TransferFrameSenderImpl.access$000(TransferFrameSenderImpl.java:100)
    at
com.ibm.wmqfte.transfer.frame.impl.TransferFrameSenderImpl$ChunkRequester.processFileIORequest(TransferFr
ameSenderImpl.java:142)
    at
com.ibm.wmqfte.transfer.frame.impl.TransferFrameIOWorker.doWorkImpl(TransferFrameIOWorker.java:318)
    at com.ibm.wmqfte.io.impl.FTEFileIOWorker.doWork(FTEFileIOWorker.java:118)
    at com.ibm.wmqfte.io.impl.FTEFileIORequestQueue.run(FTEFileIORequestQueue.java:244)
    at java.lang.Thread.run(Thread.java:825)
    at com.ibm.wmqfte.thread.FTEThread.run(FTEThread.java:70)
```

Javacore file

```
0SECTION       TITLE subcomponent dump routine
NULL           ===============================
1TICHARSET     437
1TISIGINFO     Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError" "Java heap space"
received
```

```
1TIDATETIMEUTC Date: 2022/08/10 at 09:26:53:917 (UTC)
1TIDATETIME    Date: 2022/08/10 at 10:26:53:917
1TITIMEZONE    Timezone: (unavailable)
1TINANOTIME    System nanotime: 350635184939400
1TIFILENAME    Javacore filename:
C:\ProgramData\IBM\MQ\mqft\logs\QM1\agents\AGENT1\javacore.20220810.102653.7172.0003.txt
```

This issue occurs because of exhaustion of the Java heap memory for the JVM running the agent.

See How MFT agents use Java heap and native heap memory for more information on the distinctions between Java heap memory and native heap memory.

To help reduce the likelihood of this issue occurring, complete the following steps:

Procedure

1. Increase the size of the Java heap for the JVM running the MFT agent.

   By default, the Java heap of an agent is set to 512 MB. Although this is satisfactory for small numbers of managed transfers, it might need to be increased to up to 1024MB (1GB) for production-like workload.

   Attention: When increasing the size of the Java heap for an agent, it is important to consider the other agents and applications that are running on the same system as these are using native heap. Increasing the size of the Java heap for an agent also increases its native heap usage, which in turn reduces the amount of native heap available to the other agents and applications. This means that there is an increased likelihood of agents and applications experiencing native heap exhaustion.

   - To increase or change the Java heap when running the agent as a normal process, set the BFG_JVM_PROPERTIES environment variable to pass the Java property **-Xmx** to the JVM.

     For example, on Windows, to set the maximum heap size to 1024 MB run the following command before using the **fteStartAgent** command:

     ```
     set BFG_JVM_PROPERTIES="-Xmx1024M"
     ```

     For more information about how to set Java system properties using the BFG_JVM_PROPERTIES environment variable, see Java system properties for MFT.

   - To increase or change the Java heap when running the agent as a Windows service, use the **fteModifyAgent** command and specify the **-sj** parameter to set the **-Xmx** property on the Windows service.

     The following example uses the **fteModifyAgent** command with the **-sj** parameter, to set the maximum size of the Java heap for a JVM running a Windows service configured agent to 1GB (1024MB):

     ```
     fteModifyAgent.cmd -agentName AGENT1 -s -su user1 -sp passw0rd -sj -Xmx1024M
     ```

     You can check this has been successfully set, by reviewing the `output0.log` file of the agent, after the agent has been restarted. In the *Start Display Current Environment* section, a value of 1024 MB will be reported, as follows:

     ```
     The maximum amount of memory that the Java virtual machine will attempt to use is: '1024'MB
     ```

2. Restrict Java heap usage by reducing the workload of the agent.

   Typically, `java.lang.OutOfMemoryErrors` caused by Java heap exhaustion are the result of an agent doing too much work. Every managed transfer and managed call that an agent is processing uses memory in the Java heap, as do managed transfers and managed calls that are on the backlog of an agent. Resource monitors also use Java heap memory when they perform a poll.

   This means that as the workload of an agent increases, the amount of Java heap that it is using also grows as well.

Reducing the workload of the agent can help here. To do this:

- Set the following agent properties to a lower value:
  - **maxQueuedTransfers**
  - **maxSourceTransfers**
  - **maxDestinationTransfers**
- Move some of the resource monitors of the agent to a new agent.

This reduces the number of concurrent transfers that can occur, and therefore decreases the maximum concurrent workload for the agent.

3. Enable memory allocation checking.

   The memory allocation checking functionality ensures that agents only start to process a new managed transfer if there is enough Java heap memory for it to run to completion. If there is insufficient memory, the managed transfer is rejected.

   This functionality is off by default. To enable it for an agent:

   - Add the following entry to the `agent.properties` file of the agent:

     ```
     enableMemoryAllocationChecking=true
     ```

   - Restart the agent

   Note: The memory allocation checking functionality uses the maximum amount of memory that a managed transfer requires, which might be more than the actual amount of memory used (particularly for message-to-file and file-to-message transfers). This means that turning it on can result in fewer managed transfers being processed by an agent.
4. If the agent continues to experience `java.lang.OutOfMemoryErrors` due to Java heap exhaustion, then run the **fteRas** command to collect the ABEND files, Javacores, heap dump files and snap dump files (along with other useful information about the MFT topology), and make the output available to IBM Support for analysis.

Related tasks

“Troubleshooting an MFT agent with a native heap exhaustion error” on page 155
While processing a number of managed transfer requests, such as file-to-file, message-to-file or file-to-message transfers, the agent abnormally ends (ABENDS) reporting a `java.lang.OutOfMemoryError`, and at the time your total RAM memory was not fully utilized. This exception has been caused by native memory exhaustion.

Troubleshooting an MFT agent with a native heap exhaustion error

While processing a number of managed transfer requests, such as file-to-file, message-to-file or file-to-message transfers, the agent abnormally ends (ABENDS) reporting a `java.lang.OutOfMemoryError`, and at the time your total RAM memory was not fully utilized. This exception has been caused by native memory exhaustion.

About this task

When this issue occurs, the affected agent ABENDs and generates two files that provide details on the root cause:

- An ABEND file. The name of this file conforms to the naming convention `ABEND.FTE.`*`date_timestamp.identifier`*`.log`.

  Multi On Multiplatforms, the file is written to the `MQ_DATA_PATH/mqft/logs/`*`coordination_qmgr_name`*`/agents/`*`agent_name`*`/logs/ffdc` directory.

  z/OS On z/OS, the file is written to the z/OS UNIX System Services (z/OS UNIX) location `$BFG_CONFIG/mqft/logs/`*`coordination_qmgr_name`*`/agents/`*`agent_name`*`/logs/ffdc`

- A Javacore file. The name of this file has the following format:
  javacore.*datestamp*.*timestamp*.*pid*.*identifier*.txt

  Multi On Multiplatforms, the file is written to the MQ_DATA_PATH/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.

  z/OS On z/OS, the file is written to the z/OS UNIX location $BFG_CONFIG/mqft/logs/*coordination_qmgr_name*/agents/*agent_name* directory.

The ABEND and Javacore pair contain information similar to the following examples:

Example: Pair one

Abend file

```
Filename:
C:\ProgramData\IBM\MQ\mqft\logs\COORDQM\agents\AGENT1\logs\ffdc\ABEND.FTE.20200109113518046.1764802189777
906538.log
Level:      p900-005-180821
Time:       09/01/2020 11:35:18:046 GMT
Thread:     96 (TransferSender[414d51204d44424b525030372020202045fbd6532ebfaa02])
Class:      com.ibm.wmqfte.thread.FTEThread
Instance:   55b455b4
Method:     uncaughtException
Probe:      ABEND_001
Cause:      java.lang.OutOfMemoryError: native memory exhausted

java.lang.OutOfMemoryError: native memory exhausted
    at com.ibm.mq.jmqi.local.internal.base.Native.MQPUT(Native Method)
    at com.ibm.mq.jmqi.local.LocalMQ.MQPUT(LocalMQ.java)
    at com.ibm.wmqfte.wmqiface.WMQQueueImpl.put(WMQQueueImpl.java)
    at com.ibm.wmqfte.wmqiface.WMQQueueImpl.put(WMQQueueImpl.java)
    at com.ibm.wmqfte.transfer.impl.TransferSenderRunnable.doTransfer(TransferSenderRunnable.java)
    at com.ibm.wmqfte.transfer.impl.TransferSenderRunnable.run(TransferSenderRunnable.java)
    at java.lang.Thread.run(Thread.java)
    at com.ibm.wmqfte.thread.FTEThread.run(FTEThread.java)
```

Javacore file

```
0SECTION    TITLE subcomponent dump routine
NULL        ===============================
1TISIGINFO  Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError" "native memory
exhausted" received
1TIDATETIME Date: 2020/01/09 at 11:35:18
1TIFILENAME Javacore filename:
C:\ProgramData\IBM\MQ\mqft\logs\COORDQM\agents\AGENT1\javacore.20200109.113518.14148.0002.txt
```

Example: Pair two

ABEND file

```
Filename:
C:\ProgramData\IBM\MQ\mqft\logs\COORDQM\agents\AGENT1\logs\ffdc\ABEND.FTE.20200109143700286.3177895731698
464509.log
Level:      p900-005-180821
Time:       09/01/2020 14:37:00:286 GMT
Thread:     918 (AgentStatusPublisher)
Class:      com.ibm.wmqfte.thread.FTEThread
Instance:   bc10bc1
Method:     uncaughtException
Probe:      ABEND_001
Cause:      java.lang.OutOfMemoryError: Failed to create a thread: retVal -1073741830, errno 12

java.lang.OutOfMemoryError: Failed to create a thread: retVal -1073741830, errno 12
    at java.lang.Thread.startImpl(Native Method)
    at java.lang.Thread.start(Thread.java)
```

Javacore file

```
0SECTION       TITLE subcomponent dump routine
```

```
NULL           ==============================
1TISIGINFO     Dump Event "systhrow" (00040000) Detail "java/lang/OutOfMemoryError" "Failed to create a
thread: retVal -1073741830, errno 12" received
1TIDATETIME    Date: 2020/01/09 at 14:37:00
1TIFILENAME    Javacore filename:
C:\ProgramData\IBM\MQ\mqft\logs\COORDQM\agents\AGENT1\javacore.20200109.143700.2652.0003.txt
```

This issue occurs because of exhaustion of the native heap memory on the system where the agent is running.

See How MFT agents use Java heap and native heap memory for more information on the distinctions between Java heap memory and native heap memory.

To help reduce the likelihood of this issue occurring, complete the following steps:

Procedure

1. Reduce the size of the Java heap for the JVM running the MFT agent.

   The greater the size of the allocated Java heap, the less memory is available to the native heap. Reducing the size of the Java heap used by an agent can free up more memory for the native heap.

   By default, the Java heap of an agent is set to 512 MB. If you have changed this to make it a larger value, consider reducing it, and testing with your production-like workload.

   - To lower or change the Java heap when running the agent as a normal process, set the BFG_JVM_PROPERTIES environment variable to pass options directory to the JVM.

     For example, on Windows, to set the maximum heap size to 1024 MB run the following command before using the **fteStartAgent** command:

     ```
     set BFG_JVM_PROPERTIES="-Xmx1024M"
     ```

     For more information about how to set Java system properties using the BFG_JVM_PROPERTIES environment variable, see Java system properties for MFT.

   - To lower or change the Java heap when running the agent as a Windows service, use the **fteModifyAgent** command and specify the **-sj** parameter to set the **-Xmx** property on the Windows service.

     The following example uses the **fteModifyAgent** command with the **-sj** parameter, to set the maximum size of the Java heap for a JVM running a Windows service configured agent:

     ```
     fteModifyAgent -agentName AGENT1 -s Service1 -su fteuser -sp ftepassword -sj -Xmx1024M
     ```

     You can check this has been successfully set, by reviewing the `output0.log` file of the agent, after the agent has been restarted. In the *Start Display Current Environment* section, a value of 1024 MB will be reported, as follows:

     ```
     The maximum amount of memory that the Java virtual machine will attempt to use is: '1024'MB
     ```

2. Restrict native memory use.

   Often, `java.lang.OutOfMemoryErrors` caused by native heap exhaustion are seen if an agent connects to its agent queue manager using the BINDINGS transport. When the agent has been configured to use the BINDINGS transport, the agent calls native methods whenever it needs to communicate with the queue manager.

   This means that native memory usage grows as the workload of the agent increases, due to more connections to the queue manager and increased message communication. In this situation, reducing the workload can help. To do this, set the following agent properties to a lower value than the default 25:

   - **maxSourceTransfers**
   - **maxDestinationTransfers**

This reduces the number of concurrent transfers that can occur, and therefore decreases the maximum concurrent workload for the agent.

3. Configure the agent to use the CLIENT transport when connecting to its agent queue manager.

   You can do this by setting the following agent properties:

   - **agentQMgrHost**
   - **agentQMgrPort**
   - **agentQMgrChannel**

   You can find information about these properties in The MFT agent.properties file topic.

   This ensures that all communication between the agent and the queue manager takes place over TCP/IP, rather than native code, which reduces the amount of native memory used by the agent.

   Important: Taking this action also decreases performance. By using a TCP/IP connection to the local host, rather than native code, the configuration is not as efficient when the agent requires interactions with the queue manager.

Related tasks

"Troubleshooting an MFT agent with a Java heap exhaustion error" on page 152
While processing a number of managed transfer requests, such as file-to-file, message-to-file or file-to-message transfers, the agent abnormally ends (ABENDS) reporting a `java.lang.OutOfMemoryError`, and at the time your total RAM memory was not fully utilized. This exception has been caused by Java heap exhaustion.

Troubleshooting logger problems

Use the following tasks to help you to resolve issues with loggers.

Related tasks

"Troubleshooting common MFT problems" on page 166
A checklist of common MFT problems that are usually straightforward to define and fix.

Related reference

Return codes for MFT

Troubleshooting an update to your MFT database schema on Oracle

You might receive the following error message when updating your database schema to the latest level by using the `ftelog_tables_oracle_702_703.sql` file: `ERROR at line 1: ORA-02289: sequence does not exist`. This error occurs because the sequences and triggers used by the tables are not in the same schema as the tables.

About this task

To fix this problem, you must edit the contents of the `ftelog_tables_oracle_702_703.sql` before running it.

Procedure

1. Find out which schema the sequences and triggers used by the Managed File Transfer database logger tables are located in.
   - On Db2, you can use the Control Center to view the tables and schema.
   - On Oracle, you can use the Enterprise Manager to view the tables and schema.
2. Open the `ftelog_tables_oracle_702_703.sql` file in a text editor.
3. In each occurrence of the text `SELECT FTELOG.`*`sequence_name`*`.nextval` replace the text `FTELOG` with the name of the schema where your existing sequences are located.
4. Before each occurrence of the text `CREATE OR REPLACE TRIGGER FTELOG.`*`trigger_name`*, insert the text `DROP TRIGGER` *`schema_name.trigger_name`*, where *`schema_name`* is the name of the schema where your existing triggers are located.

5. Use the edited `ftelog_tables_oracle_702_703.sql` file to update the database tables.

MFT logger error handling and message rejection

The Managed File Transfer logger identifies two types of error: per-message errors and general errors.

Per-message errors are likely to be caused by a problem with one or a few individual messages. Some examples of situations, which are identified as per-message errors are as follows:

- The result code, which is a required item of data, is missing from a message
- A transfer specifies a job name that is 3000 characters long and too large for the associated database column
- A progress message is received for a transfer, but there is no record of the transfer having been started (perhaps because of a misrouted or delayed transfer start message)
- A message is received, which is not a Managed File Transfer log message

General errors are all those errors that are not per-message errors. These are likely to be because of configuration problems or program errors.

When a per-message error is encountered, the logger rejects the message by placing the message on the reject queue. Nothing is written to the output log, so you must periodically inspect or continuously monitor the reject queue to detect rejected messages.

If too many messages are rejected consecutively, without any messages being successfully written to the database, this is treated as a general error. For example, consider a site that always uses 10 character codes as job names, but which has inadvertently reconfigured the job name column to be two characters wide. Although data that is too wide is usually a per-message error, in this case the configuration problem is general and is detected as a general error. You can tune the number of consecutive per-message errors needed to cause a general error by using the **wmqfte.max.consecutive.reject** property.

If a general error is detected the logger rolls back any messages not yet committed to the queue manager, and then retries periodically. A message identifying the problem is written to the output log and to the console if the logger was started in foreground mode with the **-F** parameter.

The location of the output logs for the logger is dependent on whether it is a stand-alone or JEE database logger. For a stand-alone database logger it is located in the directory *MQ_DATA_PATH*`/mqft/logs/`*`coordination_qmgr_name`*`/loggers/`*`logger_name`*. For a JEE database logger it is located in the standard output log of the application server.

The reject queue

Messages that result in per-message errors are moved to the reject queue. On each rejected message, a message property is set to indicate why the message was rejected. For information about viewing the contents of the reject queue, seeing why a message was rejected, and reprocessing messages, see “Reprocessing MFT messages from the reject queue” on page 159.

`Malformed` log messages in the transfer log are not logged by the logger. These messages are not viewed as being significant and so these messages are sent to the reject queue. For more information about transfer log messages, see File transfer log message formats.

Reprocessing MFT messages from the reject queue

When a per-message error is encountered, the logger rejects the message by placing the message on the reject queue. You can view the contents of the reject queue, see why a message was rejected, and (in some cases) reprocess the message.

Before you begin

This task assumes that you have read “MFT logger error handling and message rejection” on page 159, and are therefore aware of the following information:

- When a per-message error is encountered, the logger rejects the message by placing the message on the reject queue. Nothing is written to the output log, so you must periodically inspect or continuously monitor the reject queue to detect rejected messages.

- If too many messages are rejected consecutively, this is treated as a general error. You can tune the number of consecutive per-message errors needed to cause a general error by using the **wmqfte.max.consecutive.reject** property.
- For a stand-alone database logger, the output log is located in the directory *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*. For a JEE database logger, it is located in the standard output log of the application server.

About this task

Messages that result in per-message errors are moved to the reject queue. Sometimes, you might want to reprocess messages from the reject queue. In the example described in "MFT logger error handling and message rejection" on page 159, with a two-character job name column in the database, the messages could be successfully processed after the width of the database column had been increased. As another example, when a transfer-complete message is rejected because its associated transfer-start was missing, the transfer-start message might be received later. Reprocessing the transfer-complete will then be successful.

Malformed log messages in the transfer log are not logged by the logger. These messages are not viewed as being significant and so these messages are sent to the reject queue. For more information about transfer log messages, see File transfer log message formats.

Example

On each rejected message, a message property is set to indicate why the message was rejected. The full name of the property is **usr.WMQFTE_ReasonForRejection**, although usr. is omitted in some contexts (including JMS and the IBM MQ Explorer). If you are using IBM MQ Explorer, you can view the contents of the reject queue by right-clicking the queue and clicking **Browse Messages**. To see why a message was rejected, double-click the message to open its properties dialog, then select the **Named Properties** page. You will see a property called **WMQFTE_ReasonForRejection**. Alternatively, you could write or configure a monitoring tool to obtain this information automatically.

To reprocess messages, move them from the reject queue to the input queue. In a normal installation, where the logger creates its own managed subscription, the input queue is defined by the queue manager and has a name like SYSTEM.MANAGED.DURABLE.49998CFF20006204. You can identify the input queue by looking at the **Destination name** in the properties for the subscription SYSTEM.FTE.DATABASELogger.AUTO, or using the following MQSC command:

```
DISPLAY SUB(SYSTEM.FTE.DATABASELogger.AUTO) DEST
```

One way of moving messages between queues is to use the MA01 SupportPac, for example:

```
q -IFTE.REJECT -oSYSTEM.MANAGED.DURABLE.49998CFF20006204
```

The reject queue might contain messages rejected for various reasons, only some of which have been resolved. In this case you can still reprocess all the messages; those messages that can now be accepted are consumed, and those messages that cannot are again moved to the reject queue.

Troubleshooting an MFT logger that cannot write to the database tables

The database tables used by the Managed File Transfer logger require the database to have a page size of 8 KB or larger. If the page size of the database is not large enough, the tables are not created properly and you see the error SQLSTATE=42704.

About this task

If you are using the Java Platform, Enterprise Edition database logger, you might see the following message in the WebSphere Application Server system out log; if you are using the stand-alone database logger, you might see the following error in the output0.log file:

```
DB2 SQL Error: SQLCODE=-204, SQLSTATE=42704
SQLERRMC=FTELOG.TRANSFER_EVENT, DRIVER=3.40.152
```

The SQLSTATE value of 42704 indicates that a table that the logger expected to exist, in this case FTELOG.TRANSFER_EVENT, does not exist.

To fix this problem, complete the following steps:

Procedure

1. Check that the table exists and is complete.

   For information about the tables that the logger uses and their columns, see MFT database logger tables.
2. If the table does not exist or is incomplete, check the page size of the database.
3. If the database size is less than 8 KB, increase the page size of the database.

   Create a table space with a page size of at least 8 KB and an associated buffer pool with a page size of at least 8 KB. For information about how to do this, refer to the documentation for your database.

Troubleshooting the Connect:Direct bridge

Use the following tasks and examples to help you diagnose errors returned from the Connect:Direct bridge.

Log information for the Connect:Direct bridge

You can use a Connect:Direct bridge agent to transfer files between MFT agents and Connect:Direct nodes. Log information about the Connect:Direct nodes and processes involved in these transfers is displayed in the IBM MQ Explorer plug-in and is stored in your log database.

Log information about the Connect:Direct nodes and Connect:Direct processes involved in a file transfer is included in the log messages that are published to the SYSTEM.FTE topic on the coordination queue manager. For more information, see File transfer log message formats.

The following information is included in the published message:

- Connect:Direct bridge node name
- Primary node (PNODE) name
- Secondary node (SNODE) name
- Process name
- Process ID number

The Connect:Direct bridge node is the same node as either the primary node or the secondary node.

The value of the Connect:Direct bridge node name is the name that the bridge node is known to the MFT Connect:Direct bridge agent by. The primary and secondary node names are the names that are used to refer to the nodes in the network map of the Connect:Direct bridge node.

Related reference

Connect:Direct bridge transfer log message examples

z/OS *Connect:Direct file paths specified with a double forward slash*

If, as part of a file transfer, you specify a file located on a Connect:Direct node by using a file path that starts with a double forward slash (//), the file is treated as a data set.

Sources and destinations on a Connect:Direct node are specified in the format *cd_node_name*:*file_path*. If the *file_path* starts with a double forward slash (//), the source or destination is treated as a data set. This is the case even when the Connect:Direct node is not on z/OS. This can cause transfer failures if the file path is accidentally specified with a double forward slash (//) at the start and the file is not a data set.

Ensure that you do not specify a *file_path* that starts with a double forward slash (//) if you do not want the file that you specify to be treated as a data set.

Related tasks

"Troubleshooting the Connect:Direct bridge" on page 161
Use the following tasks and examples to help you diagnose errors returned from the Connect:Direct bridge.

Transferring data sets to and from Connect:Direct nodes

Tracing the Connect:Direct bridge

You can capture trace from the Connect:Direct node that is part of the Connect:Direct bridge to help with problem determination.

About this task

To enable trace, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the Connect:Direct bridge agent properties file to include the line:

```
cdTrace=true
```

3. Start the Connect:Direct bridge agent.

Results

The trace information is written to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

Related reference

The MFT `agent.properties` file

Solving permissions issues with Connect:Direct nodes

Use the information in this topic if your transfers between Managed File Transfer and Connect:Direct fail with an error about insufficient permissions.

About this task

For transfers involving the Connect:Direct bridge, the user ID that connects to the Connect:Direct node is determined by which IBM MQ Message Descriptor (MQMD) user ID is associated with the transfer request. You can map specific MQMD user IDs to specific Connect:Direct user IDs.

For more information, see Mapping credentials for Connect:Direct.

Example

You might see transfers failing with one of the following errors:

- ```
  BFGCD0001E: This task was rejected by the Connect:Direct API with the
  following error message: Connect:Direct Node detected error.
  LCCA000I The user has no functional authority to issue the selp command
  ```

- ```
  BFGCD0026I: Connect:Direct messages: The submit of the process
  succeeded. Process number 1092 (name F35079AE, SNODE MYNODE)
  executing. User fteuser does not have permission to override SNODEID.
  User fteuser does not have permission to override SNODEID. User
  fteuser does not have permission to override SNODEID.
  ```

If you see either of these errors, determine which Connect:Direct user ID is associated with the MQMD user ID that was used for the transfer request. This Connect:Direct user ID must have authority to perform the Connect:Direct operations required by the Connect:Direct bridge. For the list of functional authorities needed, and guidance on how to grant these authorities, see Mapping credentials for Connect:Direct by using the ConnectDirectCredentials.xml file.

Troubleshooting data conversion issues for text transfers to or from Connect:Direct nodes

Ensure that the network map of the Connect:Direct bridge node and any Connect:Direct nodes that are used as a transfer destination include the correct platform description.

About this task

When you transfer files in text mode between an MFT agent and a Connect:Direct node, code page and end-of-line character conversion is performed on the text. The transfer uses the operating system information in the network map of the Connect:Direct bridge node to determine the end-of-line characters of a remote node. If the information in the network map is incorrect, the end-of-line character conversion might be performed incorrectly.

Procedure

- For each remote node in your network map, select the correct operating system value.

  For example, if your Connect:Direct bridge node is on a Windows system, ensure that for each remote node in your network map you select the correct value from the **Operating System** list:

  –

If the remote node is on a Windows system, select `Windows`.

  –

If the remote node is on an AIX or Linux system, select `UNIX`.

  –

If the remote node is on a z/OS system, select `OS/390`.

  Transfers to remote nodes on other operating systems are not supported by the Connect:Direct bridge.
- For each remote node you transfer a file to or from, specify the operating system type of the remote Connect:Direct node in the `ConnectDirectNodeProperties.xml` file in the Connect:Direct bridge agent configuration directory.

  For more information, see Configure the `ConnectDirectNodeProperties.xml` file to include information about the remote Connect:Direct nodes and Connect:Direct node properties file format.

Related information

Transferring text files with MFT

###

Troubleshooting failed transfers to PDS or PDS members through the Connect:Direct bridge

If the destination of a transfer is a Connect:Direct node on z/OS and is a PDS or PDS member, the transfer fails if the **-de** parameter has not been specified with a value of overwrite.

Procedure

- If you submitted the transfer by using the **fteCreateTransfer** or **fteCreateTemplate** command, perform the following steps:
  a) Change the command that you submitted to include **-de** `overwrite`.
  b) Submit the command again.
- If you submitted the transfer by using the IBM MQ Explorer plugin, perform the following steps:
  a) Specify the source and destination information in the **Create New Managed File Transfer** wizard.
  b) Select **Overwrite files on the destination file system that have the same name**.
  c) Submit the command again.

Increasing the number of concurrent transfers for the Connect:Direct bridge

To increase the number of concurrent transfers that the Connect:Direct bridge agent can process, you must change three agent properties. You must also increase the maximum number of connections that the Connect:Direct node accepts.

About this task

The maximum number of concurrent transfers that a Connect:Direct bridge agent can process depends on the values of certain agent properties. The **maxSourceTransfers** and **maxDestinationTransfers** agent properties have a default value of five transfers for a Connect:Direct bridge agent. This default value is lower than the default of 25 transfers for other types of agent. A Connect:Direct bridge, where the agent is configured with the default values of **maxSourceTransfers** and **maxDestinationTransfers**, can process a maximum of 10 transfers at any one time: five transfers where the agent is the source, and five transfers where the agent is the destination.

These default values ensure that the Connect:Direct bridge agent does not exceed the maximum number of API connections to the Connect:Direct node. A Connect:Direct bridge agent with the default configuration uses a maximum of 10 API connections to the Connect:Direct node. The maximum number of connections accepted by a Connect:Direct node on AIX and Linux is controlled by the **api.max.connects** Connect:Direct parameter. For a Connect:Direct node on Windows, the equivalent parameter is **max.api.connects**.

If the rate at which your Connect:Direct bridge carries out large numbers of file transfers is not sufficient, you can increase the number of concurrent transfers that the Connect:Direct bridge agent processes.

Procedure

- Change the following agent properties for the Connect:Direct bridge agent:

  maxSourceTransfers
  Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

  maxDestinationTransfers
  Set this property to a value that is larger than 5, but smaller than or equal to 25. If you choose a value that is larger than 25, the agent might run out of memory unless you increase the amount of memory that is available to the JVM used by the agent.

  ioThreadPoolSize
  The default value of **ioThreadPoolSize** is 10. This property restricts the number of Connect:Direct node API connections for transfers where the Connect:Direct bridge agent is the source agent. These transfers are from Connect:Direct to Managed File Transfer. Use the following guidance to set the value of this property:

  - If the value of **maxSourceTransfers** is smaller than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to double the value of **maxSourceTransfers** or 10, whichever is the larger
  - If the value of **maxSourceTransfers** is larger than the value of **maxDestinationTransfers**, set **ioThreadPoolSize** to the sum of **maxSourceTransfers** and **maxDestinationTransfers**

- Also change the maximum number of concurrent API connections for the Connect:Direct node that is part of the Connect:Direct bridge.

  The Connect:Direct parameter that controls this number is **api.max.connects** if your node is on AIX and Linux, or **max.api.connects** if your node is on Windows.

  Make the following changes to the appropriate parameter:

api.max.connects (if the node in your Connect:Direct bridge is on AIX and Linux)
Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **api.max.connects** parameter is 16. For more information about how to set this parameter, see the Connect:Direct documentation.

max.api.connects (if the node in your Connect:Direct bridge is on Windows)
Set this parameter to a value larger than the sum of **maxSourceTransfers** and **maxDestinationTransfers**. The default value of the **max.api.connects** parameter is 10. For more information about how to set this parameter, see the Connect:Direct documentation.

Related tasks

Configuring the Connect:Direct bridge

Related reference

The MFT `agent.properties` file

Debugging a Connect:Direct process that is called by a file transfer

For a Connect:Direct process that is called by a file transfer, you can configure the Connect:Direct bridge agent to write log information to the `output0.log` file in the Connect:Direct bridge agent configuration directory.

About this task

To configure logging of the Connect:Direct processes, complete the following steps:

Procedure

1. Stop the Connect:Direct bridge agent.
2. Edit the `agent.properties` file in the *MQ_DATA_PATH*`/mqft/config/`*coordination_queue_manager*`/agents/`*bridge_agent_name* directory to include the property `logCDProcess`.

   The `logCDProcess` property can have one of the following values:

   - None - No information is logged. This is the default.
   - Failures - Information about failed Connect:Direct processes is logged.
   - All - Information about all Connect:Direct processes is logged.
3. Start the Connect:Direct bridge agent.

Results

Information about Connect:Direct processes is logged to the Connect:Direct bridge agent's `output0.log` file. The information that is logged comprises:

- MFT transfer ID
- Connect:Direct process name
- Connect:Direct process number
- Generated process definition
- File name of the process template, if the Connect:Direct process is user-defined

Related tasks

“Troubleshooting the Connect:Direct bridge” on page 161

Use the following tasks and examples to help you diagnose errors returned from the Connect:Direct bridge.

Related reference

The MFT `agent.properties` file

Troubleshooting other MFT issues

Use the following tasks to help you to diagnose general issues with Managed File Transfer (MFT).

Related reference

Return codes for MFT

Troubleshooting common MFT problems

A checklist of common MFT problems that are usually straightforward to define and fix.

Procedure

- If you see the following output from the **fteCreateAgent** command, it indicates that the coordination queue manager cannot be contacted and provides the IBM MQ reason code for why:

  ```
  BFGMQ1007I: The coordination queue manager cannot be contacted or has refused a
    connection attempt.
  The IBM MQ reason code was 2058. The agent's presence will not be published.
  ```

  Make the following checks:

  - Check that the coordination queue manager is currently available.
  - Check that you have defined the configuration correctly.
- If you are using user exit routines and there is a failure while the user exit is being called or just after the exit has been called, for example a product failure or power cut, check that the user exit is not called more than once.
- If you have an agent with a queue manager on a system with an IP address that is assigned by DHCP (rather than a static IP address), *and* the agent connects to that system by using a client TCP/IP connection, start the agent with the following system environment variable set:

  - Windows On Windows:

    ```
    set BFG_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=value"
    ```

  - Linux AIX On AIX and Linux:

    ```
    export BFG_JVM_PROPERTIES="-Dsun.net.inetaddr.ttl=value"
    ```

  where *value* is the time interval in seconds between each flush of the cached DNS values of the JVM.

  If the IP address of the queue manager system is reassigned for any reason (for example, because of a network outage, an IP lease expiry, or a system reboot), the agent reports its lost connection to the queue manager. After the JVM DNS cache is flushed, the agent can successfully reconnect. If this environment variable is not set, the agent cannot reconnect in this scenario without a JVM restart. This behavior is because the JVM internally caches the IP addresses of host names and does not refresh them by default.
- If you run the **fteStartAgent** command and see the following error message, your environment probably has additional library paths that conflict with Managed File Transfer:

  ```
  BFGCL0001E: An internal error has occurred. The exception was: 'CC=2;RC=2495;AMQ8568:
   The native JNI library 'mqjbnd' was not found. [3=mqjbnd]
  ```

  Ensure that the **javaLibraryPath** Managed File Transfer agent property references the correct location for the library. For example, for mqjbnd on AIX, set the property to `/usr/mqm/java/lib64`. For mqjbnd on Linux, set the property to `/opt/mqm/java/lib64`.

  z/OS If the LD_LIBRARY_PATH or LIBPATH environment variable is set to reference a 64-bit version of the library before the 32-bit version when the agent is running with a 32-bit version of Java, this error occurs.
- If you have enabled user authority checking by specifying `authorityChecking=true` in the agent property file, and all authority checks are failing even if the user has the required authority on the

relevant authority queue, ensure that the user that runs the agent has ALTUSER access control on the agent queue manager.

- If you have enabled user authority checking by specifying `authorityChecking=true` in the agent property file, and IBM MQ error messages are written to the agent `output0.log` file, perform one of the following actions:
  - Ignore the messages, because the agent is not affected.
  - Grant the user who runs the agent GET authority on the SYSTEM.FTE.AUTH* queues belonging to the agent.
- If you have edited the agent property file and the agent has not picked up the edits, restart the agent to ensure that the agent reads the new properties.
- z/OS

  If you are using the agent on z/OS to transfer to a PDS or PDSE data set and an abend occurs, your system might have limited disk space.

  The abend is likely to have a system completion code of B14 with a return code of 0C, indicating there is no space left.

  If you are transferring to a sequential data set, the transfer fails and indicates the out-of-space condition, but the agent remains operational.
- z/OS

  If you are using the agent on z/OS, and the WMQFTEP task generates some Java core dumps before becoming unresponsive, apply OMVS system services APAR OA43472.
- z/OS

  If you see the following output when running a configuration or administration script on z/OS, this indicates that the environment variable _BPXK_AUTOCVT=ON has not been set in the environment where the configuration or administration script is being run:

  ```
  FSUM7332 syntax error: got (, expecting Newline
  ```

  For more information about this environment variable and how to set it, see Environment variables for MFT on z/OS.

Troubleshooting MFT with JZOS

Steps to take if you encounter problems running MFT with JZOS.

Procedure

1. If the JZOS does not process successfully:
   - Add `, PARM='+T'` to the JCL. For example:

     ```
     //MQMFT EXEC PGM=JVMLDM86,REGION=0M,PARM='+T'
     ```
   - Add `set -x` to the environment file.
2. If you get either of the following errors, there is something wrong with your environment file and Managed File Transfer commands.

   JVMJZBL1038E Child shell process exited with exit code: 1
   JVMJZBL1042E JZOS batch launcher failed, return code=102

   The cause might be that specified paths are not valid.
3. From your environment file, locate the value of **BFG_PROD**.
   a) Go into OMVS and use the **`ls -ltr`** command.

   For example, if **BFG_PROD** is `/mqm/V9R2M0/mqft`, enter the following command:

   ```
   ls -ltr /mqm/V9R2M0/mqft/bin/fteBatch
   ```

b) Check this file exists, and that the batch job has read permission to the file.

c) Resolve any problems.

4. If the JCL still fails to process correctly:

a) Create a file in z/OS UNIX System Services, for example, `myenv` and use an editor to copy information from the environment file into this `myenv` file.

b) Save this file.

c) From the command line, use the following command to allow the file to run:

```
chmod +x myenv
```

d) To report any errors in the `myenv` file, run the following command:

```
. myenv
```

Note that the above command is *period blank filename*.

e) Correct any errors in both the `myenv` and environment files.

Troubleshooting MFT text data conversion error BFGIO0060E

What to do if a Managed File Transfer text transfer fails with error BFGIO0060E "Text data conversion has failed".

About this task

A text transfer can fail with error BFGIO0060E for one of two reasons:

- One or more characters in the source file cannot be converted from the source file code page to the destination file code page. This problem can occur when code pages have different character sets and certain characters cannot be converted between them.
- The source file encoding does not match the default encoding of the source agent. In this case performing a text transfer by using the default settings corrupts the character data.

Procedure

- If it is acceptable for some characters not to be converted, define a replacement character sequence at the destination agent so that the transfer does not fail.

  Specify the agent property **`textReplacementCharacterSequence`** to define a replacement character sequence. For more information, see Advanced agent properties: Input/output.
- Transfer a source file that does not have the same encoding as the source agent:
  - Specify the file encoding in a transfer definition file. For more information, see Using transfer definition files.
  - Specify the file encoding by using the **`-sce`** parameter with the **`fteCreateTransfer`** command. For more information, see **`fteCreateTransfer`**: start a new file transfer.
  - Specify the file encoding as part of an Ant move or copy task. For more information, see Using Apache Ant with MFT.
- Check that you have selected the correct source file encoding for a transfer:

  a) Set the destination file encoding to UTF-8.

  b) Transfer the file in text mode.

  c) Use a UTF-8 file viewer to view the contents of the file. If all characters in the file are correctly displayed, the source file encoding is correct.

Linux AIX *Troubleshooting an MFT agent process that disappears without logging diagnostic information*

On AIX and Linux platforms, if an agent process has disappeared but the agent log files do not contain any explanation, this might be caused by the way the agent was started.

About this task

Check for agent diagnostic information in the following ways:

Procedure

1. Check whether the agent log files say that the agent has been stopped.
2. Check whether the agent lock file `agent.lck` still exists.
3. Keep the agent running past the duration of the script that called the agent.

   If you start the agent from (for example) a shell script, all child processes associated with that script are removed when the script completes - including the agent process. To keep the agent running past the duration of the script that called the agent, complete the following step:

   a) Prefix the **fteStartAgent** command with the **nohup** command to disassociate the **fteStartAgent** process (and any child processes) from the script.

   In future, when the script terminates, the agent continues to run.

Troubleshooting an MFT agent or logger configuration that is not secure

If a Managed File Transfer process detects at startup that a configuration file contains sensitive information, is a keystore or truststore file, and has system-wide read, write, or delete permissions, the process does not start. If the condition is detected at run time, Managed File Transfer generates a warning message and ignores the contents of the configuration file. This affects the protocol bridge and Connect:Direct bridge capabilities, because they reload a configuration if it changes while the agent is running.

About this task

To determine the cause of the problem, complete the following checks:

Procedure

1. From the error message generated, identify the configuration file that has been reported as not secure.
2. Ensure that the file access permissions match the requirements needed.

   For more information, see MFT permissions to access sensitive configuration information.
3. Restart the agent or logger. For the protocol bridge or Connect:Direct credentials files, wait for the next reload.

Example

In this example error message, a database logger fails to start:

```
BFGDB0066E: The logger encountered a problem accessing its credentials file and will stop.
Reported error: BFGNV0145E: The 'Everyone' group has access to the file 'C:\mqmftcredentials.xml'.
```

In this example error message, a protocol bridge agent fails to start:

```
BFGIO0383E: The security permissions defined for credentials file 'C:\ProtocolBridgeCredentials.xml' do
not meet the
minimum requirements for a file of this type.
Reported problem: BFGNV0145E: The 'Everyone' group has access to the file
C:\ProtocolBridgeCredentials.xml'.
```

Related reference

MFT permissions to access sensitive configuration information

Troubleshooting messages that build up on your SYSTEM.MANAGED.DURABLE queues or fill your file system

If your IBM MQ Explorer plug-in uses a durable subscription on the coordination queue manager, messages can build up on the SYSTEM.MANAGED.DURABLE queues. If you have a high-volume Managed File Transfer network, use the IBM MQ Explorer plug-in infrequently, or both, this message data can fill the local file system.

Procedure

Remove the buildup of messages on the SYSTEM.MANAGED.DURABLE queues.

- Start the IBM MQ Explorer that uses the durable subscription.

  The Managed File Transfer plug-in for IBM MQ Explorer consumes the messages from the queue.
- Delete the messages from the queues manually.

Avoid the build up of messages on durable queues.

- Specify that the IBM MQ Explorer plug-in uses a non-durable subscription to the coordination queue manager.

  Perform the following steps in your IBM MQ Explorer:

  a) Select **Window** > **Preferences** > **IBM MQ Explorer** > **Managed File Transfer**

  b) From the **Transfer Log subscription type** list, choose NON_DURABLE.
- Clear durable subscriptions from the coordination queue manager that are created by the IBM MQ Explorer MFT plugin.

  The name of the durable subscription is prefixed to show that the subscription was created by the IBM MQ Explorer MFT plug-in, the host name, and the name of the user. For example `MQExplorer_MFT_Plugin_HOST_TJWatson`.

Related tasks

Retaining MFT log messages

Troubleshooting messages that build up on the SYSTEM.FTE queue on the coordination queue manager

The coordination queue manager for an IBM MQ Managed File Transfer (MFT) topology uses queued publish/subscribe to process status publications and distribute them to subscribers.

About this task

The publish/subscribe engine of the queue manager use a publication stream to monitor the SYSTEM.FTE queue for incoming publications. When it receives one, it makes copies of it to distribute to subscribers.

Under normal operation, the SYSTEM.FTE queue should be empty or contain only a handful of messages. If the queue depth continues to grow, then it usually means that the publish/subscribe engine is no longer using the publication stream. This typically happens if the coordination queue manager has recently been recreated.

To resolve this issue, complete the following steps:

Procedure

1. Check that the `SYSTEM.QPUBSUB.QUEUE.NAMELIST` namelist has been set up correctly and contains an entry for the `SYSTEM.FTE` queue.

   To do this, run the following MQSC command:

```
DISPLAY NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
```

This should generate output similar to the following example:

```
NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
NAMCOUNT(3)
NAMES(SYSTEM.BROKER.DEFAULT.STREAM
        ,SYSTEM.BROKER.ADMIN.STREAM
        ,SYSTEM.FTE)
DESCR(A list of queues for the queued Pub/Sub interface to monitor)
ALTDATE(2022-03-04)                     ALTTIME(14.34.37)
```

2. If the NAMES attribute does not include SYSTEM.FTE, add it using the following MQSC command:

```
ALTER NAMELIST(SYSTEM.QPUBSUB.QUEUE.NAMELIST)
NAMES(SYSTEM.BROKER.DEFAULT.STREAM,SYSTEM.BROKER.ADMIN.STREAM,SYSTEM.FTE)
```

Examining messages before publication

Agents send ordinary messages to the coordination queue manager that contain an MQRFH header. The MQRFH header requests that the message's payload is published. These messages are sent to the SYSTEM.FTE queue on the coordination queue manager, and the messages are typically published immediately from that queue. If error conditions stop this publication, you can examine the messages on the queue before publication is attempted.

About this task

To help diagnose the error conditions, examine the messages on the queue by completing the following steps:

Procedure

1. Disable the publish/subscribe engine in the coordination queue manager.

   Either complete this step using the IBM MQ Explorer or using an MQSC command. Be aware that this temporarily stops all publish/subscribe activity on the queue manager, including activity unrelated to Managed File Transfer if your coordination queue manager is also used for other purposes.

   IBM MQ Explorer:

   a. In the Navigator view, right-click the coordination queue manager and select **Properties**.

   b. From the **Properties** pane, select **Publish/Subscribe**.

   c. Select **Compatibility** from the **Publish/Subscribe mode** list.

   MQSC:

   ```
   ALTER QMGR PSMODE(COMPAT)
   ```

2. Send another message.

   Perform the Managed File Transfer action that has publication problems. For example, for agent registration, a message is sent whenever the agent is started (you do not need to repeatedly delete and create the agent to generate registration messages). Because the publish/subscribe engine is disabled, no publication takes place.

3. Browse the SYSTEM.FTE queue on the coordination queue manager.

   Use the IBM MQ Explorer to browse your coordination queue manager's SYSTEM.FTE queue.

   IBM MQ Explorer:

   a. In the Navigator view, expand the coordination queue manager and click **Queues**. In the Content view, right-click the SYSTEM.FTE queue and select **Browse Messages**. The **Message browser** window opens and shows the messages that would have been published.

b. The **User identifier** column shows the user ID contained in the message descriptor. A common reason for publication failure is that this user ID does not have publish authorization on the SYSTEM.FTE topic.

c. You can find out more information about each message (including the XML that will be published) by right-clicking the message and selecting **Properties**.

There is no MQSC command to inspect the contents of messages. If you are do not have the IBM MQ Explorer, use a different program that can browse queues and display all aspects of the messages found. You can use the **amqsbcg** sample program, if installed, as described in the following topic: Browsing queues. The `UserIdentifier` line shows the user ID. Alternatively, you can use **dmpmqmsg**; the user ID for a message is found in lines like these:

```
A RTM MQ24
A USR JOHNDOE
A ACC 1A0FD4D8F2F4C3C8C9D5F1F9C6F7C1C3F3F00019F7AC30000000000000000000
```

The second line in the example is the message descriptor user ID for that message.

4. Re-enable the coordination queue manager publish/subscribe engine.

You can either complete this step using the IBM MQ Explorer or using MQSC commands. After you have re-enabled the publish/subscribe engine in the coordination queue manager, any messages on the SYSTEM.FTE queue are processed immediately.

IBM MQ Explorer:

a. In the Navigator view, right-click the coordination queue manager and select **Properties**.

b. From the **Properties** pane, select **Publish/Subscribe**.

c. Select **Enabled** from the **Publish/Subscribe mode** list.

MQSC:

```
ALTER QMGR PSMODE(ENABLED)
```

Possible errors when configuring the Redistributable Managed File Transfer components

When you are configuring the Redistributable Managed File Transfer components, you might see errors "Native library for Windows could not be loaded" or "Use of bindings mode is not supported".

Native library for Windows could not be loaded

Windows You must install the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 libraries, available from Microsoft, on your system to use the Redistributable Managed File Transfer Agent. See The latest supported Visual C++ downloads.

For Continuous Delivery the Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017 and 2019 libraries are also required for the Redistributable Managed File Transfer Logger.

If these libraries are not installed and you try to run MFT commands, an error is reported:

- ```
  BFGUB0070E: Internal error: Native library for platform Windows (architecture
  amd64) could not be loaded because mqmft (Not found in java.library.path).
  ```

- ```
  BFGCL0043I: Specify the '-h' command line parameter to see more usage information.
  ```

Check that the Microsoft libraries are installed. If the libraries are not installed, install them and run the command again.

Use of bindings mode is not supported

The Redistributable Managed File Transfer Agent can only connect to IBM MQ in client mode. If you try run commands in bindings mode, an error is reported:

- ```
  BFGCL0408E: Unable to obtain IBM MQ installation information for queue manager 'queue manager
  name'.
  Reason Cannot run program "../bin/dspmq": error=2, No such file or directory
  ```

When you are issuing commands, you must provide the queue manager host, port, name, and channel name.

From IBM MQ 9.3.0, you might also see this error for the Redistributable Managed File Transfer Logger, which only supports FILE type loggers that are connecting in client mode to the coordination queue manager.

Related tasks

Downloading and configuring Redistributable Managed File Transfer components

Related reference

BFGCL0001 - BFGCL9999

BFGUB0001 - BFGUB9999

Troubleshooting using UAC and virtual store with MFT

User Account Control (UAC) is present in Windows Server 2008 R2 and other similar operating systems. This is a security infrastructure and one of its features is to divert user data stored in the central `Program Files` directory to a user location, which is known as virtual store.

About this task

If only the Managed File Transfer tools are used to manage the data structures, Managed File Transfer is not affected by UAC and virtual store. However, if the directory structure is changed or rebuilt using standard operating system tools by a non-IBM MQ administrator, the new structure might be diverted into a virtual store. This can cause one or more of the following situations:

- Users, including the IBM MQ administrator, can no longer see files in their expected location.
- An agent might fail to start, reporting message BFGCL0315 but giving no supporting reason code.
- The log files cannot be found at the location reported by the agent.
- An agent when started with the **-F** parameter might fail to start, reporting the following message:

  ```
  The current directory is invalid
  ```

  .

To correct all of these situations, complete the following steps:

Procedure

- As an IBM MQ administrator, use the **fteDeleteAgent** and **fteCreateAgent** commands to rebuild the agent structure.
- As an operating system administrator, remove the IBM MQ entries in the virtual store of the affected users.

  For example, on Windows the location of the virtual store is as follows: *%USERPROFILE%* `\AppData\Local\VirtualStore\`

Related reference

fteDeleteAgent

fteCreateAgent
  ```

Windows *Troubleshooting an MFT agent or logger running as a Windows service*

You can run a Managed File Transfer agent, a stand-alone database logger, and a stand-alone file logger, as Windows services. If you are having a problem with these Windows services, you can use the service log files and the steps in this topic to diagnose the issue.

About this task

For information about configuring your agent, stand-alone logger, or stand-alone file logger, to run as a Windows service, see Starting an MFT agent as a Windows service and fteModifyLogger: run an MFT logger as a Windows service.

Note: If the redistributable agent is going to run as a Windows service, then the **BFG_DATA** environment variable needs to be set in the system environment for the service to work.

Procedure

- Choose the level of logging.

  When you use the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**, or **fteModifyLogger** command to run an agent or logger as a Windows service, you can choose the level of logging by using the **-sl** parameter. The possible values for this parameter are `error`, `info`, `warn`, and `debug`. The default value is `info`.

  The log file for the Windows service has the file name `service`*`date`*`.log`, where *date* is the date when the service was started. The file for an agent is written to the directory *MQ_DATA_PATH*`\mqft\logs\`*coordination_qmgr_name*`\agents\`*agent_name*. This directory is the same directory that Managed File Transfer Agent trace files are written to. The file for the logger is written to the directory *MQ_DATA_PATH*`\mqft\logs\`*coordination_qmgr_name*`\loggers\`*logger_name*.

  If you have problems starting an agent, or a stand-alone logger as a Windows service, try setting the logging level to `debug` using the **-sl** parameter. Additional information is written to the `service`*`date`*`.log` file.

  Note: When the logging level is set to `debug`, the user account and password that you are using to run the Windows service are shown in the log file in plain text.

- Choose the number of log files.

  When you use the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**,or **fteModifyLogger** command to run an agent or a stand-alone logger as a Windows service, you can choose the number of log files by using the **-sj** parameter. Specify the following text as part of your command to change the number of log files: `-sj -Dcom.ibm.wmqfte.daemon.windows.windowsServiceLogFiles=`*`number`*, where *number* is the number of log files that you want expressed as a positive integer. If you do not specify the number of log files, the default is five.

- Grant "Log on as a service" authority

  The Windows account that you use to run the service must have the **Log on as a service** right. If you try to start the service, either with the **fteStartAgent**, **fteStartLogger** command, or with the Windows **Sc.exe** command, and you are using a user account that does not have this right, a **Services** window opens. If the service you wanted to start was to run an agent, this window contains the following message:

  ```
  Unable to start Windows service mqmftAgentAGENT@QMGR.
  System error 1069: The service did not start due to a logon failure.
  ```

  In this message, *AGENT* is your agent name and *QMGR* is your agent queue manager name. If you are trying to run a stand-alone logger as a service, a similar message is produced, which refers to the logger rather than an agent.

To prevent this error, give the Windows account that you use to run the service the **Log on as a service** right. For example, on Windows 10 complete the following steps:

a) From the **Start** menu, click **Administrative Tools** > **Local Security Policy**.

b) In the **Security Settings** pane, expand **Local Policies**, and then click **User Rights Assignments**.

c) In the **Policy and Security Setting** pane, double-click **Log on as a service**.

d) Click **Add User or Group**, then add the user that you want to run the service to the list of users that have the **Log on as a service** right.

You provided this user name when you ran the **fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**,**fteCreateLogger**, or **fteModifyLogger** command.

Note: The error `System error 1069: The service did not start due to a logon failure.` can also be caused by an incorrect password.

- Hide your Windows account password.

When you configure your agent or stand-alone logger to run as a Windows service, you specify a user name and password to use. In the following example, the agent AGENT1 is created, which has an agent queue manager QMGR1 and is configured to run as a Windows service:

```
fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser -sp ftepassword
```

In this example, the Windows service runs with a user name of `fteuser`, which has an associated password `ftepassword`. When you run the **fteCreateAgent** command, or one of the other commands that accepts the **-s** parameter, you specify the password for the Windows account in plain text. If you prefer not to display your password, complete the following steps:

a) Run the command (**fteCreateAgent**, **fteCreateCDAgent**, **fteCreateBridgeAgent**, **fteModifyAgent**, **fteCreateLogger**or **fteModifyLogger**) without specifying the **-sp** parameter.

For example:

```
fteCreateAgent -agentName AGENT1 -agentQMgr QMGR1 -s -su fteuser
```

Note: The command produces a message that warns you that you must set the password by using the Windows Services tool before the service starts successfully.

b) Open the Windows **Services** window.

c) In the list of services, right-click the agent or stand-alone logger service and select **Properties**.

The agent service display name is `Managed File Transfer Agent` *`AGENT @ QMGR`*, where *AGENT* is the agent name and *QMGR* is your agent queue manager name. The logger service display name is `Managed File Transfer Logger for property set` *`coordination_qmgr_name`*, where *coordination_qmgr_name* is the coordination queue manager that you specified for the stand-alone logger to use as its property set.

For more information about the property set, see fteStartLogger and fteModifyLogger.

d) In the **Properties** window, select the **Log On** tab.

e) Enter the password for the user account that runs the service in the **Password** and **Confirm password** fields. The password characters are hidden as you enter them.

f) Click **OK**.

Related tasks

Starting an MFT agent as a Windows service

Related reference

fteCreateAgent

fteModifyAgent

fteCreateCDAgent

fteCreateBridgeAgent
fteCreateLogger
fteModifyLogger
fteStartLogger

Windows *How agent and logger JVM options are processed*

If you use the **-sj** parameter of the **fteModifyAgent** or **fteModifyLogger** command to modify an existing Windows Service definition for an agent or logger by updating, adding, or removing Java system properties, the existing Windows Service is first deleted before a new one is created in its place, and the agent or logger properties file is updated with the properties for the new Windows Service. The new Windows Service definition must be consistent with the updated Windows Service properties that are defined in the agent or logger properties file.

From IBM MQ 9.0.0 Fix Pack 4, additional checks are added under APAR IT22423 such that any updates that are made to the JVM options for an agent or logger with the **-sj** parameter of the **fteModifyAgent** or **fteModifyLogger** command are verified to make sure that the options have been correctly specified. If the properties are found to be invalid, or otherwise could not be validated, the **fteModifyAgent** or **fteModifyLogger** command fails and an appropriate error message is displayed.

If the JVM properties are valid and the deletion of the existing Windows Service is successful, but a failure then arises when the **fteModifyAgent** or **fteModifyLogger** command is creating the new Windows Service, the command attempts to remove the properties that define the replacement Windows Service from the agent or logger properties file. In this case, error messages are returned to explain that the agent or logger could not be modified, the old Windows Service was deleted but a new Windows Service could not be created and the agent or logger will therefore not run as a Windows Service. You must then manually verify that the state of the Windows Service definition is consistent with the Windows Service properties that are defined in the agent or logger properties file, and take the appropriate action to correct any inconsistencies.

Related reference

fteModifyAgent: run an MFT agent as a Windows service
fteModifyLogger: run an MFT logger as a Windows service

Troubleshooting when MFT does not read keystore properties for AMS

The keystore configuration file location, if not present in the default location, must be specified by the *MQS_KEYSTORE_CONF* variable in order for the Java AMS to run in client mode. If the location is not specified, the Managed File Transfer Agent logs will show the error message: "Failed to read keystore properties from the keystore configuration file."

About this task

The default location for the keystore configuration file is *home_directory*`/.mqs/keystore.conf`. If the location of the keystore configuration file is not the default location, complete the following steps:

Procedure

1. Start the FTE agent in client mode.
2. Apply AMS security to SYSTEM.FTE.DATA.<agent name> queue.

   If the keystore configuration file is not in this location, all transfers will fail with no acknowledgment.
3. Set the system variable **BFG_JVM_PROPERTIES** to **BFG_JVM_PROPERTIES**=-**DMQS_KEYSTORE_CONF**=*path to keystore_config file* for the **fteStartAgent** command.
4. Set the system variable **MQS_KEYSTORE_CONF** to **MQS_KEYSTORE_CONF**=*path to keystore_config file* for the **fteStartAgent** command.

   This must be set to ensure all agents run, regardless of the mode they are running in.

Results

Note: If the Java AMS is running in bindings mode, error AMQ9062 is shown in the queue manager's error log if the keystore configuration file is not in the default location.

Avoiding BFGSS0023E errors when removing fix packs

If you uninstall a fix pack from an installation to move back to a previous version of the product, and an agent associated with the installation was involved with managed transfers at the time the uninstallation took place, then that agent cannot start and will report an BFGSS0023E error. You can avoid this error by completing a number of steps to prevent BFGSS0023E messages from appearing when the agents are restarted.

About this task

For every in-flight managed transfer that an agent is currently involved in, there is a message on the agent's SYSTEM.FTE.STATE.*agent_name* queue. This message stores checkpoint information on the managed transfer, and is used if the managed transfer goes into recovery. Once a managed transfer has finished, then the corresponding message on the SYSTEM.FTE.STATE.*agent_name* queue is removed.

Each state message contains some internal header information indicating which version of the Managed File Transfer component was being used by an agent when the managed transfer was running. The version information shows the specific fix pack level, so, for example, if an IBM MQ 8.0.0 Fix Pack 5 agent was running a managed transfer, then the state message for that managed transfer would contain a reference to IBM MQ 8.0.0 Fix Pack 5.

If a fix pack is uninstalled from an installation, and an agent associated with that installation has in-flight transfers associated with it, then the agent fails to start and reports the following error:

```
BFGSS0023E: The agent is configured to use IBM MQ queues that contain data created using a later
version
of the product. The agent cannot run in this configuration and will end.
```

For example, if an IBM MQ 8.0.0 Fix Pack 5 agent has some in-flight transfers running when it is stopped and downgraded to the IBM MQ 8.0.0 Fix Pack 4 level, the next time the agent is started, it checks the messages on its SYSTEM.FTE.STATE.*agent_name* queue and finds that they were written when it was using IBM MQ 8.0.0 Fix Pack 5. As it is now using IBM MQ 8.0.0 Fix Pack 4, the agent reports the BFGSS0023E error described in the previous paragraph and shuts itself down.

If you want to remove a fix pack, completing the following steps should prevent the BFGSS0023E messages from appearing when the Managed File Transfer agents are restarted:

Procedure

1. Ensure that all of the agents have completed their managed transfers.
2. Stop the agents.
3. Remove the fix pack.
4. Restart the agents.

Related tasks

Starting an MFT agent

Reverting a queue manager to a previous version on UNIX

Reverting a queue manager to a previous version on Windows

Related reference

MFT Agent queue settings

BFGSS0001 - BFGSS9999

Troubleshooting message problems

Related tasks

“Collecting information for data conversion problems on Multiplatforms” on page 330

If you need assistance to resolve a problem with data conversion on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

"Collecting information for dead-letter queue problems on Multiplatforms" on page 331
If you need assistance to resolve a problem where an IBM MQ queue manager is placing messages on its dead-letter queue (DLQ) on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

"Collecting information for error message and FFST problems on Multiplatforms" on page 333
If you need assistance to resolve a problem when IBM MQ is logging error messages or writing FFSTs (FDC files) on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Troubleshooting undelivered messages

If messages are not delivered successfully and are therefore placed on the dead-letter queue, you can browse the queue and decide how to dispose of any messages that you find.

About this task

Messages that cannot be delivered for some reason are placed on the dead-letter queue. Each queue manager typically has a local queue to use as a dead-letter queue, so that messages that cannot be delivered to their correct destination can be stored for later retrieval.

Procedure

1. Check whether the queue contains any messages by using the MQSC **DISPLAY QUEUE** command.
2. If the queue contains messages, 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.
3. 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.

   IBM MQ provides a dead-letter queue handler that allows you to specify how messages found on a dead-letter queue are to be processed or removed. For more information about dead-letter queues and handling undelivered messages, see Working with dead-letter queues.

Related concepts
Dead-letter queues
Related reference
runmqdlq (run dead-letter queue handler)

ALW Troubleshooting IBM MQ Telemetry problems

Look for a troubleshooting task to help you solve a problem with running IBM MQ Telemetry applications.
Related concepts
IBM MQ Telemetry
Related tasks
"Tracing the telemetry (MQXR) service" on page 511

The trace facility provided by the IBM MQ telemetry (MQXR) service is provided to help IBM Support diagnose customer issues related to the service.

ALW Telemetry logs, error logs, and configuration files

Enable and 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`.

Optionally, if JSON format logging for MQTT is enabled, a JSON log file is written. The log path is:

```
WMQ data directory\Qmgrs\qMgrName\errors\mqxr.json
```

The format of the log file is `mqxr_n.json`.

The IBM MQ Telemetry sample configuration created by IBM 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.

IBM 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 6. Telemetry configuration directory on Windows

```
/var/mqm/qmgrs/qMgrName/mqxr
```

Figure 7. Telemetry configuration directory on AIX or Linux

JVM

Set Java properties that are passed as arguments to the telemetry (MQXR) service in the `java.properties` file. 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 6 on page 179 and Figure 7 on page 179.

Modify `java.properties` by using the setmqxr command For example:

```
V 10.0.0 setmqxr properties -m <QueueManager> -k -Xmx1024m -v
setmqxr properties -m <QueueManager> -k -Xms1024m -v
```

JSSE

Enable a JSSE debug trace for the MQTT service, so you can see the TLS handshake.

Do this by adding the following property to the `java.properties` file, as previously described in the "JVM" on page 179 section.

```
javax.net.debug=<value>
```

where *<value>* is one or more of `all`, `true`, or `ssl`, optionally followed by debug specifiers. For more information about these options, see Debugging Utilities in the IBMJSSE2 provider documentation.

The output is sent to the `mqxr.stdout` file.

JAAS

From IBM MQ 9.3.0, the JAAS.Login module files have been moved from the `&MQ_INSTALL_DIRECTORY&/ mqxr/samples/samples` directory to the directory `&MQ_INSTALL_DIRECTORY&/ mqxr/samples/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 the following directory:

```
WMQ data directory\Qmgrs\qMgrName\service.env
```

```
CLASSPATH= WMQ Installation Directory\mqxr\samples\jaas
```

Figure 8. Sample `service.env` for Windows

Note: `service.env` must not contain any variables. Substitute the actual value of *`WMQ Installation Directory`*.

Trace

See "Tracing the telemetry (MQXR) service" on page 511. 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
```

com.ibm.mq.MQXR.GenerateJavaDump property

In some cases, it is not necessary for the MQXR service to produce a Java dump as part of First Failure Support Technology (FFST) generation.

You can set the system property **com.ibm.mq.MQXR.GenerateJavaDump** to specify whether or not Java core dumps are generated:

- Setting the property to true means that Java core dumps are always generated alongside FDCs.
- Setting the property to false means that Java core dumps are never generated alongside FDCs.

The **com.ibm.mq.MQXR.GenerateJavaDump** property is disabled by default. The property is located at `mqinstall/qmgrs/qmgr_name/mqxr/java.properties` as well as `mqinstall/qmgrs/qmgr_name/amqp/amqp_java.properties` in your IBM MQ installation. Both of these properties are required to be set to the same value.

V 10.0.0 **com.ibm.mq.MQXR.GenerateHeapDump property**

It is possible to generate a Java heap dump automatically when the IBM MQ Telemetry service reports a low memory exception.

You can set the system property **com.ibm.mq.MQXR.GenerateHeapDump** to specify whether or not a Java heap dump is generated:

- Setting the property to true means that a Java heap dump is generated when the IBM MQ Telemetry service reports a low memory exception.
- Setting the property to false means that Java heap dumps are never generated automatically.

The **com.ibm.mq.MQXR.GenerateHeapDump** property is disabled by default. The property is located at `mqinstall/qmgrs/qmgr_name/mqxr/java.properties` as well as `mqinstall/qmgrs/qmgr_name/amqp/amqp_java.properties` in your IBM MQ installation. Both of these properties are required to be set to the same value.

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 MQ Telemetry Transport, and for adding security features.
- The 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 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
      ```

ALW Enabling JSON formatted logs for MQTT

To enable JSON format error logging, you need to modify the MQTT configuration files `mqxrtraceOn.properties` and `mqxrtraceOff.properties`.

About this task

You can configure JSON format logging to be used exclusively, or simultaneously, alongside text-based error logging.

Note: Once you have modified the configuration files, you need to restart the MQTT service for any changes to come into effect.

Procedure

1. To enable JSON format logging exclusively, modify the `mqxrtraceOn.properties` and `mqxrtraceOff.properties` configuration files by using the **`setmqxr`** command and update the **`handlers`** property as follows:

   V 10.0.0

   ```
   setmqxr properties -m <QueueManager> -k handlers -v
   com.ibm.mq.util.logging.MQJSONErrorLogFileHandler
   ```

2. To enable JSON format logging simultaneously alongside text-based error logging, modify the `mqxrtraceOn.properties` and `mqxrtraceOff.properties` configuration files by using the **`setmqxr`** command and update the **`handlers`** property as follows:

   V 10.0.0

   ```
   setmqxr properties -m <QueueManager> -k
   handlers -v "com.ibm.mq.util.logging.MQErrorLogFileHandler,
   com.ibm.mq.util.logging.MQJSONErrorLogFileHandler"
   ```

Results

In either case, this enables JSON format error logging, and configures the logger with its following default properties:

- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.level`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.filter`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.limit`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.count`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.append`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.permissions`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.formatter`
- `com.ibm.mq.util.logging.MQJSONErrorLogFileHandler.pattern`

Functionally, these properties work identically to the text-based error logger configured by the `com.ibm.mq.util.logging.MQErrorLogFileHandler`.

Related information

V 10.0.0 setmqxr

V 10.0.0 ALW Enabling Java heap dump generation to troubleshoot low memory exceptions for MQTT

To enable automatic generation of a Java heap dump when the IBM MQ Telemetry service reports a low memory exception, you need to modify the MQTT Java Virtual Machine (JVM) configuration file `java.properties`.

About this task

Note: After you have modified the configuration file, you need to restart the MQTT service for any changes to come into effect.

Procedure

1. V 10.0.0
   Modify the `java.properties` file by using the following command:

   ```
   setmqxr -m <QueueManager> -k -Dcom.ibm.mq.MQXR.GenerateHeapDump -v true
   ```

2. V 10.0.0
   Restart the MQTT service.

Results

A Java heap dump is generated automatically when the MQTT service reports a low memory exception. Only one Java heap dump file is generated. To enable the creation of another file, simply restart the service.

The newly generated Java heap dump file can be found in the IBM MQ errors directory at `WMQ data directory\errors`.

Related information

Performance and scalability of IBM MQ Telemetry

V 10.0.0 setmqxr

ALW Additional diagnostics using the `controlMQXRChannel` command

Using the **`controlMQXRChannel`** command to provide additional diagnostic information about the MQXR service.

Procedure

Run the following command to provide useful diagnostic information from the MQXR service:

```
<MQ_INSTALLATION_PATH>\mqxr\bin\controlMQXRChannel -qmgr=<QMGR_NAME> -mode=diagnostics
-diagnosticstype=<number>
```

The diagnostic information generated depends on the value of the **`-diagnosticstype`**=*<number>* parameter:

`-diagnosticstype=` ***0***
: Thread dump written to the console

`-diagnosticstype=` ***1***
: FDC with some internal service statistics

`-diagnosticstype=` ***2***
: FDC with internal statistics, plus information about the clients that are currently connected

`-diagnosticstype=` ***3***
: Heap dump

-diagnosticstype= *4*
Javacore

-diagnosticstype= *5*
Full system dump

-diagnosticstype= *6*
Detailed information about a specific client. Note that you must also supply the **-clientid** parameter for that client as well.

ALW 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 179. When the telemetry (MQXR) service detects the client has timed out, it writes a first-failure data capture

Option	Description
	(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/archive/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 181.

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?

ALW 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 IBM MQ 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 179.
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`.

ALW 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. Re-create 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 Support.

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 181

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 IBM 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.

ALW 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 that are used by the telemetry (MQXR) service locate the telemetry (MQXR) service programs.

Before you begin

The IBM 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 IBM MQ Telemetry service, SYSTEM.MQXR.SERVICE, is in the **Services** folder. It is visible if the IBM MQ 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 179.

   Actions
   Install, or uninstall and reinstall the IBM MQ Telemetry feature.
2. Start the service; wait for 30 seconds; refresh the IBM MQ 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 179.
   See if the problem occurs with only the sample channel defined. Back up 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.

ALW Resolving problem: JAAS login module not called by the IBM MQ 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\jaas\LoginModule.java to create your own authentication class *WMQ installation directory*\mqxr\samples\jaas\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 190 is a sample configuration.

From IBM MQ 9.3.0, the JAAS.Login module files have been moved from the `&MQ_INSTALL_DIRECTORY&/ amqp/samples/samples` directory to the directory `&MQ_INSTALL_DIRECTORY&/ amqp/samples/jaas`

Procedure

1. Look in `mqxr.log` for an exception thrown by `javax.security.auth.login.LoginException`.

   See "Server-side logs" on page 179 for the path to `mqxr.log`, and Figure 14 on page 192 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 190.
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 9. WMQ Installation directory\data\qmgrs\qMgrName\mqxr\jaas.config

When `SYSTEM.MQTT.SERVICE` starts, it adds the path in Figure 10 on page 191 to its classpath.

```
CLASSPATH=C:\WMQTelemtryApps;
```

Figure 10. WMQ Installation directory\data\qmgrs\qMgrName\service.env

Figure 11 on page 191 shows the additional path in Figure 10 on page 191 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 11. Classpath output from runMQXRService.bat

The output in Figure 12 on page 191 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 12. 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 13 on page 191. 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 13. Exception thrown when connecting to the Eclipse Paho sample

`mqxr.log` contains additional output shown in Figure 13 on page 191.

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 14 on page 192, 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 14. Error loading JAAS configuration

Troubleshooting multicast problems

Use this information to help you to detect and deal with multicast problems on IBM MQ.

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 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)

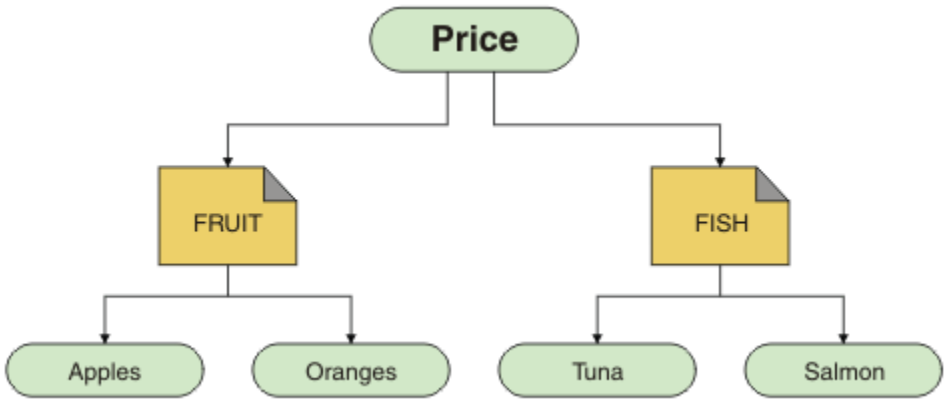
DEF COMMINFO(MC2) GRPADDR(227.20.133.2)
```

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)
```

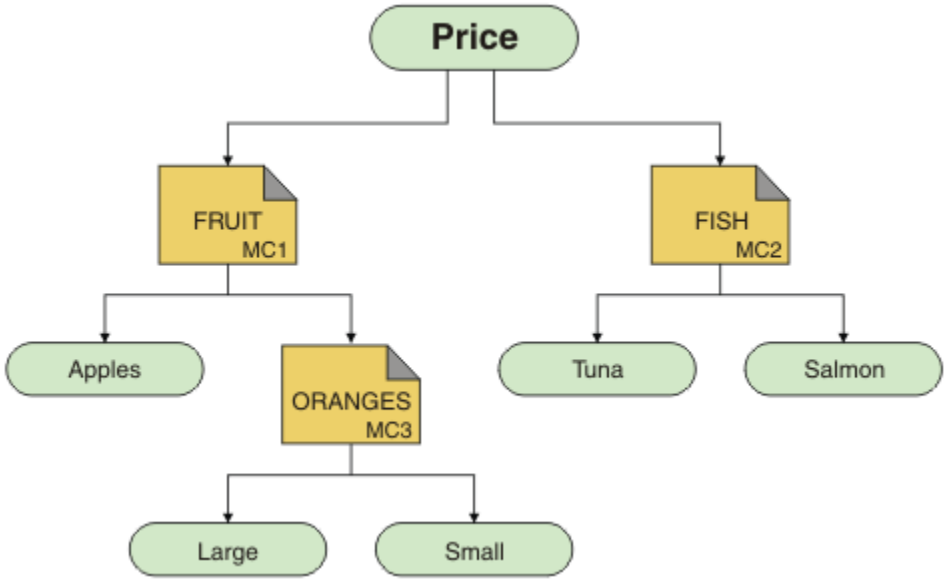
where *227.20.133.1*, *227.20.133.2*, and *227.20.133.3* are valid multicast addresses.

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.

Troubleshooting queue manager problems

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.

Log I/O operation exceeded threshold message

- **Scenario:**You receive an AMQ6729W`Log I/O operation exceeded threshold` in the queue manager error log,
- **Explanation:** IBM MQ has detected that log read and write or input and output operations are taking longer than anticipated. This might be due to problems with the operating system or storage system and can impact the queue manager performance.
- **Solution:** Use the **AMQ_IODELAY**, **AMQ_IODELAY_INMS** and **AMQ_IODELAY_FFST** environment variables to either aid in diagnosing storage performance issues, or to increase tolerance for such delays. For more information, see AMQ_IODELAY, AMQ_IODELAY_INMS and AMQ_IODELAY_FFST.

IBM MQ coordinating with Db2 as the resource manager error

- **Scenario:** You start your queue managers from the IBM MQ Explorer and are having problems when coordinating Db2. When you check your queue manager error logs, you see an error like the one shown in the following example:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
Host(HOST_1) Installation(Installation1)
VMRF(7.1.0.0) QMgr(A.B.C)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
for xa_open. The queue manager is continuing without this resource manager.
```

- **Explanation:** The user ID (default name is MUSR_MQADMIN) which runs the IBM MQ Service process `amqsvc.exe` is still running with an access token which does not contain group membership information for the group DB2USERS.
- **Solution:** After you have ensured that the IBM MQ Service user ID is a member of DB2USERS, use the following sequence of commands:

  1. Stop the service.
  2. Stop any other processes running under the same user ID.
  3. Restart these processes.

  Rebooting the machine would ensure the previous steps, but is not necessary.

Troubleshooting queue manager cluster problems

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 74.

Note: The steps in this task require you to run MQSC commands. How you do this varies by platform. See Administering IBM MQ using MQSC commands.

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 MQSC command:

  ```
  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 MQSC command:

  ```
  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 MQSC 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.

  Important: If a channel has STOPPED because it has run out of retries, restarting the queue manager does not bring the channel out of its STOPPED state.

- Check that all queue managers in the cluster are aware of all the full repositories.

  You can do this using the following MQSC command:

  ```
  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 MQSC command:

  ```
  display chstaus(*) 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 MQSC command:

  ```
  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 MQSC command:

  ```
  display ql(*) curdepth where (usage eq xmitq)
  ```

  You can check system queues using the following MQSC command:

  ```
  display ql(system*) curdepth
  ```

Related tasks

“Collecting information for IBM MQ clustering problems on Multiplatforms” on page 328
If you need assistance to resolve an issue when an IBM MQ queue manager has a problem with cluster queues, topics or channels on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Configuring a queue manager cluster

“Making initial checks” on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

“Making initial checks on z/OS” on page 50
Before you start problem determination in detail on z/OS, 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 19
Before you start problem determination in detail on IBM i, 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 reference

Messages and reason codes

Multi Application balancing troubleshooting

Use this section to help you troubleshoot problems with application balancing

Multi *Applications not balancing correctly*

Many symptoms relating to application balancing can be diagnosed using the DISPLAY APSTATUS command in various ways.

DIS APSTATUS(X) TYPE(APPL)

Symptom

The expected application is not listed.

Solution

- Verify the APPLTAG field is set correctly, either in code, or when the application is started.
- Investigate other listed applications in DIS APSTATUS(*) output to see if any are unexpected due to the name being formed incorrectly, or defaulting.
- Try running the command `DIS APSTATUS(X) TYPE(LOCAL) where(MOVABLE eq NO)` on each queue manager in the uniform cluster, to look for application instances which can not be distributed around the uniform cluster.

Symptom

The expected total number of applications are not listed.

Solution

- Verify you are actually launching the expected number of instances to connect to the uniform cluster
- Verify that the uniform cluster is communicating correctly and all queue managers are reporting application counts in DIS APSTATUS(X) TYPE(QMGR).

Symptom

The expected total number of applications are listed but some applications are flagged as not movable.

Solution

On each queue manager in the uniform cluster, use DIS APSTATUS(X) TYPE(LOCAL) where (MOVABLE equals NO) and investigate the IMMREASN field.

Symptom

The balanced state is UNKNOWN

Solution

This is a temporary state, and will resolve itself shortly. Retry the command in a while.

Symptom

The balanced state is NOTAPPLIC.

Solution

- If this queue manager is not in a uniform cluster, the balance state is always NOTAPPLIC as nothing can be rebalanced.
- In a uniform cluster, this means there has never been an application with this name connecting as movable. Information on this application is not distributed around the cluster.

Use DIS APSTATUS(X) TYPE(LOCAL) where(MOVABLE eq NO) and investigate the IMMREASN field.

Symptom

The balanced state is NO

Solution

- Monitor this output across a period of time. If applications constantly connect and disconnect this might be the appropriate answer as the instances are not given chance to rebalance.
- Use DIS APSTATUS(X) TYPE(QMGR) to investigate the numbers on each queue manager, which indicates queue managers with a surplus, or deficit, number of instances and continue the investigation on those queue managers.

DIS APSTATUS(X) TYPE(QMGR)

Symptom

Not all queue managers in the uniform cluster are listed.

Solution

- Verify the BALSTATE is not NOTAPPLIC as that prevents information being flown around the uniform cluster.

  Use DIS APSTATUS(X) TYPE(LOCAL) to look at the IMMREASN field.
- Verify any missing queue managers are running.
- Verify the state of clustering, and that channels are running between this queue manager and the missing queue manager.

Symptom

A queue manager is listed as ACTIVE(NO)

Solution

- Verify any missing queue managers are running
- Verify the state of clustering and that channels are running between this queue manager and the inactive queue manager

Symptom

A queue manager has some immovable instances of an application.

Solution

On that queue manager in the uniform cluster, Use DIS APSTATUS(X) TYPE(LOCAL) where(MOVABLE eq NO) and investigate the IMMREASN field.

Symptom

The BALSTATE is unexpected.

Solution

- Monitor this over time, as the BALSTATE is the state when the queue manager last attempted to rebalance applications, which only happens periodically
- Are applications continually connecting and disconnecting? If so, this might prevent the application ever being rebalanced into a stable state.

- If BALSTATE stays unbalanced, look at the error logs on the queue managers that are BALSTATE(HIGH) and BALSTATE(LOW), which should indicate whether they are requesting application instances, and how many were permitted to be moved.
- Verify DIS APSTATUS(X) TYPE(LOCAL) where(IMMCOUNT gt 1) to see if there are instances which are failing to move when requested.

DIS APSTATUS(X) TYPE(LOCAL)

This display command can be used to diagnose many issues which might cause an application not to rebalance as expected. Firstly, check the IMMDATE and IMMTIME fields to see if the application is only temporarily marked as immovable.

Other reasons for applications failing to rebalance are indicated by the IMMREASN. The following table shows the various causes (IMMREASN) and actions needed. Note, that in most cases, these causes need to be reviewed with the application developer or owner concerned

IMMREASN	**ACTION**
NOTCLIENT	The application is using server bindings and therefore cannot be moved to another queue manager. In most cases applications can be modified to use a client connection. This might require rebuilding the application however depending on the language and library versions in use.
NOTRECONN	The application connection is not marked as 'reconnectable'. This might be a deliberate decision in the application code, because its design requires that all messages flow to and from a single queue manager, or might indicate a configuration error or oversight (for example very old client libraries do not support reconnection). Note that for application balancing to work RECONNECT_QMGR is not sufficient, as this indicates reconnection is only permitted to the 'same' Queue Manager instance. To see the connection options in use from an application instance you can issue **DIS CONN(*) TYPE(CONN) WHERE(CONNTAG eq 'xxx') CONNOPTS**, where xxx is the CONNTAG from the DIS APSTATUS output.
APPNAMECHG	The application is making multiple connections on the same TCP connection, but with differing application names. This means that application instances cannot be reliably separated so rebalancing is prevented. If this issue occurs the application code is probably explicitly overriding the application name in the MQCONNX call.
MOVING	This should be a temporary status only as it indicates that the application instance has already been identified for rebalancing.
INTRANS	The application is currently in a transaction so rebalancing is avoiding interruption (rollback). If the application developer or deployer is not concerned about excessive rollbacks for this application, and would prefer to prioritize maintaining a consistent balance of application connections, this constraint can be ignored both in application code or configuration settings; see **BalanceOptions** for more information. Alternatively, how long the queue manager permits transactions to continue before considering interruption anyway can be modified using the **Timeout** field.
REPLY	This application has been marked as type 'request reply' and is waiting for a response to a previously dispatched request message. If you do not want to wait for responses, marking as type 'SIMPLE' prevents this wait.

IMMREASN	ACTION
	Alternatively, you can configure the extent of the waiting period using either the message expiry of the application request messages, or the **Timeout**. Note that it is often best to configure both appropriately, so that **Timeout** does not unexpectedly truncate wait times for responses.

Related reference
DISPLAYAPSTATUS

Multi *Applications continually connect or disconnect*

A symptom and solution associated with applications continually connecting or disconnecting, or fail to reach the expected balance.

First, rule out common causes using the DIS APSTATUS command described in "Applications not balancing correctly" on page 198

Symptom

All application instances appear movable, but instances are continually re-balancing and failing to reach equilibrium, and/or some queue managers do not have any instances of the application.

A likely cause is an incorrectly configured client channel definition table (CCDT). Uniform clusters require that clients connect using a CCDT which includes connection information for every individual queue manager in the cluster. When an application instance is asked to reconnect to another queue manager (to rebalance connections), the IBM MQ client code consults the CCDT to establish the route to the new destination.

If some queue managers are not included in the CCDT, or there is an error in their configuration information, when an application instance attempts to reconnect it fails part way through the move (usually reconnecting back to its previous queue manager). At some later point, it will likely be asked to attempt to reconnect again, with the same result. This results in connections frequently 'bouncing', and the application as a whole never achieving an even spread across the uniform cluster.

Solution

Ensure that all members of a uniform cluster are always represented in the CCDT provided to client applications. This should include temporarily inactive members as applications are not requested to rebalance to queue managers that are not currently running. If you identify errors in the CCDT, after making any corrections redeploy the file to all clients using your existing mechanism. There is no need to restart client applications, which locate the modified entries the next time a reconnect is attempted.

If the CCDT entries appear correct, there might be a problem with the listener or SRVCONN channel definition on some members of the cluster, which causes similar behavior for the same reasons.

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
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
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
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
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.

A2001 (07D1) (RC2001): MQRC_ALIAS_BASE_Q_TYPE_ERROR
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

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 concepts

REFRESH CLUSTER considerations for publish/subscribe clusters

Clustering: Using REFRESH CLUSTER best practices

Related reference

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: IBM 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

Multi

```
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.

Multi 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 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.

Related reference

2035 (07F3) (RC2035): MQRC_NOT_AUTHORIZED

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 IBM MQ cluster and your JMS application tries to access a cluster queue which somewhere else in the cluster. Your application needs to leave the queue manager blank if it wants to open a cluster queue located in the cluster, or specify the name of a queue manager in the cluster which hosts the cluster queue.

Related reference

2085 (0825) (RC2085): 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)
```

Related reference

2189 (088D) (RC2189): MQRC_CLUSTER_RESOLUTION_ERROR

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.

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.

Related reference

2082 (0822) (RC2082): MQRC_UNKNOWN_ALIAS_BASE_Q

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 reference
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, using 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, using 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, using 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, using 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, using 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_CLUSTER_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.

z/OS 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.

Cause 3

The new transmission queue or the storage medium hosting it has filled.

Solution

Check the queue and channel status to confirm whether administrative action is required, for example:

- Start the channel to begin moving messages
- Free space on a full remote (target) queue if this is causing the channel to back up
- Increase the MAXDEPTH attribute on the transmission queue

The switching process retries continuously and completes once the problem is resolved.

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.

Linux MQ Adv. Troubleshooting RDQM configuration problems

These topics give information that is useful for troubleshooting RDQM high availability (HA) and disaster recovery (DR) configurations.

About this task

See also the topics explaining the output of the **rdqmstatus** command for help with troubleshooting (Viewing RDQM and HA group status, Viewing DR RDQM status, and Viewing DR/HA RDQM and HA group status).

Related tasks

“Collecting information for RDQM problems on Linux” on page 356
If you need assistance when a replicated data queue manager (RDQM) is reporting a problem or failing to work properly on Linux, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Linux MQ Adv. RDQM architecture

Describes the basic architecture of replicated data queue manager high availability and disaster recovery (RDQM HA and RDQM DR) configurations to assist with troubleshooting.

- “Resource names” on page 219
- “HA architecture” on page 219
- “DRBD resource” on page 219
- “DRBD HA connections” on page 219
- “DRBD keepalive” on page 221
- “Pacemaker” on page 221
- “DR architecture” on page 221
- “DRBD DR connections” on page 222

Resource names

Various resources are created for each RDQM queue manager and these resources have names based on the Directory name of the queue manager. The name can be found in the file `/var/mqm/mqs.ini`, and is referred to here as *qm*. For example, for an RDQM HA queue manager named TMPQM1, *qm* would be `tmpqm1`.

HA architecture

The architecture of RDQM high availability (HA) involves both DRBD, for data replication, and Pacemaker, for managing where HA RDQM queue managers run. See https://linbit.com/drbd-user-guide/drbd-guide-9_0-en/ for information about DRBD and https://clusterlabs.org/pacemaker/ for information about Pacemaker.

When you create an RDQM HA queue manager, the following steps are completed:

1. A DRBD resource is created to replicate the data for the queue manager.
2. A queue manager is created and configured to use the DRBD resource for its storage.
3. A set of Pacemaker resources is created to monitor and manage the queue manager.

DRBD resource

Each RDQM HA queue manager has a DRBD resource file generated for it named `/etc/drbd.d/`*qm*`.res`. For example, when an RDQM HA queue manager named HAQM1 is created, the DRBD resource file is `/etc/drbd.d/haqm1.res`.

The most important information for troubleshooting purposes in the `.res` file is the device minor number for this particular DRBD resource. Many of the messages that DRBD logs use this minor number. For the example queue manager, HAQM1, the `.res` file contains the following information:

```
device minor 100;
```

For this queue manager, you should look for messages such as the following example:

```
Jul 31 00:17:24 mqhavm13 kernel: drbd haqm1/0 drbd100 mqhavm15.gamsworthwilliam.com:
drbd_sync_handshake:
```

The presence of the string `drbd100` indicates that the message relates to HAQM1. Not all messages logged by DRBD use the device minor number, some use the DRBD resource name, which is the same as the Directory name of the RDQM HA queue manager. For example:

```
Jul 31 00:17:22 mqhavm13 kernel: drbd haqm1 mqhavm15.gamsworthwilliam.com: Connection closed
```

DRBD HA connections

In normal operation there is a pair of DRBD connections between each pair of nodes:

- A control or meta connection.
- The main data connection.

The usual pattern is that one of these connections is established by one node of a pair and the other connection is established by the other node in the pair.

For RDQM HA, the DRBD listener port for the queue manager is chosen automatically when the queue manager is created. The port numbers begin at 7000. You can find the port for a queue manager by looking for the address lines in the appropriate DRBD .res file, for example:

```
address ipv4 10.51.7.58:7000;
```

When you know the appropriate port number, you can find the TCP connections relating to that queue manager with the **ss** command, for example:

```
ss -nt '( src :7000 or dst :7000 )'
```

An example RDQM HA configuration consists of the following nodes:

- node-1, IP address 10.51.6.190
- node-2, IP address 10.51.6.210
- node-3, IP address 10.51.7.19

There is a single RDQM HA queue manager configured and running, with all DRBD connections connected. The **ss** command run on node-3 gives the following output:

```
State        Recv-Q         Send-Q                Local Address:Port                 Peer
Address:Port         Process
ESTAB        0              0                      10.51.7.19:36111
10.51.6.210:7000
ESTAB        0              0                      10.51.7.19:7000
10.51.6.210:50901
ESTAB        0              0                      10.51.7.19:7000
10.51.6.190:42379
ESTAB        0              0                      10.51.7.19:54205
10.51.6.190:7000
```

The first connection listed was established by node-3 to node-2.

The second connection listed was established by node-2 to node-3. As mentioned above, the two DRBD connections between a pair of nodes are usually established in opposite directions.

The third connection listed was established by node-1 to node-3.

The fourth connection listed was established by node-3 to node-1.

The easiest way to distinguish the control connection from the data connection for an active queue manager is by the amount of data flowed over the connection, which is usually smaller for the control Connection.

To see the number of bytes flowed over each connection, use the **ss** command with the `i` option. In the example, running the following command:

```
ss -nit '( src :7000 or dst :7000 )'
```

Produces the following output:

```
State          Recv-Q          Send-Q                    Local
Address:Port                    Peer Address:Port          Process
ESTAB          0               0
10.51.7.19:36111                    10.51.6.210:7000
cubic wscale:7,7 rto:201 rtt:0.209/0.085 ato:40 mss:1448 pmtu:1500 rcvmss:1448
advmss:1448 cwnd:10 ssthresh:27 bytes_sent:2587329 bytes_retrans:20 bytes_acked:2587310
bytes_received:3704817 segs_out:2980 segs_in:1566 data_segs_out:2046 data_segs_in:932 send
554258373bps lastsnd:229936217 lastrcv:3102096 lastack:3102101 pacing_rate 1106531336bps
delivery_rate 1672556144bps delivered:2047 busy:131ms retrans:0/1 dsack_dups:1 rcv_rtt:7283.92
rcv_space:277568 rcv_ssthresh:2662500 minrtt:0.128
ESTAB          0               0
10.51.7.19:7000                     10.51.6.210:50901
cubic wscale:7,7 rto:201 rtt:0.3/0.027 ato:40 mss:1448 pmtu:1500 rcvmss:536 advmss:1448
cwnd:10 bytes_sent:401848 bytes_acked:401848 bytes_received:388128 segs_out:23848 segs_in:46420
data_segs_out:23343 data_segs_in:23086 send 386133333bps lastsnd:9624 lastrcv:9624 lastack:9624
pacing_rate 770340808bps delivery_rate 176406088bps delivered:23344 app_limited busy:6597ms
rcv_rtt:68387.1 rcv_space:28976 rcv_ssthresh:64336 minrtt:0.097
ESTAB          0               0
10.51.7.19:7000                     10.51.6.190:42379
cubic wscale:7,7 rto:201 rtt:0.538/0.059 ato:40 mss:1448 pmtu:1500 rcvmss:536 advmss:1448
cwnd:10 bytes_sent:885176 bytes_acked:885176 bytes_received:415992 segs_out:53022 segs_in:27238
data_segs_out:29765 data_segs_in:23599 send 215315985bps lastsnd:8025 lastrcv:8025 lastack:8025
pacing_rate 430531936bps delivery_rate 132237440bps delivered:29766 app_limited busy:7935ms
rcv_rtt:299060 rcv_space:28960 rcv_ssthresh:68624 minrtt:0.116
ESTAB          0               0
10.51.7.19:54205                    10.51.6.190:7000
     cubic wscale:7,7 rto:201 rtt:0.245/0.079 ato:40 mss:1448 pmtu:1500 rcvmss:1448
```

```
advmss:1448 cwnd:10 ssthresh:28 bytes_sent:2947194 bytes_retrans:20 bytes_acked:2947175
bytes_received:3170253590 segs_out:67287 segs_in:78314 data_segs_out:3556 data_segs_in:76191
send 472816327bps lastsnd:229936217 lastrcv:229936217 lastack:2573657 pacing_rate 942267408bps
delivery_rate 875365232bps delivered:3557 app_limited busy:1187ms retrans:0/1 dsack_dups:1
rcv_rtt:17.234 rcv_space:3043672 rcv_ssthresh:3145728 minrtt:0.078
```

There are a number of values in the second line of output for each connection that you can use to decide which is the control connection and which is the data connection:

- `bytes_sent` (only shown on RHEL 8 and RHEL 9 systems)
- `bytes_acked`
- `bytes_received`

If a queue manager is idle then it is possible that the connection with the greater amount of data flowing is actually the control connection, because of the DRBD ping that is sent every 10 seconds and the response to it. If you have an idle queue manager then the most reliable way to identify the control connection is to run **tcpdump** on one of the connections using a command such as **tcpdump src port 36111**. If the connection with the specified local port is the control connection then you should see some activity every 10 seconds.

If there is no output from **tcpdump** then the connection is almost certainly the data connection. You can verify that by copying a file into the `userdata` directory of the queue manager, which will cause the data to be sent over the data connection.

DRBD keepalive

DRBD has its own keepalive implementation that is used on the control connection. The data connection relies on the standard Linux keepalive feature.

RDQM uses the following default DRBD configuration:

```
ping-int     10; # seconds, default
ping-timeout  5; # 1/10 seconds, default
```

This configuration means that DRBD sends a ping request every 10 seconds and expects a response within half a second. If a response is not received within that time, you will see a message in the syslog saying that `PingAck` did not arrive in time.

Pacemaker

There are a number of Pacemaker resources generated for an RDQM HA queue manager:

qm
: This is the main resource representing the RDQM HA queue manager.

p_rdqmx_*qm*
: This is an internal resource.

p_fs_*qm*
: This is a standard filesystem resource that mounts the volume for the queue manager onto `/var/mqm/vols/`*qm*`.`

ms_drbd_*qm*
: This is the master/slave resource for the DRBD resource for the RDQM.

p_drbd_*qm*
: This is the primitive resource for the DRBD resource for the RDQM.

If a floating IP address is configured for an HA RDQM then an additional resource is configured:

p_ip_*qm*

DR architecture

The architecture of RDQM DR is simpler as Pacemaker is not involved, only DRBD.

The architecture of RDQM DR/HA is a combination of the architecture for DR and the architecture for HA.

DRBD DR connections

The DRBD DR connections for a DR/HA queue manager are between the node where the queue manager is running and the node that is the HA preferred location in the RDQM HA Group that is currently the DR Secondary. There is only one pair of connections between the HA Groups for each queue manager. The node in the DR Secondary HA Group is responsible for synchronously replicating the updates it receives to the other two nodes in the HA Group.

The DR replication is described as asynchronous but it is not fully asynchronous. Asynchronous replication is mapped to the DRBD Protocol A. DRBD regards an update as complete as soon as the data has been copied to the local TCP sendbuffer of the connection, assuming the connection is established. If the connection is not established, the update is remembered and the update completes immediately, or in the case of DR/HA, as soon as the update is complete on the other HA nodes.

The following example DR/HA configuration comprises two HA groups, each using different IP addresses for HA and DR. The first HA group has the following nodes:

Node	HA IP Address	DR IP Address
node-a1	10.51.20.185	192.168.198.50
node-a2	10.51.20.192	192.168.198.66
node-a3	10.51.20.203	192.168.199.236

The second HA group has the following nodes:

Node	HA IP Address	DR IP Address
node-b1	10.51.7.58	192.168.208.25
node-b2	10.51.7.190	192.168.208.26
node-b3	10.51.15.65	192.168.208.28

The queue manager QM1 is running on node-a1.The **rdqmstatus -m QM1** command produces the following output:

```
Node:
node-a1
Queue manager status:                   Running
CPU:                                    0.00%
Memory:                                 181MB
Queue manager file system:              58MB used, 2.9GB allocated [2%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA blocked location:                    None
HA floating IP interface:               None
HA floating IP address:                 None
DR role:                                Primary
DR status:                              Normal
DR port:                                8001
DR local IP address:                    192.168.198.50
DR remote IP address list:              192.168.208.25,192.168.208.26,192.168.208.28
DR current remote IP address:           192.168.208.25

Node:
node-a2
HA status:                              Normal

Node:
node-a3
HA status:                              Normal
```

The value of DR `current remote IP address` is the DR IP address of the node where the DR Secondary instance is currently receiving the data from the DR Primary, in this case node-b1. Running **`rdqmstatus -m QM1`** on node-b1 results in the following output:

```
Node:
node-b1
Queue manager status:                   Ended immediately
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA blocked location:                    None
HA floating IP interface:               None
HA floating IP address:                 None
DR role:                                Secondary
DR status:                              Normal
DR port:                                8001
DR local IP address:                    192.168.208.25
DR remote IP address list:              192.168.198.50,192.168.198.66,192.168.199.236
DR current remote IP address:           192.168.198.50

Node:
node-b2
HA status:                              Normal

Node:
node-b3
HA status:                              Normal
```

The value of DR `port` is the port number used for the DR replication of this particular queue manager, as specified on the original **`crtmqm`** command.

In the example, the following connections use this port on node-a1:

```
ss -nt '( src :8001 or dst :8001 )'
State      Recv-Q Send-Q                  Local Address:Port
Peer Address:Port
ESTAB      0      0                      192.168.198.50:8001
192.168.208.25:57373
ESTAB      0      0                      192.168.198.50:59630
192.168.208.25:8001
```

As with the HA connections, the DR connections are normally established in opposite directions. Here, the first connection was established by node-b1 to node-a1 and the second connection was established by node-a1 to node-b1. One of the connections is the control or meta connection and the other is the data connection. The following example includes the information about each connection to see the bytes received:

```
ss -int '( src :8001 or dst :8001 )'
State      Recv-Q Send-Q                  Local Address:Port
Peer Address:Port
ESTAB      0      0                      9.20.198.50:8001
9.20.208.25:57373
     cubic wscale:7,7 rto:202 rtt:1.648/2.188 ato:40 mss:1448 rcvmss:592 advmss:1448 cwnd:10
bytes_acked:733 bytes_received:1181 segs_out:14 segs_in:16 send 70.3Mbps lastsnd:4841379
lastrcv:4841380 lastack:4841373 pacing_rate 140.6Mbps rcv_rtt:6 rcv_space:28960
ESTAB      0      0                      9.20.198.50:59630
9.20.208.25:8001
     cubic wscale:7,7 rto:201 rtt:0.61/0.044 ato:40 mss:1448 rcvmss:536 advmss:1448 cwnd:10
bytes_acked:7785 bytes_received:7728 segs_out:970 segs_in:487 send 189.9Mbps lastsnd:1397
lastrcv:1397 lastack:1397 pacing_rate 379.4Mbps rcv_space:29200
```

The first connection has a `bytes_received` value of 1181 and the second connection has a `bytes_received` value of 7728 which suggests that the first connection is the control connection and the second connection is the data connection, although these values are similar enough that it might be the case that the connection with the higher value is actually the control connection that has overtaken a recently-idle data connection due to the DRBD ping traffic.

Linux MQ Adv. Example RDQM HA configurations and errors

An example RDQM HA configuration, complete with example errors and information on how to resolve them.

The example RDQM HA group consists of three nodes:

- mqhavm13.gamsworthwilliam.com (referred to as vm13).
- mqhavm14.gamsworthwilliam.com (referred to as vm14).
- mqhavm15.gamsworthwilliam.com (referred to as vm15).

Three RDQM HA queue managers have been created:

- HAQM1 (created on vm13)
- HAQM2 (created on vm14)
- HAQM3 (created on vm15)

Initial conditions

The initial condition on each of the nodes is given in the following listings:

vm13

```
[midtownjojo@mqhavm13 ~]$ rdqmstatus -m HAQM1
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.00%
Memory:                                 135MB
Queue manager file system:              51MB used, 1.0GB allocated [5%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm13 ~]$ rdqmstatus -m HAQM2
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm14.gamsworthwilliam.com
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm13 ~]$ rdqmstatus -m HAQM3
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm15.gamsworthwilliam.com
HA preferred location:                  mqhavm15.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
```

```
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

vm14

```
[midtownjojo@mqhavm14 ~]$ rdqmstatus -m HAQM1
Node:                                   mqhavm14.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm13.gamsworthwilliam.com
HA preferred location:                  mqhavm13.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm14 ~]$ rdqmstatus -m HAQM2
Node:                                   mqhavm14.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.00%
Memory:                                 135MB
Queue manager file system:              51MB used, 1.0GB allocated [5%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm14 ~]$ rdqmstatus -m HAQM3
Node:                                   mqhavm14.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm15.gamsworthwilliam.com
HA preferred location:                  mqhavm15.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

vm15

```
[midtownjojo@mqhavm15 ~]$ rdqmstatus -m HAQM1
Node:                                   mqhavm15.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm13.gamsworthwilliam.com
HA preferred location:                  mqhavm13.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal
```

```
Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm15 ~]$ rdqmstatus -m HAQM2
Node:                                   mqhavm15.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm14.gamsworthwilliam.com
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.

[midtownjojo@mqhavm15 ~]$ rdqmstatus -m HAQM3
Node:                                   mqhavm15.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.02%
Memory:                                 135MB
Queue manager file system:              51MB used, 1.0GB allocated [5%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm13.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

DRBD scenarios

RDQM HA configurations use DRBD for data replication. The following scenarios illustrate the following possible problems with DRBD:

- Loss of DRBD quorum
- Loss of a single DRBD connection
- Synchronization stuck

DRBD Scenario 1: Loss of DRBD quorum

If the node running an RDQM HA queue manager loses the DRBD quorum for the DRBD resource corresponding to the queue manager, DRBD immediately starts returning errors from I/O operations, which will cause the queue manager to start producing FDCs and eventually stop.

If the remaining two nodes have a DRBD quorum for the DRBD resource then Pacemaker chooses one of the two nodes to start the queue manager. Because there were no updates on the original node from the time where the quorum was lost, it is safe to start the queue manager somewhere else.

The two main ways that you can monitor for a loss of DRBD quorum are:

- By using the **rdqmstatus** command.
- By monitoring the syslog of the node where the RDQM HA queue manager is initially running.

rdqmstatus

If you use the **rdqmstatus** command, if the node vm13 loses DRBD quorum for the DRBD resource for HAQM1, you might see status similar to the following example:

```
[midtownjojo@mqhavm13 ~]$ rdqmstatus -m HAQM1
Node:                                 mqhavm13.gamsworthwilliam.com
Queue manager status:                 Running elsewhere
HA role:                              Secondary
HA status:                            Remote unavailable
HA control:                           Enabled
HA current location:                  mqhavm14.gamsworthwilliam.com
HA preferred location:                This node
HA floating IP interface:             None
HA floating IP address:               None

Node:                                 mqhavm14.gamsworthwilliam.com
HA status:                            Remote unavailable
HA out of sync data:                  0KB

Node:                                 mqhavm15.gamsworthwilliam.com
HA status:                            Remote unavailable
HA out of sync data:                  0KB
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

Notice that the `HA status` has changed to `Remote unavailable`, which indicates that both DRBD connections to the other nodes have been lost.

In this case the other two nodes have DRBD quorum for the DRBD resource so the RDQM is running somewhere else, on mqhavm14.gamsworthwilliam.com as shown as the value of `HA current location`.

monitoring syslog

If you monitor syslog, you will see that DRBD logs a message when it loses quorum for a resource:

```
Jul 30 09:38:36 mqhavm13 kernel: drbd haqm1/0 drbd100: quorum( yes -> no )
```

When quorum is restored a similar message is logged:

```
Jul 30 10:27:32 mqhavm13 kernel: drbd haqm1/0 drbd100: quorum( no -> yes )
```

DRBD Scenario 2: Loss of a single DRBD connection

If only one of the two DRBD connections from a node running an RDQM HA queue manager is lost then the queue manager does not move.

Starting from the same initial conditions as in the first scenario, after blocking just one of the DRBD replication links, the status reported by **rdqmstatus** on vm13 is similar to the following example:

```
Node:                                 mqhavm13.gamsworthwilliam.com
Queue manager status:                 Running
CPU:                                  0.01%
Memory:                               133MB
Queue manager file system:            52MB used, 1.0GB allocated [5%]
HA role:                              Primary
HA status:                            Mixed
HA control:                           Enabled
HA current location:                  This node
HA preferred location:                This node
HA floating IP interface:             None
HA floating IP address:               None

Node:                                 mqhavm14.gamsworthwilliam.com

HA status:                            Remote unavailable
HA out of sync data:                  0KB

Node:                                 mqhavm15.gamsworthwilliam.com
HA status:                            Normal
Command '/opt/mqm/bin/rdqmstatus' run with sudo.
```

DRBD Scenario 3: Synchronization stuck

Some versions of DRBD had an issue where a synchronization would appear to be stuck and this prevented an RDQM HA queue manager from failing over to a node when the sync to that node is still in progress.

One way to see this is to use the `drbdadm status` command. When operating normally a response similar to the following example is output:

```
[midtownjojo@mqhavm13 ~]$ drbdadm status
haqm1 role:Primary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate

haqm2 role:Secondary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Primary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate

haqm3 role:Secondary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Primary
    peer-disk:UpToDate
```

If synchronization gets stuck, the response is similar to the following example:

```
[midtownjojo@mqhavm13 ~]$ drbdadm status
haqm1 role:Primary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Secondary
    replication:SyncSource peer-disk:Inconsistent done:90.91

haqm2 role:Secondary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Primary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate

haqm3 role:Secondary
  disk:UpToDate
  mqhavm14.gamsworthwilliam.com role:Secondary
    peer-disk:UpToDate
  mqhavm15.gamsworthwilliam.com role:Primary
    peer-disk:UpToDate
```

In this case the RDQM HA queue manager HAQM1 cannot move to vm15 as the disk on vm15 is `Inconsistent`.

The done value is the percentage complete. If that value is not increasing you could try disconnecting that replica then connecting it again with the following commands (run as `root`) on vm13:

```
drbdadm disconnect haqm1:mqhavm15.gamsworthwilliam.com
drbdadm connect haqm1:mqhavm15.gamsworthwilliam.com
```

If the replication to both Secondary nodes is stuck, you can do the **`disconnect`** and **`connect`** commands without specifying a node and that will disconnect both connections:

```
drbdadm disconnect haqm1
drbdadm connect haqm1
```

Pacemaker scenarios

RDQM HA configurations use Pacemaker to determine where an RDQM HA queue manager runs. The following scenarios illustrate the following possible problems that involve Pacemaker:

- Corosync main process not scheduled
- RDQM HA queue manager not running where it should

Pacemaker scenario 1: Corosync main process not scheduled

If you see a message in the syslog similar to the following example this indicates that the system is either too busy to schedule CPU time to the main Corosync process or, more commonly, that the system is a Virtual Machine and the Hypervisor has not scheduled any CPU time to the entire VM.

```
corosync[10800]:  [MAIN  ] Corosync main process was not scheduled for 3787.0891 ms (threshold
is 2400.0000 ms). Consider token timeout increase.
```

Both Pacemaker (and Corosync) and DRBD have timers that are used to detect loss of quorum, so messages like the example indicate that the node did not run for so long that it would have been dropped from the quorum. The Corosync timeout is 3000 ms and the threshold of 2400 ms is 80% of that, so the message shown in the example is printed when the delay in the scheduling of the main Corosync process hits 80% of the timeout. In the example the process was not scheduled for nearly three seconds. Whatever is causing such a problem must be resolved. One thing that might help in a similar situation is to reduce the requirements of the VM, for example, reducing the number of vCPUs required, as this makes it easier for the Hypervisor to schedule the VM.

Pacemaker scenario 2: An RDQM HA queue manager is not running where it should be

The main tool to help troubleshooting in this scenario is the **rdqmstatus** command. The following example shows a response for the configuration when everything is working as expected. The commands are run on VM13:

```
%rdqmstatus -m HAQM1

Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.00
Memory:                                 123MB
Queue manager file system:              606MB used, 1.0GB allocated [60%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  This node
HA preferred location:                  This node
HA blocked location:                    None
HA floating IP interface:               eth4
HA floating IP address:                 192.0.2.4


%rdqmstatus -m HAQM2

Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm14.gamsworthwilliam.com
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA blocked location:                    None
HA floating IP interface:               eth4
HA floating IP address:                 192.0.2.6



%rdqmstatus -m HAQM3

Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
```

```
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm15.gamsworthwilliam.com
HA preferred location:                  mqhavm15.gamsworthwilliam.com
HA blocked location:                    None
HA floating IP interface:               eth4
HA floating IP address:                 192.0.2.8
```

Note the following points:

- All three nodes are shown with an HA status of `Normal`.
- Each RDQM HA queue manager is running on the node where it was created, for example, HAQM1 is running on vm13 and so on.

This scenario is constructed by preventing HAQM1 from running on vm14, and then attempting to move HAQM1 to vm14. HAQM1 cannot run on vm14 because the file `/var/mqm/mqs.ini` on vm14 has an invalid value for the Directory of queue manager HAQM1.

The preferred location for HAQM1 is changed to vm14 by running the following command on vm13:

```
rdqmadm -m HAQM1 -n mqhavm14.gamsworthwilliam.com -p
```

This command would normally cause HAQM1 to move to vm14 but in this case checking the status on vm13 returns the following information:

```
$ rdqmstatus -m HAQM1
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.15%
Memory:                                 133MB
Queue manager file system:              52MB used, 1.0GB allocated [5%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA blocked location:                    mqhavm14.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
```

HAQM1 is still running on vm13, it has not moved to vm14 as requested and the cause needs investigating. Examining the status and including failed resource actions gives the following response:

```
$ rdqmstatus -m HAQM1 -a

Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running
CPU:                                    0.15%
Memory:                                 133MB
Queue manager file system:              52MB used, 1.0GB allocated [5%]
HA role:                                Primary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    This node
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA blocked location:                    mqhavm14.gamsworthwilliam.com
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal

Failed resource action:                 Start
Resource type:                          Queue manager
Failure node:                           mqhavm14.gamsworthwilliam.com
```

```
Failure time:                           2022-01-01 12:00:00
Failure reason:                         Generic error
Blocked location:                       mqhavm14.gamsworthwilliam.com
```

Take note of the `Failed resource action` section that has appeared.

The entry shows that when Pacemaker tried to check the state of HAQM1 on vm14 it got an error because HAQM1 is not configured, which is because of the deliberate misconfiguration in `/var/mqm/mqs.ini`.

Correcting the failure

To correct the failure you must correct the underlying problem (in this case restoring the correct directory value for HAQM1 in `/var/mqm/mqs.ini` on vm14). Then you must clear the failed action by using the command **rdqmclean** on the appropriate resource, which in this case is the resource haqm1 as that is the resource mentioned in the failed action. For example:

```
$ rdqmclean -m HAQM1
```

Then check the failed resource action status again:

```
$ rdqmstatus -m HAQM1 -a
```

The failed action has disappeared and HAQM1 is now running on vm14 as expected. The following example shows the RDQM status:

```
$ rdqmstatus -m HAQM1
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm14.gamsworthwilliam.com
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA blocked location:                    None
HA floating IP interface:               None
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
```

Correcting the failure

To correct the failure you must correct the underlying problem (in this case restoring the correct directory value for HAQM1 in `/var/mqm/mqs.ini` on vm14). Then you must clear the failed action by using the command **rdqmclean** on the appropriate resource, which in this case is the resource haqm1 as that is the resource mentioned in the failed action. For example:

```
$ rdqmclean -m HAQM1
```

Then check the failed resource action status again:

```
$ rdqmstatus -m HAQM1 -a
```

The failed action has disappeared and HAQM1 is now running on vm14 as expected. The following example shows the RDQM status:

```
$ rdqmstatus -m HAQM1
Node:                                   mqhavm13.gamsworthwilliam.com
Queue manager status:                   Running elsewhere
HA role:                                Secondary
HA status:                              Normal
HA control:                             Enabled
HA current location:                    mqhavm14.gamsworthwilliam.com
HA preferred location:                  mqhavm14.gamsworthwilliam.com
HA blocked location:                    None
HA floating IP interface:               None
```

```
HA floating IP address:                 None

Node:                                   mqhavm14.gamsworthwilliam.com
HA status:                              Normal

Node:                                   mqhavm15.gamsworthwilliam.com
HA status:                              Normal
```

Pacemaker scenario 3: An RDQM HA node is offline

V 10.0.0

If one of the three nodes unexpectedly returns an offline status when you use the **rdqmstatus -n** command, and you have enabled heartbeat encryption in your group, then it might be that you have a mismatch of the `authkey` files on the nodes. Before you enable heartbeat encryption, you must create a shared secret and distribute it to all three nodes (see ../configure/config_encrypt_heartbeat.dita). Check that all three nodes have the same `authkey` file, and remedy the situation if necessary.

Linux MQ Adv. Problems after upgrading RDQM

If you encounter problems after upgrading one or more nodes in your RDQM configuration, you should check that your installed DRBD kernel matches the OS kernel version.

RDQM configurations rely upon the DRBD module to provide data replication. When you upgrade RDQM, it is important to ensure that the correct version of the DRBD kernel module is installed for the version of RHEL kernel running on the system. If the versions do not match, you might experience problems of varying severity. It might be that queue manager will not start, or that it does not run on the upgraded node, even if that node is the preferred location.

You can use the **rdqmstatus** command to view information about installed versions, see Viewing RDQM and HA group status, Viewing DR RDQM status, and Viewing DR/HA RDQM and HA group status.

For guidance on selecting the correct DRBD module to install, see Installing RDQM (replicated data queue managers).

The following examples illustrate some potential mismatches and their effects.

Example 1

```
Node:                                   mqhavm57.exampleco.com
OS kernel version:                      5.14.0-362.18.1
DRBD OS kernel version:                 5.14.0-70.13.1
DRBD version:                           9.2.7+ptf.14
DRBD kernel module status:              Partially loaded

Queue manager name:                     QM1
Queue manager status:                   Running elsewhere
HA status:                              Unknown
HA current location:                    mqhavm58.exampleco.com
HA preferred location:                  This node
HA blocked location:                    None
DR role:                                Primary
DR status:                              Unknown

Queue manager name:                     QM2
Queue manager status:                   Running elsewhere
HA status:                              Unknown
HA current location:                    mqhavm58.exampleco.com
HA preferred location:                  This node
HA blocked location:                    None

Queue manager name:                     QM3
Queue manager status:                   Status not available
DR role:                                Secondary
DR status:                              Unknown
```

This summary status shows that the DRBD kernel module for RHEL 9.0 has been installed, while the OS kernel version is for RHEL 9.3. As a result of this mismatch, the DRBD module is only partially loaded. QM1 is an HA/DR queue manager and moves to another node, its HA status is unknown and its DR status

is unknown. QM2 is an HA queue manager, it fails to start on its preferred node and its HA status in Unknown. QM3 is a DR queue manager and is meant to be the Primary instance, but since the DRBD kernel module did not fully load, it reports as Secondary with a DR status of Unknown. To resolve these failures the DRBD kernel module must be updated with the version target for the running OS kernel.

Example 2

```
Node:                                   mqhavm07.exampleco.com
OS kernel version:                      5.14.0-362.18.1
DRBD OS kernel version:                 5.14.0-284.11.1
DRBD version:                           9.2.7+ptf.14
DRBD kernel module status:              Loaded

Queue manager name:                     RDQM7
Queue manager status:                   Running
HA current location:                    This node
HA preferred location:                  This node
HA blocked location:                    None
```

This summary status shows that the DRBD kernel module for RHEL 9.2 has been installed, while the OS kernel version is for RHEL 9.3. This is not such a serious mismatch, the DRBD module is loaded, and queue manager RDQM7 is running on this node. It is, however, recommended that you install the correct DRBD module for your RHEL installation to ensure correct operation.

Troubleshooting Native HA and Native HA CRR configurations on Linux

Information about troubleshooting Native HA and Native HA CRR configurations on Linux.

Error logs

Consult the error logs located in `/var/mqm/qmgrs/<QMname>/errors`. A Native HA queue manager consists of multiple instances, the messages that might be helpful in resolving an issue could be spread across multiple instances. Errors and warnings related to Native HA operation are in the range AMQ3201 through AMQ3399.

Table 4. Error messages

Message IDs	Action
AMQ3201 AMQ3206 AMQ3262 AMQ3309	Check spelling of stanzas and attributes in qm.ini, check for duplication of stanzas.
AMQ3209 AMQ3210	Check all instances were created using the same log extent parameters, check that each instance's qm.ini file provides a consistent configuration. (Messages contain more detail.)
AMQ3216	Check network replication port number is not already in use on this node, check that the correct address is used.
AMQ3228	Check TLS configuration, check that the instance has read access to keystores and so on.
AMQ3231 AMQ3236	CipherSpec or protocol mismatch. Check for consistent configuration and consider using “ANY_” to negotiate strongest common protocol and algorithm.
AMQ3232	Bad TLS certificate. Check expiry, revocation status, and CA trust chain.

Table 4. Error messages (continued)	
Message IDs	**Action**
AMQ3233	Unknown instance. Check spelling of instance names.
AMQ3246	Failed self-check. Check the local instance's replication address is resolvable, and correct it if necessary.
AMQ3251	Live instance failed to connect to recovery group. Check the possible locations of the recovery group leader are correct, and check that there are enough replicas started in the recovery group to elect a leader.
AMQ3256 AMQ3257	Configuration mismatch between groups. Check detail of message (similar to AMQ3209 and AMQ3210).
AMQ3267 AMQ3273	Inconsistent group roles within the group. Correct mismatch between requested group roles in qm.ini.
AMQ3274 AMQ3275 AMQ3289 AMQ3296	Insufficient log filesystem space. Increase available log filesystem space.
AMQ3283 AMQ3295	Check group replication port number is not already in use on this machine.
AMQ3285	Duplicate group name. Check each group has a unique group name.
AMQ3297	Recovery group VRM must be equal to or greater than the VRM of the Live group. Update the Recovery group to the same or higher version than the Live group.
AMQ3298	VRM must be the same in a planned role swap
AMQ3303	VRM of an instance within a group does not match, this warning can be ignored if rolling out a new maintenance level to each instance one at a time.

System clock synchronization

It is best practice is to ensure that the system clocks on each instance are synchronized with each other as much as possible, and updated via NTP. Ensuring that each instance in a group is configured in the same timezone, and is synchronized within at least a few seconds of the other system clocks, helps when cross referencing messages in the logs.

DISPLAY QMSTATUS and dspmq

The information displayed by using **dspmq** with the **-o nativeha** option represents status at a point in time. For a queue manager that is actively replicating data between instances and between groups, the **dspmq** values change on a frequent basis. While **dspmq** can be issued against any instance, whether it is inactive or running as a replica, the best data is usually available from the elected active instance in a live group or the elected group leader in a recovery group.

QUORUM and BACKLOG

The QUORUM field is available if there is an elected active instance (it is displayed when you use **dspmq** or DISPLAY QMSTATUS via PCF, MQSC, and REST). The QUORUM field gives two numbers that provide a simple traffic light indicator on the health of the queue manager. The first number is the number of instances within the group that are in-sync and able to provide quorum on log updates, the second number is the number of instances in the group.

A QUORUM(3/3) value is the green light that you would expect to see when every instance in the group is connected to each other and a new log record is able to be written to at least two replicas and be acknowledged.

A QUORUM(2/3) value is an amber light, indicating that there are still a sufficient number of replicas able to acknowledge a new log record, however there is no redundancy and if one of the instances were to end or lose connectivity, the queue manager would no longer be able to provide messaging.

QUORUM(2/3) can also indicate that all three instances are connected, but that one of the instances is currently in a catch up process, where older log data is being sent. Until the instance has caught up, it is not possible for it to provide the positive acknowledgements required to form quorum. The amount of log data that needs to be replicated to catch up is reported in Kilobytes in the BACKLOG field.

Note that, in a rolling update across multiple instances, it is normal for there to be a small amount of BACKLOG following an instance stopping to apply maintenance, however, if a non-zero backlog is observed over a long period of time after the restart of the instance, this could indicate a performance issue. Querying the BACKLOG over time reveals the rate at which catch up is being achieved, or if there is a performance issue this value might not drop or might even grow.

V 10.0.0 If a performance problem is suspected, use the IBM MQ performance tool **mqperfck** or the **amqsrua** command to examine detailed statistics for the replica's network round trip times, and log write latency. See mqperfck and Monitoring system resource usage by using the amqsrua command.

During the catch up process, some throttling might be applied to the rate at which new log records are acknowledged. This provides an opportunity for the out-of-date replica to catch up at the cost of slowing the rate that new work is accepted.

A QUORUM(1/3) value is a red light and indicates that a group lacks the ability to progress new log writes.

When there are three instances in a newly-created recovery group available, an election takes place to choose a nominal group leader. The recovery group leader opens its group replication port and must wait for a live group to contact its group replication port and rebase the recovery group, during this wait the QUORUM(1/3) is displayed.

Network connectivity

A queue manager instance with a replicated log is reliant on being able to accept and establish network connections to its peer members of the group. Regardless of the group role, each instance in a group attempts to establish a connection to every other instance in the same group, including itself.

The self-connection check is not used for replication of any log data, but it provides validation of the `NativeHAInstance` configuration in the qm.ini file which is expected to be the same for each member of the group. A successful self-connection check establishes that the network address and port are reachable and, if TLS is configured, that the certificates and CipherSpecs are compatible combinations.

If a self-check fails, an AMQ3246 error message is reported to the error logs and the queue manager instance fails to start. The preceding messages are helpful in understanding the configuration issue, for example if the port is already in use. Network firewall rules are another potential reason that can prevent connectivity between instances and groups, check firewall logs to ensure traffic is not being blocked.

Standard network diagnostics like ping and traceroute can be used to check connectivity between instances and groups.

Troubleshooting security problems

Troubleshooting information to help you solve problems relating to security.

Related tasks

“Collecting information for security problems on Multiplatforms” on page 357
If an IBM MQ is incorrectly allowing or denying access to a user or application on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

“Collecting information for TLS channel problems on Multiplatforms” on page 359
If an IBM MQ queue manager or client application is failing to establish a secure channel using TLS on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Troubleshooting channel authentication record problems

If you are having problems with 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 that is 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, that is, `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 might be operable initially, the effect of the command will not persist over a queue manager restart. You should investigate, ensuring that 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

V 10.0.0 From IBM MQ 10.0.0, the maximum size of the **ADDRESS** field is 264 characters.

Some IPv6 address patterns could be longer than the 48 character limit, for example `'0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff:0000-ffff'`. If the length is limited to 48 characters, you could use `'*'` instead.

Or, 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'`

On z/OS, the maximum size of the **ADDRLIST** field is 48 characters.

Related information

SET CHLAUTH (create or modify a channel authentication record)

CipherSpec mismatches

Both ends of an IBM MQ TLS channel must use compatible CipherSpecs. Mismatches can be detected during the TLS handshake or during channel startup.

A CipherSpec identifies the combination of the encryption algorithm and hash function. Both ends of an IBM MQ TLS channel must use compatible CipherSpecs, although they can specify those CipherSpecs in a different manner. Mismatches can be detected at two stages:

During the TLS handshake

The TLS handshake fails when the CipherSpec specified by the TLS client is unacceptable to the TLS support at the TLS server end of the connection. A CipherSpec failure during the TLS handshake arises when the TLS client proposes a CipherSpec that is not supported by the TLS provision on the TLS server. For example, when a TLS client running on AIX proposes the DES_SHA_EXPORT1024 CipherSpec to a TLS server running on IBM i.

During channel startup

Channel startup fails when there is a mismatch between the CipherSpec defined for the responding end of the channel and the CipherSpec defined for the calling end of channel. Channel startup also fails when only one end of the channel defines a CipherSpec.

See Specifying CipherSpecs for more information.

Note: If Global Server Certificates are used, a mismatch can be detected during channel startup even if the CipherSpecs specified on both channel definitions match.

Global Server Certificates are a special type of certificate which require that a minimum level of encryption is established on all the communications links with which they are used. If the CipherSpec requested by the IBM MQ channel configuration does not meet this requirement, the CipherSpec is renegotiated during the TLS handshake. This is detected as a failure during IBM MQ channel startup as the CipherSpec no longer matches the one specified on the channel.

In this case, change the CipherSpec at both sides of the channel to one which meets the requirements of the Global Server Certificate. To establish whether a certificate that has been issued to you is a Global Server Certificate, contact the certificate authority which issued that certificate.

TLS servers do not detect mismatches when an TLS client channel on AIX, Linux, and Windows systems specifies the DES_SHA_EXPORT1024 CipherSpec, and the corresponding TLS server channel on AIX, Linux, and Windows systems is using the DES_SHA_EXPORT CipherSpec. In this case, the channel runs normally.

Authentication failures during TLS handshake

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

These reasons include, but are not limited to, those in the following list:

A certificate has been found in a Certificate Revocation List or Authority Revocation List
You can check certificates against the revocation lists published by the Certificate Authorities.

A Certificate Authority can revoke a certificate that is no longer trusted by publishing it in a Certificate Revocation List (CRL) or Authority Revocation List (ARL). For more information, see Working with revoked certificates.

An OCSP responder has identified a certificate as Revoked or Unknown
You can check certificates using OCSP. An OCSP responder can return a response of Revoked, indicating that a certificate is no longer valid, or Unknown, indicating that it has no revocation data for that certificate. For more information, see Working with revoked certificates.

A certificate has expired or is not yet active
Each digital certificate has a date from which it is valid and a date after which it is no longer valid, so an attempt to authenticate with a certificate that is outside its lifetime fails.

A certificate is corrupted
If the information in a digital certificate is incomplete or damaged, authentication fails.

A certificate is not supported
If the certificate is in a format that is not supported, authentication fails, even if the certificate is still within its lifetime.

The TLS client does not have a certificate
The TLS server always validates the client certificate if one is sent. If the TLS client does not send a certificate, authentication fails if the end of the channel acting as the TLS server is defined:

- With the SSLCAUTH parameter set to REQUIRED or
- With an SSLPEER parameter value

There is no matching CA root certificate or the certificate chain is incomplete
Each digital certificate is issued by a Certificate Authority (CA), which also provides a root certificate that contains the public key for the CA. Root certificates are signed by the issuing CA itself. If the key repository on the computer that is performing the authentication does not contain a valid root certificate for the CA that issued the incoming user certificate, authentication fails.

Authentication often involves a chain of trusted certificates. The digital signature on a user certificate is verified with the public key from the certificate for the issuing CA. If that CA certificate is a root certificate, the verification process is complete. If that CA certificate was issued by an intermediate CA, the digital signature on the intermediate CA certificate must itself be verified. This process continues along a chain of CA certificates until a root certificate is reached. In such cases, all certificates in the chain must be verified correctly. If the key repository on the computer that is performing the authentication does not contain a valid root certificate for the CA that issued the incoming root certificate, authentication fails.

However, certain TLS implementations such as IBM Global Security Kit (GSKit), DCM, and RACF validate the certificates as long as the trust anchor (ROOT CA) is present, with some of the intermediate CA not present in the trust chain. Therefore, it is important to ensure that the server-side certificate store contains the complete trust chain. Also, the technique of selectively removing signer (CA) certificates must not be used to control connectivity to the queue manager.

For more information, see How certificate chains work.

For more information about the terms used in this topic, see:

- Transport Layer Security (TLS) concepts
- Digital certificates

Linux AIX Troubleshooting authentication token problems

A list of common authentication token problems, and information to help you resolve them.

The list of problems is divided according to the user who is most likely to benefit from the troubleshooting advice.

Advice for the administrator

Is the queue manager configured to accept authentication tokens?
The queue manager must be configured to accept authentication tokens. For more information, see Configuring a queue manager to accept authentication tokens using a JWKS endpoint.

Is the queue manager at a version that supports authentication tokens?
You can use authentication tokens with queue managers that run version IBM MQ 9.3.4 or later on AIX or Linux platforms.

Has the security configuration been refreshed after changing the token authentication configuration?
Changes to the token authentication configuration take effect when the queue manager's security configuration is refreshed, or when the queue manager is restarted. To refresh the queue manager's security configuration, issue the following MQSC command:

```
REFRESH SECURITY TYPE(CONNAUTH)
```

Are there messages in the queue manager error log?
If an authentication token is rejected, check for messages in the queue manager error log. The following messages might be issued when an authentication token is rejected:

- AMQ5783E
- AMQ5784E
- AMQ5785I
- AMQ5786E
- AMQ5788E

These messages might contain an error code that describes the cause of the error. For more information, see Token authentication error codes.

The following messages might be issued if the token authentication configuration in the **AuthToken** stanza of the `qm.ini` file is invalid:

- AMQ7006
- AMQ7076

Is the token issuer's certificate or symmetric key in the key repository?
The token issuer's public key certificates or symmetric keys must be imported into the token authentication key repository for the queue manager to validate the token signature. Public key certificates must not be expired.

Is there a problem accessing the token authentication key repository?
Check that the **KeyStore** attribute of the **AuthToken** stanza of the `qm.ini` file references the correct key repository. The key repository must be either a CMS key repository with the file extension `.kdb` or a PKCS#12 key repository with the file extension `.p12`.

For more information, see Configuring a queue manager to accept authentication tokens using a JWKS endpoint.

Is there a problem with the key repository password?
If you encounter problems with the password to access the token authentication key repository, check the following items:

- If using a local keystore, the encrypted key repository password is stored in a file that is referenced in the **AuthToken** stanza of the `qm.ini` file, or in a stash file.

  - If the **KeyStorePwdFile** attribute of the **AuthToken** stanza of the `qm.ini` file is specified, it must reference the correct path to the file that contains the encrypted key repository password. The key repository password must be encrypted by using the **runqmcred** command. The key repository password file must contain only the encrypted password as a single line of text. If an initial key is specified when the password is encrypted, the same initial key must be specified in the queue manager **INITKEY** attribute. For more information, see runqmcred (protect authentication token keystore password).
  - If a key repository password file is not specified by using the **KeyStorePwdFile** attribute of the **AuthToken** stanza, a key repository stash file must be present. The stash file has the same name as the key repository, with a `.sth` file name extension. The stash file is created by the **runmqakm** command.
- The mqm user has read access to the key repository files, the file that contains the key repository password, and the directory that contains the key repository.

Is the user ID in the token not being adopted?

To adopt the user ID in the token user claim for subsequent authorization checks in IBM MQ, you must configure the following:

- Set the **ADOPTCTX** attribute of the authentication information object that is referenced by the queue manager's **CONNAUTH** attribute to YES.
- Set the **UserClaim** attribute of the **AuthToken** stanza of the `qm.ini` file to the name of the claim in the token that contains the user ID to be adopted.

For more information, see the AuthToken stanza or the JWKS stanzaof the `qm.ini`file.

Is the SecurityPolicy attribute of the qm.ini Service stanza set to group?

The **SecurityPolicy** attribute of the **Service** stanza in the `qm.ini` file for the queue manager must either be set or allowed to default to `UserExternal`. If the **SecurityPolicy** attribute is explicitly set to `Group`, the authentication token configuration is rejected. For more information, see SecurityPolicy.

Advice for the developer

Is the application receiving an error code when it connects to the queue manager?

If an error occurs when the application attempts to connect to a queue manager with an authentication token, a reason code that describes the error is returned to the application. For more information, see the descriptions for the following reason codes:

- 2035 MQRC_NOT_AUTHORIZED
- 2063 MQRC_SECURITY_ERROR
- 2064 MQRC_TOKEN_TIMESTAMP_NOT_VALID
- 2595 MQRC_CSP_ERROR
- 2298 MQRC_FUNCTION_NOT_SUPPORTED

V 10.0.0 **Have any of the following JMS exception messages been returned to the application?**

- JMSCC0005
- JMSCMQ1068
- JMSCMQ1069
- JMSCMQ1070
- JMSCMQ1071
- JMSCMQ1072
- JMSCMQ1073
- JMSCMQ1074
- JMSCMQ1075
- JMSCMQ1076

- JMSFMQ1006
- JMSFMQ6319
- JMSFMQ6320
- JMSFMQ6321
- JMSFMQ6322
- V 10.0.0 JMSFMQ6326

For more information about these messages, see JMS exception messages.

Do the client and the queue manager support authentication tokens?
You can use authentication tokens from IBM MQ 9.3.4. The queue manager must run on AIX or Linux platforms. Applications must be IBM MQ MQI clients that connect in client, not bindings mode.

Are you using authentication tokens in a Java application?
You can use authentication tokens with your Java applications, but only with a security exit that uses the MQCSP structure that contains token information. For more information, see Using authentication tokens in an application and Java class MQCSP.

Make sure that your application has `INQ` authority on the IBM MQ object that it accesses. For more information, see Objects used by JMS that require authorization for non-privileged users.

Are you using the MQCSP structure to pass the authentication token to the queue manager?
You can use authentication tokens with IBM MQ MQI clients that use the MQCSP structure on the MQCONNX call. For more information, see Using authentication tokens in an application.

If any of the fields in the MQCSP structure that is provided by the application contain an invalid value, reason code MQRC_CSP_ERROR is returned to the application. Message AMQ8960E might also be written to the client error log to provide more information about the cause of the error.

For applications that use IBM MQ classes for JMS/Jakarta Messaging, equivalent messages from the `com.ibm.mq.exits.MQCSP` class are written to the IBM MQ classes for JMS log. For more information, see "Error logs in IBM MQ classes for JMS" on page 413.

Is the structure and contents of the authentication token correct?
An authentication token is a structure with three base64url encoded elements. The elements are the header, payload, and the signature. Each element is separated by a period (.). When decoded, the header and payload must be valid JSON objects. The header and the payload must contain the required header parameters and claims. For more information, see Requirements for authentication tokens.

Does the authentication token contain a valid user claim?
The authentication token can contain a user claim that specifies the user ID that is to be adopted for authorization checks to access IBM MQ objects. The **`UserClaim`** attribute of the **`AuthToken`** stanza of the `qm.ini` file must specify the name of the token claim that contains the user ID to be adopted. For example, if your token has the user claim `"AppUser": "MyUserName"`, then you must specify `UserClaim=AppUser` in the **`AuthToken`** stanza of the `qm.ini` file. For more information, see Requirements for authentication tokens.

Is the authentication token not yet valid or expired?
The authentication token must have a valid expiry (**exp**) claim. The value of the claim must be after the time when the token is presented to the queue manager. If the token has a not before (**nbf**) claim, the time that the token is valid from must be before the time that the token is presented to the queue manager. For more information, see Requirements for authentication tokens.

Is a valid authentication token being rejected?
If the token meets all the requirements for authentication tokens, and the queue manager is configured correctly, check for error messages in the queue manager's error log. The error messages might contain an error code that describes the cause of the error. For more information, see Token authentication error codes.

Is the application attempting to use client reconnection?
You cannot use tokens to connect to IBM MQ if the reconnect option is specified. If an application supplies an authentication token, and specifies the `MQCNO_RECONNECT` or

MQCNO_RECONNECT_Q_MGR option in the MQCNO structure, the connection fails and reason code 2547 MQRC_RECONNECT_INCOMPATIBLE is returned to the application.

From IBM MQ 9.4.3, JMS applications can configure connection factory properties such that the IBM MQ JMS library retrieves a token on the application's behalf from a token provider URL. This also applies to reconnectable clients, so MQRC_RECONNECT_INCOMPATIBLE is not thrown if **WMQ_CLIENT_RECONNECT_OPTIONS** is set to WMQ_CLIENT_RECONNECT or WMQ_CLIENT_RECONNECT_Q_MGR.

Related concepts

Working with authentication tokens

Requirements for authentication tokens

Related tasks

Configuring a queue manager to accept **AuthTokens**

Using authentication tokens in an application

Related reference

AuthToken stanza of the qm.ini file

MQCSP - Security parameters

Troubleshooting TLS problems

Use the information listed here to help you solve problems with your TLS 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 compatible CipherSuite or CipherSpec pairs.

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 TLS 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.

Troubleshooting WCF custom channel for IBM MQ problems

Troubleshooting information to help you solve problems with running Microsoft Windows Communication Foundation (WCF) custom channel for IBM MQ applications.

Related concepts

"FFST: WCF XMS First Failure Support Technology" on page 425

You can collect detailed information about what various parts of the IBM MQ code is doing by using IBM MQ trace. XMS FFST has its own configuration and output files for the WCF custom channel.

Related tasks

"Tracing the WCF custom channel for IBM MQ" on page 514

You can use IBM MQ trace to collect detailed information about what various parts of the IBM MQ code is doing. When using Windows Communication Foundation (WCF), a separate trace output is generated for the Microsoft Windows Communication Foundation (WCF) custom channel trace integrated with the Microsoft WCF infrastructure trace.

"Collecting troubleshooting information" on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Developing Microsoft Windows Communication Foundation applications with IBM MQ

WCF custom channel exception hierarchy

The exceptions types thrown by the custom channel are consistent with WCF and are typically a TimeoutException or CommunicationException (or a subclass of CommunicationException). Further details of the error condition, where available, are provided using linked or inner exceptions.

SOAP/JMS interface

The following exceptions are typical examples, and each layer in the architecture of the channel contributes an additional linked exception, for example CommunicationsException has a linked XMSException, which has a linked MQException:

1. System.serviceModel.CommunicationsExceptions
2. IBM.XMS.XMSException
3. IBM.WMQ.MQException

Key information is captured and provided in the data collection of the highest CommunicationException in the hierarchy. This capture and provision of data prevents the need for the applications to link to each layer in the architecture of the channel in order to interrogate the linked exceptions, and any additional information they might contain. The following key names are defined:

- IBM.XMS.WCF.ErrorCode: The error message code of the current custom channel exception.
- IBM.XMS.ErrorCode: The error message of the first XMS exception in the stack.
- IBM.WMQ.ReasonCode: The underlying IBM MQ reason code.
- IBM.WMQ.CompletionCode: The underlying IBM MQ completion code.

Non-SOAP/Non-JMS interface

The following exceptions are typical examples, and each layer in the architecture of the channel contributes an additional linked exception, for example CommunicationsException has a linked MQException:

1. System.ServiceModel.CommunicationsExceptions
2. IBM.WMQ.MQException

Key information is captured and provided in the data collection of the highest CommunicationException in the hierarchy. This capture and provision of data prevents the need for the applications to link to each layer in the architecture of the channel in order to interrogate the linked exceptions, and any additional information they might contain. The following key names are defined:

- IBM.WMQ.WCF.ErrorCode: The error message code of the current custom channel exception.
- IBM.WMQ.ReasonCode: The underlying IBM MQ reason code.
- IBM.WMQ.CompletionCode: The underlying IBM MQ completion code.

WCF version information

WCF version information aids with problem determination and is included in the assembly metadata of the custom channel.

The IBM MQ custom channel for WCF version metadata can be retrieved in one of three ways:

- Using the IBM MQ utility **dspmqver**. For information about how to use dspmqver, see: dspmqver
- Using the Windows Explorer properties dialog: In the Windows Explorer, right-click **IBM.XMS.WCF.dll** > **Properties** > **Version**.
- From the header information of any of the channels FFST or trace files. For more information about the FFST header information, see: “FFST: WCF XMS First Failure Support Technology” on page 425

WCF hints and tips

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 might save you time if they are relevant to the work that you are doing.

Externalizing exceptions from the WCF service host

For services hosted using the WCF service host; any unhandled exceptions thrown by the service, WCF internals, or channel stack are not externalized by default. To be informed of these exceptions, an error handler must be registered.

The following code provides an example of defining the error handler service behavior which can be applied as an attribute of a service:

```
using System.ServiceModel.Dispatcher;
using System.Collections.ObjectModel;
....
    public class ErrorHandlerBehaviorAttribute : Attribute, IServiceBehavior, IErrorHandler
    {
        //
```

```
        // IServiceBehavior Interface
        //
        public void AddBindingParameters(ServiceDescription serviceDescription,
           ServiceHostBase serviceHostBase, CollectionServiceEndpoint endpoints,
           BindingParameterCollection bindingParameters)
        {
        }
        public void ApplyDispatchBehavior(ServiceDescription serviceDescription,
            ServiceHostBase serviceHostBase)
        {
            foreach (ChannelDispatcher channelDispatcher in serviceHostBase.ChannelDispatchers)
{
                channelDispatcher.ErrorHandlers.Add(this);
            }
        }
        public void Validate(ServiceDescription serviceDescription, ServiceHostBase
serviceHostBase)
        {
        }

        //
        // IErrorHandler Interface
        //
        public bool HandleError(Exception e)
        {
            // Process the exception in the required way, in this case just outputting to the
console
            Console.Out.WriteLine(e);

            // Always return false to allow any other error handlers to run
            return false;
        }
        public void ProvideFault(Exception error, MessageVersion version, ref Message fault)
        {
        }
    }
```

Troubleshooting XMS .NET problems

Troubleshooting information to help you solve a problem with running XMS .NET applications.

Related tasks

“Tracing XMS .NET applications” on page 515

If you are using IBM MQ classes for XMS .NET Framework, you can configure trace from an application configuration file as well as from the XMS environment variables. If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. You can select the components that you want to trace. Trace is normally used under the guidance of IBM Support.

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Troubleshooting IBM MQ .NET problems” on page 98

Troubleshooting information to help you solve a problem with running IBM MQ .NET applications.

Developing XMS .NET applications

Troubleshooting tips for XMS .NET problems

Use these tips to help you troubleshoot problems with using XMS .NET.

An XMS application cannot connect to a queue manager (MQRC_NOT_AUTHORIZED)

The XMS .NET client may have different behavior from the behavior of the IBM MQ JMS client. Therefore, you may find that your XMS application cannot connect to your queue manager, although your JMS application can.

- A simple solution to this problem is to try using a user ID that is no more than 12 characters long and is authorized completely in the queue manager's authority list. If this solution is not ideal, a different but

more complex approach would be to use security exits. If you need further help on this issue, contact IBM Support for assistance.

- If you set the XMSC_USERID property of the connection factory, it must match the user ID and password of the logged on user. If you do not set this property, the queue manager uses the user ID of the logged on user by default.
- User authentication for IBM MQ is performed by using the details of the user currently logged on and not the information provided in the XMSC.USERID and XMSC.PASSWORD fields. This is designed to maintain consistency with IBM MQ. For more information on authentication, see Connection authentication.

Connection redirected to the messaging engine

When you connect to a WebSphere Application Server 6.0.2 service integration bus, all connections may be redirected from the original provider endpoint to the messaging engine that the bus chooses for that client connection. When doing so, it will always redirect the connection to a host server specified by the host name, rather than by an IP address. Therefore, you may experience connection problems if the host name cannot be resolved.

To successfully connect to WebSphere Application Server 6.0.2 service integration bus, you may need to provide a mapping between the host names and IP addresses on your client host machine. For example you can specify the mapping in a local hosts table on your client host machine.

Support for telnet-like password authentication

The XMS .NET Real Time Transport protocol supports only simple telnet-like password authentication. The XMS .NET Real Time Transport protocol does not support Quality Of Protection.

Setting values for property type double

On a Windows 64-bit platform, the SetDoubleProperty() or GetDoubleProperty() methods may not work correctly when setting or getting values for the property type double, if the values are smaller than Double.Epsilon.

For example, if you try to set a value of 4.9E-324 for a property with type double, the Windows 64-bit platforms treat it as 0.0. So, in a distributed messaging environment, if a JMS or another application sets the value for a double property as 4.9E-324 on any AIX, Linux, and Windows, Linux, or Windows 32-bit machine, and XMS .NET runs on a 64-bit machine, the value returned by GetDoubleProperty() is 0.0. This is a known issue with Microsoft .NET Framework 2.0.

Related tasks

Developing XMS .NET applications

Common SSL error codes thrown by XMS .NET client libraries

From IBM MQ 9.4.0, improvements to the XMS .NET client (`amqmxmsstd.dll`) diagnostic messages provide a more specific exception mechanism for SSL-related issues.

From IBM MQ 9.4.0, the XMS .NET client libraries provide meaningful and specific errors for SSL-related errors. The MQRC reason codes are in line with the other .NET client libraries, such as C.

The following table shows some of the many scenarios where, from IBM MQ 9.4.0, XMS .NET client libraries throw errors that are aligned with other client .NET libraries:

Table 5. Comparison of error messages		
Scenario	**Error message before IBM MQ 9.4.0**	**Error message from IBM MQ 9.4.0**
When the parameter provided for the SSL key repository, MQC.MQCA_SSL_KEY_REPOSITORY or MQEnvironment.SSLKeyRepository, is given incorrectly.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2381 – MQRC_KEY_REPOSITORY_ERROR
When an invalid cipher suite is set on the client while connecting to the IBM MQ server using SSL.	2538 – MQRC_HOST_NOT_AVAILABLE	2393 – MQRC_KEY_REPOSITORY_ERROR
When an invalid peer name is set on the client while connecting to the IBM MQ server using SSL.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2399 – MQRC_SSL_PEER_NAME_ERROR
When peer names do not match while connecting to the IBM MQ server using SSL.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2398 – MQRC_SSL_PEER_NAME_ERROR
When an invalid certificate is set on either the client or server during the SSL handshake.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_KEY_REPOSITORY_ERROR
When the channel that is used to provide an SSL connectivity between the client and server does not provide a cipher suite.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_KEY_REPOSITORY_ERROR
When there is a mismatch between the cipher suite algorithms set from the client and the IBM MQ server.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_KEY_REPOSITORY_ERROR
When there is no permission to the folder where IBM MQ key repository is created.	2059 – MQRC_Q_MGR_NOT_AVAILABLE	2393 – MQRC_KEY_REPOSITORY_ERROR

For a more detailed explanation of the errors, refer to the queue manager logs found in *MQ_DATA_DIRECTORY*`/qmgrs/errors/AMQERR*.log`.

Related concepts

Installing IBM MQ classes for XMS .NET

V 10.0.0 Troubleshooting issues with .NET 10 runtime for IBM MQ classes for XMS .NET

IBM MQ supports .NET 10 at runtime for applications using IBM MQ classes for XMS .NET.

About this task

When IBM MQ classes for XMS .NET applications run against the new runtime or component, existing binaries might encounter a breaking change in behavior, such as failure to load or execute, and they might require recompilation.

- If you are using a .NET 6 or .NET 8 application, you can run this application without any recompilation being required by making a small edit in the `runtimeconfig` file to set the `targetframeworkversion` to `"net10.0"`. For more information, see Breaking changes in .NET 10.

z/OS Troubleshooting IBM MQ for z/OS problems

IBM MQ for z/OS, CICS, Db2, and IMS produce diagnostic information which can be used for problem determination.

About this task

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 subtopics for more information about problem determination and diagnostic information on IBM MQ for z/OS.

- “IBM MQ for z/OS performance constraints” on page 255
- “IBM MQ for z/OS recovery actions” on page 257
- “IBM MQ for z/OS abends” on page 258
- “Diagnostic information produced on IBM MQ for z/OS” on page 261
- “Other sources of problem determination information for IBM MQ for z/OS” on page 263
- “Diagnostic aids for CICS” on page 266
- “Diagnostic aids for IMS” on page 276
- “Diagnostic aids for Db2” on page 276
- “IBM MQ for z/OS dumps” on page 276
- “Dealing with performance problems on z/OS” on page 296
- “Dealing with incorrect output on z/OS” on page 302

Related concepts

“Using error logs” on page 405
There are a variety of error logs that you can use to help with problem determination and troubleshooting.

“First Failure Support Technology (FFST)” on page 414
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

“IBM MQ troubleshooting and support” on page 5
If you are having problems with your queue manager network or IBM MQ applications, you can use the information in this section to help you diagnose and either solve the problem yourself or obtain help to resolve it.

“Tracing” on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

z/OS 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
V 10.0.0 V 10.0.0 This parameter controls the size of the output buffer (in the range 40 KB through 40000 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
- “Queue manager detected errors” on page 258

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 "Tracing on z/OS" on page 447 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.

z/OS 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 an 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 an X'6C6' is issued, IBM MQ has determined that continued operation could result in the loss of data integrity. Errors associated with an 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 6 on page 259 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 261.

Table 6. 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 an 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
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 abends on IBM MQ for z/OS” on page 260
Abends can occur with applications and other z/OS systems. Use this topic to investigate program abends, batch abends, CICS transaction abends, and IMS transaction abends.

“CICS, IMS, and z/OS abends” on page 261
Use this topic to investigate abends from CICS, IMS, and z/OS.

“Diagnostic information produced on IBM MQ for z/OS” on page 261
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 276

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.

z/OS *Dealing with abends on IBM MQ for z/OS*

Abends can occur with applications and other z/OS systems. Use this topic to investigate program abends, batch abends, CICS transaction abends, and IMS transaction abends.

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 documentation 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 *z/OS MVS System Codes* manual
- For CICS abends, CICS Messages
- For IMS abends, *IMS Messages and Codes*
- For Db2 abends, *Db2 Messages*
- For RRS abends, see ATR messages in *z/OS MVS System Messages, Volume 3*
- For XES abends, see IXL messages in *z/OS MVS System Messages, Volume 10*

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 276).

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 z/OS” on page 300 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 *CICS Troubleshooting*, formerly 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 Troubleshooting for IMS.

z/OS *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 Transaction abend 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 information is 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.

See SDWA information for more details.

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 276.

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 276 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 15 on page 263 shows an example of a symptom string.

```
PIDS/ 5655R3600 RIDS/CSQMAIN1 AB/S6C6 PRCS/0E30003
```

Figure 15. Sample symptom string

The symptom string provides a number of keywords that IBM Support can use to search the software support database. If you report a problem, 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

“Other sources of problem determination information for IBM MQ for z/OS” on page 263
Use this topic to investigate other sources of information for IBM MQ for z/OS problem determination.

“Diagnostic aids for CICS” on page 266
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 276
Use this topic to investigate IMS diagnostic facilities.

“Diagnostic aids for Db2” on page 276
Use this topic to investigate references for Db2 diagnostic tools.

Related tasks

“Tracing on z/OS” on page 447
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.

z/OS Other sources of problem determination information for IBM MQ for z/OS

Use this topic to investigate other sources of information for IBM MQ for z/OS 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.

z/OS Troubleshooting zHyperLink

There are a few ways you can troubleshoot zHyperLink performance issues.

Issue the command:

```
D SMS,DSNAME(dsname),STATS(ZHLWRITE)
```

to display zHyperLink write statistics, where *dsname* is the name of the data set that is used for log writes.

```
D SMS,DSNAME,STATS(ZHLWRITE) Start of Report
DATA SET VICY.MQ57.LOGCOPY1.DS01.DATA
STATISTICS Since 04/08/2024 10:51:57.096121
SUMMARY
        TOTAL     %SYNC -------------%ASYNC---------------
  WRITE REQUESTS WRITES   SKIP LNKBSY   ¬EST   MISC DISABL
           1146   4.36  79.66   0.00   0.08   0.00   4.01
                       -------%ASYNC-------
                         MISS  DELAY   DUAL
                        11.86   0.00   0.00
DEVICE STATISTICS
            TOTAL  %SYNC ---------------%ASYNC---------------------
SSID DEVNO WRITES WRITES   SKIP LNKBSY    EST   MISC   MISS  DELAY
5144 0619C   2288   3.05  39.90   0.00   0.04  16.21  40.77   0.00
D SMS,DSNAME,STATS(ZHLWRITE) End of Report
```

The preceding output shows what percentage of writes were completed synchronously, and what percentage of writes were completed asynchronously, broken down by reason for not using synchronous writes.

The output displays:

SKIP
Percentage of zHyperLink writes for which zHyperLink was not attempted due to a previous zHyperLink failure to the same track that is being written to.

LINKBSY
Percentage of zHyperLink writes that encountered a busy link condition.

EST
Percentage of zHyperLink writes where zHyperLink write access was not established.

MISC
Percentage of zHyperLink writes that were completed asynchronously due to other reasons.

MISS
Percentage of zHyperLink writes that were completed asynchronously due to a cache miss.

DELAY
Percentage of zHyperLink writes that were completed asynchronously due to an operation delay.

and, if present

DISABL
The percentage of zHyperLink writes where zHyperLink was not enabled.

DUAL
Percentage of zHyperLink writes that were completed asynchronously due to the other data set in a dual logging request.

See Appendix B of Getting Started with IBM zHyperLink for z/OS for more information and various commands and processes for problem determination with zHyperLink.

Finding the serial number of the device

You can find the serial number and SCU number by running the following command:

```
DS QD,nnnn
```

where *nnnn* is the device number. For example:

```
DS QD,9200
IEE459I 07.04.45 DEVSERV QDASD 742
UNIT VOLSER  SCUTYPE DEVTYPE CYL  SSID SCU-SERIAL DEV-SERIAL EFC
09200 CONCD1 2107981 2107900 1113 0230 0175-FCC41 0175-FCC41 *OK
**** 1 DEVICE(S) MET THE SELECTION CRITERIA
**** 0 DEVICE(S) FAILED EXTENDED FUNCTION CHECKING
```

Use the output of this command to get the SCU-SERIAL number required for displaying the number of zHyperLink sessions.

Sample DS QD,scu-serial,ZHL command

```
DS QD,MACH=0175-FCC41,ZHL
IEE459I 04.10.26 DEVSERV QDASD 419
UNIT  VOLSER SCUTYPE DEVTYPE CYL  SSID SCU-SERIAL DEV-SERIAL EFC
09200 CONCD1 2107981 2107900 1113 0230 0175-FCC41 0175-FCC41 *OK
ZHYPERLINK WRITE TOKENS ASSIGNED
TOKEN DATE TIME UNIT DATA SET NAME
00AD 07/07/21 15:10 09C16 DB2BL.D2B1.LOGCOPY2.DS02.DATA
01AE 07/07/21 15:10 09B96 DB2BL.D2B1.LOGCOPY1.DS02.DATA
02AF 07/07/21 15:10 09B96 DB2BL.D2B2.LOGCOPY1.DS02.DATA
03B0 07/07/21 15:10 09639 DB2BL.D2B2.LOGCOPY2.DS02.DATA
**** 4 WRITE TOKEN(S) MET THE SELECTION CRITERIA
**** 1 DEVICE(S) MET THE SELECTION CRITERIA
**** 0 DEVICE(S) FAILED EXTENDED FUNCTION CHECKING
```

Each token represents a connected zHyperLink session, showing which data sets are using zHyperLink.

z/OS 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 from MQI calls” on page 266.

The CICS adapter also writes trace entries to the CICS trace. These entries are described in “CICS adapter trace entries” on page 455.

Additional trace and dump data is available from the CICS region. These entries are as described in *Troubleshooting CICS*.

Examples of CEDF output from MQI calls

Examples of the output produced by the CICS execution diagnostic facility (CEDF) when using IBM MQ.

These examples show the data produced on entry to and exit from the following MQI calls, in both hexadecimal and character format. Other MQI calls produce similar data.

Related reference

Function calls

Example CEDF output for the MQOPEN call

The parameters for this call are as follows:

Table 7. Parameters for the MQOPEN call

Parameter	Description
ARG 000	Connection handle
ARG 001	Object descriptor
ARG 002	Options
ARG 003	Object handle

Table 7. Parameters for the MQOPEN call (continued)	
Parameter	**Description**
ARG 004	Completion code
ARG 005	Reason code

```
  STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 001 (X'D6C4404000000001000000001C3C5C4C6')           AT X'00144910'
 001: ARG 002 (X'00000072000000000000000000000000')            AT X'001445E8'
 001: ARG 003 (X'00000000000000720000000000000000')            AT X'001445E4'
 001: ARG 004 (X'00000000000000000000000000000000')            AT X'001445EC'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'001445F0'
```

Figure 16. Example CEDF output on entry to an MQOPEN call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 001 (X'D6C4404000000001000000001C3C5C4C6')           AT X'00144910'
 001: ARG 002 (X'00000072000000000000000000000000')            AT X'001445E8'
 001: ARG 003 (X'00000001000000720000000000000000')            AT X'001445E4'
 001: ARG 004 (X'00000000000000000000000000000000')            AT X'001445EC'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'001445F0'
```

Figure 17. Example CEDF output on exit from an MQOPEN call (hexadecimal)

```
  STATUS:  ABOUT TO EXECUTE COMMAND
  CALL TO RESOURCE MANAGER MQM
   001: ARG 000 ('............. .')
   001: ARG 001 ('OD  ........CEDF')
   001: ARG 002 ('................')
   001: ARG 003 ('................')
   001: ARG 004 ('................')
   001: ARG 005 ('................')
```

Figure 18. Example CEDF output on entry to an MQOPEN call (character)

```
 STATUS:  COMMAND EXECUTION COMPLETE
 CALL TO RESOURCE MANAGER MQM
  001: ARG 000 ('............. .')
  001: ARG 001 ('OD  ........CEDF')
  001: ARG 002 ('................')
  001: ARG 003 ('................')
  001: ARG 004 ('................')
  001: ARG 005 ('................')
```

Figure 19. Example CEDF output on exit from an MQOPEN call (character)

Related reference

MQOPEN - Open object

Example CEDF output for the MQCLOSE call

The parameters for this call are:

Table 8. Parameters for the MQCLOSE call	
Parameter	**Description**
ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Options
ARG 003	Completion code
ARG 004	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 003 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 004 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 20. Example CEDF output on entry to an MQCLOSE call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000000000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000000000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 003 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 004 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 21. Example CEDF output on exit from an MQCLOSE call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.............. .')
 001: ARG 003 ('................')
 001: ARG 004 ('...............-')
```

Figure 22. Example CEDF output on entry to an MQCLOSE call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.............. .')
 001: ARG 003 ('................')
 001: ARG 004 ('...............-')
```

Figure 23. Example CEDF output on exit from an MQCLOSE call (character)

Related reference

MQCLOSE - Close object

Example CEDF output for the MQPUT call

The parameters for this call are:

Table 9. Parameters for the MQPUT call	
Parameter	**Description**
ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Message descriptor
ARG 003	Put message options
ARG 004	Buffer length
ARG 005	Message data
ARG 006	Completion code
ARG 007	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'D4C4404000000001000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D64000000001000000024000000')             AT X'00144B48'
 001: ARG 004 (X'00000080000000000000000000040000')            AT X'001445F4'
 001: ARG 005 (X'5C5CC8C5D3D3D640E6D6D9D3C45C5C5C')            AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000000')            AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000000000000')            AT X'001445F0'
```

Figure 24. Example CEDF output on entry to an MQPUT call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'D4C4404000000001000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D64000000001000000024000000')             AT X'00144B48'
 001: ARG 004 (X'00000080000000000000000000040000')            AT X'001445F4'
 001: ARG 005 (X'5C5CC8C5D3D3D640E6D6D9D3C45C5C5C')            AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000000')            AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000000000000')            AT X'001445F0'
```

Figure 25. Example CEDF output on exit from an MQPUT call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO .............')
 001: ARG 004 ('................')
 001: ARG 005 ('**HELLO WORLD***')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
```

Figure 26. Example CEDF output on entry to an MQPUT call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO .............')
 001: ARG 004 ('................')
 001: ARG 005 ('**HELLO WORLD***')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
```

Figure 27. Example CEDF output on exit from an MQPUT call (character)

Related reference

MQPUT - Put message

Example CEDF output for the MQPUT1 call

The parameters for this call are:

Table 10. Parameters for the MQPUT1 call

Parameter	Description
ARG 000	Connection handle
ARG 001	Object descriptor
ARG 002	Message descriptor
ARG 003	Put message options
ARG 004	Buffer length
ARG 005	Message data
ARG 006	Completion code
ARG 007	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000000000007200000000')            AT X'001445E0'
 001: ARG 001 (X'D6C4404000000010000001C3C5C4C6')              AT X'00144910'
 001: ARG 002 (X'D4C4404000000010000000000000008')             AT X'001449B8'
 001: ARG 003 (X'D7D4D6400000000100000024000000000')           AT X'00144B48'
 001: ARG 004 (X'00000080000000800000006000040000')            AT X'001445F4'
 001: ARG 005 (X'5C5CC8C5D3D3D640E6D6D9D3C45C5C5C')            AT X'00144BF8'
 001: ARG 006 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 007 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 28. Example CEDF output on entry to an MQPUT1 call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'0000000000000000000007200000000')            AT X'001445E0'
 001: ARG 001 (X'D6C4404000000000100000001C3C5C4C6')          AT X'00144910'
 001: ARG 002 (X'D4C4404000000001000000000000000008')        AT X'001449B8'
 001: ARG 003 (X'D7D4D64000000001000000024000000000')        AT X'00144B48'
 001: ARG 004 (X'0000000800000008000000060000040000')        AT X'001445F4'
 001: ARG 005 (X'5C5CC8C5D3D3D640E6D6D9D3C45C5C5C')          AT X'00144BF8'
 001: ARG 006 (X'0000000000000000000000800000008')           AT X'001445EC'
 001: ARG 007 (X'0000000000000008000000800000060')           AT X'001445F0'
```

Figure 29. Example CEDF output on exit from an MQPUT1 call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('...........-....')
 001: ARG 005 ('**HELLO WORLD***')
 001: ARG 006 ('................')
 001: ARG 007 ('...............-')
```

Figure 30. Example CEDF output on entry to an MQPUT1 call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('OD  ........CEDF')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('PMO ............')
 001: ARG 004 ('...........-....')
 001: ARG 005 ('**HELLO WORLD***')
 001: ARG 006 ('................')
 001: ARG 007 ('...............-')
```

Figure 31. Example CEDF output on exit from an MQPUT1 call (character)

Related reference

MQPUT1 - Put one message

Example CEDF output for the MQGET call

The parameters for this call are:

Table 11. Parameters for the MQGET call

Parameter	Description
ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Message descriptor
ARG 003	Get message options
ARG 004	Buffer length
ARG 005	Message buffer
ARG 006	Message length

Table 11. Parameters for the MQGET call (continued)	
Parameter	**Description**
ARG 007	Completion code
ARG 008	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')            AT X'001445E4'
 001: ARG 002 (X'D4C4404000000001000000000000000')             AT X'001449B8'
 001: ARG 003 (X'C7D4D64000000000100004044FFFFFFFF')           AT X'00144B00'
 001: ARG 004 (X'00000008000000000000000000040000')            AT X'001445F4'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'00144C00'
 001: ARG 006 (X'00000000000000000000040000000000')            AT X'001445F8'
 001: ARG 007 (X'00000000000000000000000800000000')            AT X'001445EC'
 001: ARG 008 (X'00000000000000080000000000000000')            AT X'001445F0'
```

Figure 32. Example CEDF output on entry to an MQGET call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'00000001000000720000000000000000')            AT X'001445E4'
 001: ARG 002 (X'D4C4404000000001000000000000008')             AT X'001449B8'
 001: ARG 003 (X'C7D4D64000000000100004044FFFFFFFF')           AT X'00144B00'
 001: ARG 004 (X'00000008000000080000000000040000')            AT X'001445F4'
 001: ARG 005 (X'5C5CC8C5D3D3D640E6D6D9D3C45C5C5C')            AT X'00144C00'
 001: ARG 006 (X'00000008000000000000040000000000')            AT X'001445F8'
 001: ARG 007 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 008 (X'00000000000000080000000800000000')            AT X'001445F0'
```

Figure 33. Example CEDF output on exit from an MQGET call (hexadecimal)

```
 STATUS:  ABOUT TO EXECUTE COMMAND
 CALL TO RESOURCE MANAGER MQM
  001: ARG 000 ('................')
  001: ARG 001 ('................')
  001: ARG 002 ('MD  ............')
  001: ARG 003 ('GMO ...... .....')
  001: ARG 004 ('................')
  001: ARG 005 ('................')
  001: ARG 006 ('................')
  001: ARG 007 ('................')
  001: ARG 008 ('................')
```

Figure 34. Example CEDF output on entry to an MQGET call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('MD  ............')
 001: ARG 003 ('GMO ...... .....')
 001: ARG 004 ('................')
 001: ARG 005 ('**HELLO WORLD***')
 001: ARG 006 ('................')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
```

Figure 35. Example CEDF output on exit from an MQGET call (character)

Related reference

MQGET - Get message

Example CEDF output for the MQINQ call

The parameters for this call are:

Table 12. Parameters for the MQINQ call

Parameter	Description
ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Count of selectors
ARG 003	Array of attribute selectors
ARG 004	Count of integer attributes
ARG 005	Integer attributes
ARG 006	Length of character attributes buffer
ARG 007	Character attributes
ARG 008	Completion code
ARG 009	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')            AT X'05ECAFCC'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'0000000200000404485ECA00885ECA220')           AT X'05ECAFD4'
 001: ARG 003 (X'0000000D0000000C0000000000000000')            AT X'00144C08'
 001: ARG 004 (X'0000000200000404485ECA00885ECA220')           AT X'05ECAFD4'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')            AT X'05ECAFCC'
 001: ARG 007 (X'00000000000000000000000000000000')            AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800040000')            AT X'001445F0'
```

Figure 36. Example CEDF output on entry to an MQINQ call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000000200004044')            AT X'05ECAFCC'
 001: ARG 001 (X'0000000100000072000000000000000'0')           AT X'001445E4'
 001: ARG 002 (X'0000000200000404485ECA00885ECA220')            AT X'05ECAFD4'
 001: ARG 003 (X'0000000D0000000C00400000000000000')           AT X'00144C08'
 001: ARG 004 (X'0000000200000404485ECA00885ECA220')            AT X'05ECAFD4'
 001: ARG 005 (X'00400000000000000000000000000000')            AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')            AT X'05ECAFCC'
 001: ARG 007 (X'00000000000000000000000000000000')            AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800040000')            AT X'001445F0'
```

Figure 37. Example CEDF output on exit from an MQINQ call (hexadecimal)

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('................')
 001: ARG 002 ('...... .e...e.s.')
 001: ARG 003 ('................')
 001: ARG 004 ('...... .e...e.s.')
 001: ARG 005 ('................')
 001: ARG 006 ('.............. .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('................')
```

Figure 38. Example CEDF output on entry to an MQINQ call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('.............. .')
 001: ARG 001 ('................')
 001: ARG 002 ('...... .e...e.s.')
 001: ARG 003 ('......... ......')
 001: ARG 004 ('...... .e...e.s.')
 001: ARG 005 ('. ..............')
 001: ARG 006 ('.............. .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('................')
```

Figure 39. Example CEDF output on exit from an MQINQ call (character)

Related reference

MQINQ - Inquire object attributes

Example CEDF output for the MQSET call

The parameters for this call are:

Table 13. Parameters for the MQSET call

Parameter	Description
ARG 000	Connection handle
ARG 001	Object handle
ARG 002	Count of selectors
ARG 003	Array of attribute selectors
ARG 004	Count of integer attributes

Table 13. Parameters for the MQSET call (continued)	
Parameter	**Description**
ARG 005	Integer attributes
ARG 006	Length of character attributes buffer
ARG 007	Character attributes
ARG 008	Completion code
ARG 009	Reason code

```
STATUS:  ABOUT TO EXECUTE COMMAND
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'0000000100000002000404485ECA008')             AT X'05ECAFDC'
 001: ARG 003 (X'00000018000007DF0000000000000000')            AT X'00144C08'
 001: ARG 004 (X'0000000100000002000404485ECA008')             AT X'05ECAFDC'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 007 (X'00000000000000000000000000000000')            AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 40. Example CEDF output on entry to an MQSET call (hexadecimal)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 (X'00000000000000010000007200000000')            AT X'001445E0'
 001: ARG 001 (X'0000000100000072000000000000000')             AT X'001445E4'
 001: ARG 002 (X'0000000100000002000404485ECA008')             AT X'05ECAFDC'
 001: ARG 003 (X'00000018000007DF0000000000000000')            AT X'00144C08'
 001: ARG 004 (X'0000000100000002000404485ECA008')             AT X'05ECAFDC'
 001: ARG 005 (X'00000000000000000000000000000000')            AT X'00144C10'
 001: ARG 006 (X'00000000000000010000000200004044')            AT X'05ECAFD8'
 001: ARG 007 (X'00000000000000000000000000000000')            AT X'00144C18'
 001: ARG 008 (X'00000000000000000000000800000008')            AT X'001445EC'
 001: ARG 009 (X'00000000000000080000000800000060')            AT X'001445F0'
```

Figure 41. Example CEDF output on exit from an MQSET call (hexadecimal)

```
 STATUS:  ABOUT TO EXECUTE COMMAND
 CALL TO RESOURCE MANAGER MQM
  001: ARG 000 ('................')
  001: ARG 001 ('................')
  001: ARG 002 ('.......... .e...')
  001: ARG 003 ('................')
  001: ARG 004 ('.......... .e...')
  001: ARG 005 ('................')
  001: ARG 006 ('............... .')
  001: ARG 007 ('................')
  001: ARG 008 ('................')
  001: ARG 009 ('................-')
```

Figure 42. Example CEDF output on entry to an MQSET call (character)

```
STATUS:  COMMAND EXECUTION COMPLETE
CALL TO RESOURCE MANAGER MQM
 001: ARG 000 ('................')
 001: ARG 001 ('................')
 001: ARG 002 ('.......... .e...')
 001: ARG 003 ('................')
 001: ARG 004 ('.......... .e...')
 001: ARG 005 ('................')
 001: ARG 006 ('............... .')
 001: ARG 007 ('................')
 001: ARG 008 ('................')
 001: ARG 009 ('................-')
```

Figure 43. Example CEDF output on exit from an MQSET call (character)

Related reference

MQSET - Set object attributes

z/OS 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. See *Troubleshooting IMS* for more information.

z/OS Diagnostic aids for Db2

Use this topic to investigate references for Db2 diagnostic tools.

See the Db2 Information Center for help in diagnosing Db2 problems.

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.

Use the instructions in the following topics to get and process a dump:

- “Getting a dump with IBM MQ for z/OS” on page 277
- “Using the z/OS DUMP command” on page 278
- “Processing a dump using the IBM MQ for z/OS dump display panels” on page 279
- “Processing an IBM MQ for z/OS dump using line mode IPCS” on page 283
- “Processing an IBM MQ for z/OS dump using IPCS in batch” on page 290

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 z/OS” on page 291. For information about the IBM MQ for z/OS abend codes, see “IBM MQ for z/OS abends” on page 258, 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 z/OS” on page 293
- “Snap dumps on z/OS” on page 294
- “SYS1.LOGREC information on z/OS” on page 294
- “SVC dumps on z/OS” on page 295

Related concepts

“IBM MQ for z/OS abends” on page 258
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 261
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.

Related tasks

“Tracing on z/OS” on page 447
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.

z/OS *Getting a dump with IBM MQ for z/OS*

Use this topic to understand the different dump types for IBM MQ for z/OS 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 14. 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 z/OS” on page 295. 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.

See SVC dump in the *z/OS MVS Diagnosis: Tools and Service Aids* manual for more information.

z/OS 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 44 on page 278 through to Figure 48 on page 279 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 44. 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 45. 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 46. 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 47. 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 48. Dumping a coupling facility structure

See DUMP command for more information.

z/OS *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 required.  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  IBM MQ dump formatter panel interface
  CSQWDMP  IBM MQ dump formatter
  CTRACE   Component trace summary
  DAEDATA  DAE header data
  DIVDATA  Data-in-virtual storage
```

Figure 49. IPCS MVS Dump Component Data Analysis panel

3. Select **CSQMAIN IBM MQ dump formatter panel interface** by typing s next to 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 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, 2026. 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 280 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 end 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 an IBM MQ for z/OS dump using line mode IPCS" on page 283
Use the IPCS commands to format a dump.

"Processing an IBM MQ for z/OS dump using IPCS in batch" on page 290
Use this topic to understand how IBM MQ for z/OS dumps can be formatted by IPCS commands in batch mode.

"Analyzing the dump and interpreting dump titles on z/OS" on page 291
Use this topic to understand how IBM MQ for z/OS dump titles are formatted, and how to analyze a dump.

z/OS *Processing an IBM MQ for z/OS 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.

Formatting a dump from the channel initiator on z/OS

how to format a channel initiator dump for IBM MQ for z/OS 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.

Table 15 on page 283 describes the keywords that you can specify with CSQXDPRD.

Table 15. 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

Table 15. Keywords for the IPCS VERBEXIT CSQXDPRD (continued)	
Keyword	**What is formatted**
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.
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

“Processing a dump using the IBM MQ for z/OS dump display panels” on page 279
You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

“Processing an IBM MQ for z/OS dump using IPCS in batch” on page 290
Use this topic to understand how IBM MQ for z/OS dumps can be formatted by IPCS commands in batch mode.

“Analyzing the dump and interpreting dump titles on z/OS” on page 291
Use this topic to understand how IBM MQ for z/OS dump titles are formatted, and how to analyze a dump.

“Formatting an IBM MQ for z/OS dump” on page 284
Use this topic to understand how to format a queue manager dump using line mode IPCS commands.

z/OS *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 *z/OS MVS IPCS Commands* manual.

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 16 on page 285 explains each keyword that you can specify in the control statement for formatting dumps.

Table 16. 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.
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 17 on page 286 details 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 in Table 16 on page 285.

Table 17. 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 page set, if the page is present in a buffer. (The difference between BMC=3 and BMC=4 is the route taken to the page.)
BUFL= *nnnnnnnnnn*	Storage access buffer allocation sz.
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).

Table 17. Resource manager dump formatting keywords (continued)	
Keyword	**What is formatted**
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 18. 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= *nnnnnnnn*	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).

Table 18. Resource manager dump formatting keywords (J -P) (continued)	
Keyword	**What is formatted**
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
MT	Message properties trace
MQVCX	MQCHARVs in hexadecimal format
PROPS= *nnnnnnnnnnnnnnnn*	Message properties pointer
PSID= *nnnnnnnn*	Page set to format page
PSTRX	Properties strings in hex format

Table 19. 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

Table 19. Resource manager dump formatting keywords (R -Z) (continued)	
Keyword	**What is formatted**
TOP= *nnnnnnnnnnnnnnn* /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 = *nnnnnnnnnnnnnnn*	ZST message handle

If the dump is initiated by the operator, there is no information in the summary portion of the dump. Table 20 on page 289 shows additional keywords that you can use in the CSQWDMP control statement.

Table 20. 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 IBM Support.)

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 21 on page 290 shows some other commands that are used frequently for analyzing dumps. For more information about these sub commands, see the *z/OS MVS IPCS Commands* manual.

Table 21. 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.
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 information

“Formatting a dump from the channel initiator on z/OS” on page 283

z/OS *Processing an IBM MQ for z/OS dump using IPCS in batch*

Use this topic to understand how IBM MQ for z/OS 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 50 on page 291).

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 50. 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 279
You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

"Processing an IBM MQ for z/OS dump using line mode IPCS" on page 283
Use the IPCS commands to format a dump.

"Analyzing the dump and interpreting dump titles on z/OS" on page 291
Use this topic to understand how IBM MQ for z/OS dump titles are formatted, and how to analyze a dump.

z/OS *Analyzing the dump and interpreting dump titles on z/OS*

Use this topic to understand how IBM MQ for z/OS 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 51 on page 291 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 51. 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 52 on page 292 illustrates this format.

```
ssnm,ABN=compltn-reason,U=userid,C=compid.release.comp-function,
 M=module,PSW=psw_contents,ASID=address_space_id
```

Figure 52. 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 279

You can use commands available through IPCS panels to process dumps. Use this topic to understand the IPCS options.

"Processing an IBM MQ for z/OS dump using line mode IPCS" on page 283
Use the IPCS commands to format a dump.

"Processing an IBM MQ for z/OS dump using IPCS in batch" on page 290
Use this topic to understand how IBM MQ for z/OS dumps can be formatted by IPCS commands in batch mode.

z/OS *SYSUDUMP information on z/OS*

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 *z/OS MVS Diagnosis: Tools and Service Aids* manual.

Figure 53 on page 293 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 53. Sample beginning of a SYSUDUMP

z/OS *Snap dumps on z/OS*

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.

z/OS *SYS1.LOGREC information on z/OS*

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 *z/OS 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 to format records in the SYS1.LOGREC data set.

  Following is some example JCL to get software LOGREC report:

```
//STEP EXEC PGM=IFCEREP1,PARM='CARD'
//ACCIN DD DSN=EREP.HISTORY,DISP=(OLD,PASS)
//DIRECTWK DD UNIT=SYSDA,SPACE=(CYL,10,,CONTIG)
//EREPPT DD SYSOUT=A,DCB=BLKSIZE=133
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133
//SYSIN DD *
PRINT=PS
HIST ****don't use this line if record is in SYS1.LOGREC****
ACC=N
TYPE=S
ENDPARM
/*
```

  Notes:

  – PRINT=PS is for Detail and Summary, TYPE=S is for software.

    You can also add a time range like TIME=(2000-2005) to get only the records you need.

  - AS for TIME, data can be extracted using DATE as well. See EREP Selection Parameters, Parameter descriptions, and Syntax rules for information on how you set the selection of the date range.
- 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 279).

Only records available in storage when the dump was requested are included. Each formatted record follows the heading *****LOGDATA*****.

z/OS *SVC dumps on z/OS*

Use this topic to understand how to suppress SVC dumps on z/OS, 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 *z/OS 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 *z/OS 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.

z/OS Dealing with performance problems on z/OS

Use this topic to investigate IBM MQ for z/OS 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.

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.

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 for z/OS performance. Use this information for further details 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.

Related concepts

“Dealing with incorrect output on z/OS” on page 302
Incorrect output can be missing, unexpected, or corrupted information.

“Dealing with applications that are running slowly or have stopped on z/OS” on page 297
Waits and loops can exhibit similar symptoms. Use the links in this topic to help differentiate between waits and loops on z/OS.

Related tasks

“Making initial checks” on page 6

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

z/OS *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 on z/OS.

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

Distinguishing between waits and loops on z/OS

Waits and loops on IBM MQ for z/OS can present similar symptoms. Use this information to help determine if you are experiencing a wait or 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 appropriate for your problem. The following information 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

To perform the tests shown in these topics, you need access to the z/OS console, and to be able to issue operator commands.

- “Dealing with waits on z/OS” on page 298
- “Dealing with loops on z/OS” on page 300

Related tasks

“Making initial checks” on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

z/OS *Dealing with waits on z/OS*

Waits can occur in batch or TSO applications, CICS transactions, and other components on IBM MQ for z/OS. 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 z/OS" on page 303), or in IBM MQ itself (see "Is IBM MQ waiting for z/OS ?" on page 300). See "IBM MQ for z/OS dumps" on page 276 (specifically Figure 44 on page 278) for information about obtaining a dump.

If the problem persists, see "Collecting troubleshooting information" on page 310 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 z/OS" on page 303). 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 z/OS" on page 305).

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. See "Dealing with loops on z/OS" on page 300 for information about waits. Refer to "IBM MQ for z/OS dumps" on page 276 (specifically Figure 44 on page 278) for information about obtaining a dump.

If the problem persists, refer to "Collecting troubleshooting information" on page 310 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? in "Examining the problem in greater depth on z/OS" on page 56.
   - 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, see "Collecting troubleshooting information" on page 310 for further guidance.

Related concepts

"Dealing with loops on z/OS" on page 300
Loops can occur in different areas of a z/OS system. Use this topic to help determine where a loop is occurring.

Related information

"Distinguishing between waits and loops on z/OS" on page 297

z/OS *Dealing with loops on z/OS*

Loops can occur in different areas of a z/OS system. Use this topic to help determine where a loop is occurring.

The following sections 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? in "Examining the problem in greater depth on z/OS" on page 56 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 "Collecting troubleshooting information" on page 310 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

"Dealing with waits on z/OS" on page 298

Waits can occur in batch or TSO applications, CICS transactions, and other components on IBM MQ for z/OS. Use this topic to determine where waits can occur.

Related information

"Distinguishing between waits and loops on z/OS" on page 297

z/OS *Dealing with improving the MQGET performance on z/OS*

Check for the CSQI004I message in the MSTR job log to know whether MQGET performance can be improved by adding a queue index for a local queue.

Attention: The lack of queue indexing, when it is needed, can also result in excessively high CPU usage.

The common MQGET performance problems are characterized by the following:

- The queue has a large depth
- The MQGET call uses MQMO_MATCH_MSG_ID or MQMO_MATCH_CORREL_ID (the default is for both to be set)
- The MQGET call sets a nonnull value or a nonblank value for the corresponding MQMD.MsgId or MQMD.CorrelId output fields

An example of a message with CSQI004I is:

```
CSQI004I +CSQ1 CSQIMGE3
Consider indexing TESTQ by
MSGID for BATCH connection GETJOB, 110 messages skipped
```

If both MSGID and CORRELID are matched, the recommendation is to index by MSGID. The queue is scanned sequentially searching for the requested message. The message does not in itself indicate a problem, but can be useful in tracking because it does indicate a potential issue, which you need to address.

To set the queue index that is indicated by the output you receive for this message, use one of the following commands:

- **ALTER QLOCAL(qname) INDXTYPE(MSGID)**
- **ALTER QLOCAL(qname) INDXTYPE(CORRELID)**

The description of the **INDXTYPE** parameter has a table to see when index changes can be made. For more information, see Index type change permitted depending upon queue-sharing and presence of messages in the queue.

Note: By default, the warning is issued once, per queue per key type, within a 60-minute interval when the MQGET call skips 100 messages.

There can still be performance impact if fewer than 100 messages are skipped and the CSQI004I message does not appear. To see whether messages are being skipped, you can collect the SMF 116 CLASS(3) accounting data to verify the current accounting trace data collection. The `WQ->getsmsg` field indicates the number of messages that are skipped during the MQGET calls.

In SupportPac MP1B output, the `get skipped message count` is available in the TASK output. Details of the structures and fields can be found in IBM MQ SupportPac MP1B.

Related information

Data manager messages (CSQI...)

z/OS Dealing with incorrect output on z/OS

Incorrect output can be missing, unexpected, or corrupted information.

The term "incorrect output' can be interpreted in many different ways. Points to consider if you have obtained what you believe to be some incorrect output are:

- When to classify output as incorrect

"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, see Are there any error messages, return codes or other error conditions? in “Identifying characteristics of the problem on z/OS” on page 51.

- Unexpected messages

  Your application might not have received a message that it was expecting, or 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).

The subtopics contain 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 z/OS” on page 303
- “Problems with missing messages when using distributed queuing on z/OS” on page 305
- “Problems with getting messages when using message grouping on z/OS” on page 306
- “Finding messages sent to a cluster queue on z/OS” on page 307
- “Finding messages sent to the IBM MQ - IMS bridge” on page 307
- “Messages contain unexpected or corrupted information on z/OS” on page 308

Related concepts

“Dealing with performance problems on z/OS” on page 296
Use this topic to investigate IBM MQ for z/OS performance problems in more detail.

Related tasks

“Making initial checks” on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

z/OS *Messages do not arrive when expected on z/OS*

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 z/OS" on page 308.

z/OS *Problems with missing messages when using distributed queuing on z/OS*

Use this topic to understand possible causes of missing messages when using distributed queuing on IBM MQ for z/OS.

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.

z/OS *Problems with getting messages when using message grouping on z/OS*

Use this topic to understand some of the issues with getting messages when using message grouping on IBM MQ for z/OS.

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 z/OS” on page 305. 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.

z/OS *Finding messages sent to a cluster queue on z/OS*

Use this topic to understand some of the issues involved with finding messages sent to a cluster queue on IBM MQ for z/OS.

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.

z/OS *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?

z/OS *Messages contain unexpected or corrupted information on z/OS*

Use this topic to understand some of the issues that can cause unexpected or corrupted output on z/OS.

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.

z/OS 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 concepts

Layout of SMF records for the channel initiator

z/OS *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 queue manager attribute 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, data is written to SMF when any of the following conditions occur:
   - From IBM MQ for z/OS 9.3.0, a timed interval, depending on the values of the STATIME and ACCTIME queue manager system parameters. Use the **DISPLAY SYSTEM** command to display the value of STATIME and ACCTIME.
   - From IBM MQ for z/OS 9.3.0, the **SET SYSTEM** command is issued to change the value of the STATIME or ACCTIME system parameters.
   - The channel initiator is shut down.
   - The **STOP TRACE(A) CLASS(4)** command is issued.
   - From IBM MQ 9.3.0, if you specify a value between 0 and 1440 for the ACCTIME parameter, that value is used instead of STATIME.
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. The MAXDORM value is displayed in the format *mmss*, where *mm* is the time in minutes and *ss* is seconds. SMF can keep the data in memory for the MAXDORM period before writing it out.

Related tasks

Planning for channel initiator SMF data

Interpreting IBM MQ performance statistics

z/OS *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 to gather channel initiator statistics SMF data.
2. If the trace is enabled, data is written to SMF when any of the following conditions occur:
   - On a timed interval, depending on the value of the STATIME queue manager system parameter. A value of zero means that the SMF statistics broadcast is used. Use the **DISPLAY SYSTEM** command to display the value of STATIME.
   - The **SET SYSTEM** command is issued to change the value of the STATIME system parameter.
   - The channel initiator is shut down.
   - The **STOP TRACE(S) CLASS(4)** command is issued.
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. The MAXDORM value is displayed in the

format *mmss*, where *mm* is the time in minutes and *ss* is seconds. SMF can keep the data in memory for the MAXDORM period before writing it out.

Collecting troubleshooting information

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

About this task

If you need assistance with a problem that you are having with IBM MQ, you can obtain help through the following web pages:

- Multi IBM MQ for Multiplatforms Support web page
- z/OS IBM MQ for z/OS Support web page

The support pages provide links to a number of different resources to help you with troubleshooting. You can also subscribe to notifications about IBM MQ fixes, troubleshooting, and other news.

If you are unable to resolve an issue yourself and need assistance, you can open a case. This section explains how to collect troubleshooting information to include with the case for a number of different types of problem that you might encounter with IBM MQ for Multiplatforms or IBM MQ for z/OS. In addition to the information described in this section, IBM might request further information on a case by case basis.

Note: Multi Running the **runmqras** command will help you with collecting troubleshooting information before you send it to IBM. For more information, see runmqras (collect IBM MQ troubleshooting information).

For more information, including how to register for support, see the IBM Support Guide.

Collecting troubleshooting information on Multiplatforms

An overview of how to collect troubleshooting information for IBM MQ on Multiplatforms.

About this task

Note: In addition to the information described in this section, IBM might request further information on a case by case basis.

Procedure

- For general information on how to collect troubleshooting information and send it to IBM, see:
  - “Collecting troubleshooting information automatically with runmqras” on page 311
  - “Collecting troubleshooting information manually” on page 315
  - “Sending troubleshooting information to IBM” on page 404
- For information on how to collect troubleshooting and diagnostic information for a specific problem area for IBM MQ for Multiplatforms, see:
  - MQ Adv. Advanced Message Security (AMS)
  - C, C++, COBOL, .NET, pTAL, RPG and Visual Basic client applications
  - Channels
  - IBM MQ Clustering
  - Data conversion
  - Dead-letter queue messages

  - Error Messages and FFST Files
  - IBM WebSphere MQ File Transfer Edition (FTE): see Managed File Transfer (MFT)
  - Hang and high CPU problems
  - IBM MQ Explorer
  - “Collecting information for MQIPT problems on Multiplatforms” on page 354
  - Installation and uninstallation
  - Java and JMS
  - Logging and recovery
  - MQ Adv. Managed File Transfer
  - Microsoft Cluster Service
  - Performance
  - Publish/subscribe
  - Linux MQ Adv. Replicated data queue manager (RDQM)
  - Security
  - TLS channels (formerly SSL)
  - Triggering
- MQ Appliance
  For IBM MQ Appliance, see Collect IBM MQ Appliance MustGather data to solve problems.
- OpenShift CP4I
  For IBM MQ in containers, see Collecting troubleshooting information for queue managers deployed with the IBM MQ Operator.
- For all other problems see Collect IBM MQ MustGather data to solve all other problems on Linux, UNIX, Windows, and IBM i.

Related tasks

“Collecting troubleshooting information on z/OS” on page 361
An overview of how to collect troubleshooting information for IBM MQ for z/OS.

Multi Collecting troubleshooting information automatically with runmqras

If you need to collect troubleshooting information for a support case, you can use the **runmqras** command to gather the information together into a single archive.

Before you begin

The **runmqras** command is a Java application for collecting IBM MQ troubleshooting information. If your IBM MQ installation includes the Java JRE component, **runmqras** will use it, otherwise make sure a recent Java runtime environment (JRE) is in your **PATH** to avoid the following error:

```
AMQ8599E: The runmqras command was unable to locate a JRE
```

Ensure that your environment is set up for your IBM MQ installation before starting **runmqras**. For example,:

- Linux UNIX On UNIX and Linux:

```
sh> PATH="$PATH":/path/to/java/bin (only if needed)
sh> . /opt/mqm/bin/setmqenv -n Installation1
```

- Windows On Windows:

```
C:\> SET PATH=%PATH%;C:\path\to\java\bin; (only if needed)
C:\> C:\Program Files\IBM\MQ\bin\setmqenv -n Installation2
```

- IBM i On IBM i (Qshell):

```
PATH="$PATH":/QOpenSys/QIBM/ProdData/JavaVM/jdk80/64bit (only if needed)
```

  Optionally, you can add the `/QIBM/ProdData/mqm/bin` directory to your **PATH** so you can use **runmqras** without typing its full path. To do so, enter one of the following commands in the Qshell or add it to the `.profile` file in your home directory so it will run automatically every time you start the Qshell:

```
===> . /QIBM/ProdData/mqm/bin/setmqenv -s
```

If you are unable to use the **runmqras** tool to collect the information automatically, for example if you are running an older version of IBM MQ or cannot use **runmqras** for any other reason, you can instead collect the information manually as described in “Collecting troubleshooting information manually” on page 315.

Tip: Before using **runmqras**, you might wish to clean up IBM MQ files to reduce the amount of data collected. For more information, see Cleaning up IBM MQ files.

About this task

You can use the **runmqras** command to gather troubleshooting information about an application or IBM MQ failure into a single archive that you can submit to IBM when you report a problem.

By default, **runmqras** gathers information such as:

- IBM MQ FDC files.
- Error logs (from all queue managers as well as the machine-wide IBM MQ error logs).
- Product versioning, status information, and output from various other operating system commands.

If IBM Support ask you for more detailed information, you can add this by specifying the required options with the **-section** parameter.

Procedure

1. To specify that the output file name starts with your case number, use the **-caseno** parameter.
   For example:
   - Linux UNIX On UNIX and Linux:

     ```
     sh> runmqras -caseno TS123456789
     ```

   - Windows On Windows:

     ```
     C:\> runmqras -caseno TS123456789
     ```

   - IBM i On IBM i (Qshell):

     ```
     /QIBM/ProdData/mqm/bin/runmqras -caseno TS123456789
     ```

   If you are using an earlier version of the product that does not support the **-caseno** parameter, use the **-zipfile** option instead of the **-caseno** option to make the output file name start with your case number.

- Linux UNIX On UNIX and Linux:

```
sh> runmqras -zipfile TS123456789
```

- Windows On Windows:

```
C:\> runmqras -zipfile TS123456789
```

- IBM i On IBM i (Qshell):

```
/QIBM/ProdData/mqm/bin/runmqras -zipfile TS123456789
```

2. Choose the sections that you want to gather data for.

   The **runmqras** command uses a configuration file called `isa.xml` that describes which files to collect and which commands to run. This file is organized into sections that identify the information needed to solve different kinds of problems, and IBM adds new sections as needed.

   To choose the required sections, specify the **-section** parameter with the appropriate options. For example:

   - Linux UNIX On UNIX and Linux:

```
sh> runmqras -caseno TS123456789 -section defs,cluster,trace
```

   - Windows On Windows:

```
C:\> runmqras -caseno TS123456789 -section defs,cluster,trace
```

   - IBM i On IBM i (Qshell):

```
/QIBM/ProdData/mqm/bin/runmqras -caseno TS123456789 -section defs,cluster,trace
```

3. Choose the queue managers that you want to gather data for.

   By default the **runmqras** command tries to collect information about all queue managers. Use the **-qmlist** option to provide a comma-separated list of the queue managers in your current installation that **runmqras** should examine. For example:

   - Linux UNIX On UNIX and Linux:

```
sh> runmqras -caseno TS123456789 -section defs,cluster,trace -qmlist QMA,QMB,QMC
```

   - Windows On Windows:

```
C:\> runmqras -caseno TS123456789 -section defs,cluster,trace -qmlist QMA,QMB,QMC
```

   - IBM i On IBM i (Qshell):

```
===> /QIBM/ProdData/mqm/bin/runmqras -caseno TS123456789 -section defs,cluster,trace
-qmlist QMA,QMB,QMC
```

   Important: Do not use the **-qmlist** option on IBM MQ client installations.

   If you have multiple IBM MQ installations, do not use the **runmqras** command from one installation to collect information about a queue manager in a different installation. While the **runmqras** command will not fail outright, some of the commands issued by **runmqras** will fail with the error:

   `AMQ6292: The queue manager is associated with a different installation`

   Instead, first use the **setmqenv** command to switch between installations. Then, in each installation, use the **-qmlist** option of the **runmqras** command to collect information from the queue managers associated with that installation.

The queue managers you choose should be running, or else some the commands issued by the **runmqras** command will fail with the error

```
AMQ8146: IBM MQ queue manager not available
```

However, the **runmqras** command is still useful if you have a queue manager that you cannot start.

4. Choose a different directory for handling large files.

   If your system has lots of FDCs or trace files to collect, or if you collect the **all** or **QMGR** sections, the archive that the **runmqras** command creates can be very large. Normally, **runmqras** uses space in a temporary directory to collect and zip up the files. To choose a different directory on a file system or disk with more free space, use the **-workdirectory** option. The directory that you specify must be empty. If it does not exist yet, **runmqras** will create it. For example,:

   - Linux UNIX On UNIX and Linux:

     ```
     sh> runmqras -caseno TS123456789 -section defs,cluster,trace,QMGR -qmlist QMA,QMB,QMC
     -workdirectory /var/bigdata/2019-07-27
     ```

   - Windows On Windows:

     ```
     C:\> runmqras -caseno TS123456789 -section defs,cluster,trace,QMGR -qmlist QMA,QMB,QMC
     -workdirectory G:\BigData\2019-07-27
     ```

   - IBM i On IBM i (Qshell):

     ```
     ===> /QIBM/ProdData/mqm/bin/runmqras -caseno TS123456789 -section defs,cluster,trace,QMGR
     -qmlist QMA,QMB,QMC -workdirectory /QIBM/bigdata/2019-07-27
     ```

5. Send the troubleshooting information that you have collected to IBM.

   Make sure that the **runmqras** archive file starts with your IBM case number, for example TS123456789-runmqras.zip then send the file to IBM. For more information, see "Sending troubleshooting information to IBM" on page 404.

What to do next

Important: After sending your **runmqras** archive file to IBM, keep a copy of it until your problem is resolved and you have tested the solution to your satisfaction.

The **runmqras** command does not delete any files from your system, neither IBM MQ logs, nor FDCs, job logs, dumps, or trace files. After collecting these files with **runmqras**, consider archiving or deleting them as described in Cleaning up IBM MQ files. If you then need to collect troubleshooting information with **runmqras** again at a later time, the new **runmqras** file will be smaller and easier to analyze because it does not contain duplicate files and old information.

Related tasks

"Collecting troubleshooting information manually" on page 315
In some cases, you might need to collect troubleshooting information manually, for example if you are running an older version of IBM MQ or cannot use the **runmqras** command to collect troubleshooting information automatically.

"Sending troubleshooting information to IBM" on page 404

After you have generated and collected troubleshooting information for a problem, you can send it to IBM to help with problem determination for a support case.

Multi Collecting troubleshooting information manually

In some cases, you might need to collect troubleshooting information manually, for example if you are running an older version of IBM MQ or cannot use the **runmqras** command to collect troubleshooting information automatically.

About this task

If you need to collect troubleshooting information for a support case, you should in most cases use the **runmqras** tool, which automates the task of collecting troubleshooting information, rather than collecting this information manually.

These manual instructions are provided for you to use if you are unable to use the **runmqras** tool to collect the information automatically, for example if you are running an older version of IBM MQ or cannot use **runmqras** for any other reason.

Tip: Consider cleaning up IBM MQ files before packaging data in order to reduce the size the data and speed up its transfer to IBM. For more information, see Cleaning up IBM MQ files.

Procedure

1. If your system has more than one IBM MQ installation, use the **setmqenv** command to choose the installation with the problem before proceeding:
   - Linux UNIX On UNIX and Linux:

     ```
     sh> . /path/to/mqm/bin/setmqenv -n InstallationX
     ```

   - Windows On Windows:

     ```
     C:\> "C:\Program Files\IBM\MQ\bin\setmqenv" -n InstallationX
     ```

2. Record the IBM MQ version and maintenance level.

   You can use the **dspmqver** command to display these details. For more information, see Displaying the IBM MQ version. If you are collecting troubleshooting information for an AMS, channel, data conversion, dead-letter queue, error message and FFST, security, or TLS channel problem, record the version and maintenance level on both sides of the channel. Alternatively, collect the IBM MQ data manually on both sides of the channel.
3. Record the Operating System version and maintenance level.

   If you are collecting troubleshooting information for an AMS, channel, data conversion, dead-letter queue, error message and FFST, security, or TLS channel problem, record this information for both sides of the channel.
4. If you are collecting troubleshooting information for an AMS, channel, data conversion, dead-letter queue, error message and FFST, security, or TLS channel problem, record the IP addresses and hostnames of the systems on both sides of the channel.
5. Save the IBM MQ configuration information, for example, registry keys and `.ini` files.
6. If your system has more than one IBM MQ installation, use the **dspmqinst** command to record your IBM MQ installation details:
   - Linux UNIX On UNIX and Linux:

     ```
     sh> dspmqinst > /tmp/dspmqinst.txt
     ```

- Windows On Windows:

```
C:\>dspmqinst > %TEMP%/dspmqinst.txt
```

7. On IBM MQ server installations, use the **dspmq** command to record the status of your queue managers.
   This step does not apply to hang and high cpu, publish/subscribe, or triggering problems.
   - Linux UNIX On UNIX and Linux:

```
sh> dspmq -a > /tmp/dspmq.txt
```

   - Windows On Windows:

```
C:\> dspmq -a > %TEMP%/dspmq.txt
```

   - IBM i On IBM i (command line):

```
===> WRKMQM
```

   - IBM i On IBM i (Qshell):

```
===> /QSYS.LIB/QMQM.LIB/DSPMQ.PGM -a > /tmp/dspmq.txt
```

8. On IBM MQ server installations, record the IBM MQ processes that are active on your system.
   This step does not apply to triggering problems.
   - Linux UNIX On UNIX and Linux:

```
sh> ps -ef | grep mq > /tmp/ps.txt
```

   - Windows On Windows:

```
C:\> TASKLIST /V > %TEMP%/tasklist.txt
```

   - IBM i On IBM i (command line):

```
===> WRKACTJOB SBS(QMQM)
```

   - IBM i On IBM i (Qshell):

```
===> ps -ef | grep mq > /tmp/ps.txt
```

9. ULW
   For a logging or recovery problem only, collect the following troubleshooting information:
   a) ULW
   On UNIX, Linux, and Windows, list the contents of the queue manager `LogPath` directory.
   For example:
   - Linux UNIX On UNIX and Linux:

```
sh> ls -ltR /var/mqm/log/QMA > /tmp/QMA.logfiles.txt
```

   - Windows On Windows:

```
C:\> DIR /s "C:\ProgramData\IBM\MQ\log\QMA" > %TEMP%/QMA.logfiles.txt
```

   b) ULW

On UNIX, Linux, and Windows, make sure that the file system or disk holding the logs is not full.

For example:

- Linux UNIX On UNIX and Linux:

```
sh> df -k > /tmp/filesystems.txt
```

- Windows On Windows:

```
C:\> DIR C: > %TEMPT/diskuage.txt
```

c) ULW

On UNIX, Linux, and Windows, run the **amqldmpa** program against the queue manager to gather details about the logger.

The command must be run by an IBM MQ administrator and the output file should be in a location to which the queue manager has permission to write. For example:

- Linux UNIX On UNIX and Linux:

```
sh> amqldmpa -m QMA -c H -d 8 -f /tmp/QMA.amqldmpa.logger.txt
```

- Windows On Windows:

```
C:\> amqldmpa -m QMA -c H -d 8 -f %TEMP%\QMA.amqldmpa.logger.txt
```

d) On all systems, run the **amqldmpa** program against the queue manager to gather details about the persistence layer.

The command must be run by an IBM MQ administrator and the output file should be in a location to which the queue manager has permission to write. For example:

- Linux UNIX On UNIX and Linux:

```
sh> amqldmpa -m QMA -c A -d 8 -f /tmp/QMA.amqldmpa.dap.txt
```

- Windows On Windows:

```
C:\> amqldmpa -m QMA -c A -d 8 -f %TEMP%\QMA.amqldmpa.dap.txt
```

- IBM i On IBM i (Qshell):

```
===> /QSYS.LIB/QMQM.LIB/AMQLDMPA.PGM -m QMA -c A -d 8 -f /tmp/QMA.amqldmpa.dap.txt
```

e) ULW

On UNIX, Linux, and Windows, collect the log file header, `amqhlctl.lfh`, which is found in the active subdirectory of the queue manager's LogPath.

For example:

- Linux UNIX On UNIX and Linux:

```
/var/mqm/log/QMA/active/amqhlctl.lfh
```

- Windows On Windows:

```
C:\ProgramData\IBM\MQ\Log\QMA\active\amqhlctl.lfh
```

f) IBM i

On IBM i, find the **Library** attribute from the queue manager's `qm.ini` file and display its contents. For more information about the queue manager library, see Object names on IBM i.

Alternatively, display the library QM* and select your queue manager from the list to display its contents.

- To display the library for a given queue manager, for example QMA:

```
===> WRKLIB LIB(QMQMA)
```

- To display the library for all queue managers:

```
===> WRKLIB LIB(QM*)
```

g) IBM i

On IBM i, use the same **Library** value to work with the queue manager's journals.

Save the output, then use F17 to display attached journal receivers and save the output from those screens as well. For example, to display the journals and journal receivers for queue manager QMA:

```
===> WRKJRNA JRN(QMQMA/AMQAJRN)
```

10. On IBM MQ server installations, use the **dmpmqcfg** command to record the queue manager configuration:

    This step does not apply to logging or recovery problems.

    - Linux UNIX On UNIX and Linux:

      ```
      sh> dmpmqcfg -m QMA >/tmp/QMA.config.txt
      ```

    - Windows On Windows:

      ```
      C:\> dmpmqcfg -mQMA>%TEMP%\QMA.config.txt
      ```

    - IBM i On IBM i (Qshell):

      ```
      ===> /QSYS.LIB/QMQM.LIB/DMPMQCFG.PGM -mQMA > /tmp/QMA.config.txt
      ```

11. On IBM MQ server installations, use the **runmqsc** command to record status information from the queue manager. For more information, see Saving IBM MQ MQSC output.

    This step does not apply to logging or recovery problems.

    If any command returns an error, carry on with the others:

```
DISPLAY PUBSUB ALL
DISPLAY QMSTATUS ALL
DISPLAY CHSTATUS(*) ALL
DISPLAY LSSTATUS(*) ALL
DISPLAY SVSTATUS(*) ALL
DISPLAY SBSTATUS(*) ALL
DISPLAY CONN(*) TYPE(*) ALL
DISPLAY QSTATUS(*) TYPE(QUEUE) ALL
DISPLAY QSTATUS(*) TYPE(HANDLE) ALL
DISPLAY TPSTATUS('#') TYPE(PUB) ALL
DISPLAY TPSTATUS('#') TYPE(SUB) ALL
DISPLAY TPSTATUS('#') TYPE(TOPIC) ALL
```

12. For an IBM MQ clustering or hang and high CPU problem only, record information about the cluster objects known to the queue manager.

    For an IBM MQ clustering problem, also dump the contents of the cluster repository cache.

    a) Use the **runmqsc** command to record information about cluster objects known to the queue manager.

    If any command returns an error, carry on with the others:

```
DISPLAY CLUSQMGR(*) ALL
DISPLAY QCLUSTER(*) ALL
DISPLAY TCLUSTER(*) ALL
```

b) Dump the contents of the cluster repository cache using the **amqrfdm** utility.

Be sure to use the correct input file for your platform. For example:

- Linux UNIX To dump the cluster repository cache for queue manager QMA on UNIX and Linux:

```
sh> amqrfdm -m QMA < cluster-unix.txt > /tmp/QMA.cluster.txt
```

- Windows To dump the cluster repository cache for queue manager QMA on Windows:

```
C:\> amqrfdm -m QMA < %TEMP%\cluster-win.txt > %TEMP%\QMA.cluster.txt
```

- IBM i To dump the cluster repository cache for queue manager QMA on IBM i (Qshell):

```
===> /QSYS.LIB/QMQM.LIB/AMQRFDM.PGM -m QMA < cluster-IBMi.txt > /tmp/QMA.cluster.txt
```

13. For problems with publish/subscribe only, complete the following steps:

a) On all systems, run the **amqldmpa** program against the queue manager to gather details about the topics.

The command must be run by an IBM MQ administrator and the output file should be in a location to which the queue manager has permission to write. For example:

- Linux UNIX On UNIX and Linux:

```
sh> amqldmpa -m QMA -c T -d 8 -f /tmp/QMA.amqldmpa.topic.txt
```

- Windows On Windows:

```
C:\> amqldmpa -m QMA -c T -d 8 -f %TEMP%\QMA.amqldmpa.topic.txt
```

- IBM i On IBM i (Qshell):

```
===> /QSYS.LIB/QMQM.LIB/AMQLDMPA.PGM -m QMA -c T -d 8 -f /tmp/QMA.amqldmpa.topic.txt
```

b) If your system has queued publish/subscribe enabled, browse the publish subscribe system queues by using a program like the **amqsbcg** sample.

For example:

```
amqsbcg SYSTEM.PENDING.DATA.QUEUE QMA > QMA.PENDING.DATA.browse.txt
amqsbcg SYSTEM.JMS.ND.SUBSCRIBER.QUEUE QMA > QMA.JMS.ND.SUB.browse.txt
amqsbcg SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE QMA > QMA.JMS.ND.CC.SUB.browse.txt
amqsbcg SYSTEM.JMS.D.SUBSCRIBER.QUEUE QMA > QMA.JMS.D.SUB.browse.txt
amqsbcg SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE QMA > QMA.JMS.D.CC.SUB.browse.txt
```

14. For a problem where a channel or client application is having difficulty connecting, use your operating system tools to list network connections on both sides immediately before and after the connection attempt.

This step applies to collecting troubleshooting information for the following types of problem: AMS, channel, client application, data conversion, dead-letter queue, error message and FFST, Java and JMS, security, or TLS channel.

- Linux UNIX To display network connections on UNIX and Linux:

```
sh> netstat -an
```

- Windows To display network connections on Windows:

```
C:\>NETSTAT -AN
```

- IBM i To display IPv4 and IPv6 network connections at the IBM i command line:

```
===> NETSTAT OPTION(*CNN)
===> NETSTAT OPTION(*CNN6)
```

15. Manually package your files for IBM:
    - Linux UNIX "Manually packaging information on UNIX and Linux" on page 320
    - Windows "Manually packaging information on Windows" on page 321
    - IBM i "Manually packaging information on IBM i" on page 322

Related tasks

"Collecting troubleshooting information automatically with runmqras" on page 311
If you need to collect troubleshooting information for a support case, you can use the **runmqras** command to gather the information together into a single archive.

"Sending troubleshooting information to IBM" on page 404
After you have generated and collected troubleshooting information for a problem, you can send it to IBM to help with problem determination for a support case.

Linux UNIX *Manually packaging information on UNIX and Linux*

On UNIX and Linux, you first select a directory with enough free space to hold all the data that you need to collect. You then add the required files to a compressed file with a name beginning with your IBM case number.

Procedure

1. Find a directory with enough free space to hold all the IBM MQ data.

   The contents of the `/var/mqm/errors` and `/var/mqm/trace` directories typically make up most of the IBM MQ data, so check the disk usage of those directories against the free space in your file systems using the **du** (disk usage) and **df** (display file systems) commands. For example:

```
sh> du -sk /var/mqm/errors /var/mqm/trace
384     /var/mqm/errors
189496  /var/mqm/trace

sh> df -k
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           393216    256536   35%     8641    12% /
/dev/hd2          8257536   1072040   88%    70803    21% /usr
/dev/hd9var        393216    126792   68%     6694    16% /var
/dev/hd3         12582912  12441980   99%     5108     2% /tmp
/dev/hd1          1310720    162560   88%      439     2% /home
/proc                   -         -    -        -     -  /proc
/dev/hd10opt       7208960     97180   99%    64796    65% /opt
/dev/fslv00       16777216  15405312    9%    12415     1% /var/mqm
```

2. In the directory you chose, create a new tar file whose name begins with your IBM case number and add the contents of the IBM MQ `errors` directory to it.

   For example:

```
sh> tar -cf /tmp/TS001234567-mqdata.tar /var/mqm/errors
```

3. Add the IBM MQ configuration files to the tar file. Include the `mqinst.ini` file only if you have installed IBM WebSphere MQ 7.1 or later on the system:

```
sh> tar -uf /tmp/TS001234567-mqdata.tar /var/mqm/mqs.ini /etc/opt/mqm/mqinst.ini
```

4. Add the IBM MQ configuration files and error logs for your queue managers.

   For example:

```
sh> tar -uf /tmp/TS001234567-mqdata.tar /var/mqm/qmgrs/QMA/qm.ini /var/mqm/qmgrs/QMA/errors/
*.LOG
```

5. Add any additional files as shown in "Collecting troubleshooting information on Multiplatforms" on page 310 and as requested by IBM Support, including files that contain output from IBM MQ and system commands.

   For example:

```
sh> tar -uf /tmp/TS001234567-mqdata.tar /tmp/ps.txt /tmp/ipcs.txt /tmp/mqconfig.txt
```

6. If you gathered an IBM MQ trace, add the trace files last of all:

```
sh> tar -uf /tmp/TS001234567-mqdata.tar /var/mqm/trace
```

7. Compress the tar file using any available compression tool on your system.

   For example:

   - Using **compress**: creates a `.tar.Z` file

```
sh> compress /tmp/TS001234567-mqdata.tar
```

   - Using **gzip**: creates a `.tar.gz.` file

```
sh> gzip /tmp/TS001234567-mqdata.tar
```

   - Using **bzip2**: creates a `.tar.bz2` file

```
sh> bzip2 /tmp/TS001234567-mqdata.tar
```

8. After sending your data to IBM as described in "Sending troubleshooting information to IBM" on page 404, take a back up copy of your file to keep until your case is resolved then delete the file from the system to save space.

```
sh> rm /tmp/TS001234567-mqdata.*
```

Windows *Manually packaging information on Windows*

On Windows, you first select a directory in which to package the IBM MQ files. You then add the required files to a compressed folder with a name beginning with your IBM case number.

About this task

There are a number of third party utilities for creating archives on Windows. Feel free to use one of these if you wish, but be sure to include the case number at the beginning of the file name, for example `TS001234567-mqdata.zip`. The following instructions demonstrate how to package files using only the capabilities of Windows.

Procedure

1. Open the Windows Explorer and navigate to a directory where you will package the IBM MQ files.

   For example, if you want to do so in your personal temporary directory, you can enter %TEMP% in the Windows Explorer location bar.

   Right-click in the directory and choose **New > Compressed (zipped) Folder**. Include the case number at the beginning of the file name, for example `TS001234567-mqdata`. Windows automatically adds the `.zip` extension.

2. Open a second Windows Explorer window and use it to find the Windows directories and files you wish to include.

Most IBM MQ files will be located under a directory identified by the "WorkPath" registry key. To determine this directory, use the amquregn program shipped with Windows and ignore the double backslash characters in the path it returns:

```
C:\Program Files\IBM\MQ\bin> amquregn amquregn.ctl | FINDSTR WorkPath
    .."WorkPath"="C:\\ProgramData\\IBM\\MQ"
    ...."WorkPath"="C:\\ProgramData\\IBM\\MQ"
```

If your system consists of a new installation of IBM MQ 8.0, the WorkPath might point to a directory under `C:\ProgramData` rather than `C:\Program Files (x86)`. Windows hides the `C:\ProgramData` directory by default, so you must enter %PROGRAMDATA% in the Windows Explorer location bar to navigate to that directory. Alternatively, you can modify your personal settings in the Control Panel so that the Windows Explorer will show hidden files.

3. Add a directory or file by dragging it on top of the new compressed folder. Start by including the top-level IBM MQ `errors` directory.
4. If your system has only IBM WebSphere MQ 7.1 or later installed, drag the IBM MQ `.ini` files to the compressed folder.
5. Drag the IBM MQ configuration files and error logs for your queue managers to the compressed folder.
6. Add any additional files as shown in “Collecting troubleshooting information on Multiplatforms” on page 310 and as requested by IBM Support, including files that contain output from IBM MQ and system commands.
7. If you gathered an IBM MQ trace, add the trace files last of all.
8. After sending your data to IBM as described in “Sending troubleshooting information to IBM” on page 404, take a back up copy of your file to keep until your case is resolved then use the Windows Explorer to delete the file from the system to save space.

IBM i *Manually packaging information on IBM i*

On IBM i, you package the IBM MQ files by running commands at the IBM i command line. You need to include your IBM case number at the beginning of each save file name.

Procedure

1. Create a save file containing the top-level IBM MQ configuration files and errors directory, which might include IBM MQ FFST files, error logs, and JOB files:

```
===> CRTSAVF FILE(QGPL/P12345A) TEXT('Top-level files for PMR 12345,67R,890')
===> SAV DEV('/QSYS.LIB/QGPL.LIB/P12345A.FILE') OBJ(('/QIBM/UserData/mqm/*.ini' *INCLUDE) ('/
QIBM/UserData/mqm/errors/*' *INCLUDE)) DTACPR(*MEDIUM)
```

2. Create a save file which includes the `qm.ini` file and error logs of any queue managers involved in the problem.

   For example:

```
===> CRTSAVF FILE(QGPL/P12345B) TEXT('QMB files for PMR 12345,67R,890')
===> SAV DEV('/QSYS.LIB/QGPL.LIB/P12345B.FILE') OBJ(('/QIBM/UserData/mqm/qmgrs/QMB/qm.ini'
*INCLUDE) ('/QIBM/UserData/mqm/qmgrs/QMB/errors/*' *INCLUDE))
===> CRTSAVF FILE(QGPL/P12345C) TEXT('QMC files for PMR 12345,67R,890')
===> SAV DEV('/QSYS.LIB/QGPL.LIB/P12345C.FILE') OBJ(('/QIBM/UserData/mqm/qmgrs/QMC/qm.ini'
*INCLUDE) ('/QIBM/UserData/mqm/qmgrs/QMC/errors/*' *INCLUDE))
```

3. Create a save file which includes the system history log:

   a) First, create a data base file:

```
===> CRTPF FILE(QGPL/QHIST) RCDLEN(132) MAXMBRS(*NOMAX) SIZE(10000 1000 100)
```

   b) Display the system history log for the period you want to show. For example:

```
===> DSPLOG PERIOD(('12:00:00' '05/16/2014') ('23:59:59' '05/30/2014')) OUTPUT(*PRINT)
```

   c) Work with spool files to find the QPDSPLOG history log information:

```
===> WRKSPLF
```

d) Copy the history log spool file into the data base file.

For example:

```
===> CPYSPLF FILE(QPDSPLOG) TOFILE(QGPL/QHIST) TOMBR(HISTORY)
```

e) Create a save file and save the data base file to it:

```
===> CRTSAVF FILE(QGPL/P12345H) TEXT('History log for PMR 12345,67R,890')
===> SAVOBJ OBJ(QHIST) LIB(QGPL) DEV(*SAVF) SAVF(QGPL/P12345H)
```

4. Create a save file which includes the IBM MQ job logs:

a) First, create a data base file:

```
===> CRTPF FILE(QGPL/JOBLOGS) RCDLEN(132) MAXMBRS(*NOMAX) SIZE(10000 1000 100)
```

b) Work with the QMQM spool files, then press F11 twice to get the job log information (File Nbr, Job, User, and Number, listed in that order on the screen):

```
===> WRKSPLF SELECT(QMQM)
```

c) Copy each job log into the data base file. The **JOB** parameter for each job log should consist of the values Number/User/Job, while the **SPLNBR** parameter should contain just the File Nbr value.

For example:

```
===> CPYSPLF FILE(QPJOBLOG) TOFILE(QGPL/JOBLOGS) JOB(135383/QMQM/RUNMQCHL) SPLNBR(1)
===> CPYSPLF FILE(QPJOBLOG) TOFILE(QGPL/JOBLOGS) JOB(135534/QMQM/AMQZXMA0) SPLNBR(1)
     ...
```

d) Create a save file and save the data base file to it.

For example:

```
===> CRTSAVF FILE(QGPL/P12345J) TEXT('Job logs for PMR 12345,67R,890')
===> SAVOBJ OBJ(JOBLOGS) LIB(QGPL) DEV(*SAVF) SAVF(QGPL/P12345J)
```

5. Create a save file that includes the trace files, if you generated a trace:

```
===> CRTSAVF FILE(QGPL/P12345T) TEXT('Trace files for PMR 12345,67R,890')
===> SAV DEV('/QSYS.LIB/QGPL.LIB/P12345T.FILE') OBJ(('/QIBM/UserData/mqm/trace/*' *INCLUDE)
DTACPR(*MEDIUM)
```

6. Add any additional files as shown in “Collecting troubleshooting information on Multiplatforms” on page 310 and as requested by IBM Support, including files that contain output from IBM MQ and system commands.

```
===> CRTSAVF FILE(QGPL/P12345X) TEXT('Extra files for PMR 12345,67R,890')
===> SAV DEV('/QSYS.LIB/QGPL.LIB/P12345X.FILE') OBJ(('/tmp/QMA.mqsc.txt' *INCLUDE) ('/tmp/
ipcs.txt' *INCLUDE))
```

7. While sending your data to IBM as described in “Sending troubleshooting information to IBM” on page 404, be sure to rename the files so that they contain your full problem record number, for example from P12345A to P12345,67R,890A.SAVF and so on. This is necessary because IBM i libraries limit names to only ten characters, but the IBM ECuRep site needs the full PMR number to associate files with your problem record.
8. After sending your data to IBM, take back up a copy of your save files to keep until your case is resolved then delete the save files using **WRKOBJ** option 4 to save space.

```
===> WRKOBJ OBJ(QGPL/P12345*)
```

Multi MQ Adv. Collecting information for AMS problems on Multiplatforms

If you need assistance to resolve a problem with AMS on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What AMS error did you observe on the system?
- What is the detailed AMS message flow?
- How is AMS implemented in your design? (client-side AMS or channel, MCA interception AMS)?
- What time did the AMS problem start and when did it stop?
- Which specific users or applications and queue manager queues are involved? The IBM MQ security policy, `keystore.conf` file, and certificate keystores are important for AMS to work. Provide details on how these files are set up.
- Provide the type and full version of the IBM MQ client.

About this task

If the AMS problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

Generate the troubleshooting information.

1. Generate a trace of the queue manager in which the security problem occurs.

   If client-side AMS is also implemented, an IBM MQ client trace might also be needed.

   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433

2. Display information about the AMS security policy, `keystore.conf` file and keystores involved.

   a) Display the AMS security policies.

   Run the **dspmqspl** command as shown in the following example:

   ```
   dspmqspl -m QMGRNAME
   ```

   where *QMGRNAME* is the name of the queue manager in which the problem occurs.

   b) Provide a detailed file listing showing the `keystore.conf` and the certificate keystores.

   The default location for the `keystore.conf` file is the user's home `.mqs` directory. If your `keystore.conf` file is located elsewhere, show this location, and explain how you are telling IBM MQ to find the `keystore.conf` file.

   Linux UNIX On UNIX and Linux, use the following command:

   ```
   ls -alR ~/.mqs
   ```

   c) Provide the contents of the `keystore.conf` file.

d) Provide the full type and version of the IBM MQ client. (If Java is used, provide the Java version details also.)

e) Provide a listing of certificates and certificate details for the AMS keystores involved.

- To list the labels of the certificates in a CMS or PKCS #12 keystore, run the following **runmqakm** command:

```
runmqakm -cert -list -db keystorefilename -pw keystorepassword
```

- To list the labels of the certificates in a JKS keystore, run the following **runmqktool** command:

```
runmqktool -list -keystore keystorefilename
```

- To show the details of all certificates in a CMS or PKCS #12 keystore, run the following **runmqakm** command for each certificate label:

```
runmqakm -cert -details -db keystorefilename -pw keystorepassword -label labelname
```

- To show the details of all certificates in a JKS keystore, run the following **runmqktool** command for each certificate label:

```
runmqktool -list -keystore keystorefilename -alias labelname -v
```

Update the case and collect the troubleshooting information.

3. Update the case with your answers to the initial questions.

   Place the outputs/ information from Step 1 directly in the top-level IBM MQ errors directory. Both the **runmqras** automation tool and the manual collection steps below collect files found there.

4. Collect the IBM MQ troubleshooting information.

   You can do this either automatically or manually.

   - Collect the troubleshooting information automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs`, `logger` and `trace` sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,logger,trace -qmlist QMA -caseno TS123456789
     ```

   - Alternatively, collect the troubleshooting information manually as described in "Collecting troubleshooting information manually" on page 315.

Send the troubleshooting information to IBM.

5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting AMS problems" on page 69

Troubleshooting information to help you identify and resolve problems relating to Advanced Message Security (AMS).

Collecting information for channel problems on Multiplatforms

If you need assistance to resolve a problem when an IBM MQ channel is reporting a problem or failing to run on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What channel problem did you observe on the system?
- What time did the channel problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?

About this task

If the channel problem is happening right now, or if you can reproduce the problem, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

For more information about troubleshooting channel problems, see Troubleshooting MQ Channels.

Procedure

1. Generate a trace of the queue manager while the channel problem is happening:
   - Linux UNIX Linux and UNIX
   - Windows Windows
   - IBM i IBM i
2. Generate IBM MQ trace simultaneously at the other end of the channel, whether it is a remote queue manager, a native client application, or a JMS or Java client:
   - Linux UNIX Linux and UNIX
   - Windows Windows
   - IBM i IBM i
   - Java and JMS client
   - z/OS z/OS CHIN trace
3. Linux UNIX
   On UNIX and Linux systems, save the output from the **mqconfig** command.
4. Collect the IBM MQ data.
   You can do this either automatically or manually:
   - Collect the data automatically by using the **runmqras** command as described in “Collecting troubleshooting information automatically with runmqras” on page 311. Be sure to collect the **runmqras** `defs` and `trace` (if the issue was traced) sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```

- Alternatively, collect the data manually as described in “Collecting troubleshooting information manually” on page 315.

5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

“Troubleshooting distributed queue management problems” on page 78
Troubleshooting information to help you solve problems relating to distributed queue management (DQM).

Multi Collecting information for client application problems on Multiplatforms

If you need assistance to resolve a problem with an IBM MQ C, C++, COBOL, .NET, pTAL, RPG or Visual Basic client application on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What client application problem did you observe on the system?
- What time did the client application problem start and when did it stop?
- What is the client application name, and to which queue manager does it connect?
- Which SVRCONN channel, queues, and other objects does the client application use?

About this task

If the client application problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the client application while the problem is happening:
   - Linux UNIX “Tracing on AIX and Linux” on page 427
   - Windows “Tracing on Windows” on page 443
   - IBM i “Tracing on IBM i” on page 433
2. If the client application is receiving an unexpected error from a remote queue manager, generate a simultaneous IBM MQ trace of that queue manager:
   - Linux UNIX “Tracing on AIX and Linux” on page 427
   - Windows “Tracing on Windows” on page 443
   - IBM i “Tracing on IBM i” on page 433

3. Linux UNIX
   On Linux and UNIX systems, save the output from the **mqconfig** command and place this **mqconfig** data directly in the top-level IBM MQ errors directory.

   The automatic and manual data collection processes in Step "4" on page 328 both collect files found in this directory.
4. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs` and `trace` (if the issue was traced) sections, and to specify your case number as shown in the following example for collecting output from queue manager QMA:

     ```
     runmqras -section defs,trace -qmlist QMA -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Multi Collecting information for IBM MQ clustering problems on Multiplatforms

If you need assistance to resolve an issue when an IBM MQ queue manager has a problem with cluster queues, topics or channels on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What IBM MQ clustering problem did you observe on the system?
- What time did the IBM MQ clustering problem start and when did it stop?
- What does your cluster topology look like, and where are the full repositories?
- Which cluster queue managers, channels, queues and topics are involved in the problem?

About this task

If the IBM MQ clustering problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager while the IBM MQ clustering problem is happening:
   - Linux UNIX "Tracing on AIX and Linux" on page 427

- Windows "Tracing on Windows" on page 443
- IBM i "Tracing on IBM i" on page 433

2. If the problem involves other queue managers in the cluster, such as the cluster full repositories, generate IBM MQ trace simultaneously on those queue managers:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
3. Linux UNIX
   On Linux and UNIX systems, save the output from the **mqconfig** command and place this **mqconfig** data directly in the top-level IBM MQ errors directory.

   The automatic and manual data collection processes in Step "4" on page 329 both collect files found in this directory.
4. Collect the IBM MQ data.

   You can do this either automatically or manually:
   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs`, `cluster`, and `trace` (if the issue was traced) sections, and to specify your case number as shown in the following example for collecting **runmqras** output from queue managers QMA and REPOS1:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA,REPOS1 -caseno TS001234567
     ```

     The **runmqras** output will include all of your cluster definitions as well as the contents of your cluster repository cache.
   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting queue manager cluster problems" on page 195

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.

Multi Collecting information for data conversion problems on Multiplatforms

If you need assistance to resolve a problem with data conversion on Multiplatforms, you might need to collect additional troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What data conversion problem did you observe on the system?
- What is the MQMD.Format of the message and its original MQMD.CodedCharSetId (CCSID)?
- What is the intended MQMD.CodedCharSetId to which the message should be converted?
- Which specific characters in the message are invalid, and which did you expect to see instead?

About this task

If the data conversion problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Browse the message immediately after it has been put to an IBM MQ queue using a sample program such as amqsbcg.

   It is important to see the message in hexadecimal in order to examine the MQMD header and the byte values of the message data. For example, to browse messages on a queue called 'Target.Queue' on queue manager called 'QMA', enter this command:

   ```
   amqsbcg Source.Queue QMA > Source.Queue.browse.txt
   ```

2. Generate a trace of the queue manager while the application is putting the message:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
3. If the message contents are corrupted while flowing over an IBM MQ channel with **CONVERT(YES)**, generate trace of the queue manager while the message is flowing across the sending channel:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
4. Browse the message using a sample program such as amqsbcg just before it is retrieved by the target application.

   For example, to browse messages on a queue called 'Target.Queue' on queue manager called 'QMA', enter this command:

   ```
   amqsbcg Target.Queue QMA > Target.Queue.browse.txt
   ```

5. If the message contents are corrupted when the target application gets the message, generate a trace of the queue manager while the application is getting the message:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
6. Collect the IBM MQ data.

   You can do this either automatically or manually:
   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311 to collect the data for both sides of the channel. Be sure to collect the **runmqras** `defs` and `trace` sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```
   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
7. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting message problems" on page 177

Multi Collecting information for dead-letter queue problems on Multiplatforms

If you need assistance to resolve a problem where an IBM MQ queue manager is placing messages on its dead-letter queue (DLQ) on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What dead-letter queue problem did you observe on the system?
- What time did the dead-letter queue problem start and when did it stop?
- Where are the dead-letter messages coming from, and what is their intended route?

About this task

If the messages are going to the dead-letter queue right now, or if you can reproduce the problem that causes the messages to go there, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager while messages are going to the dead-letter queue:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
2. Browse the messages on the dead-letter queue using a sample program such as `amqsbcg` just before it is retrieved by the target application.

   For example, to browse messages on a queue called 'Target.Queue' on queue manager called 'QMA', enter this command:

   ```
   amqsbcg Target.Queue QMA > Target.Queue.browse.txt
   ```

   Place the browse output file, that is `QMA.DLQ.browse.txt` directly in the high-level error log directory, that is:

   - Linux `var/mqm/errors` on Linux.
   - Windows *MQ_INSTALLATION_PATH*`\errors` on Windows.

   Both the automatic and the manual collection processes described in Step 3 collect files found in this directory.
3. Collect the IBM MQ data.

   You can collect do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs`, `cluster`, and `trace` sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```
   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
4. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting message problems" on page 177

Multi Collecting information for error message and FFST problems on Multiplatforms

If you need assistance to resolve a problem when IBM MQ is logging error messages or writing FFSTs (FDC files) on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What unexpected error messages or FFSTs did you observe on the system?
- What time did the error messages or FFSTs start and when did they stop?
- Were there any changes made to the system before the problem started?

About this task

If the error message or FFST problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager while error messages or FFSTs are being logged. Consider generating a high detail trace if you have plenty of disk space.
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
2. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs` and `trace` sections, and to specify your case number as shown in the following example for collecting **runmqras** output from queue manager QMA:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
3. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Multi Collecting information for hang and high CPU problems on Multiplatforms

If you need assistance to resolve a problem with IBM MQ performance, hanging or excessively high CPU usage on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What performance problem or hang did you observe on the system?
- What time did the problem start and when did it stop?
- Which processes were involved in the performance problem or hang?
- Were there any recent changes to the system or to your applications before the problem?

About this task

In order to identify the cause of the problem, it is essential to gather information from the system when the performance problem or hang is happening, including stack dumps and other debugging data from the queue managers and applications that are showing the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate data from Managed File Transfer processes:

   Generate three javacores from the hanging Managed File Transfer process, delaying approximately one minute between each one.

   a) Generate three agent javacores with the **fteSetAgentTraceLevel** command as shown in the following example:

   Linux UNIX
   ```
   fteSetAgentTraceLevel -jc AGENTNAME
   ...
   fteSetAgentTraceLevel -jc AGENTNAME
   ...
   fteSetAgentTraceLevel -jc AGENTNAME
   ```

   where *AGENTNAME* is the name of the Managed File Transfer agent that is hanging.

   b) Generate three logger javacores with the **fteSetLoggerTraceLevel** command as shown in the following example:

   Linux UNIX
   ```
   fteSetLoggerTraceLevel -jc LOGGERNAME
   ...
   fteSetLoggerTraceLevel -jc LOGGERNAME
   ...
   fteSetLoggerTraceLevel -jc LOGGERNAME
   ```

   where *LOGGERNAME* is the name of the Managed File Transfer logger that is hanging.

   The javacores generated by this method are stored in the Managed File Transfer data directory, based on the coordination queue manager name and the agent name. For example:

   Linux UNIX **On UNIX and Linux**

   `/var/mqm/mqft/logs/COORDQMNAME/loggers/LOGGERNAME`

   `/var/mqm/mqft/logs/COORDQMNAME/agents/AGENTNAME`

   Windows **On Windows**

   `C:\Program Files\IBM\MQ\mqft\logs\COORDQMNAME\agents\AGENTNAME`

`C:\Program Files\IBM\MQ\mqft\logs\`*COORDQMNAME*`\loggers\`*LOGGERNAME*

This location might vary, depending on which version of IBM MQ you are using. For more information, see Program and data directory locations on Windows.

In these examples, *AGENTNAME* or *LOGGERNAME* is the name of the Managed File Transfer agent or logger that is hanging, and *COORDQMNAME* is the name of the coordination queue manager.

2. For all other Managed File Transfer commands, generate three javacores from the process as shown in the following examples.

   The javacores or thread dumps in this case are typically written to the working directory of the command.

   a) Linux UNIX

   On UNIX and Linux, list the Java virtual machines that are using **ps** and find the one that is running the hanging Managed File Transfer command. Then send **SIGQUIT** to that process identifier (PID) to generate a javacore or thread dump.

   The **kill -QUIT** command does not terminate Java virtual machines on UNIX and Linux, but instead makes them create a javacore or thread dump. For example:

   ```
   sh> ps -ef | egrep 'PID|StartAgent'
     UID   PID  PPID   C STIME   TTY          TIME CMD
    7001 37789     1   0 Sun03PM ??         3:07.35 java ... com.ibm.wmqfte.api.StartAgent
   AGENT1
    7001 69177 64373   0  2:35PM ttys003    0:00.00 egrep PID|StartAgent
   sh> kill -QUIT 37789
       ...
   sh> kill -QUIT 37789
       ...
   sh> kill -QUIT 37789
   ```

   b) Windows

   On Windows, start the Managed File Transfer command from the Windows command prompt.

   Be sure to add the `-F` option to the **fteStartAgent** and **fteStartLogger** commands so that they will run in the foreground and not in the background or as a Windows service. Then type the Ctrl+Break keyboard sequence to generate a javacore from the process. For example:

   ```
   C:\> fteStartLogger -F LOGGER1
   ...
   Ctrl+Break
   ...
   Ctrl+Break
   ...
   Ctrl+Break
   ```

   c) IBM i

   On IBM i, list the Java virtual machine jobs in the system using WRKJVMJOB option 7 to find the one running the hanging managed file transfer command. Then press F3 to exit and use the job Number, User and Job name to generate a Java thread dump from the job.

   For example:

   ```
   ===> WRKJVMJOB

   Opt  Job Name    User        Number  Function          Status
        QJVACMDSRV  QMQM        136365  PGM-StartAgent     THDW
        QYPSJSVR    QYPSJSVR    136415  PGM-jvmStartPa     SIGW
   ```

   Use option 7 to find the right job and F3 to return to the command line:

   ```
   ===> GENJVMDMP JOB(136365/QMQM/QJVACMDSRV) TYPE(*JAVA)
   ```

3. Linux UNIX

   On UNIX and Linux, use the `stackit` and `sigdump` scripts to generate debugging data from processes.

a) Download the IBM `stackit` and `sigdump` scripts. On Linux systems you must install the GNU debugger (GDB), even if temporarily, for `stackit` to work:

- Linux Download stackit
- Linux Download GDB for Linux

b) Run the `stackit` script three times against the affected IBM MQ queue managers and applications, with a delay of a minute or less between each run.

For example:

```
sh> stackit -m QMA -m QMB -n myapp -f /var/mqm/errors/stackit-1.txt
sh> sleep 30
sh> stackit -m QMA -m QMB -n myapp -f /var/mqm/errors/stackit-2.txt
sh> sleep 30
sh> stackit -m QMA -m QMB -n myapp -f /var/mqm/errors/stackit-3.txt
```

c) Run the `sigdump` script once against the affected IBM MQ queue managers. The `sigdump` script will cause each queue manager to generate diagnostic FFST files.

For example:

```
sh> sigdump -m QMA -m QMB
```

4. Windows

On Windows, generate debugging from processes by using debugging utilities.

a) Download the following debugging utilities from Microsoft if you do not have them on your system:

- The latest version of the debugging tools for Windows, obtained from Debug Diagnostic Tool
- Download Microsoft PsList
- Download Microsoft Handle
- Download Microsoft Process Monitor

b) Display the list of processes:

```
C:\> tasklist -v
```

c) Display additional information about each process:

```
C:\> pslist -x
```

d) Display information about IBM MQ processes and any affected applications by passing the first few characters of each process name to the handle program, for example:

```
C:\> handle -a -p amq
C:\> handle -a -p runmq
C:\> handle -a -p myapp
```

e) Gather data from hangs (or even crashes) of IBM MQ processes and any affected applications, for example:

```
C:\> adplus -hang -pn amqzxma0.exe
C:\> adplus -hang -pn amqzlaa0.exe
C:\> adplus -crash -pn runmqchi.exe
```

f) Use the Microsoft Process Monitor tool to provide real-time stack data, loaded modules, environment information, files accessed, libraries used, registry keys accessed, and more information.

This tool can be very CPU intensive, even with filtering options set. See the section "Scripting Process Monitor" in the included `procmon.chm` help file for information on using it in a script or batch file.

5. IBM i

On IBM i, generate debugging data from processes by using the MQSTACK and SERVICEDOCS tools:

a) Download and run the IBM MQSTACK tool. MQSTACK will show the status of all threads for all queue manager processes, however it does not show information about non-IBM processes.

b) For processes that are not part of the queue manager, such as application programs, run the SERVICEDOCS utility. SERVICEDOCS will show the stack for the main thread of every process on the system.

6. Generate an IBM MQ trace while the problem is happening:
   - Linux UNIX **Linux and UNIX**
   - Windows **Windows**
   - IBM i **IBM i**

   To avoid worsening the system performance, stop the trace after a short period of time (for example, after a minute or less).
7. If the hang or high CPU usage is happening inside WebSphere Application Server, complete the WebSphere Application Server MustGather instructions for your platform:
   - AIX AIX
   - Linux Linux
   - Windows Windows
   - IBM i IBM i
8. Linux UNIX
   On UNIX and Linux systems, save the output from the **mqconfig** command.
9. Place the following information directly in the top-level IBM MQ errors directory:
   - The debug files that you collected in Step 1.
   - Linux UNIX The output from the **mqconfig** command that you collected in Step 4.

   The automatic and manual data collection processes in Step "10" on page 337 both collect files found in this directory.
10. Collect the IBM MQ data.

    You can do this either automatically or manually:
    - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs`, `cluster`, and `trace` sections, and to specify your case number as shown in the following example:

      ```
      runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
      ```
    - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
11. Send the information that you have collected to IBM.

    A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

    For FTP and email instructions, see Exchanging information with IBM Software Support.

    To open or update a case, go to the IBM My Support site.

    Note: Always update your case to indicate that data was sent.

    If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

“Troubleshooting message problems” on page 177

Multi Collecting information for IBM MQ Explorer problems

If you need assistance to resolve a problem with IBM MQ Explorer when administering a queue manager , you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What IBM MQ Explorer problem did you observe on the system?
- Which queue managers are you trying to administer, and on what systems are they located?
- Which operating system version and IBM MQ version are the remote queue managers running?

From IBM MQ 9.3.0, the IBM MQ Explorer feature is no longer included in the Server product on Windows or Linux x86_64. It is still available as a separate download for those platforms. For more information, see Installing and uninstalling IBM MQ Explorer as a stand-alone application on Linux and Windows.

About this task

IBM MQ Explorer is available for Linux and Windows systems as a standalone installation through Fix Central. IBM MQ Explorer can administer local queue managers where it is installed as well as remote queue managers on all platforms.

It is important to gather information from the IBM MQ Explorer when the problem is happening in order to identify the cause.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate an IBM MQ Explorer trace which shows the problem when you try to use IBM MQ Explorer to administer the queue manager.
2. Generate a trace of the queue manager while the application is putting the message:
   - Linux UNIX “Tracing on AIX and Linux” on page 427
   - Windows “Tracing on Windows” on page 443
   - IBM i “Tracing on IBM i” on page 433
3. If there is a graphical problem in IBM MQ Explorer, take a screen shot or use a camera phone to capture an image of the problem.
4. Collect the IBM MQ data.
   a) Record the MQ Explorer version and maintenance level.
   b) Record the MQ version and maintenance level of the target queue manager.
   c) Record the operating system version and maintenance level where both the IBM MQ Explorer and the target queue manager are running.
   d) If you are using the stand-alone IBM MQ Explorer installed from Fix Central, list the contents of its installation directory, for example:

      Linux
      ```
      sh> ls -alR "/opt/ibm/wmq-explorer"
      ```

      Windows
      ```
      C:\> DIR /S "C:\Program Files\IBM\MQ Explorer"
      ```

Note: The directory name is chosen during installation and might differ from these examples.

e) Find the IBM MQ Explorer `.log` file.

When IBM MQ Explorer encounters an error, it might create a file called just `.log` with more information. Look for the `.log` file in the appropriate directory based on the IBM MQ Explorer installation type and the user who encountered the problem, and collect the `.log` file as well as all other files in the `.metadata` directory. In the following examples, $HOME and %USERPROFILE% are user-specific environment variables used to locate the files.

To find the `.log` file for the stand-alone IBM MQ Explorer:

Linux
```
sh> ls -al "$HOME"/IBM/*MQ/workspace/.metadata/.log
```

Windows
```
C:\> DIR "%USERPROFILE%\IBM\*MQ\workspace\.metadata\.log"
```

f) If IBM MQ Explorer is having difficulty connecting, use your operating system tools to list network connections on both sides immediately before and after the connection attempt:

- Linux UNIX To display network connections on UNIX and Linux:

```
sh> netstat -an
```

- Windows To display network connections on Windows:

```
C:\>NETSTAT -AN
```

g) Manually package your files for IBM:

- Linux "Manually packaging information on UNIX and Linux" on page 320
- Windows "Manually packaging information on Windows" on page 321

5. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

Troubleshooting problems with IBM MQ Explorer

Multi Collecting information for installation and uninstallation problems on Multiplatforms

If you need assistance to resolve a problem when IBM MQ or one of its fix packs is failing to install or uninstall properly on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What are you trying to install or uninstall?
- What account are you using to perform the installation or uninstallation?

About this task

It is helpful to gather information from the system when the installation or uninstallation problem is happening in order to identify the cause.

After collecting the data, you can then send the information that you have collected to IBM.

Procedure

1. Run the installation or uninstallation with debug logging enabled in order to gather more detailed information about the failure:

   a) AIX

   To generate debug installation and uninstallation data on AIX:

   i) Export the environment variable INST_DEBUG=YES, which directs AIX to log extra debugging information. Then run the installation or uninstallation, either through SMIT or by running the **installp** command directly. For example:

   ```
   sh> export INST_DEBUG=YES
   sh> installp...
   ```

   ii) Unset the INST_DEBUG variable when done:

   ```
   sh> unset INST_DEBUG
   ```

   The `smit.log` file, located in the root directory of the system, will contain the debugging information from the installation or uninstallation attempt.

   b) Linux

   To generate debug installation and uninstallation data on Linux, add the `-vv` option to the **rpm** command and capture all output (stdout and stderr) to a file.

   For example:

   ```
   sh> rpm -vv ... 2>&1 | tee mqinstall.log
   ```

   c) Windows

   To generate debug installation and uninstallation data on Windows, use the **msiexec** command with the option **/l*vx** to log debugging output to a file.

   To determine what additional parameters to use to install or uninstall IBM MQ with **msiexec**, see Installing the server using msiexec. For example:

   ```
   C:\> msiexec /l*vx "C:\mqinstall.log" ...
   ```

   d) IBM i

   To generate debug installation and uninstallation data on IBM i, specify the **OUTPUT(*PRINT)** option on the **RSTLICPGM** or **DLTLICPGM** commands to ensure that a job log is spooled.

   For example:

   ```
   ===> RSTLICPGM ... OUTPUT(*PRINT)
   ```

   Then use WRKSPLF option 5 to display the joblog.

2. Collect the IBM MQ data.

   Save the output of any errors reported by the installation or uninstallation process. Take a screen shot of the error, or use a camera phone to capture an image of the problem.

   a) Record the MQ version and maintenance level currently on the system or identify the version you are attempting to install.

   b) Record the operating system version and maintenance level.

   c) If your system has more than one IBM MQ installation, record your IBM MQ installation details:

- Linux UNIX On UNIX and Linux:

```
sh> dspmqinst > /tmp/dspmqinst.txt
```

- Windows On Windows:

```
C:\> dspmqinst > %TEMP%/dspmqinst.txt
```

d) Linux UNIX
On UNIX and Linux systems, include the `/etc/opt/mqm/mqinst.ini` file, if it exists.

e) Windows
On Windows systems, save a copy of the IBM MQ information from the Windows registry information using the amquregn program, if you have an IBM MQ installation available to run it.

f) Record the precise commands that you used to start the installation or uninstallation process.

Linux On Linux systems, include the **crtmqpkg** command that you used to repackage IBM MQ, if you are working with multiple installations.

g) AIX
On AIX systems, collect the `smit.log` and `smit.script` files found in the root directory of the system.

h) Windows
On Windows systems, collect the MSI installer log file. If you used msiexec, then you will have selected the file name on the command line. Otherwise, include all files named `MSI*.*`, `MQ*.*`, and `amq*.*` located in the %TEMP% directory of the user who attempted the installation or uninstallation. Include the files `amqmsccw.txt` and `amqmjpse.txt` from the IBM MQ data directory, if they exist.

i) On all systems, include the `mqpatch.dat` and `mqpatch.log` files from the IBM MQ installation directory, if they exist.

j) On all systems, list the contents (if any) of the directory where you were trying to install, update, or remove IBM MQ. For example:

- Linux UNIX On UNIX and Linux:

```
sh> ls -alR /path/to/mq > mqfiles.txt
```

- Windows On Windows:

```
C:\> DIR /S "C:\Program Files\IBM\MQ" > %TEMP%/mqfile.txt
```

- IBM i On IBM i Qshell:

```
===> ls -alR /QIBM/UserData/mqm /QIBM/ProdData/mqm /QSYS.LIB/QMQM.LIB > /tmp/mqfile.txt
```

k) Manually package your files for IBM, including files containing the output from the commands listed in Steps 1 and 2. For new installations, skip over any directories or files that do not yet exist on the system:

- Linux UNIX "Manually packaging information on UNIX and Linux" on page 320
- Windows "Manually packaging information on Windows" on page 321
- IBM i "Manually packaging information on IBM i" on page 322

3. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting message problems" on page 177

Multi Collecting information for Java and JMS application problems on Multiplatforms

If you need assistance from to resolve a problem with a Java or JMS application on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

IBM recommends using the IBM MQ classes for Java in Java Platform, Enterprise Edition (Java EE) application servers such as WebSphere Application Server. If you are using the IBM MQ classes for Java in a Java EE environment, review the restrictions and other considerations for their usage.

Before you start this task, answer the following questions about the problem:

- What Java or JMS problem did you observe on the system?
- What time did the Java or JMS problem start and when did it stop?
- Were any Java exceptions reported, and did they include a Java call stack?
- Which queue managers, queues and topics does the Java or JMS application use?

About this task

It is essential to gather information from the system when the Java or JMS problem is happening in order to identify the cause.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate an IBM MQ classes for Java trace or an IBM Java Message Service trace, depending on whether your application uses the IBM MQ Java or JMS interface.

   If your application is running under WebSphere Application Server, follow the trace instructions for that environment.
2. Generate a trace of the client application while the problem is happening:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
3. Collect the following information for the Java or JMS application:
   a) If your application is running in WebSphere Application Server, use its collector tool to gather information about the application server and its configuration, JNDI definitions, FFDC files, logs, and any traces generated in Steps 1 and 2:
      - WebSphere Application Server 9.0.5
      - WebSphere Application Server 8.5.5

b) If your application is running in another Java application server or in a Java Platform, Standard Edition (Java SE) environment, collect the following files:

- The standard output stream data (for example, `System.out` or similar files).
- The standard error stream data (for example, `System.err` or similar files).
- The Java virtual machine log files (for example, `native_stdout.log` and `native_stderr.log` or similar files).
- The `mqjms.log` file, found by default in the application's current working directory.
- The `mqjms_PID.trc` file, named for the process ID of the Java virtual machine, found in the same directory.
- Any FFST files found in the FFDC subdirectory of the application's current working directory.

4. Place the Java or JMS traces and logs from Steps 1 to 3, and, where applicable, the WebSphere Application Server collector, in the top-level IBM MQ errors directory.

   The automatic and manual data collection processes in Step "5" on page 343 both collect files found in this directory.

5. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `trace` section, and from queue managers the `defs` and `topic` sections as well, and to specify your case number as shown in the following example for collecting output from queue manager QMA:

     ```
     runmqras -section defs,topic,trace -qmlist QMA -caseno TS001234567
     ```

     To collect output from a client, specify the `trace` section and your case number as shown in the following example:

     ```
     runmqras -section trace -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Multi Collecting information for logging and recovery problems on Multiplatforms

If you need assistance to resolve a problem where an IBM MQ queue manager is reporting errors with logging data or recovering information from its logs on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What logging or recovery problem did you observe on the system?

- What time did the logging or recovery problem start and when did it stop?
- What other details can you provide to help determine the cause of the problem?

About this task

If the logging or recovery problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager while the problem is happening.

   Consider gathering a high detail trace if you have plenty of disk space:

   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433

2. ULW

   On UNIX, Linux, and Windows, dump the contents of the queue manager logs.

   This is particularly useful if you suspect a problem with the amount of data being logged.

   Note: You must stop the queue manager in question in order to dump its logs. You must also provide the log path for the queue manager. The log path is defined with the **LogPath** attribute of the Log stanza of the `qm.ini` file.

   The commands in the following examples use the **dmpmqlog** command to dump the contents of the logs for queue manager QMA:

   - Linux UNIX On UNIX and Linux:

     ```
     sh> endmqm -i QMA
     sh> dmpmqlog -b -m QMA -f /var/mqm/log/QMA > /tmp/QMA.dmpmqlog.txt
     sh> strmqm QMA
     ```

   - Windows On Windows:

     ```
     C:\> endmqm -i QMA
     C:\> dmpmqlog -b -m QMA -f "C:\ProgramData\IBM\MQ\log\QMA" > %TEMP%\QMA.dmpmqlog.txt
     C:\> strmqm QMA
     ```

3. Linux UNIX

   On Linux and UNIX systems, save the output from the **mqconfig** command.

4. Place the output from the **dmpmqlog** command and the **mqconfig** command that you generated in Steps 2 and 3 in the top-level IBM MQ errors directory.

   The automatic and manual data collection processes in Step "5" on page 344 both collect files found in this directory.

5. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `trace` section, and from queue managers the `defs` and `topic` sections as well, and to specify your case number as shown in the following example for collecting output from queue manager QMA:

     ```
     runmqras -section defs,topic,trace -qmlist QMA -caseno TS001234567
     ```

To collect output from a client, specify the `trace` section and your case number as shown in the following example:

```
runmqras -section trace -caseno TS001234567
```

- Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Multi Collecting information for Managed File Transfer problems on Multiplatforms

If you need assistance to resolve a problem when a Managed File Transfer (MFT) agent, logger or command is reporting a problem or failing to work properly on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem. The information that is needed depends on the problem that you are seeing.

Procedure

1. Collect the information that is needed for the type of problem that you are seeing:
   - Managed File Transfer agent problems
   - Managed File Transfer protocol bridge agent problems
   - Managed File Transfer resource monitor problems
   - Managed File Transfer managed transfer problems
   - Managed File Transfer database logger problems
   - Managed File Transfer file logger problems
   - Managed File Transfer command problems
2. After you have collected the Managed File Transfer data that is needed to investigate the problem, create an archive containing all of the relevant files.

   For more information, see "Creating an archive of MFT troubleshooting information" on page 352.
3. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting Managed File Transfer problems" on page 127

Use this information to help you to diagnose errors in Managed File Transfer (MFT).

Multi *Collecting information for MFT agent problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) agent problem.

About this task

Managed File Transfer agent problems include:

- The agent failing to connect to, or being disconnected from, its agent queue manager.
- The agent hanging.
- The agent stopping unexpectedly.
- The agent going into recovery.
- The **fteListAgents** or **fteShowAgentDetails** commands, or the IBM MQ Explorer Managed File Transfer plug-in, showing incorrect, or out of date, status information for the agent.
- The agent failing to report any status information.

Procedure

1. Initially, review the following topics to see whether they help you to resolve the problem:
   - “Troubleshooting MFT agent status problems” on page 128
   - “Troubleshooting java.lang.OutOfMemoryError problems” on page 152
   - “Troubleshooting the Connect:Direct bridge” on page 161
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the agent.
   - The name of the agent queue manager.
   - The version of Managed File Transfer that the agent is using.
   - The version of IBM MQ for the agent queue manager.
   - The installation type for the agent (that is, was the agent installed from the IBM MQ product installation media or via the Managed File Transfer redistributable agent package?).
   - Any error messages that are seen in the agent's event log (`output0.log`) when the issue occurs.
   - An agent trace covering the time of the issue. For more information about how to collect the trace, see “Tracing Managed File Transfer agents on Multiplatforms” on page 490.
     - If the agent is experiencing the problem when communicating with its agent queue manager (for example, the agent's event log contains error messages that include an IBM MQ reason code such as `2009 – MQRC_CONNECTION_BROKEN`), collect the trace using the trace specification `=all`.
     - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
   - Three Javacores taken 30 seconds apart, if the agent is hanging. In order to do this, run the **fteSetAgentTraceLevel** command with the `-jc` option set, as shown in the following example:

     ```
     fteSetAgentTraceLevel -jc <agent_name>
     ```

     If the command does not cause the agent to generate a Javacore, then you should send a SIGQUIT signal to the agent process.
   - An archive containing the agent's log files, configuration files, trace files and Javacores, if applicable. For more information about how to create the archive, see “Creating an archive of MFT troubleshooting information” on page 352.
   - **runmqras** output for the coordination queue manager and agent queue managers. For more information about how to create the output, see “Collecting troubleshooting information automatically with runmqras” on page 311.

Related tasks

"Troubleshooting MFT agent status problems" on page 128
Use these tasks, logs and links to help you to resolve issues with the status of Managed File Transfer (MFT) agents.

Configuring MFT Agent queue managers

Multi *Collecting information for MFT protocol bridge agent problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) protocol bridge agent problem.

About this task

Managed File Transfer protocol bridge agent problems include:

- The agent failing to connect to, or being disconnected from, a remote file server.
- Managed transfers to or from a remote file server failing.

Procedure

1. Initially, review the information in "Troubleshooting a protocol bridge agent that reports a file not found" on page 143 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the protocol bridge agent.
   - The name of the protocol bridge agent queue manager.
   - The version of Managed File Transfer that the protocol bridge agent is using.
   - The version of IBM MQ for the protocol bridge agent queue manager.
   - The hostname of the remote file server system.
   - Product and version information for the remote file server.
   - The protocol that the agent is using to communicate with the remote file server (that is, FTP, FTPS or SFTP).
   - The entry for the remote file server in the protocol bridge agent configuration file (`ProtocolBridgeProperties.xml`).
   - Any error messages that are seen in the agent's event log (`output0.log`) when the issue occurs.
   - A protocol bridge agent log file, where the log level for the protocol being used is set to on. For more information about how to set the log level, see fteSetAgentLogLevel (Turn on or turn off logging to file of certain MFT agent operations).
   - An archive that contains the protocol bridge agent's log files and configuration files. For more information about how to create the archive, see "Creating an archive of MFT troubleshooting information" on page 352.

Multi *Collecting information for MFT resource monitor problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) resource monitor problem.

About this task

Managed File Transfer resource monitor problems include:

- A resource monitor stops polling.
- A resource monitor is polling, and not triggering on any items (either files or messages).
- A resource monitor is not submitting managed transfer requests to the agent.
- A resource monitor stops unexpectedly.

Procedure

1. Initially, review the information in “Troubleshooting resource monitor problems” on page 144 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the agent.
   - The name of the agent queue manager.
   - The version of Managed File Transfer that the agent is using.
   - The version of IBM MQ for the agent queue manager.
   - The name of the resource monitor.
   - The name of the resource (either a queue or directory) that the monitor is polling.
   - The monitor's trigger condition.
   - The monitor's task XML.
   - Details of any items that the monitor is not triggering on.
   - A resource monitor log file (for example, `resmonevent0.log`), where the log level for the resource monitor is set to VERBOSE. For more information about how to create the log file, see Logging MFT resource monitors.

     If the monitor is polling, and has not got stuck, then the log file should include entries for at least three polls.
   - An archive containing the agent's configuration files, and the log files for the agent and resource monitor. For more information about how to create the archive, see “Creating an archive of MFT troubleshooting information” on page 352.

Multi *Collecting information for MFT managed transfer problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) managed transfer problem.

About this task

Problems related to Managed File Transfer managed transfers include:

- A managed transfer failing unexpectedly.
- A managed transfer going into recovery and not completing.
- A managed transfer getting stuck.

Procedure

1. Initially, review the information in “Troubleshooting managed transfer problems” on page 136 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the source agent for the managed transfer.
   - The name of the source agent queue manager.
   - The version of Managed File Transfer or Managed File Transfer for z/OS that the source agent is using.
   - The version of IBM MQ or IBM MQ for z/OS for the source agent queue manager.
   - The name of the destination agent for the managed transfer.
   - The name of the destination agent queue manager.
   - The version of Managed File Transfer or IBM MQ for z/OS that the destination agent is using.
   - The version of IBM MQ or IBM MQ for z/OS for the destination agent queue manager.

- If the source and destination agent queue managers are different, details of how the queue managers are connected together (that is, through sender/receiver channels or an IBM MQ cluster).
- The transfer identifier for the managed transfer.
- Details of how the managed transfer request was created (that is, was it generated by a resource monitor, the **fteCreateTransfer** command, the IBM MQ Explorer Managed File Transfer plug-in, or something else?).
- Details of any error messages that are related to the managed transfer in either the source agent or destination agent's event log (`output0.log`).
- If the source and/or destination agents, are running Managed File Transfer or IBM MQ for z/OS at IBM MQ 9.3 or later, a VERBOSE transfer log that covers the time when the issue occurred. For more information about how create the transfer log, see fteSetAgentLogLevel (Turn on or turn off logging to file of certain MFT agent operations).
- A trace from both the source and destination agents that covers the time when the issue occurred. For more information about how to collect the trace, see "Tracing Managed File Transfer agents on Multiplatforms" on page 490. The trace should be collected using the trace specification `com.ibm.wmqfte=all`.
- An archive from the source agent containing the agent's log files and configuration files and an archive from the destination agent, containing the agent's log files and configuration files. For more information about how to collect the archives for the source and destination agents, see "Creating an archive of MFT troubleshooting information" on page 352.
- **runmqras** output for the source agent queue manager and the destination agent queue manager. For more information about how to collect the **runmqras** output for the source agent queue manager and the destination agent queue manager, see "Collecting troubleshooting information automatically with runmqras" on page 311.

When you are investigating issues related to managed transfers, it is often useful to draw a simple diagram, as shown in the following example, that shows the agents and the agent queue managers. This diagram allows you and IBM Support to see how the agents and the agent queue managers are connected, which can help to identify possible issues within the IBM MQ network that might cause managed transfers to enter recovery or get stuck.

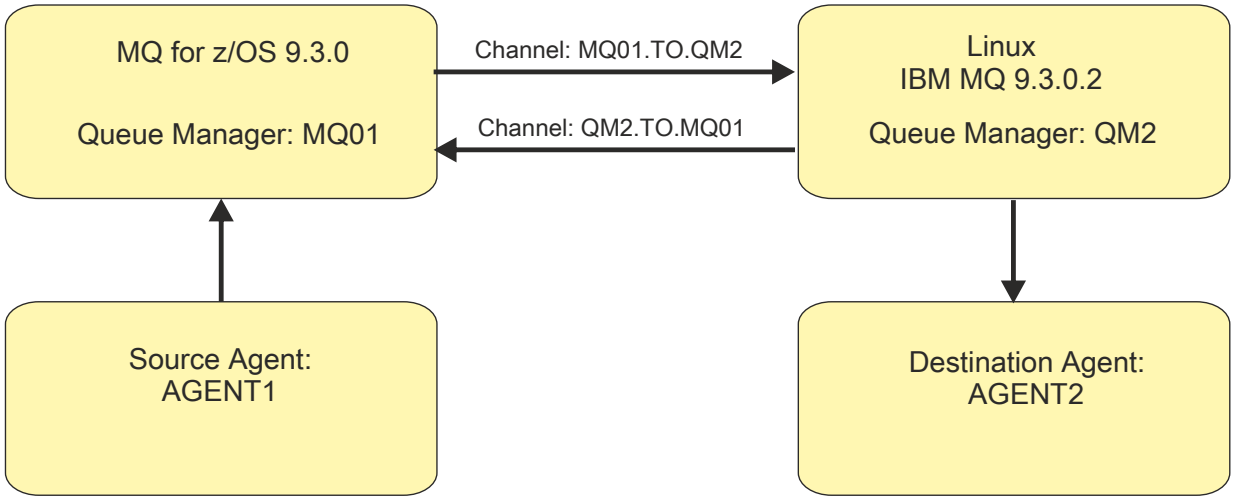


Figure 54. Example of a simple diagram showing how a source agent, AGENT1, and a destination agent, AGENT2, and their agent queue managers, MQ01 and QM2 are connected.

Multi *Collecting information for MFT database logger problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) database logger problem.

About this task

Managed File Transfer database logger problems include:

- The database logger fails to connect to the coordination queue manager.
- The database logger fails to connect to the database.
- The database logger doesn't update the database.

Procedure

1. Initially, review the information in "Troubleshooting logger problems" on page 158 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the database logger.
   - The name of the coordination queue manager that the database logger is connecting to.
   - The version of Managed File Transfer that the database logger is using.
   - The version of IBM MQ for the coordination queue manager.
   - The type of database that the database logger is using.
   - Details of any error messages that appear in the database logger's event log when the issue occurs.
   - A database logger trace that covers the time of the issue. For more information about how to collect this trace, see "Tracing Managed File Transfer standalone loggers on Multiplatforms" on page 494.
     - If the database logger is experiencing the problem when communicating with the coordination queue manager (for example, the database logger's event log contains error messages that include an IBM MQ reason code such as `2009 – MQRC_CONNECTION_BROKEN`), collect the trace using the trace specification `=all`.
     - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
   - An archive that contains the database logger log files and configuration files. For more information about how to create the archive, see "Creating an archive of MFT troubleshooting information" on page 352.
   - **runmqras** output for the coordination queue manager. For more information about how to create the output, see "Collecting troubleshooting information automatically with runmqras" on page 311.

Multi *Collecting information for MFT file logger problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) file logger problem.

About this task

Managed File Transfer file logger problems include:

- The file logger fails to connect to the coordination queue manager.
- The file logger fails to log any data.

Procedure

1. Initially, review the information in "Troubleshooting logger problems" on page 158 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:

- The name of the file logger.
- The name of the coordination queue manager that the file logger is connecting to.
- The version of Managed File Transfer that the file logger is using.
- The version of IBM MQ for the coordination queue manager.
- The type of database that the database logger is using.
- Details of any error messages that appear in the file logger's event log when the issue occurs.
- A file logger trace that covers the time of the issue. For more information about how to collect this trace, see "Tracing Managed File Transfer standalone loggers on Multiplatforms" on page 494.
  - If the file logger is experiencing the problem when communicating with the coordination queue manager (for example, the file logger's event log contains error messages that include an IBM MQ reason code such as `2009 – MQRC_CONNECTION_BROKEN`), collect the trace using the trace specification `=all`.
  - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
- An archive containing the file logger log files and configuration files. For more information about how to create the archive, see "Creating an archive of MFT troubleshooting information" on page 352.
- **`runmqras`** output for the coordination queue manager. For more information about how to create the output, see "Collecting troubleshooting information automatically with runmqras" on page 311.

Multi *Collecting information for MFT command problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) command problem.

About this task

Problems with Managed File Transfer commands include:

- A command failing to connect to the queue manager.
- A command timing out.
- A command reporting an error.

Procedure

To investigate these, provide the following information:

- The command that is being run.
- The username for the user that is logged in when the command is run.
- The output from the command.
- The version of Managed File Transfer that the command is using.
- A trace of the command, covering the time when the issue occurred. For information on how to collect this trace, see "Tracing Managed File Transfer commands on Multiplatforms" on page 492.
  - If the command is experiencing the problem when communicating with a queue manager (for example, the command reports an error containing an IBM MQ reason code), collect the trace using the trace specification `=all`.
  - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
- An archive containing the configuration files on the system where the command is being run. For more information about how to create the archive, see "Creating an archive of MFT troubleshooting information" on page 352.

Multi *Creating an archive of MFT troubleshooting information*

After you have collected the data needed to investigate the Managed File Transfer (MFT) problem that you are seeing, you need to create an archive that contains all of the relevant files and send it in to IBM. You can either create the archive manually, or by using the **fteRAS** utility.

About this task

Warning: If there are a large number of Managed File Transfer agents configured on a system, the **fteRAS** command can take a long time to complete. If that happens, you should create the archive manually by compressing the contents of the Managed File Transfer agent's logs and configuration directories into a zip file.

From IBM MQ 9.4.0, you can collect troubleshooting information for a specific Managed File Transfer agent or agents instead of for all agents on the system. You do this by running the **fteRAS** command with the **-agents** parameter.

Procedure

- To archive the Managed File Transfer files automatically using the **fteRAS** command:
  - Linux UNIX On UNIX and Linux, copy any interactive command traces and javacores to the /var/mqm/errors directory and then run the **fteRAS** command as shown in the following example:

    ```
    sh> fteRAS /var/mqm/errors
    ...
    BFGCL0604I: fteRAS command completed successfully. Output is stored in /var/mqm/errors/
    fteRAS.zip
    ```

  - Windows On Windows, copy any interactive command traces and javacores to the top-level IBM MQ errors directory. The actual path name of this directory depends on which version of IBM MQ you are using. For more information, see Program and data directory locations on Windows. Run the **fteRAS** command with the correct path name for your system, for example:

    ```
    C:\> fteRAS "C:\ProgramData\IBM\MQ\errors"
    ...
    BFGCL0604I: fteRAS command completed successfully. Output is stored in
    C:\ProgramData\IBM\MQ\errors\fteRAS.zip
    ```

  - IBM i On IBM i, copy any interactive command traces and javacores you created (that is, spool files from the **GENJVMDMP** command) to `/QIBM/UserData/mqm/errors` and then run the **fteRAS** command from the Qshell as shown in the following example:

    ```
    ===> /QIBM/ProdData/mqm/bin/fteRAS /QIBM/UserData/mqm/errors
    ...
    BFGCL0604I: fteRAS command completed successfully. Output is stored in /QIBM/UserData/mqm/
    errors/fteRAS.zip
    ```

- To archive the Managed File Transfer files manually:
  - Linux UNIX On UNIX and Linux, copy the agent and logger javacores, traces, logs, properties and FFST files. Include any interactive command traces and javacores written to the current directory or to other directories as well as the contents of:

    ```
    /var/mqm/mqft/logs/COORDQMNAME/*
    /var/mqm/mqft/config/COORDQMNAME/*
    ```

– Windows On Windows, copy the agent and logger Javacores, traces, logs, properties and FFST files. Include any interactive command traces and javacores written to the current directory or to other directories as well as the contents of the following directories.

```
C:\Program Files\IBM\MQ\mqft\logs\COORDQMNAME\*
C:\Program Files\IBM\MQ\mqft\config\COORDQMNAME\*
```

The actual path name of these directories depends on which version of IBM MQ you are using. For more information, see Program and data directory locations on Windows.

– IBM i On IBM i, copy the agent and logger JVM dumps, traces, logs, properties and FFST files. Include any interactive command traces and javacores you created (that is, spool files from the **GENJVMDMP** command), as well as the contents:

```
/QIBM/UserData/mqm/mqft/logs/COORDQMNAME/*
/QIBM/UserData/mqm/mqft/config/COORDQMNAME/*
```

What to do next

Send the information that you have collected to IBM. For more information, see Step "3" on page 345 of "Collecting information for Managed File Transfer problems on Multiplatforms" on page 345.

Windows Collecting information for Microsoft Cluster Service problems

If you need assistance to resolve a problem where an IBM MQ queue manager is not failing over properly under Microsoft Cluster Service (MSCS) on Windows, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What MSCS problem did you observe on the cluster?
- What time did the MSCS problem start and when did it stop?
- What are the names and addresses of the cluster members?
- Is this a new cluster, or were there any changes made to either cluster member before the problem started?

About this task

It is essential to gather data from the system at the time of the cluster failure in order to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate an MSCS cluster log after the problem occurs.

   On one of the cluster members:

   a) Start PowerShell (or run the 'PowerShell' command in a DOS prompt).

   b) Go to the IBM MQ top-level errors directory.

   For example:

   ```
   PS C:\> CD $env:ProgramData\IBM\MQ\Errors
   ```

   c) Run the Get-ClusterLog cmdlet to generate cluster logs for the nodes in the cluster:

   ```
   PS C:\ProgramData\IBM\MQ\Errors> Get-ClusterLog -Destination
   ```

2. Run the IBM MQ **amqmsysn** utility to display information about all IBM MQ executables and libraries on both members of the cluster.

   Use the **To File** button to save this information to a file, for example `%TEMP%\MQ.exeinfo.txt`.
3. Generate a high detail MQ trace on both members of the cluster during the problem. For example:

```
C:\> strmqtrc -e -t all -t detail
     ...
C:\> endmqtrc -a
```

4. Show the registry checkpoints on both members of the cluster:

```
C:\> CLUSTER RESOURCE /CHECKPOINTS > %TEMP%\Cluster.checkpoints.txt
```

5. Check the registry checkpoints on both cluster members.
6. Save all three Event Viewer logs (System, Application and Security) on both members of the cluster.
7. Place the cluster logs, error logs and command outputs from Steps 1 and 2 directly in the top-level IBM MQ errors directory on each member of the cluster.

   The automatic and manual data collection processes in Step "8" on page 354 both collect files found in this directory.
8. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command on both cluster members as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `trace` section, and to specify your case number as shown in the following example for collecting output from queue manager QMA:

     ```
     runmqras -section trace -qmlist QMA -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
9. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Collecting information for MQIPT problems on Multiplatforms

If you need to report a problem with MQIPT to IBM, send relevant information that will help to resolve the problem more quickly.

About this task

Complete the following steps to obtain the required information.

Procedure

1. Synchronize the system clock on each computer involved, including all those running IBM MQ and MQIPT.

   This operation helps to match trace entries in different trace files.
2. Move old trace files to a backup directory so that new trace files contain information related only to this problem.

3. Turn on trace for all routes that are affected by the problem.

   For more information, see “Tracing errors in IBM MQ Internet Pass-Thru” on page 469.
4. Run the client to reproduce the problem and create new trace files.
5. Send a copy of all MQIPT `.TRC`, `.FDC`, and `.log` files.

   Also send a simple network diagram of all the computers used between the IBM MQ endpoints, including firewalls, routers, load balancers, and servers. For each computer, include its name, IP address, and relevant port numbers.
6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“Troubleshooting IBM MQ Internet Pass-Thru problems” on page 92
There are a number of steps you can follow to help determine the nature of any problems you might encounter when using IBM MQ Internet Pass-Thru (MQIPT).

Multi Collecting information for publish/subscribe problems on Multiplatforms

If you need assistance to resolve a problem where IBM MQ publish/subscribe is not delivering messages properly or reporting a problem on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What publish subscribe problem did you observe on the system?
- What time did the publish subscribe problem start and when did it stop?
- Which specific topics and subscriber applications are involved in the problem?

About this task

It is important to gather information from the system when the publish/subscribe problem is happening in order to identify the cause.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

Generate the troubleshooting information.

1. If the publish subscribe problem is affecting an IBM MQ classes for Java or IBM MQ classes for JMS application, generate an IBM MQ classes for Java trace or a Java Message Service trace, as appropriate.

If your application is running under WebSphere Application Server, follow the trace instructions for that environment.

2. Generate a trace of the queue manager when the publish/subscribe problem occurs.

   If you are generating a Java or JMS trace, do this at the same time.

   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433

3. Linux UNIX

   On Linux and UNIX systems, save the output from the **mqconfig** command.

Collect the troubleshooting information.

4. Place the output from the **mqconfig** command that you generated in Step 3 in the top-level IBM MQ errors directory.

   The automatic and manual data collection processes in Step "5" on page 356 both collect files found in this directory.

5. Collect the IBM MQ data.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `trace` section, and from queue managers the `defs` and `topic` sections as well, and to specify your case number as shown in the following example for collecting output from queue manager QMA:

     ```
     runmqras -section defs,topic,trace -qmlist QMA -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

Send the troubleshooting information to IBM.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Linux MQ Adv. Collecting information for RDQM problems on Linux

If you need assistance when a replicated data queue manager (RDQM) is reporting a problem or failing to work properly on Linux, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

About this task

If you need to collect troubleshooting information to send to IBM Support when reporting a problem with RDQM, you can use the **runmqras** command to collect the diagnostic data.

These instructions apply to IBM MQ 9.0.0 Fix Pack 4 and later on Linux.

Procedure

1. Collect the **runmqras** output from each RDQM node using an mqm user:

   ```
   sudo runmqras -qmlist rdqmName -section defs,trace -caseno casenumber
   ```

   where *rdqmName* is the name of the queue manager and *casenumber* is the case number, for example TS001234567.

   Notes:

   - **-caseno** only works in IBM MQ 9.0.0 Fix Pack 5 or later, IBM MQ 9.1.0 Fix Pack 1 or later, IBM MQ 9.1.1 or later, and IBM MQ 9.2.0 or later.
   - You can omit the `trace` attribute from the **-section** parameter if you are using IBM MQ 9.1.5 or later.

   For more information about using the **runmqras** command, see "Collecting troubleshooting information automatically with runmqras" on page 311.
2. Provide the `/var/log/messages` file from all three nodes.

   Include any archived `syslog` files that might contain activity from the date of the problem.
3. Provide the `/var/log/pacemaker.log` from all three nodes.

   Include any archived `pacemaker.log` files that might contain activity from the date of the problem.

   Note: The **-section** `trace` option collects files in the `/var/mqm/trace` folder, which is where `root-RDQM.log` and `mqm-RDQM.LOG` files are located. DRBD logs are written to the `/var/log/messages` (`syslog`) file.

   One small caveat is that `/var/log/messages` is the location of the default `syslog` output. If a non-default location is used for the `syslog` target, locate the `syslog` in the custom location.

Related tasks

"Troubleshooting RDQM configuration problems" on page 218
These topics give information that is useful for troubleshooting RDQM high availability (HA) and disaster recovery (DR) configurations.

Related reference

rdqm high availability

Multi Collecting information for security problems on Multiplatforms

If an IBM MQ is incorrectly allowing or denying access to a user or application on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What security problem did you observe on the system?
- What time did the security problem start and when did it stop?
- Which specific users or applications and queue manager objects are involved?
- Was this system previously working?
- What changed since it was working?
- How long is your username and password that you are attempting to use?

About this task

If the security problem is happening right now or you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager when the security problem occurs.

   If possible, issue the **runmqsc** command **REFRESH SECURITY** just before tracing so that the trace will show the queue manager querying the operating system for details about the user.

   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433

2. Display information about the user, particularly the groups to which the user belongs.

   For example:

   - Linux UNIX To display user watson on UNIX and Linux:

     ```
     sh> id watson > /tmp/watson.id.txt
     sh> groups watson > /tmp/watson.groups.txt
     ```

   - Windows To display user "Thomas Watson" on Windows:

     ```
     C:\> NET USER "Thomas Watson" > %TEMP%\watson.user.txt
     ```

   - IBM i To display user WATSON at the IBM i command line:

     ```
     ===> DSPUSRPRF USER(WATSON) OUTPUT(*PRINT)
     ```

     Then use **WRKSPLF** option 5 to display the joblog from QPUSRPRF

3. Collect the IBM MQ data.

   You can collect do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs` and `trace` (if the issue was traced) sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

   Note: If one of the sides of this connection is not a queue manager, collect that client's applicable logs.

4. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting security problems" on page 236

Troubleshooting information to help you solve problems relating to security.

Multi Collecting information for TLS channel problems on Multiplatforms

If an IBM MQ queue manager or client application is failing to establish a secure channel using TLS on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What TLS channel problem did you observe on the system?
- What time did the TLS channel problem start and when did it stop?
- Which specific channels and certificates are involved in the problem?
- Was this channel previously working with TLS or is this a new configuration?
- If the channel was previously working what changed?
- Does the channel work without TLS?

Submit the outputs from both sides of the IBM MQ connection. The following examples are from KDB keystore based systems. For clients using other formats, see the documentation for the appropriate format for information on how to list the keystores.

- Keystore Location and Permissions
  - Linux UNIX UNIX and Linux command line:

    ```
    ls -la <DIRECTORY OF KEYSTORE>
    ```

  - Windows Windows Powershell command:

    ```
    Get-Acl <DIRECTORY OF KEYSTORE> |
    Format-List
    ```

- ULW Keystore Certificate Listing UNIX, Linux and Windows:

  ```
  runmqakm -cert -list -v -db <KEYSTORE FILE> -stashed
  ```

- Certificates expired or expiring in the next 90 days:

  ```
  runmqakm -cert -list -expiry 90 -db <KEYSTORE NAME> -stashed
  ```

About this task

If the TLS channel problem is happening right now or you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager when the TLS problem occurs.

   Unless your support representatives inform you differently the correct options for a queue manager TLS trace are `-t all -t detail`:

   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443

- IBM i "Tracing on IBM i" on page 433

2. Generate IBM MQ trace simultaneously at the other end of the channel, whether it is another queue manager or a client application:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
   - Java and JMS Client Applications
   - z/OS z/OS CHIN trace
3. Collect the IBM MQ data.

   You can collect do this either automatically or manually:
   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. Be sure to collect the **runmqras** `defs` and `trace` (if the issue was traced) sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,cluster,trace -qmlist QMA -caseno TS001234567
     ```
   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

   Note: If one of the sides of this connection is not a queue manager, collect that client's applicable logs.
4. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting security problems" on page 236
Troubleshooting information to help you solve problems relating to security.

Multi Collecting information for triggering problems on Multiplatforms

If you need assistance to resolve a problem where IBM MQ is not triggering an application or a channel properly on Multiplatforms, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What triggering problem did you observe on the system?
- What time did the triggering problem start and when did it stop?
- Which queue did not trigger, and which channel or processs should have been started?

About this task

If the triggering problem is happening right now, or if you are able to reproduce it, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate a trace of the queue manager when the triggering problem occurs:
   - Linux UNIX "Tracing on AIX and Linux" on page 427
   - Windows "Tracing on Windows" on page 443
   - IBM i "Tracing on IBM i" on page 433
2. Collect the IBM MQ data.

   You can do this either automatically or manually:
   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311 to collect the data for both sides of the channel. Be sure to collect the **runmqras** `defs` and `trace` sections, and to specify your case number as shown in the following example:

     ```
     runmqras -section defs,trace -qmlist QMA -caseno TS001234567
     ```
   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.
3. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting message problems" on page 177

z/OS Collecting troubleshooting information on z/OS

An overview of how to collect troubleshooting information for IBM MQ for z/OS.

About this task

Note: In addition to the information described in this section, IBM might request further information on a case by case basis.

Procedure

- For information on how to collect troubleshooting and diagnostic information for a specific problem area for IBM MQ for z/OS, see the following topics:
  - Abend problems
  - MQ Adv. VUE MQ Adv. z/OS AMS problems
  - C client problems

  - Channel problems
  - CICS adapter problems
  - CICS bridge problems
  - Cluster problems
  - Data conversion problems
  - Database problems
  - Dead letter queue problems
  - Error messages
  - IBM MQ Explorer remote administration problems
  - IBM WebSphere MQ File Transfer Edition (FTE): see Managed File Transfer for z/OS problems
  - IMS OTMA bridge or IMS adapter problems
  - Install problems
  - Java and JMS application problems
  - Managed File Transfer for z/OS problems
  - Performance problems
  - Publish/subscribe problems
  - Security problems
  - Shared channel problems
  - Shared queue problems
  - Shutdown problems
  - Startup problems
  - TLS problems
  - Triggering channels problems
  - Triggering programs problems
  - Wait, hang, or loop problems
- For all other problems, see Collect troubleshooting data for a general, or unknown problem in WebSphere MQ for z/OS.

Related tasks

“Collecting troubleshooting information on Multiplatforms” on page 310
An overview of how to collect troubleshooting information for IBM MQ on Multiplatforms.

z/OS Collecting information for abend problems on z/OS

If you need assistance to resolve an abend problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Search the IBM Support site for known problems.

You can search by using symptoms like the message number and error codes.

Procedure

1. Collect the following troubleshooting (MustGather) information for this problem:
   a) Collect the following required information:

Job logs
You can find the IBM MQ for z/OS job logs in the Syslog, MSTR job log, and CHIN job log. The job logs are named *xxxx*MSTR and *xxxx*CHIN, where *xxxx* is the IBM MQ subsystem identifier (SSID). For more information, see Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

Dumps generated at point of failure
IBM MQ dumps are located in a system dump data set (see Step "4" on page 363).

b) Optionally, also collect the z/OS LOGREC report.

See A LOGREC report and "SYS1.LOGREC information on z/OS" on page 294.

2. Record the level of the product for IBM MQ for z/OS and the product at the other end of the channel.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS, and for other platforms, see Displaying the IBM MQ version.

3. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

4. Review the dumps generated at point of failure.

   IBM MQ dumps are located in a system dump data set and can be identified by their title. The title for a dump requested by IBM MQ starts with the four-character subsystem name of the queue manager. For example:

   ```
   CSQ1,ABN=5C6-00E20016,U=SYSOPR ,C=MQ900.910.DMC
   -CSQIALLC,M=CSQGFRCV,LOC=CSQSLD1 .CSQSVSTK+00000712
   ```

   The dump title might provide sufficient information in the abend and reason codes to resolve the problem. For more information, see "Analyzing the dump and interpreting dump titles on z/OS" on page 291.

   For more information about the two system abend completion codes X'5C6' and X'6C6' that IBM MQ for z/OS uses, see "IBM MQ for z/OS abends" on page 258. You can also search for known problems at the IBM support site by using abend codes, reason codes, and program names listed in the dump.

5. Check the system log (syslog).

   Comm dumps might not contain the queue manager name, depending on the comment specified in the dump command. Check the syslog for an IEA611I or IEA911E message to determine the dump data set name and also to see whether the dump is complete or partial. For example:

   ```
   IEA611I COMPLETE DUMP ON DUMP.MQT1MSTR.DMP00074
   DUMPID=074 REQUESTED BY JOB(MQT1MSTR)
   FOR ASID(005E)

   IEA911E PARTIAL DUMP ON SYS1.MCEVS4.DMP00039
   DUMPID=039 REQUESTED BY JOB(DMSGTODI)
   FOR ASID(00D2)
   ```

   If insufficient disk space is the reason for the problem, there might not be sufficient information in the dump to diagnose the problem.

   Dumps might be suppressed by Dump Analysis and Elimination (DAE). In this case, some symptoms might not appear in the system log (syslog) or joblog, but they appear in Logrec (see "SYS1.LOGREC information on z/OS" on page 294). For more information about management of DAE, see Generating a suppressed dump.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS MQ Adv. z/OS Collecting information for AMS problems on z/OS

If you need assistance to resolve a problem with Advanced Message Security (AMS) on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Search the IBM Support site for known problems.

You can search by using symptoms like the message number and error codes.

Before you start this task, answer the following initial questions about the problem:

- What AMS error did you observe on the system?
- What is the detailed AMS message flow?
- What time did the AMS problem start and when did it stop?
- Which specific users or applications and queue manager queues are involved?
- What IBM MQ security policy is being used, and what files is AMS using?

  Provide details on how the AMS files are set up.

Procedure

1. For configuration problems, gather:
   - A RACF listing of `<user>\drq.ams.keyring` for each application user involved, for example, put and get applications.
   - A RACF listing of `<AMSUSER>\drq.ams.keyring`
   - RACDCERT ID(user ID) LISTRING(`drq.ams.keyring`)
   - RACDCERT ID(CSQ1AMSM) LISTRING(`drq.ams.keyring`) - replace CSQ1 with the name of your queue manager. For RACDCERT information, see TLS Digital Certificate Commands for External Security Managers:
   - CSQ0UTIL list of policies :
     - **dspmqspl -m** "CSQ1" - replace CSQ1 with the name of your queue manager.
     - **dspmqspl -m** "CSQ1" **-p** "PROBLEMQ" - replace CSQ1 with the name of your queue manager and PROBLEMQ with the name of your queue.
2. For issues to do with AMS server-to-server Message Channel Agent interception, gather channel definitions and display the output.
3. For other errors and/or abend failures, additionally gather:
   - Dumps of the queue manager, channel initiator, AMSM, and putting/getting application address spaces.

     See “IBM MQ for z/OS dumps” on page 276 for more information.
   - Job logs for the queue manager, channel initiator, AMSM address space, and putting/getting application jobs (as applicable).
   - AMS (and/or IBM MQ) internal trace.

     See “Tracing on z/OS” on page 447 for more information.
   - AMS debug trace (written to SYSOUT of the AMSM address space or to the putting/getting application job logs).

     See step “6” on page 365 for information on how you capture an AMS trace.

- IBM Global Security Kit (GSKit) trace.

  See step “7” on page 365 for information on how you capture a GSKit trace.

4. Record the version and release number of IBM MQ for z/OS and the product at the other end of the channel.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS, and for other platforms, see Displaying the IBM MQ version.

5. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

6. Capture an AMS trace.

   For more information, see “Enabling internal trace for the AMSM address space” on page 458.

7. Capture a GSKit trace on your system to help you diagnose problems with keystores and certificates.

   For more information, see “Using GSKit trace for problems related to certificates and keys when using AMS on z/OS” on page 459.

8. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

“Troubleshooting AMS problems” on page 69
Troubleshooting information to help you identify and resolve problems relating to Advanced Message Security (AMS).

z/OS Collecting information for C client problems on z/OS

If you need assistance to resolve a client problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, check:

- The level of IBM MQ client code that is installed on the client machine.
- That the client connection channel on the client machine is properly defined.

About this task

If you can reproduce the client problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following traces while the problem is happening:

   a. Generate a GTF trace.

   b. Generate a MSTR internal trace.

c. Generate a CHIN trace.

2. Optionally, use the following information:
   - Generated dumps
   - “Using the z/OS DUMP command” on page 278
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and the product at the other end of the channel.

      For IBM MQ for z/OS, either look for message CSQY000I in the MSTR job log, or use the **DISPLAY QMGR VERSION** command. For other platforms, see Displaying the IBM MQ version.

   b) Record the operating system version and maintenance level on both sides of the channel.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF. For other platforms, see Operating system version and maintenance level.

   c) Provide the client connection and server connection channel definitions.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for channel problems on z/OS

If you need assistance to resolve a channel problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What channel problem did you observe on the system?
- What time did the channel problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?

About this task

If you can reproduce the channel problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.

2. Generate an IBM MQ trace simultaneously at the other end of the channel.

   This applies, whether it is a remote queue manager, a native client application, or a JMS or Java client:

   - Linux UNIX Linux and UNIX
   - Windows Windows
   - IBM i IBM i
   - Java and JMS client

3. Linux UNIX

   On UNIX and Linux systems, save the output from the **mqconfig** command.

   For more information, see mqconfig (check system configuration).

4. Collect the IBM MQ data.

   a) Record the version and release number of IBM MQ for z/OS and the product at the other end of the channel.

   For IBM MQ for z/OS, either look for message CSQY000I in the MSTR job log, or use the **DISPLAY QMGR VERSION** command. For other platforms, see Displaying the IBM MQ version.

   b) Record the operating system version and maintenance level on both sides of the channel.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF. For other platforms, see Operating system version and maintenance level.

   c) Record the IP addresses and host names of the systems on both sides of the channel.

   d) If your channel is having difficulty connecting, use your operating system tools to list network connections on both sides immediately before and after the connection attempt:

   - To display the network connections on z/OS, issue the following command, `TSO NETSTAT CONN`.
   - To display the device links on z/OS, issue the following command, `TSO NETSTAT DEVLINK`.
   - To display the local interfaces on z/OS, issue the following command, `TSO NETSTAT HOME`.

5. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for CICS adapter problems on z/OS

If you need assistance to resolve a CICS adapter problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

The components to connect CICS and IBM MQ for z/OS are integrated with CICS Transaction Server. CICS-IBM MQ Adapter modules and error messages begin with DFHMQ.

Before you start this task, answer the following questions about the problem:

- What is the problem symptom?

- When did the problem first occur?
- Is the problem a one time failure or reoccurring?
- Was software or hardware maintenance applied?
- What are the names of the queue managers, queues, CICS regions and transactions involved in the problem?

About this task

If you can reproduce the CICS-IBM MQ Adapter problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the data requested in the MustGather topic from the CICS documentation.

   See Collecting CICS troubleshooting data (CICS MustGather) for IBM Support.
2. If additional IBM MQ is required, generate the following traces while the problem is happening:

   a. Generate a MSTR internal trace.

   b. Optionally, Generate a CHIN trace if messages are wing put or got from a channel.

   c. Optionally, Generate a GTF trace while the problem is happening.

      If you need to include IBM MQ and CICS trace entries in GTF, you should update the JOBNAME and USR parameters referenced in those instructions, as follows:

      ```
      JOBNAME=(ssidMSTR,cicsjob,appjob),USR=(5E9,5EA,5EE,F6C)
      ```

      where:

      ssid
      Is the subsystem id of the queue manager

      cicsjob
      Is replaced with your CICS job name

      appjob
      Is any other job batch, IMS, or `ssidCHIN` job involved in putting or getting the messages.
3. Collect the IBM MQ data.

   a) Record the version and release number of IBM MQ for z/OS and CICS.

      See message CSQY000I in the MSTR job log for IBM MQ for z/OS, and message DFHSI1500 in the CICS job log for the CICS version information.

   b) Record the operating system version and maintenance level of IBM MQ and CICS.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

   c) Record the version, release, and maintenance level of any other products involved with the problem.

   d) Collect the IBM MQ and CICS joblogs.

   e) Collect the IBM MQ and CICS dump.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for CICS bridge problems on z/OS

If you need assistance to resolve a CICS bridge problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

The components to connect CICS and IBM MQ for z/OS are integrated with CICS Transaction Server. CICS-IBM MQ Bridge modules, whose names begin with DFHMQBR, and error messages begin with DFHMQ07.

Before you start this task, answer the following questions about the problem:

- What is the problem symptom?
- When did the problem first occur?
- Is the problem a one time failure or reoccurring?
- Was software or hardware maintenance applied?
- Are you using the 3270 or DPL Bridge?
- Are the fields set correctly in the MQMD and MQCIH structures?
- What are the names of the queue managers, queues, CICS regions and transactions involved in the problem?

About this task

If you can reproduce the CICS-IBM MQ Bridge problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the data requested in the MustGather topic from the CICS documentation.

   See Collecting CICS troubleshooting data (CICS MustGather) for IBM Support.
2. If additional IBM MQ data is required, generate the following traces while the problem is happening:
   a. Generate a MSTR internal trace.
   b. Optionally, Generate a CHIN trace if messages are wing put or got from a channel.
   c. Optionally, Generate a GTF trace while the problem is happening.

      If you need to include IBM MQ and CICS trace entries in GTF, you should update the JOBNAME and USR parameters referenced in those instructions, as follows:

      ```
      JOBNAME=(ssidMSTR,cicsjob,appjob),USR=(5E9,5EA,5EE,F6C)
      ```

      where:

      ssid
      Is the subsystem id of the queue manager

      cicsjob
      Is replaced with your CICS job name

appjob
Is any other job batch, IMS, or `ssidCHIN` job involved in putting or getting the messages.

3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and CICS.
      See message CSQY000I in the MSTR job log for IBM MQ for z/OS, and message DFHSI1500 in the CICS job log for the CICS version information.
   b) Record the Operating system version and maintenance level of IBM MQ and CICS.
      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
   c) Record the version, release, and maintenance level of any other products involved with the problem.
   d) Collect the IBM MQ and CICS joblogs.
   e) Collect the IBM MQ and CICS dump.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.
   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.
   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!
   For FTP and email instructions, see Exchanging information with IBM Software Support.
   To open or update a case, go to the IBM My Support site.
   Note: Always update your case to indicate that data was sent.
   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for cluster problems on z/OS

If you need assistance to resolve cluster queues, topics, or channels problems on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What cluster problem did you observe on the system?
- What time did the cluster problem start?
- Which queue managers, channels, remote queues, and transmission queues are involved?

About this task

If you can reproduce the cluster problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information while the problem is happening:
   a) Information on the default cluster objects
      See Default cluster objects
   b) Dumps generated at point of failure.

See “IBM MQ for z/OS dumps” on page 276

c) Dumps taken at the point of failure.

See “Using the z/OS DUMP command” on page 278

2. Generate the following optional information while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS.

   For IBM MQ for z/OS, either look for message CSQY000I in the MSTR job log, or use the **DISPLAY QMGR VERSION** command. For other platforms, see Displaying the IBM MQ version.

   b) Record the operating system version and maintenance level of all the queue managers involved.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF. For other platforms, see Operating system version and maintenance level.

   c) Provide feedback regarding cluster errors, if these occur with sample applications as well.

   d) Provide output from the following commands:

   DISPLAY CLUSQMGR(*) ALL
   Produces information about queue manager interconnection in a cluster

   DISPLAY QUEUE(*) CLUSINFO
   Produces information about cluster queues

   DISPLAY CHSTATUS(*) ALL
   Produces information about channel status
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for data conversion problems on z/OS

If you need assistance to resolve a data conversion problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What data conversion problem did you observe on the system?
- What time did the data conversion problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?

Investigate the following:

- An IBM MQ message consists of two parts, the control information in a message descriptor and the application data.

  Application data is converted using one of the following methods:

  – In an application program when the MQGMO_CONVERT option is specified on an MQGET call.
  – In the channel program by specifying CONVERT(YES) keyword on the sender or server channel.
- The **Format** field in the MQMD structure associated with the message must contain a valid format:
  – MQFMT_NONE is the initial setting and data conversion does not occur with this setting.
  – The built in format (MQFMT_STRING) should be used if the message is string data. IBM MQ data conversion programs convert the data.
  – If your message contains numeric data, then you need to have your own format. You also need to write your own exit program to do the data conversion.
  – The built in format (MQFMT_CICS) can be used with CICS messages, however messages in that format can only be converted on IBM MQ on host systems. When sending messages to a different platform you should configure the sender channel process to do the data conversion. See RC 2110 (MQRC_FORMAT_ERROR) for more information.
- Conversion of EBCDIC newline characters

  If you need to ensure that the data you send from an EBCDIC platform to an ASCII one is identical to the data you receive back again, you must control the conversion of EBCDIC newline characters. This can be done using a platform-dependent switch that forces IBM MQ to use the unmodified conversion tables but you must be aware of the inconsistent behavior that can result.

  The problem arises because the EBCDIC newline character is not converted consistently across platforms or conversion tables. As a result, if the data is displayed on an ASCII platform, the formatting can be incorrect. This makes it difficult, for example, to administer an iSeries system remotely from an ASCII platform using RUNMQSC.

  For further information about converting EBCDIC-format data to ASCII format, see **ConvEBCDICNewline**.

About this task

If you can reproduce the data conversion problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log.

      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) A LOGREC report

      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

   c) Gather the following information for the Sending and Receiving queue manager:

      Sending queue manager

```
Queue Manager CCSID:
Putting application setting for MQMD CCSID:
Putting application setting for MQMD Format:
Use CSQ4BCG1 to capture the message on the transmission queue:
What is the character and its Hex representation and offset within the message:
```

Receiving queue manager

```
Queue Manager CCSID:
Getting Application Setting for MQMD CCSID:
Use CSQ4BCG1 to capture the message on the destination/local queue:
What is the character and its Hex representation and offset within the message:
```

2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.
   See message CSQY000I in the MSTR job log for IBM MQ for z/OS
   b) Record the operating system version and maintenance level of your system.
   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for database problems on z/OS

If you need assistance to resolve a database problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What database problem did you observe on the system?
- What time did the database problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- Search for the most recent database related APARs.

About this task

If you can reproduce the database problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs
      You require the Syslog, MSTR job log, and CHIN job log.
      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b) A LOGREC report
      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.
      See message CSQY000I in the MSTR job log for IBM MQ for z/OS
   b) Record the operating system version and maintenance level of your system.
      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.
   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.
   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!
   For FTP and email instructions, see Exchanging information with IBM Software Support.
   To open or update a case, go to the IBM My Support site.
   Note: Always update your case to indicate that data was sent.
   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for dead letter queue problems on z/OS

If you need assistance to resolve a dead letter queue problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What dead letter queue problem (DLQ) did you observe on the system?
- What time did the DLQ problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- Determine why a message is on the DLQ:

1. Browse the message on the dead letter queue by using a program that shows headers, for example, CSQ4BCG0 in SCSQLOAD with sample JCL in SCSQPROC.
2. Locate the dead letter header, which starts with the characters "**DLH**":

```
444C 4820 0000 0001 0000 010A 4D59 2E51 'DLH ........MY.Q'
```

3. Locate the dead letter Reason code, that is the reason the message arrived on the DLQ, in the third word in the dead letter header.

   In the preceding example the dead letter Reason is x'**0000 010A**'
4. Convert hexadecimal x'**0000 010A**' to decimal 266.
5. Use the dead letter reason code to determine why the message was placed on the dead-letter queue. The dead letter reason code is a Reason Code (MQRC_*) or a Feedback Code (MQFB_*).For example:

```
MQFB_TM_ERROR  266  X'0000010A'
MQRC_Q_FULL   2053  X'00000805'
```

MQRC_* lists the reason codes, and MQFB_* lists the feedback values.

Additionally, use the Fields information in the MQDLH structure to determine why the message was routed to the dead letter queue (DLQ)

About this task

If you can reproduce the dead letter queue problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log.

      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) A LOGREC report

      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.

   This data includes the reason code or feedback code in the dead letter header.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

      See message CSQY000I in the MSTR job log for IBM MQ for z/OS
   b) Record the operating system version and maintenance level of your system.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for error messages on z/OS

If you need assistance to resolve an error message problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What error message did you observe on the system?
- What time did the error message occur?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- Search IBM MQ for z/OS messages, completion, and reason codes for the z/OS information, and messages for any other platform your enterprise is using in the product documentation.

About this task

If you can reproduce the error message or the message is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs
      You require the Syslog, MSTR job log, and CHIN job log.
      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b) A LOGREC report
      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.
      See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

b) Record the operating system version and maintenance level of your system.

For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for IMS OTMA bridge or IMS adapter problems on z/OS

If you need assistance to resolve an IMS Open Transaction Manager Access (OTMA) bridge or IMS adapter problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What IMS problem did you observe on the system?
- What time did the problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

For both IMS OTMA bridge or IMS adapter problems, search the IBM MQ Support site for known problems.

For IMS OTMA bridge problems, check that:

- The queue is a bridge queue, that is, the queue has been defined with an XCF STGCLASS.
- IMS is running.
- OTMA has been started.
- IBM MQ is connected to OTMA.

and look at:

- The latest IBM MQ IMS technical support information - see Products and services
- The latest IBM MQ IMS maintenance information - see IMS service considerations
- “Finding messages sent to the IBM MQ - IMS bridge” on page 307
- MP16: Capacity Planning and Tuning for IBM MQ for z/OS

For the IMS Adapter, check that the IMS subsystem is started and look at

- “IMS-related problems” on page 546.
- Controlling the IMS trigger monitor.

For security considerations for using IBM MQ with IMS, look at:

- Security considerations for connecting to IMS

- Application access control for the IMS bridge
- Security checking on IMS
- Security checking done by the IMS bridge
- Using RACF PassTickets in the IMS header
- Connection security profiles for IMS connections

About this task

If you can reproduce the IMS problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs for IMS.

      Collect the job logs for IMSCNTL, IMSMSG and IMSBATCH.

   b) Job logs for IBM MQ for z/OS.

      You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID). For information on how to use XDC in SDSF to create a print data set that contains the JES2 job log for the CHIN or MSTR address space, see Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   c) A LOGREC report.

      For more information, see A LOGREC report and "SYS1.LOGREC information on z/OS" on page 294.

   d) Save any z/OS dumps generated from IBM MQ. For more information, see "IBM MQ for z/OS dumps" on page 276 and IBM MQ Error Log, FFST/FDC and Dump Locations.
2. Collect the following data:
   a) Record the version and release number of IBM MQ for z/OS, IMS, and any other related product.

      For more information, see message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
3. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace. Use `JOBNAME=(ssidMSTR,IMSregions)` where *IMSregions* are the IMS control region plus the BMP, IFP or MPP that are involved.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace if a channel is part of the message flow.
   d. Generate an IMS trace if needed.
4. If a dump was not automatically created, capture a dump of the queue manager and IMS:

   ```
   DUMP COMM=(dump Title)
   Rnn,JOBNAME=(ssidMSTR,IMSjobs),cont
   Rnn,SDATA=(CSA,RGN,PSA,SQA,LSQA,TRT,SUM,GRSQ),END
   ```

   where *ssidMSTR* is the IBM MQ queue-manager MSTR address space and *IMSjobs* is the IMS Control region plus any other IMS region(s) such as the message or batch processing region associated with the problem.
5. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
6. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for install problems on z/OS

If you need assistance to resolve an install problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Review the IBM MQ for z/OS Migration Guide and also Migrating IBM MQ on z/OS.

If there is still a problem, answer the following questions:

- What install problem did you observe on the system, including any error messages?
- Is there is an error in the Causer section of one of the following SMP/E installation jobs?
  - SMP/E RECEIVE
  - SMP/E APPLY
  - SMP/E ACCEPT
- Did you use the GROUPEXTEND option as recommended in the Program Directory? For download links for the Program Directories, see IBM MQ for z/OS Program Directory PDF files.

Procedure

1. Perform the product installation.

   See Installing IBM MQ for z/OS and the IBM MQ for z/OS Program Directory.
2. Collect the following troubleshooting (MustGather) information for this problem:

   a) Collect the following required information:

   Job logs

   You can find the IBM MQ for z/OS job logs in the Syslog, MSTR job log, and CHIN job log. The job logs are named *xxxx*MSTR and *xxxx*CHIN, where *xxxx* is the IBM MQ subsystem identifier (SSID). For more information, see Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) Optionally, also collect the z/OS LOGREC report.

   See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294).
3. Search the IBM Support site for known problems.

   You can search by using symptoms like the message number and error codes.
4. Record the version and release number of IBM MQ for z/OS and any other products related to the problem.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS, and for other platforms, see Displaying the IBM MQ version.
5. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
6. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for Java and JMS application problems on z/OS

If you need assistance to resolve a JMS or Java problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What JMS or Java problem did you observe on the system?
- What time did the problem start and when did it stop?
- Were any Java exceptions reported, and did they include a Java call stack?
- Which queue managers, queues and topics does the JMS or Java application use?

Search the IBM MQ Support site for known problems using, for example, any error messages or return codes.

The majority of IBM MQ for z/OS Java failures are caused by setup or configuration problems. You must configure the path, classpath, and environment variables correctly. See:

- Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging
- Environment variables relevant to IBM MQ classes for Java
- Configuring the Java Native Interface (JNI) libraries
- IBM MQ classes for Java libraries
- Using WebSphere Application Server with IBM MQ

About this task

If you can reproduce the problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log.

      The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) Dumps generated at point of failure.

      See “IBM MQ for z/OS dumps” on page 276.

   c) Dumps taken at the point of failure.

      See “Using the z/OS DUMP command” on page 278.

d) A LOGREC report

See A LOGREC report.

2. Optionally, generate the following while the problem is happening:
   a. Generate a GTF trace
   b. Generate a MSTR internal trace
   c. Generate a CHIN trace
   d. “Tracing IBM MQ classes for JMS applications” on page 476
   e. “Tracing IBM MQ classes for Java applications” on page 481
   f. Enabling JMS Trace for WebSphere Application Server
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF. For other platforms, see Operating system version and maintenance level.
4. For standalone applications, record the version of the IBM MQ classes for Java or IBM MQ classes for JMS that are being used.

   Finding the build level for the classes on the current classpath

   Issue the following command:

   ```
   java com.ibm.mq.MQJavaLevel
   ```

   or

   ```
   java com.ibm.mq.jms.MQJMSLevel
   ```

   Finding the build level for a jar files in a specific directory:

   Change to the directory containing the jars, and issue the following command:

   ```
   java -cp com.ibm.mq.jar com.ibm.mq.MQJavaLevel
   ```

   or

   ```
   java -cp com.ibm.mqjms.jar:com.ibm.mq.jar com.ibm.mq.jms.MQJMSLevel
   ```

   If you receive *java.lang.NoClassDefFoundError* for **MQJavaLevel** or **MQJMSLevel**, you need provide a path to the class. You can use one of the following methods:

   a. Specify the CLASSPATH in the command:

      ```
      java -cp path/com.ibm.mq.jar com.ibm.mq.MQJavaLevel
      ```

      where `path` is the directory path which contains the jar file of interest.

      For example:

      ```
      java -cp ./com.ibm.mq.jar com.ibm.mq.MQJavaLevel
      ```

      gives version information for the `com.ibm.mq.jar` file in the current directory, or

      ```
      java -cp /usr/lpp/mqm/V9R3M0/java/lib/com.ibm.mq.jar com.ibm.mq.MQJavaLevel
      ```

      gives version information for the file `/usr/lpp/mqm/V9R3M0/java/lib/com.ibm.mq.jar`.

      Note: `java -cp /usr/lpp/mqm/V9R3M0/java/lib/com.ibm.mq.jar com.ibm.mq.MQJavaLevel` must be on one line.

b. Set up the CLASSPATH and LIBPATH environment variables.

   See Setting environment variables for IBM MQ classes for JMS/Jakarta Messaging and Configuring the Java Native Interface (JNI) libraries to match the settings used by the application or WebSphere Application Server.

   The "**env**" command in OMVS will show you what your current settings are.

   If you need to set the variables for your OMVS session, the following export commands are an example of what to issue from the OMVS command line or add to your `.profile`:

```
export LIBPATH=/usr/lpp/mqm/V9R3M0/java/lib:$LIBPATH
export CLASSPATH=/usr/lpp/mqm/V9R3M0/java/lib/com.ibm.mq.jar:$CLASSPATH
export CLASSPATH=/usr/lpp/mqm/V9R3M0/java/lib/com.ibm.mqjms.jar:$CLASSPATH
```

   Notes:

   - `/usr/lpp` is an install directory that might be different for your environment
   - These variables are the minimum needed to display the levels
   - The run-time CLASSPATH contains other libraries.

5. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

6. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for Managed File Transfer for z/OS problems

If you need assistance to resolve a problem when a Managed File Transfer (MFT) for z/OS agent, logger or command is reporting a problem or failing to work properly you might need to collect troubleshooting information to include with your support case to help find a solution to the problem. The information that is needed depends on the problem that you are seeing.

Procedure

1. Collect the information that is needed for the type of problem that you are seeing:
   - Managed File Transfer for z/OS agent problems
   - Managed File Transfer for z/OS protocol bridge agent problems
   - Managed File Transfer for z/OS resource monitor problems
   - Managed File Transfer for z/OS managed transfer problems
   - Managed File Transfer for z/OS database logger problems
   - Managed File Transfer for z/OS command problems
2. After you have collected the Managed File Transfer data that is needed to investigate the problem, create an archive containing all of the relevant files.

   For more information, see “Creating an archive of troubleshooting information for MFT for z/OS” on page 388.

3. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting Managed File Transfer problems" on page 127
Use this information to help you to diagnose errors in Managed File Transfer (MFT).

z/OS *Collecting information for MFT for z/OS agent problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) agent problem on z/OS.

About this task

Managed File Transfer agent problems include:

- The agent failing to connect to, or being disconnected from, its agent queue manager.
- The agent hanging.
- The agent stopping unexpectedly.
- The agent going into recovery.
- The **fteListAgents** or **fteShowAgentDetails** commands, or the IBM MQ Explorer Managed File Transfer plug-in, showing incorrect, or out of date, status information for the agent.
- The agent failing to report any status information.

Procedure

1. Initially, review the following topics to see whether they help you to resolve the problem:
   - "Troubleshooting MFT agent status problems" on page 128
   - "Troubleshooting java.lang.OutOfMemoryError problems" on page 152
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the agent.
   - The name of the agent queue manager.
   - The version of Managed File Transfer for z/OS that the agent is using.
   - The version of IBM MQ for z/OS for the agent queue manager.
   - Details of how the agent is started (for example, is it running as a started task?).
   - Any error messages that are seen in the agent's event log (`output0.log`) when the issue occurs.
   - An agent trace covering the time of the issue. For more information about how to collect the trace, see "Tracing Managed File Transfer for z/OS agents" on page 497.
     - If the agent is experiencing the problem when communicating with its agent queue manager (for example, the agent's event log contains error messages that include an IBM MQ reason code such as `2009 – MQRC_CONNECTION_BROKEN`), collect the trace using the trace specification `=all`.
     - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
   - Three Javacores taken 30 seconds apart, if the agent is hanging. In order to do this, run the **fteSetAgentTraceLevel** command with the `-jc` option set, as shown in the following example:

```
fteSetAgentTraceLevel -jc <agent_name>
```

If the command does not cause the agent to generate a Javacore, then you should send a SIGQUIT signal to the agent process.

- An archive containing the agent's log files, configuration files, trace files and Javacores, if applicable. For more information about how to create the archive, see "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.
- The job logs for the coordination queue manager and agent queue manager.

z/OS *Collecting information for MFT for z/OS protocol bridge agent problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) protocol bridge agent problem on z/OS.

About this task

Managed File Transfer protocol bridge agent problems include:

- The agent failing to connect to, or being disconnected from, a remote file server.
- Managed transfers to or from a remote file server failing.

Procedure

1. Initially, review the information in "Troubleshooting a protocol bridge agent that reports a file not found" on page 143 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the protocol bridge agent.
   - The name of the protocol bridge agent queue manager.
   - The version of Managed File Transfer for z/OS that the protocol bridge agent is using.
   - The version of IBM MQ for z/OS for the protocol bridge agent queue manager.
   - The hostname of the remote file server system.
   - Product and version information for the remote file server.
   - The protocol that the agent is using to communicate with the remote file server (that is, FTP, FTPS or SFTP).
   - The entry for the remote file server in the protocol bridge agent configuration file (`ProtocolBridgeProperties.xml`).
   - Any error messages that are seen in the agent's event log (`output0.log`) when the issue occurs.
   - A protocol bridge agent log file, where the log level for the protocol being used is set to `on`. For more information about how to set the log level, see fteSetAgentLogLevel (Turn on or turn off logging to file of certain MFT agent operations).
   - An archive that contains the protocol bridge agent's log files and configuration files. For more information about how to create the archive, see "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.

z/OS *Collecting information for MFT for z/OS resource monitor problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) resource monitor problem on z/OS.

About this task

Managed File Transfer resource monitor problems include:

- A resource monitor stops polling.
- A resource monitor is polling, and not triggering on any items (either files or messages).
- A resource monitor is not submitting managed transfer requests to the agent.

- A resource monitor stops unexpectedly.

Procedure

1. Initially, review the information in "Troubleshooting resource monitor problems" on page 144 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the agent.
   - The name of the agent queue manager.
   - The version of Managed File Transfer for z/OS that the agent is using.
   - The version of IBM MQ for z/OS for the agent queue manager.
   - The name of the resource monitor.
   - The name of the resource (either a queue or directory) that the monitor is polling.
   - The monitor's trigger condition.
   - The monitor's task XML.
   - Details of any items that the monitor is not triggering on.
   - A resource monitor log file (for example, `resmonevent0.log`), where the log level for the resource monitor is set to VERBOSE. For more information about how to create the log file, see Logging MFT resource monitors.

     If the monitor is polling, and has not got stuck, then the log file should include entries for at least three polls.
   - An archive containing the agent's configuration files, and the log files for the agent and resource monitor. For more information about how to create the archive, see "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.

z/OS *Collecting information for MFT for z/OS managed transfer problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) managed transfer problem on z/OS.

About this task

Problems related to Managed File Transfer managed transfers include:

- A managed transfer failing unexpectedly.
- A managed transfer going into recovery and not completing.
- A managed transfer getting stuck.

Procedure

1. Initially, review the information in "Troubleshooting managed transfer problems" on page 136 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the source agent for the managed transfer.
   - The name of the source agent queue manager.
   - The version of Managed File Transfer or Managed File Transfer for z/OS that the source agent is using.
   - The version of IBM MQ or IBM MQ for z/OS for the source agent queue manager.
   - The name of the destination agent for the managed transfer.
   - The name of the destination agent queue manager.

- The version of Managed File Transfer or Managed File Transfer for z/OS that the destination agent is using.
- The version of IBM MQ or IBM MQ for z/OS for the destination agent queue manager.
- If the source and destination agent queue managers are different, details of how the queue managers are connected together (that is, through sender/receiver channels or an IBM MQ cluster).
- The transfer identifier for the managed transfer.
- Details of how the managed transfer request was created (that is, was it generated by a resource monitor, the **fteCreateTransfer** command, the IBM MQ Explorer Managed File Transfer plug-in, or something else?).
- Details of any error messages that are related to the managed transfer in either the source agent or destination agent's event log (`output0.log`).
- If the source and/or destination agents, are running Managed File Transfer or Managed File Transfer for z/OS at IBM MQ 9.3 or later, a VERBOSE transfer log that covers the time when the issue occurred. For more information about how create the transfer log, see fteSetAgentLogLevel (Turn on or turn off logging to file of certain MFT agent operations).
- A trace from both the source and destination agents that covers the time when the issue occurred. For more information about how to collect the trace, see "Tracing Managed File Transfer agents on Multiplatforms" on page 490 or "Tracing Managed File Transfer for z/OS agents" on page 497. The trace should be collected using the trace specification `com.ibm.wmqfte=all`.
- An archive from the source agent containing the agent's log files and configuration files and an archive from the destination agent, containing the agent's log files and configuration files. For more information about how to collect the archives for the source and destination agents, see "Creating an archive of MFT troubleshooting information" on page 352 or "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.
- **runmqras** output for the source agent queue manager and the destination agent queue manager, if they are running on a platform other than z/OS. For more information about how to collect the **runmqras** output for the source agent queue manager and the destination agent queue manager, see "Collecting troubleshooting information automatically with runmqras" on page 311.

When you are investigating issues related to managed transfers, it is often useful to draw a simple diagram, as shown in the following example, that shows the agents and the agent queue managers. This diagram allows you and IBM Support to see how the agents and the agent queue managers are connected, which can help to identify possible issues within the IBM MQ network that might cause managed transfers to enter recovery or get stuck.

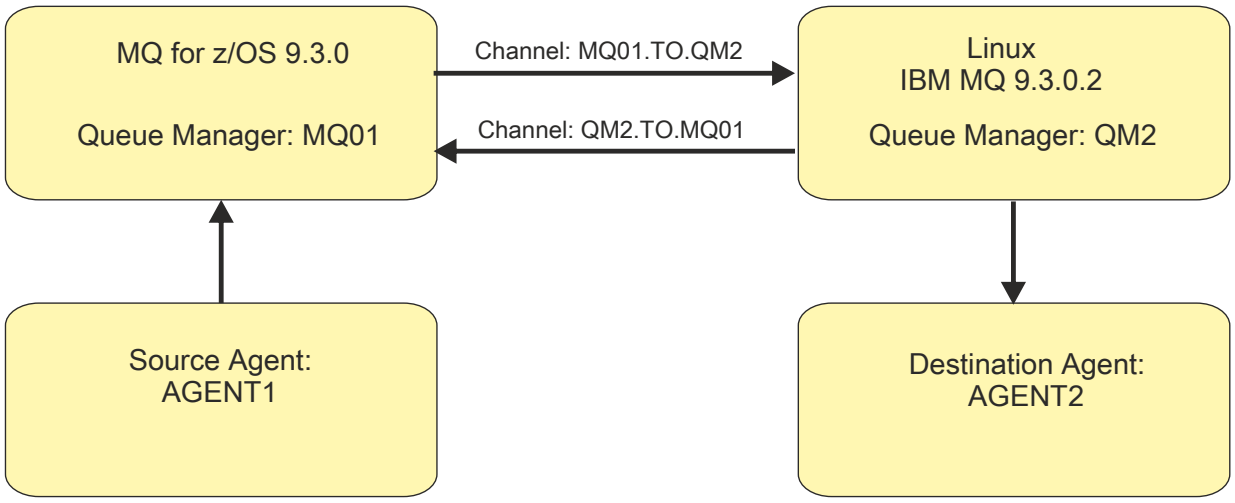


Figure 55. Example of a simple diagram showing how a source agent, AGENT1, and a destination agent, AGENT2, and their agent queue managers, MQ01 and QM2 are connected.

z/OS *Collecting information for MFT for z/OS database logger problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) database logger problem on z/OS.

About this task

Managed File Transfer database logger problems include:

- The database logger fails to connect to the coordination queue manager.
- The database logger fails to connect to the database.
- The database logger doesn't update the database.

Procedure

1. Initially, review the information in "Troubleshooting logger problems" on page 158 to see whether that helps you to resolve the problem.
2. If you still require assistance, collect the following information and send it to IBM Support:
   - The name of the database logger.
   - The name of the coordination queue manager that the database logger is connecting to.
   - The version of Managed File Transfer for z/OS that the database logger is using.
   - The version of IBM MQ for z/OS for the coordination queue manager.
   - The type of database that the database logger is using.
   - Details of any error messages that appear in the database logger's event log when the issue occurs.
   - A database logger trace that covers the time of the issue. For more information about how to collect this trace, see "Tracing Managed File Transfer for z/OS standalone database loggers" on page 505.
     - If the database logger is experiencing the problem when communicating with the coordination queue manager (for example, the database logger's event log contains error messages that include an IBM MQ reason code such as `2009 – MQRC_CONNECTION_BROKEN`), collect the trace using the trace specification `=all`.
     - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
   - An archive that contains the database logger log files and configuration files. For more information about how to create the archive, see "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.
   - The job logs for the coordination queue manager.

z/OS *Collecting information for MFT for z/OS command problems*

The troubleshooting information that you need to collect and send to IBM if you need assistance with a Managed File Transfer (MFT) command problem on z/OS.

About this task

Problems with Managed File Transfer commands include:

- A command failing to connect to the queue manager.
- A command timing out.
- A command reporting an error.

Procedure

To investigate these, provide the following information:

- The command that is being run.

- Whether the command is being run from z/OS UNIX System Services (USS) or via JCL.
- The username for the user that is logged in when the command is run.
- The output from the command.
- The version of Managed File Transfer for z/OS that the command is using.
- A trace of the command, covering the time when the issue occurred. For information on how to collect this trace, see "Tracing Managed File Transfer for z/OS commands" on page 503.
  - If the command is experiencing the problem when communicating with a queue manager (for example, the command reports an error containing an IBM MQ reason code), collect the trace using the trace specification `=all`.
  - For all other issues, collect the trace using the trace specification `com.ibm.wmqfte=all`.
- An archive containing the configuration files on the system where the command is being run. For more information about how to create the archive, see "Creating an archive of troubleshooting information for MFT for z/OS" on page 388.

z/OS *Creating an archive of troubleshooting information for MFT for z/OS*

After you have collected the data needed to investigate the Managed File Transfer (MFT) problem that you are seeing on z/OS, you need to create an archive that contains all of the relevant files and send it in to IBM. You can either create the archive manually, or by using the **fteRAS** utility.

About this task

Warning: If there are a large number of Managed File Transfer for z/OS agents configured on a system, the **fteRAS** command can take a long time to complete. If that happens, you should create the archive manually by compressing the contents of the Managed File Transfer agent's logs and configuration directories.

From IBM MQ 9.3.4, you can collect troubleshooting information for a specific Managed File Transfer agent or agents instead of for all agents on the system. You do this by running the **fteRAS** command with the **-agents** parameter.

Procedure

- To archive the Managed File Transfer files automatically using the **fteRAS** command:
  - Copy any trace files that were generated when running a command into the BFG_DATA directory for your Managed File Transfer for z/OS installation.
  - If you are using z/OS UNIX System Services (USS), run the **fteRAS** command.
  - If you are using JCL:
    - Locate the data set containing the JCL for the installation.
    - Submit the BFGRAS member within the data set.
- To archive the Managed File Transfer files manually:
  - Copy the agent and logger javacores, traces, logs, properties and FFST files into a temporary directory. Include any interactive command traces and javacores written to the current directory or to other directories as well as the contents of the following directories:

    BFG_DATA/mqft/config/*coordination_qmgr_name*
    BFG_DATA/mqft/logs/*coordination_qmgr_name*
  - Create an archive containing the contents of the temporary directory.

What to do next

Send the information that you have collected to IBM. For more information, see Step "3" on page 345 of "Collecting information for Managed File Transfer for z/OS problems" on page 382.

z/OS Collecting information for performance problems on z/OS

If you need assistance to resolve a performance problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What effect is the problem having, for instance high CPU or response delays?
- When did the problem first occur?
- Was software or hardware maintenance applied?
- Is the problem a one time failure or reoccurring?
- What are the names of the queue managers, queues, channels, or other jobs involved in the problem?
- Have you reviewed the information in Troubleshooting MQ performance problems.

About this task

The IBM Software Support Handbook states that analyzing performance is one of the activities that often require some form of Advance Support Offering. If analysis reveals a suspected defect in the product and you can reproduce the performance problem or the problem is happening right now, you can generate data to provide more information about the problem so that the IBM MQ Support team can diagnose your problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. If you can reproduce the performance problem or the problem is happening right now, generate data to provide more information about the problem:
   a. Generate a GTF trace while the problem is happening.
   b. Generate a MSTR internal trace and capture it in a dump while the problem is happening.
   c. Generate a CHIN trace and capture it in the same dump with the MSTR trace while the problem is happening.
   d. If the network performance is in question, generate a z/OS TCP/IP packet trace and an MQ CHIN trace simultaneously at the other end of the channel while the problem is happening.
2. Collect the data.
   a) Record the version, release, and maintenance levels your software:
      - IBM MQ: find the version in the CSQY000I message in the MSTR job log.
      - The z/OS operating system: find the version in the output of /D IPLINFO in SDSF.
      - For any other products involved with the problem, look for the version in the job log for the product.
   b) Collect the IBM MQ MSTR and CHIN joblogs and, optionally, collect the syslog.
   c) Collect the z/OS dump that you generated when collecting the traces.
   d) Collect a z/OS LOGREC report.
      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
3. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
4. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Troubleshooting distributed queue management problems" on page 78
Troubleshooting information to help you solve problems relating to distributed queue management (DQM).

z/OS Collecting information for publish/subscribe problems on z/OS

If you need assistance to resolve a publish/subscribe problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What publish/subscribe problem did you observe on the system?
- What time did the publish/subscribe problem start and has it stopped?
- Which queue managers are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- See Troubleshooting MQ channels for help in correcting problems with any channels..

About this task

If you can reproduce the publish/subscribe problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following traces and job logs while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a CHIN trace.
   c. You require the Syslog, MSTR job log, and CHIN job log.

      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID). See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
2. You might require the following information:
   a) IBM MQ system dumps

      See "IBM MQ for z/OS dumps" on page 276 for more information.
   b) A MSTR internal trace

      See Generate a MSTR internal trace

c) A LOGREC report

See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

3. Collect the IBM MQ data.

a) Include the following information:

- Publish/Subscribe status information, by issuing the command **DISPLAY PUBSUB TYPE**(ALL) **ALL**
- Topic attributes, by issuing the command **DISPLAY TOPIC**(*) **TYPE**(ALL) **ALL**
- Subscription attributes, by issuing the command **DISPLAY SUB**(*) **SUBTYPE**(ALL) **ALL**

b) Record the version and release number of IBM MQ for z/OS and any other related product.

See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

c) Record the operating system version and maintenance level of your system.

For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

5. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for security problems on z/OS

If you need assistance to resolve a security problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What security problem did you observe on the system?
- What time and date did the security problem start and when did it stop?
- Which queue managers are involved and, if relevant, channels, remote queues and transmission queues?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- See Troubleshooting MQ channels for help in correcting problems with any channels..

About this task

If you can reproduce the security problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
2. You might need to collect the following information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b) A RACF trace

      See the z/OS Security Server RACF Diagnosis Guide for further information.
   c) IBM MQ system dumps

      See “IBM MQ for z/OS dumps” on page 276 for more information.
   d) The output from the command DISPLAY SECURITY ALL

      See Displaying security status for more information.
   e) Results of SETROPTS LIST and RACLIST commands

      See Using SETROPTS RACLIST and SETROPTS GENLIST for more information.
   f) A LOGREC report

      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

      See message CSQY000I in the MSTR job log for IBM MQ for z/OS.
   b) Record the operating system version and maintenance level of your system.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for shared channel problems on z/OS

If you need assistance to resolve a shared channel problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What shared channel problem did you observe on the system?

- What time did the shared channel problem start and has it stopped?
- Which queue managers and channel initiators are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- See Shared channels for help in correcting problems with any shared channels.
- See DVIPA (Dynamic Virtual IP Address) for information on shared inbound channels.
- Performance issues:
  - Do DISPLAY CHSTATUS on both ends of the channel and check STATUS, SUBSTATE and INDOUBT.
  - Use a monitor or Resource Management Facility (RMF) data to confirm that the channel initiator or TCP/IP task is not suffering dispatching delays due to WLM settings or CPU constraint.
  - Check the send and receive buffer sizes, as the TCP/IP sockets created by the IBM MQ channel initiator take the default sizes from the TCP/IP profile.
- For more information, see Distributed queuing and queue sharing groups and Setting up communication for IBM MQ for z/OS using queue sharing groups.

About this task

If you can reproduce the shared channel problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following job logs and dumps while the problem is happening:
   a. You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b. IBM MQ system dumps, see “IBM MQ for z/OS dumps” on page 276 for more information.
2. You might require the following information:
   a) A MSTR and CHIN internal trace

      See Generate a MSTR internal trace and Generate a CHIN trace
   b) A LOGREC report

      See A LOGREC report
   c) Output from IBM MQ commands, where `cpf` is the command prefix for the queue manager:

      `/cpf` DISPLAY CHINIT
      This shows:
      - Which listeners are started, and information about them.
      - How many dispatchers, adapter subtasks, and SSL subtasks are started, and how many were requested.
      - The TCP system name.
      - How many channel connections are current, and whether they are active, stopped, or retrying.
      - The maximum number of current connections.

      `/cpf` DISPLAY CHSTATUS(xx) CURRENT
      To obtain the current status.

      `/cpf` DISPLAY CHSTATUS(xx) SAVED
      To obtain the saved status.

/cpf DISPLAY CHANNEL(xx)
To obtain the channel definition.

d) Copy SYSTEM.QSG.CHANNEL.SYNCQ

This is where shared channels write their synchronize queue information. See Copying queues into a data set while the queue manager is running for details and an example of the output.

e) A dump of the ADMIN structure as shared channels are serialized applications.

Serialized information is stored in the QSG Admin structure. Issue, for example, the following commands:

```
/dump comm=(title)
/r xx,strlist=(strname=QSGnameStructurename,(listnum=all,
adjunct=capture,entrydata=unser),eventqs,(emcontrols=all))
```

An example of `QSGnameStructurename` is CSQ1CSQ_ADMIN where CSQ1 is the queue manager subsystem name.

f) For outbound channels, a dump of the structure where the shared transmission queue is defined.

g) SPUFI output of the shared channel status Db2 tables. Issue the following commands:

```
SELECT * FROM CSQ.ADMIN_B_SCST ;
SELECT * FROM CSQ.ADMIN_B_SSKT ;
```

See Executing SQL by using SPUFI for more information.

In addition, CSQ45STB in SCSQPROC has sample JCL that you can customize to do the Db2 SELECT commands.

h) z/OS TCP/IP PKTTRACE.

See How to collect Packet Traces and other TCP/IP related traces on z/OS for more information.

3. Collect the IBM MQ data.

a) Include the following information:

- Publish/Subscribe status information, by issuing the command **DISPLAY PUBSUB TYPE**(ALL)
- Topic attributes, by issuing the command **DISPLAY TOPIC**(*) **TYPE**(ALL)
- Subscription attributes, by issuing the command **DISPLAY SUB**(*) **SUBTYPE**(ALL)

b) Record the version and release number of IBM MQ for z/OS and any other related product.

See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

c) Record the operating system version and maintenance level of your system.

For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

5. Send the information that you have collected to IBM.

A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for shared queue problems on z/OS

If you need assistance to resolve a shared queue problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What is the name of the shared queue having a problem?
- What is the name of the IBM MQ Coupling Facility structure (CFSTRUCT) associated with the problematic shared queue?
- What is the message id associated with the problem?
- What is the name of the queue sharing group?
- What time did the problem occur?
- Which queue manager in the queue sharing group is involved?

About this task

If you can reproduce the shared queue problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
2. Collect the data.

   The following steps include an example of dumping both application structure and the IBM MQ Administration structure.

   a) Record the version, release, and maintenance levels your software:
      - For IBM MQ, find the version in the CSQY000I message in the MSTR job log.
      - For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
      - For any other products involved with the problem, look for the version in the job log for the product.

   b) Collect the IBM MQ MSTR and CHIN joblogs and, optionally, collect the syslog.

   c) Collect a z/OS LOGREC report.

      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

   d) Save any z/OS dumps that you generated from IBM MQ.

      IBM MQ dumps are located in a system dump data set and can be identified by their title. The title for a dump requested by IBM MQ starts with the four-character subsystem name of the queue manager. For example:

```
CSQ1,ABN=5C6-00E20016,U=SYSOPR ,C=MQ900.910.DMC
-CSQIALLC,M=CSQGFRCV,LOC=CSQSLD1 .CSQSVSTK+00000712
```

Comm dumps might not contain the queue manager name, depending on the comment specified in the dump command. Check the syslog for an IEA611I or IEA911E message to determine the dump data set name and also to see whether the dump is complete or partial. For example:

```
IEA611I COMPLETE DUMP ON DUMP.MQT1MSTR.DMP00074
DUMPID=074 REQUESTED BY JOB(MQT1MSTR)
FOR ASID(005E)

IEA911E PARTIAL DUMP ON SYS1.MCEVS4.DMP00039
DUMPID=039 REQUESTED BY JOB(DMSGTODI)
FOR ASID(00D2)
```

e) Collect Coupling Facility Structure dumps for the application Structure and the IBM MQ Administration Structure:

```
/DUMP COMM=(title)
/R nnn,SDATA=(ALLNUC,LPA,PSA,RGN,SQA,TRT,CSA,XESDATA,COUPLE,GRSQ),CONT
/R nnn,JOBNAME=(ssidMSTR),CONT
/R nnn,STRLIST=(STRNAME=QSGnameStructurename,(LISTNUM=ALL,
ADJUNCT=CAPTURE,ENTRYDATA=UNSER),EVENTQS,(EMCONTROLS=ALL),
/R nnn,STRNAME=QSGnameCSQ_ADMIN,(LISTNUM=ALL,ADJUNCT=CAPTURE,
ENTRYDATA=UNSER),EVENTQS,(EMCONTROLS=ALL)),END
```

where *ssid* is the subsystem ID for the queue manager.

An example of *QSGnameStructurename* is QSG1APPLICATION where QSG1 is the queue sharing group name.

An example of *QSGnameCSQ_ADMIN* is QSG1CSQ_ADMIN.

The following example shows dumping the application structure solely:

```
/DUMP COMM=(title)
/R xx,STRLIST=(STRNAME=QSGnameStructurename,(LISTNUM=ALL,
ADJUNCT=CAPTURE,ENTRYDATA=UNSER),EVENTQS,(EMCONTROLS=ALL))
```

An example of *QSGnameStructurename* is QSG1APPLICATION where QSG1 is the queue sharing group name.

3. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

4. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for shutdown problems on z/OS

If you need assistance to resolve a shutdown problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- Was shutdown accompanied by an abend in either the queue manager or channel initiator ?
- Was shutdown associated with an apparent wait or hang ?

- Was any rise in CPU noted or possible loop condition?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- See Troubleshooting MQ channels for help in correcting problems with any channels..

About this task

If you can reproduce the shutdown problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. If shutdown abends, provide the dump, otherwise generate the dumps and job logs while the problem is happening:

   See “IBM MQ for z/OS abends” on page 258 for information on the IBM MQ for z/OS abend codes issued.

   a. “Using the z/OS DUMP command” on page 278. See “IBM MQ for z/OS dumps” on page 276 for additional information.

   b. SLIP dump - see Collecting documentation to submit to the IBM Support Center.

   c. You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

2. You might require the following information:

   a) An operations log

   See OPERLOG for more information.

   b) A LOGREC report

   See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

3. Collect the IBM MQ data.

   a) Record the version and release number of IBM MQ for z/OS and any other related product.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for startup problems on z/OS

If you need assistance to resolve a startup problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What startup problem did you observe on the system?
- What changes might have been made prior to the problem?
- Was this a new install or migration, and is the within a queue-sharing group?

Investigate the following:

- Search the IBM MQ Support site for known problems.

About this task

If you can reproduce the startup problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. If startup abends, provide the dump, otherwise generate the dumps and job logs while the problem is happening:

   See “IBM MQ for z/OS abends” on page 258 for information on the IBM MQ for z/OS abend codes issued.

   a. “Using the z/OS DUMP command” on page 278. See “IBM MQ for z/OS dumps” on page 276 for additional information.

   b. A LOGREC report, see A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

   c. You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

2. You might require a SLIP trap, which will be supplied by IBM Support, if you cannot generate any dumps.

   For more information, see Collecting documentation to submit to the IBM Support Center.

3. Collect the IBM MQ data.

   a) Record the version and release number of IBM MQ for z/OS and any other related product.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.

4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.

5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for TLS problems on z/OS

If you need assistance to resolve a TLS problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- Was the TLS issue associated with a channel ?
- If not associated with a channel, what specific error (message) occurred ?
- Did the issue seem to be associated with the currency of the TLS key ring ?

Investigate the following:

- Search the IBM MQ Support site for known problems.
- Provide the expected chain-of-trust on both sides of the TLS channel.
- For messages CSQX633E or CSQX634E displays of the key ring are useful.

This task assumes that you have already configured the SSL started task, which you must do one time before SSL/TLS trace can be run. See Configuring the SSL started task in the z/OS documentation.

About this task

If you can reproduce the TLS problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Generate the following information while the problem is happening:
   a. A CHIN internal trace. See Generate a CHIN trace.
   b. A TLS trace. See Generating System TLS (formerly SSL) trace on IBM MQ for z/OS.
   c. RACDCERT output. See TLS Digital Certificate Commands for External Security Managers:.
2. You might require the following information:
   a) The Syslog, MSTR job log, and CHIN job log.

      The job logs are named xxxxMSTR and xxxxCHIN, where xxxx is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b) z/OS TCP/IP PKTTRACE.

      See How to collect Packet Traces and other TCP/IP related traces on z/OS for more information.
   c) A LOGREC report

      See A LOGREC report.
   d) Output from the following commands:

      Displaying the RACF view of the channel initiator key ring:
      Issue the following command:

      ```
      RACDCERT ID(chinit-user-id) LISTRING(key ring)
      ```

      Displaying the RACF view of a specific certificate:
      Issue the following command:

```
RACDCERT ID(chinit-user-id) list(LABEL('...'))
```

Displaying certificate authorities:
Issue the following command:

```
RACDCERT CERTAUTH LIST
```

3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.
      See message CSQY000I in the MSTR job log for IBM MQ for z/OS.
   b) Record the operating system version and maintenance level of your system.
      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for triggering channels problems on z/OS

If you need assistance to resolve a triggering channels problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What problem did you observe on the system?
- What time did the problem start and has it stopped?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems. using, for example, any error messages or return codes.
- Set a short disconnect interval on the associated channel, The disconnect interval setting stops the channel quickly, with triggering enabled, and make debugging easier.
- For the channel initiator:
  - Make sure that the channel initiator is running as it is the most important piece of the triggering process.
  - On z/OS channel initiator processing is provided by the CHIN started task. You can start the channel initiator using the START CHINIT command, for example:

```
START CHINIT INITQ(SYSTEM.CHANNEL.INITQ)
```

  - Make sure that the channel initiator is monitoring the initiation queue, not the transmission queue.
  - Check the channel initiator log for channel error messages.
- Try to start the channel manually. If the channel fails to starts, or does not successfully move the message from the transmission queue to the remote queue manage, then this is channel problem. See "Collecting information for channel problems on z/OS" on page 366 for more information.

About this task

If you can reproduce the problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) Dumps generated at point of failure.

      See "IBM MQ for z/OS dumps" on page 276

   c) Dumps taken at the point of failure.

      See "Using the z/OS DUMP command" on page 278

   d) A LOGREC report

      See A LOGREC report and "SYS1.LOGREC information on z/OS" on page 294.

   e) A full display of the related object definitions:

      - Queue
      - Remote Transmission queue
      - Initiation queue
      - Sender channel
      - Process, if a process definition is used , although you do not need the actual object definition.
2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

      See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

To open or update a case, go to the IBM My Support site.

Note: Always update your case to indicate that data was sent.

If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for triggering programs problems on z/OS

If you need assistance to resolve a triggering programs problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What problem did you observe on the system?
- What time did the problem start and has it stopped?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Investigate the following:

- Search the IBM MQ Support site for known problems. using, for example, any error messages or return codes.
- For the trigger monitor:
  - Make sure that the trigger monitor is running and monitoring the initiation queue, not the transmission queue.
  - Verify that the user ID used to start the trigger monitor has the authority to access the entire path to the executable code.
- Verify that your applications are putting their messages to the trigger queue, not the initiation queue.
- Try to start the trigger program manually using the string specified in the APPLICID property of your process definition. See DEFINE PROCESS for more information.
- Check for security errors in your External Security Manager, for example, RACF.
- Check logs for your External Security Manager to see if there is a security failure.

About this task

If you can reproduce the problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.

   b) Dumps generated at point of failure.

      See “IBM MQ for z/OS dumps” on page 276.

   c) Dumps taken at the point of failure.

      See “Using the z/OS DUMP command” on page 278.

d) A LOGREC report

See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.

e) A full display of the related object definitions:

- Trigger queue
- Initiation queue
- Process

2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

   See message CSQY000I in the MSTR job log for IBM MQ for z/OS.

   b) Record the operating system version and maintenance level of your system.

   For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

z/OS Collecting information for wait, hang, or loop problems on z/OS

If you need assistance to resolve a wait, hang, or loop problem on IBM MQ for z/OS, you might need to collect troubleshooting information to include with your support case to help find a solution to the problem.

Before you begin

Before you start this task, answer the following questions about the problem:

- What wait, hang, or loop problem did you observe on the system?
- What time did the problem start and when did it stop?
- Which queue managers, channels, remote queues and transmission queues are involved?
- What message number and error codes did you receive?

Search the IBM MQ Support site for known problems. using, for example, any error messages or return codes.

About this task

If you can reproduce the problem or the problem is happening right now, you can generate data to provide more information about the problem.

After collecting the troubleshooting information, you can send it to IBM.

Procedure

1. Collect the following required information:
   a) Job logs

      You require the Syslog, MSTR job log, and CHIN job log. The job logs are named `xxxxMSTR` and `xxxxCHIN`, where `xxxx` is the IBM MQ subsystem identifier (SSID).See Creating a print data set containing the JES2 joblog for the IBM MQ for z/OS jobs.
   b) Dumps generated at point of failure.

      See “IBM MQ for z/OS dumps” on page 276.
   c) Dumps taken at the point of failure.

      See “Using the z/OS DUMP command” on page 278.
   d) A LOGREC report

      See A LOGREC report and “SYS1.LOGREC information on z/OS” on page 294.
2. Optionally, generate the following traces while the problem is happening:
   a. Generate a GTF trace.
   b. Generate a MSTR internal trace.
   c. Generate a CHIN trace.
3. Collect the IBM MQ data.
   a) Record the version and release number of IBM MQ for z/OS and any other related product.

      See message CSQY000I in the MSTR job log for IBM MQ for z/OS.
   b) Record the operating system version and maintenance level of your system.

      For the z/OS operating system, find the version in the output of `/D IPLINFO` in SDSF.
4. Use the AMATERSE utility before uploading to ECUREP, and ensure you specify the case number with which the data is associated.

   For more information, see Using AMATERSE in the *z/OS Communications Server: IP Diagnosis Guide*.
5. Send the information that you have collected to IBM.

   A good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

   For FTP and email instructions, see Exchanging information with IBM Software Support.

   To open or update a case, go to the IBM My Support site.

   Note: Always update your case to indicate that data was sent.

   If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Support in the US, you can call 1-800-IBM-SERV.

Sending troubleshooting information to IBM

After you have generated and collected troubleshooting information for a problem, you can send it to IBM to help with problem determination for a support case.

About this task

When you are sending troubleshooting information, a good description of the problem and the data is the most important information you can provide to IBM. Do not send data without providing a description!

Procedure

- For FTP and email instructions, see Exchanging information with IBM Software Support for problem determination.
- Go to the IBM My Support site to open or update a case.

  Note: Always update your case to indicate that data was sent.

  For more information about IBM Support, including how to register for support, see the IBM Support Guide.
- If you need to speak with IBM Software Support, contact your country representative. If you need to speak with IBM Software Support in the US, you can call 1-800-IBM-SERV.

Related tasks

"Collecting troubleshooting information automatically with runmqras" on page 311
If you need to collect troubleshooting information for a support case, you can use the **runmqras** command to gather the information together into a single archive.

"Collecting troubleshooting information manually" on page 315
In some cases, you might need to collect troubleshooting information manually, for example if you are running an older version of IBM MQ or cannot use the **runmqras** command to collect troubleshooting information automatically.

Using error logs

There are a variety of error logs that you can use to help with problem determination and troubleshooting.

Multi On Multiplatforms, use the following links to find out about the error logs available for your platform and how to use them:

- ALW "Error logs on AIX, Linux, and Windows" on page 407
- IBM i "Error logs on IBM i" on page 410

z/OS On z/OS error messages are written to:

- The z/OS system console
- The channel-initiator job log

For information about error messages, console logs, and dumps on IBM MQ for z/OS, see "Troubleshooting IBM MQ for z/OS problems" on page 255.

For information about error logs in IBM MQ classes for JMS, see "Error logs in IBM MQ classes for JMS" on page 413.

Suppressing or excluding messages from error logs

It is possible to suppress or exclude some messages on both Multiplatforms and z/OS systems:

- Multi For information on suppressing some messages on Multiplatforms, see "Suppressing channel error messages from error logs on Multiplatforms" on page 413.
- z/OS On z/OS, if you are using the z/OS message processing facility to suppress messages, the console messages can be suppressed. For more information, see IBM MQ for z/OS concepts.

AMQ_DIAGNOSTIC_MSG_SEVERITY environment variable

Multi

If the environment variable **AMQ_DIAGNOSTIC_MSG_SEVERITY** is set for an IBM MQ process, when that IBM MQ process writes a message to an error log or to the console, the message severity is appended to the message number as a single uppercase alphabetic character as follows:

Table 22. Key to message severity identifiers	
Type of message	**Character**
Informational (0)	I
Warning (10)	W
Error (20 or 30)	E
Severe (40)	S
Termination (50)	T

For example:

```
AMQ5051I: The queue manager task 'LOGGER-IO' has started.
AMQ7075W: Unknown attribute foo at /var/mqm/qmgrs/QM1/qm.ini in
the configuration data.
AMQ9510E: Messages cannot be retrieved from a queue.
AMQ8506S: Command server MQGET failed with reason code 2009.
AMQ8301T: IBM MQ storage monitor job could not be started.
```

Notes:

1. Because the queue manager writes messages, the environment variable has to be set in the environment where the queue manager is started. This is especially important on Windows, where it might be the Windows service that starts the queue manager.
2. **AMQ_DIAGNOSTIC_MSG_SEVERITY** also affects messages printed by a program.

The behavior that **AMQ_DIAGNOSTIC_MSG_SEVERITY** enables is set by default. You can turn off this behavior by setting the environment variable to `0`.

Note that the new services always add the severity character.

ISO 8601 Time

Multi

The message time is included in ISO 8601 format, rather than in local time.

When IBM MQ processes write a message to an error log, the message time in ISO 8601 format, in Coordinated Universal Time (UTC), is included as a `Time()` attribute.

For example, where the Z time zone indicates UTC:

```
11/04/2017 07:37:59 - Process(1) User(X) Program(amqzmuc0.exe)
                      Host(JOHNDOE) Installation(MQNI09000200)
                      VRMF(9.0.2.0) QMgr(QM1)
                      Time(2017-04-11T07:37:59.976Z)
```

Rename on Rollover

Multi

Prior to IBM MQ 9.1, when AMQERR01.LOG reaches the maximum configured size, AMQERR02.LOG is renamed to be AMQERR03.LOG. The contents of AMQERR01.LOG are then copied into AMQERR02.LOG, and AMQERR01.LOG is truncated to empty. This meant that it was possible for certain tools to miss messages that the tool has not processed, before those messages were copied into AMQERR02.LOG.

From IBM MQ 9.1, the logic is changed, so that AMQERR01.LOG is renamed to AMQERR02.LOG.

Related concepts

“First Failure Support Technology (FFST)” on page 414

First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

"Tracing" on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

ALW Error logs on AIX, Linux, and Windows

The `errors` subdirectory, which is created when you install IBM MQ on AIX, Linux, and Windows, can contain up to three error log files.

At installation time, an `errors` subdirectory is created in the `/var/mqm` file path under AIX and Linux systems, and in the installation directory, for example `C:\Program Files\IBM\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 AIX, Linux, and Windows" on page 409.

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 32 MB (33554432 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 AIX and Linux systems, and `C:\Program Files\IBM\MQ\qmgrs\` *qmname* `\errors` file path under Windows systems.

As error messages are generated, they are placed in AMQERR01. When AMQERR01 gets bigger than 32 MB it is renamed to AMQERR02.

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.

V 10.0.0 You can retrieve the contents of an error log file programmatically by using the PCF command MQCMD_INQUIRE_ERROR_LOG (see MQCMD_INQUIRE_ERROR_LOG (inquire error log) on Multiplatforms).

An example of an error log

Figure 56 on page 408 shows an extract from an IBM MQ error log:

```
17/11/2014 10:32:29 - Process(2132.1) User(USER_1) Program(runmqchi.exe)
Host(HOST_1) Installation(Installation1)
VRMF(8.0.0.0) 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 56. 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 AIX and Linux systems

Linux AIX On AIX 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

Windows 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

"Error logs on z/OS" on page 413
On z/OS, error messages are written to the z/OS system console and the channel-initiator job log.

"First Failure Support Technology (FFST)" on page 414
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

"Tracing" on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

Related reference

"Error logs on IBM i" on page 410
Use this information to understand the IBM MQ for IBM i error logs.

Messages and reason codes

ALW Error log directories on AIX, 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 23 on page 409.

Table 23. Queue manager error log directory

Platform	Directory
Linux AIX AIX and Linux systems	`/var/mqm/qmgrs/ qmname /errors`
Windows 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 24 on page 409.

Table 24. System error log directory

Platform	Directory
Linux AIX AIX and Linux systems	`/var/mqm/errors`
Windows 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 25 on page 410.

Table 25. Client error log directory	
Platform	**Directory**
Linux AIX AIX and Linux systems	`/var/mqm/errors`
Windows Windows systems	`MQ_DATA_PATH\ERRORS\AMQERR01.LOG`

Windows 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\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\MQ`, errors are logged in `C:\Program Files\IBM\MQ\errors`.

For further information about configuration files, see Changing IBM MQ configuration information on Multiplatforms..

Note: Errors in the Windows Registry are notified by messages when a queue manager is started.

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 in the IFS depends on whether the queue manager name is known:

- 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 57 on page 412 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: IBM MQ queue manager 'mick' ended.
EXPLANATION:
Cause . . . . . :   IBM MQ queue manager 'mick' ended.
Recovery  . . . :   None.
Technical Description . . . . . :   None.
----------------------------------------------------------------------------
07/19/02  11:16:48 AMQ7163: IBM 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: IBM 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: IBM 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: IBM 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 57. Extract from an IBM MQ error log

Related concepts

“Error logs on AIX, Linux, and Windows” on page 407
The `errors` subdirectory, which is created when you install IBM MQ on AIX, Linux, and Windows, can contain up to three error log files.

“Error logs on z/OS” on page 413
On z/OS, error messages are written to the z/OS system console and the channel-initiator job log.

“First Failure Support Technology (FFST)” on page 414
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

“Tracing” on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

Related reference

Messages and reason codes

z/OS Error logs on z/OS

On z/OS, error messages are written to the z/OS system console and 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.

Related concepts

"Diagnostic information produced on IBM MQ for z/OS" on page 261
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.

"Other sources of problem determination information for IBM MQ for z/OS" on page 263
Use this topic to investigate other sources of information for IBM MQ for z/OS problem determination.

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

"Error logs on AIX, Linux, and Windows" on page 407
The `errors` subdirectory, which is created when you install IBM MQ on AIX, Linux, and Windows, can contain up to three error log files.

Related reference

"Error logs on IBM i" on page 410
Use this information to understand the IBM MQ for IBM i error logs.

Error logs in IBM MQ classes for JMS

Information about runtime problems that might require corrective action by the user is written to the IBM MQ classes for JMS log.

For example, if an application attempts to set a property of a connection factory, but the name of the property is not recognized, IBM MQ classes for JMS writes information about the problem to its log.

By default, the file containing the log is called `mqjms.log` and is in the current working directory. However, you can change the name and location of the log file by setting the com.ibm.msg.client.commonservices.log.outputName property in the IBM MQ classes for JMS configuration file. For more information about the IBM MQ classes for JMS configuration file, see The IBM MQ classes for JMS/Jakarta Messaging configuration file. For more information about valid values for the com.ibm.msg.client.commonservices.log.outputName property, see "Logging errors for IBM MQ classes for JMS" on page 100.

Related reference

JMS Exception Messages

Multi Suppressing channel error messages from error logs on Multiplatforms

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. This method enables you to suppress the error messages listed in Diagnostic message service stanzas.
- By using the environment variables **MQ_CHANNEL_SUPPRESS_MSGS** and **MQ_CHANNEL_SUPPRESS_INTERVAL**. This method enables you to suppress any channel messages.

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 IBM 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_INTERVAL**, 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 restrictive 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 AIX On AIX 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 concepts

QMErrorLog stanza on AIX, Linux, and Windows

Related reference

Environment variables descriptions

Queue manager properties

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 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 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 420
- Windows "FFST: IBM MQ for Windows" on page 418
- Linux AIX "FFST: IBM MQ for AIX or Linux" on page 415
- IBM i "FFST: IBM MQ for IBM i" on page 416

Related concepts

"Using error logs" on page 405
There are a variety of error logs that you can use to help with problem determination and troubleshooting.

Related tasks

"Tracing" on page 426
You can use different types of trace to help you with problem determination and troubleshooting.

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Linux AIX FFST: IBM MQ for AIX or Linux

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for AIX and Linux systems.

On IBM MQ for AIX or 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.

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 IBM Support.

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 AIX 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).

IBM i 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.

```
------------------------------------------------------------------------------
| IBM 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 :- IBM 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 IBM Support.

Windows 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\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 58 on page 419.

```
+------------------------------------------------------------------------------+
| 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 7 Build 2600: Service Pack 1)          |
| PIDS              :- 5724H7200                                               |
| LVLS              :- 7.0.0.0                                                 |
| Product Long Name :- IBM 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\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 58. 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 IBM Support.

In certain circumstances a small dump file can be generated in addition to an FFST file and placed in the `C:\Program Files\IBM\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 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. The FFST information is recorded whether you are using Jakarta Messaging 3.0 or JMS 2.0.

V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, if the property **com.ibm.msg.client.commonservices.trace.outputName** has been set in the IBM MQ classes for JMS configuration file or as Java system property, the FFST output is directed according to the value specified in the property:

- If `com.ibm.msg.client.commonservices.trace.outputName=stdout`, FFST output is directed to stdout.
- If `com.ibm.msg.client.commonservices.trace.outputName=stderr`, FFST output is directed to stderr.
- If `com.ibm.msg.client.commonservices.trace.outputName=<file>`, FFST output is created as a file in the FFDC directory that is a subdirectory of the directory that this property points to.
- If **com.ibm.msg.client.commonservices.trace.outputName** is not set, FFST output is created as a file in the FFDC directory that is a subdirectory of the current working directory for the IBM MQ classes for JMS application that was running when the FFST was generated.

For more information, see Java Standard Environment Trace stanza for JMS trace and The IBM MQ classes for JMS/Jakarta Messaging configuration file.

You can prevent **com.ibm.msg.client.commonservices.trace.outputName** from determining the FFST destination by overriding it using the `com.ibm.mq.commonservices` properties file, which allows you to redirect FFST output to a different location. For more information, see “Using com.ibm.mq.commonservices” on page 488.

Note: Although it is possible to override this FFST behavior by using the `com.ibm.mq.commonservices` properties file, this is typically only used for tracing IBM MQ Explorer.

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

```
----------------------------------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 2019
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
-----------------------------

Overview of JMS System
Num. Connections : 3
Num. Sessions  : 3
Num. Consumers  : 0
Num. Producers  : 0

Detailed JMS System Information
Connections   :
|  Instance          :- com.ibm.msg.client.wmq.internal.WMQXAConnection@bd4b665a
|  connectOptions       :- version:5 options:64 clientConn:
[channelName:'MY.SVRCONN' version:10 channelType:6 transportType:2 desc:'<null>'
qMgrName:'test' xmitQName:'<null>' connectionName:'9.20.124.119(1414)' mcaName:'<null>'
modeName:'<null>' tpName:'<null>' batchSize:50 discInterval:6000 shortRetryCount:10
shortRetryInterval:60 longRetryCount:999999999 longRetryInterval:1200
seqNumberWrap:99999999 maxMsgLength:104857600 putAuthority:1 dataConversion:0
userIdentifier:'<null>' password:'<null>' mcaUserIdentifier:'<null>' mcaType:1
remoteUserIdentifier:'' msgRetryExit:'<null>' msgRetryUserData:'<null>' msgRetryCount:10
heartbeatInterval:1 batchInterval:0 nonPersistentMsgSpeed:2 clustersDefined:0
networkPriority:0
mcaSecurityId:0000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000 remoteSecurityId:0000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000 sslCipherSpec:'<null>' sslPeerName:'<null>' sslClientAuth:0 keepAliveInterval:-1
localAddress:'<null>' batchHeartbeat:0 hdrCompList:(0,-1)msgCompList:(0,-1,-1,-1,-1,-1,-
1,-1,-1,-1,-1,-1,-1,-1,-1)clwlChannelRank:0 clwlChannelPriority:0 clwlChannelWeight:50
channelMonitoring:0 channelStatistics:0 exitNameLength:128 exitDataLength:32
sendExitsDefined:0 sendExit:'<null>'
sendUserData:0000000000000000000000000000000000000000000000000000000000000000000000000000
sendExitPtr:<null> sendUserDataPtr:<null> receiveExitsDefined:0 receiveExit:'<null>'
receiveUserData:0000000000000000000000000000000000000000000000000000000000000000000000000000
receiveExitPtr:<null> ReceiveUserDataPtr:<null> SharingConversations:999999999
propertyControl:0 maxInstances:999999999 maxInstancesPerClient:999999999
clientChannelWeight:0 connectionAffinity:1 batchDataLimit:5000 useDLQ:2 defReconnect:0 ]
connTag:0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
sslConfig:[version:1 keyRepository:'<null>' cryptoHardware:'<null>' authInfoRecCount:0
keyResetCount:0 fipsRequired:0 encryptionPolicySuiteB:(1,0,0,0)certificateValPolicy:0 ]
connectionId:414D514374657374202020202020208CA3E2522028FD02 securityParms:[<null>]
|  exceptionListener      :-
com.ibm.msg.client.jms.internal.JmsProviderExceptionListener@f17b3583
```

```
|  helper           :-
com.ibm.msg.client.wmq.internal.WMQConsumerOwnerShadow@adabbe93
|  queueManagerName      :- test
...

Sessions     : 3
|  Instance      :- com.ibm.msg.client.wmq.internal.WMQXASession@f5c63f0a
|  Parent Connection :- com.ibm.msg.client.wmq.internal.WMQXAConnection@228b45cb
|  ackMode      :- 0
|  asfConsumer    :- <null>
|  asfDestination   :- <null>
|  asfSubName     :- <null>
|  asyncPutCounter  :-
com.ibm.msg.client.wmq.internal.WMQSession$AsyncPutCounter@88db6ec0
|  didRecovAsyncPut  :- false
|  helper       :-
com.ibm.msg.client.wmq.internal.WMQConsumerOwnerShadow@28192ad1
|  inSyncpoint    :- false
|  queueManagerName  :- test
...

Consumers    :
Producers    :

All Thread Information
Name : DispatchThread:
[com.ibm.mq.jmqi.remote.impl.RemoteSession[connectionId=414D514374657374202020202020202020208
CA3E2522028FA01]]
Priority : 5
ThreadGroup : java.lang.ThreadGroup[name=JMSCCThreadPool,maxpri=10]
ID : 86
State : TIMED_WAITING
Stack : java.lang.Object.wait(Object.java:-2)
: java.lang.Object.wait(Object.java:196)
:
com.ibm.mq.jmqi.remote.impl.RemoteDispatchThread.waitOnSleepingEvent(RemoteDispatchThread
.java:151)
:
com.ibm.mq.jmqi.remote.impl.RemoteDispatchThread.sleepPhase(RemoteDispatchThread.java:636)
:
com.ibm.mq.jmqi.remote.impl.RemoteDispatchThread.run(RemoteDispatchThread.java:385)
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueItem.runTask(WorkQueueItem.java:214)
:
com.ibm.msg.client.commonservices.workqueue.SimpleWorkQueueItem.runItem(SimpleWorkQueueIt
em.java:105)
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueItem.run(WorkQueueItem.java:229)
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueManager.runWorkQueueItem(WorkQueueMa
nager.java:303)
:
com.ibm.msg.client.commonservices.j2se.workqueue.WorkQueueManagerImplementation$ThreadPoo
lWorker.run(WorkQueueManagerImplementation.java:1219)
Name : RcvThread:
com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection@269522111[qmid=test_2014-01-
24_15.55.24,fap=10,channel=MY.SVRCONN,ccsid=850,sharecnv=10,hbint=300,peer=/9.20.124.119(
1414),localport=65243,ssl=no,hConns=0,LastDataSend=1391436871409 (0ms ago
),LastDataRecv=1391436871409 (0ms ago),]
Priority : 5
ThreadGroup : java.lang.ThreadGroup[name=JMSCCThreadPool,maxpri=10]
ID : 84
State : RUNNABLE
Stack :
java.net.SocketInputStream.socketRead0(SocketInputStream.java:-2)
:
java.net.SocketInputStream.read(SocketInputStream.java:163)
:
java.net.SocketInputStream.read(SocketInputStream.java:133)
:
com.ibm.mq.jmqi.remote.impl.RemoteTCPConnection.receive(RemoteTCPConnection.java:1545)
:
com.ibm.mq.jmqi.remote.impl.RemoteRcvThread.receiveBuffer(RemoteRcvThread.java:794)
:
com.ibm.mq.jmqi.remote.impl.RemoteRcvThread.receiveOneTSH(RemoteRcvThread.java:757)
:
com.ibm.mq.jmqi.remote.impl.RemoteRcvThread.run(RemoteRcvThread.java:150)
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueItem.runTask(WorkQueueItem.java:214)
:
com.ibm.msg.client.commonservices.workqueue.SimpleWorkQueueItem.runItem(SimpleWorkQueueIte
m.java:105)
```

```
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueItem.run(WorkQueueItem.java:229)
:
com.ibm.msg.client.commonservices.workqueue.WorkQueueManager.runWorkQueueItem(WorkQueueManager.j
ava:303)
:
com.ibm.msg.client.commonservices.j2se.workqueue.WorkQueueManagerImplementation$ThreadPoo
lWorker.run(WorkQueueManagerImplementation.java:1219)


...
First Failure Symptom Report completed at Mon Feb 03 14:14:46 GMT 2014
-----------------------------------END FFST-----------------------------------
```

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 IBM Support.

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: WCF XMS First Failure Support Technology

You can collect detailed information about what various parts of the IBM MQ code is doing by using IBM MQ trace. XMS FFST has its own configuration and output files for the WCF custom channel.

XMS FFST trace files are traditionally named using the base name and process ID format of: `xmsffdc` *`pid_date`*`.txt`, where *pid* is the process ID and *date* is the time and date.

As XMS FFST trace files can still be produced in parallel with WCF custom channel XMS FFST files, the WCF custom channel XMS FFST output files have the following format to avoid confusion: `wcf ffdc` *`pid_date`*`.txt`, where *pid* is the process ID and *date* is the time and date.

This trace output file is created in the current working directory by default, but this destination can be redefined if necessary.

The WCF custom channel with XMS .NET trace header is similar to the following example:

```
************* Start Display XMS WCF Environment *************
Product Name :- value
WCF Version  :- value
Level  :- value
************* End Display XMS WCF Environment *************
```

The FFST trace files are formatted in the standard way, without any formatting that is specific to the custom channel.

Related tasks

“Tracing the WCF custom channel for IBM MQ” on page 514

You can use IBM MQ trace to collect detailed information about what various parts of the IBM MQ code is doing. When using Windows Communication Foundation (WCF), a separate trace output is generated

for the Microsoft Windows Communication Foundation (WCF) custom channel trace integrated with the Microsoft WCF infrastructure trace.

“Troubleshooting WCF custom channel for IBM MQ problems” on page 250
Troubleshooting information to help you solve problems with running Microsoft Windows Communication Foundation (WCF) custom channel for IBM MQ applications.

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Developing Microsoft Windows Communication Foundation applications with IBM MQ

FFDC configuration for XMS .NET applications

For the .NET implementation of XMS, one FFDC file is produced for each FFDC.

First Failure Data Capture (FFDC) files are stored in human readable text files. These files have names of the form `xmsffdc`*`processID_Date`*`T`*`Timestamp`*`.txt`. An example of a file name is `xmsffdc264_2006.01.06T13.18.52.990955.txt`. The timestamp contains microseconds resolution.

Files start with the date and time that the exception occurred, followed by the exception type. The files include a unique short probeId, which can be used to locate where this FFDC occurred.

You do not need to carry out any configuration to turn on FFDC. By default, all FFDC files are written to the current directory. However, if required, you can specify a different directory by changing ffdcDirectory in the Trace section of the application configuration file. In the following example, all trace files are logged to the directory `c:client\ffdc:`.

```
<IBM.XMS>
      <Trace ffdc=true ffdcDirectory="c:\client\ffdc"/>
</IBM.XMS>
```

You can disable trace by setting FFDC to false in the Trace section of the application configuration file.

If you are not using an application configuration file, FFDC is on and trace is off.

Tracing

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.

- Linux AIX “Tracing on AIX and Linux” on page 427
- IBM i “Tracing on IBM i” on page 433
- Windows “Tracing on Windows” on page 443
- z/OS “Tracing on z/OS” on page 447
- “Tracing the Advanced Message Queuing Protocol (AMQP) Service” on page 464
- “Tracing the IBM MQ Console” on page 466
- “Tracing errors in IBM MQ Internet Pass-Thru” on page 469
- “Tracing IBM MQ .NET applications” on page 471
- “Tracing JMS/Jakarta Messaging and Java applications” on page 476
- Multi “Tracing Managed File Transfer resources on Multiplatforms” on page 490
- z/OS “Tracing Managed File Transfer for z/OS resources” on page 496

- “Tracing the REST API” on page 513
- “Tracing runmqakm” on page 514
- “Tracing the WCF custom channel for IBM MQ” on page 514
- “Tracing XMS .NET applications” on page 515
- “Enabling dynamic tracing of LDAP client library code” on page 522

Related concepts

“Using error logs” on page 405
There are a variety of error logs that you can use to help with problem determination and troubleshooting.

“First Failure Support Technology (FFST)” on page 414
First Failure Support Technology (FFST) for IBM MQ provides information about events that, in the case of an error, can help IBM Support to diagnose the problem.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Linux AIX Tracing on AIX and Linux

On AIX and Linux, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file. On AIX, you can use AIX system trace in addition to using the **strmqtrc** and **endmqtrc** commands.

Before you begin

To format or view a trace file, you must be either the creator of the trace file, or a member of the mqm group.

Before you enable tracing in IBM MQ, ensure that there is plenty of free disk space in the trace directory, otherwise, if this directory gets filled with trace files, it could impact the running of the queue managers due lack of available disk space.

When taking long traces of an IBM MQ queue manager or a C-based IBM MQ client application, you can mitigate any potential disk space issues by using the environment variable **MQTRACEPATH** to point to a directory that has more disk space as shown in the following example:

```
export MQTRACEPATH=/other/mytrace
```

Notes:

- Before setting **MQTRACEPATH**, make sure that the queue manager is not already running, otherwise setting this environment variable will NOT change the location when traces are generated.
- You must ensure that the directory has the same file permissions and ownership as `/var/mqm/trace`.

For more information, see the IBM Support technote Approaches to mitigate the disk space issue when taking long traces for IBM MQ

About this task

Traces can capture the root cause of many IBM MQ problems. It is important to limit how long tracing is active in order to avoid affecting performance or generating excessively large trace files and filling up your disk.

On AIX and Linux, you use the following commands to control the trace facility:

strmqtrc
On AIX and Linux systems, you enable or modify tracing using the **strmqtrc** control command. The command has optional parameters that specify the level of tracing you want. For example:

- For selective component tracing, use the **-t** and **-x** parameters to control the amount of trace detail to record. Specify the points that you want to trace by using the **-t** parameter or the points you do not want to trace by using the **-x** parameter. You can run multiple traces with different parameters at the same time, if necessary.
- For selective process tracing, use the **-p** parameter to restrict trace generation to specified named processes.
- If you need to prevent IBM MQ from including any message data in its traces, use the **-d 0** parameter.

For more information, see strmqtrc (start trace).

endmqtrc
To stop tracing, you use the **endmqtrc** control command. The **endmqtrc** command ends only the trace that is described by its parameters. Using **endmqtrc** with no parameters ends early tracing of all processes. For more information, see endmqtrc (end trace).

dspmqtrc *filename*
Trace output is unformatted; use the **dspmqtrc** control command to format trace output before viewing. For more information, see dspmqtrc (display formatted trace).

Linux On Linux x86-64 systems, you can alternatively use IBM 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**.

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.

Note: Make sure that the `/var/mqm/trace` directory is on a local file system that is separate from the file systems containing your queue managers. Otherwise, you risk an outage if IBM MQ trace fills the file system that your queue managers are using. The trace directory must have sufficient free space since a partial trace might not contain the information that IBM Support needs to solve the problem.

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 this example.
2. There is one trace file for each process running as part of the entity being traced.

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.

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.

Procedure

1. If you have multiple installations of IBM MQ on a single AIX or Linux system, make sure that your environment is set for the installation that you want to trace.

For example, if you are tracing a queue manager or application associated with Installation1, then you must use the **setmqenv** command to switch to Installation1 before running any of the commands to start, stop, or format traces, otherwise, your trace will capture no useful information.

Check the `/etc/opt/mqm/mqinst.ini` file to determine the location of the IBM MQ installations on your system. Use the **setmqenv** command to set your environment for the installation you want to trace.

For example:

Setting your environment for the installation with queue manager QMA

```
/opt/mq93/bin/setmqenv -m QMA
```

Setting your environment for installation Installation2

```
/opt/mq93/bin/setmqenv -n Installation2
```

2. Start tracing with the **strmqtrc** command.

   By default all trace points are enabled and a default-detail trace is generated. The parameters passed to **strmqtrc** control which processes generate trace files and what kinds of data they trace. For example:

   Starting a default detail trace of queue manager QMA

   ```
   strmqtrc -m QMA
   ```

   Starting a trace of queue manager QMA with only output data associated with using Transport Layer Security (TLS) channel security

   ```
   strmqtrc -m QMA -t ssl
   ```

   Starting a default detail trace of queue manager QMB with no message data included

   ```
   strmqtrc -m QMB -d 0
   ```

   Starting an API trace of processes named amqsput with all message data included

   ```
   strmqtrc -t api -p amqsput -d all
   ```

   Starting a high detail trace of everything with files that wrap at 100MB

   ```
   strmqtrc -e -t all -t detail -l 100
   ```

   AIX For selective component tracing on 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 enabled. 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.

   Note: Only set the environment variable **MQS_TRACE_OPTIONS** if you have been instructed to do so by IBM Support.

3. Reproduce the problem while tracing is active, making every attempt to keep the trace as short as possible.

   To check trace status use the **-s** parameter of the **strmqtrc** command:

   ```
   strmqtrc -s
   ```

4. Stop tracing with the **endmqtrc** command.

As soon as the problem has occurred, stop tracing immediately. If you wait to stop the trace, the data that IBM Support needs might be lost or overwritten. You can stop all traces at once, or you can stop each trace individually. For example:

Stopping all traces in an installation

```
endmqtrc -a
```

Stopping a trace of queue manager QMA

```
endmqtrc -m QMA
```

Stopping a trace of processes named amqsput

```
endmqtrc -p amqsput
```

5. Format the traces with the **dspmqtrc** command.

   On AIX and Linux, the trace files that IBM MQ generates are binary, and must be formatted before they can be read. The binary trace files are named like `AMQ*.TRC`, and if you started a wrapping trace, some files might also be named like `AMQ*.TRS`:

   ```
   cd /var/mqm/trace
   dspmqtrc AMQ*.TR?
   ```

6. Clean up the trace directory.

   Clean up the `/var/mqm/trace` directory to reclaim space and also to make sure that you are ready to generate new traces in the future, if necessary. Save a copy of the binary trace files for as long as you are working with IBM Support, but do not send both binary and formatted traces to IBM. Delete the binary trace files (except `AMQ.SSL.TRC` and `AMQ.SSL.TRC.1`, if present) before sending your trace files to IBM.

   ```
   cd /var/mqm/trace
   rm -f AMQ*.?.TR?
   ```

7. Collect the traces and any other troubleshooting information relating to your problem.

   If you are using the **runmqras** command to collect IBM MQ troubleshooting information and you need to include trace data, you must include `trace` in the section names that you specify for the **-section** parameter. If the traces are very large, you might need to use the **-workdirectory** option to package the files in a directory with plenty of free space. For more information, see "Collecting troubleshooting information automatically with runmqras" on page 311 and "Sending troubleshooting information to IBM" on page 404.

8. Send the traces and any other troubleshooting information that you have collected for the same problem to IBM.

   For more information, see "Sending troubleshooting information to IBM" on page 404.

Related concepts

"FFST: IBM MQ for AIX or Linux" on page 415
Describes the name, location, and contents of the First Failure Support Technology (FFST) files for AIX and Linux systems.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

AIX 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.

Note: You should use the *aix* option, only when directed to do so by IBM service personnel.

AIX system tracing is a three-step process:

1. Set the **-o** parameter on the strmqtrc command to *aix*.
2. Gather the data, then run the endmqtrc command.
3. Format 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 Support 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.

Linux AIX Example trace data for AIX and Linux

Extracts from traces file for AIX and Linux.

Example for AIX

AIX

Figure 59 on page 432 shows an extract from an IBM MQ for AIX trace:

```
Timestamp        Process.Thread  Trace Ident  Trace Data
============================================================
 12:06:32.904335   622742.1           :           Header.v02:9.0:AIX 7.2:64:-1:1:GMT
 12:06:32.904427   622742.1           :           Version : 9.0.0.0    Level : p000-L090514
 12:06:32.904540   622742.1           :           UTC   Date : 05/15/16  Time :
11:06:32.904302
 12:06:32.904594   622742.1           :           Local Date : 05/15/16  Time :
12:06:32.904302  GMT
 12:06:32.904697   622742.1           :           PID : 622742 Process : dltmqm_nd (64-bit)
 12:06:32.904728   622742.1           :           Host : dynamo
 12:06:32.904755   622742.1           :           Operating System : AIX 7.3
 12:06:32.904781   622742.1           :           Product Long Name : IBM MQ for AIX
 12:06:32.904806   622742.1           :           --------------------------------
 12:06:32.904832   622742.1           :           xtrNullFd: 3, xihTraceFileNum: 5
 12:06:32.904916   622742.1           :           Data: 0x00000000
 12:06:32.904952   622742.1           :           Thread stack
 12:06:32.904982   622742.1           :           -> InitProcessInitialisation
 12:06:32.905007   622742.1           :      {  InitProcessInitialisation
 12:06:32.905033   622742.1           :      -{  xcsIsEnvironment
 12:06:32.905062   622742.1           :           xcsIsEnvironment[AMQ_NO_CS_RELOAD] = FALSE
 12:06:32.905088   622742.1           :      -}  xcsIsEnvironment rc=OK
 12:06:32.905117   622742.1           :      -{  xcsLoadFunction
 12:06:32.905145   622742.1           :           LibName(libmqmcs_r.a(shr.o))
LoadType(2097200)
 12:06:32.905178   622742.1           :           General, comms, CS, OAM, or WAS
 12:06:32.905204   622742.1           :      --{  xcsQueryValueForSubpool
 12:06:32.905282   622742.1           :      --}  xcsQueryValueForSubpool rc=OK
 12:06:32.905504   622742.1           :           FullPathLibName(/usr/mqm/lib64/
libmqmcs_r.a(shr.o)) loaded with load
 12:06:32.905540   622742.1           :      --{  xcsGetMem
 12:06:32.905575   622742.1           :           component:24 function:176 length:2088
options:0 cbmindex:-1 *pointer:110011408
 12:06:32.905601   622742.1           :      --}  xcsGetMem rc=OK
 12:06:32.905638   622742.1           :           Handle(0) Function(0)
FullPathLibName(/usr/mqm/lib64/libmqmcs_r.a(shr.o))
 12:06:32.905665   622742.1           :      -}  xcsLoadFunction rc=OK
```

Figure 59. Sample IBM MQ for AIX trace

Example for Linux

Linux

Figure 60 on page 433 shows an extract from an IBM MQ for Linux trace:

```
Timestamp        Process.Thread  Trace Ident  Trace Data
============================================================
 11:02:23.643879     1239.1           :           Header.v02:9.0:Linux RHEL Server 7
7.2:64:-1:1:GMT
 11:02:23.643970     1239.1           :           Version : 9.0.0.0    Level : p000-L090514
 11:02:23.644025     1239.1           :           UTC   Date : 05/15/16  Time :
10:02:23.643841
 11:02:23.644054     1239.1           :           Local Date : 05/15/16  Time :
11:02:23.643841  GMT
 11:02:23.644308     1239.1           :           PID : 1239 Process : dltmqm (64-bit)
 11:02:23.644324     1239.1           :           Host : hal1
 11:02:23.644334     1239.1           :           Operating System : RHEL Server 7 7.2
 11:02:23.644344     1239.1           :           Product Long Name : IBM MQ for Linux (x86
platform)
 11:02:23.644353     1239.1           :           --------------------------------
 11:02:23.644363     1239.1           :           xtrNullFd: 3, xihTraceFileNum: 4
 11:02:23.644394     1239.1           :           Thread stack
 11:02:23.644412     1239.1           :           -> InitProcessInitialisation
 11:02:23.644427     1239.1           :      {  InitProcessInitialisation
 11:02:23.644439     1239.1           :      -{  xcsIsEnvironment
 11:02:23.644469     1239.1           :           xcsIsEnvironment[AMQ_NO_CS_RELOAD] = FALSE
 11:02:23.644485     1239.1           :      -}  xcsIsEnvironment rc=OK
 11:02:23.644504     1239.1           :      -{  xcsLoadFunction
 11:02:23.644519     1239.1           :           LibName(libmqmcs_r.so) LoadType(2097200)
 11:02:23.644537     1239.1           :           General, comms, CS, OAM, or WAS
 11:02:23.644558     1239.1           :      --{  xcsQueryValueForSubpool
 11:02:23.644579     1239.1           :      --}  xcsQueryValueForSubpool rc=OK
 11:02:23.644641     1239.1           :           FullPathLibName(/opt/mqm/lib/
libmqmcs_r.so) loaded with dlopen
 11:02:23.644652     1239.1           :      --{  xcsGetMem
 11:02:23.644675     1239.1           :           component:24 function:176 length:8212
options:0 cbmindex:-1 *pointer:0x8065908
 11:02:23.644685     1239.1           :      --}  xcsGetMem rc=OK
 11:02:23.644722     1239.1           :           Handle((nil)) Function((nil))
FullPathLibName(/opt/mqm/lib/libmqmcs_r.so)
 11:02:23.644732     1239.1           :      -}  xcsLoadFunction rc=OK
 11:02:23.644753     1239.1           :           SystemPageSize is 4096.
```

Figure 60. Sample IBM MQ for Linux trace

IBM i Tracing on IBM i

On IBM i, tracing is nearly identical between server and client installations. However, some tracing options are available on server installations only and some options do not apply to a stand-alone client.

About this task

On IBM i, both the server and client support tracing at the IBM i command line by calling the **QMQM/STRMQTRC** and **QMQM/ENDMQTRC** programs, and both support tracing at the IBM i Qshell using the **STRMQTRC**, **ENDMQTRC** and **DSPMQTRC** commands.

However, only the IBM MQ server installation for IBM i provides the **TRCMQM** command. Furthermore, a stand-alone client does not support the **-m** parameter on either the start or end trace commands, since there are no queue managers. The **runmqras -qmlist** parameter is not valid on a stand-alone client for the same reason.

Related concepts

“FFST: IBM MQ for IBM i” on page 416
Describes the name, location, and contents of the First Failure Support Technology (FFST) files for IBM i systems.

Related tasks

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

IBM i Using trace with IBM MQ server on IBM i

If you have an IBM MQ server installation on IBM i, you can use the **TRCMQM** command to start and stop tracing and specify the type of trace that you require. Alternatively, you can control tracing from the IBM i command line by calling the **QMQM/STRMQTRC** and **QMQM/ENDMQTRC** programs, or in the IBM i Qshell using the **STRMQTRC**, **ENDMQTRC** and **DSPMQTRC** commands.

About this task

Traces can capture the root cause of many IBM MQ problems. It is important to limit how long tracing is active in order to avoid affecting performance or generating excessively large trace files and filling up your disk.

Early tracing lets you trace the creation and startup of queue managers. However, early trace can easily generate large amounts of trace, because it is implemented by tracing all jobs for all queue managers.

If you have an IBM MQ server installation on IBM i, the **TRCMQM** command is the most commonly used way of starting and stopping trace. You can use the optional parameters to specify the level of tracing you want. For example:

- To specify the level of detail you want, use the **TRCLEVEL** parameter.
- To specify the type of trace output you want, use the **OUTPUT** parameter.
- For selective component tracing, use the **TRCTYPE** and **-EXCLUDE** parameters to reduce the amount of trace data being saved, improving runtime performance. Specify the points that you want to trace by using the **TRCTYPE** parameter or the points you do not want to trace by using the **-EXCLUDE** parameter. If you omit the **TRCTYPE** parameter, all trace points are enabled. If you omit the **-EXCLUDE** parameter, all trace points specified in **TRCTYPE** are enabled.
- To wrap trace, and to specify the maximum size of storage to be used for the collected trace records, use the **MAXSTG** parameter.
- For selective process tracing, use the **JOB** parameter to restrict trace generation to specified named processes.
- If you need to prevent IBM MQ from including any message data in its traces, use the **DATASIZE(*NONE)** parameter.

For more information, see TRCMQM (Trace MQ).

Note: You can either enter the **TRCMQM** command manually as shown in the examples in this task, or you can type TRCMQM and press F4 followed by F9 to enable tracing using the TRCMQM panel.

You can also use the following commands to control the trace facility:

Starting a trace

You can start tracing at the IBM i command line by calling the **QMQM/STRMQTRC** program, or at the IBM i Qshell by using the **STRMQTRC** command. You can use the optional parameters to specify the level of tracing you want. For example:

- For selective component tracing, use the **-t** and **-x** parameters to control the amount of trace detail to record. Specify the points that you want to trace by using the **-t** parameter or the points you do not want to trace by using the **-x** parameter. You can run multiple traces with different parameters at the same time, if necessary. If you supply multiple trace types, each must have its own **-t** or **-x** flag. You can include any number of **-t** or **-x** 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** or **-x** flags.
- For selective process tracing, use the **-p** parameter to restrict trace generation to specified named processes.
- If you need to prevent IBM MQ from including any message data in its traces, use the **-d 0** parameter.

For more information, see strmqtrc (start trace).

Ending a trace

You can end tracing at the IBM i command line by calling the **QMQM/ENDMQTRC** program, or at the IBM i Qshell by using the **ENDMQTRC** command. You can end early tracing of all processes by specifying no parameters, or you can end a specific type of trace by including the appropriate parameter. For more information, see endmqtrc (end trace).

Formatting trace

The trace files generated by IBM MQ are binary. You can use the **DSPMQTRC** command at the IBM i Qshell to format trace output before sending it to IBM. For more information, see dspmqtrc (display formatted trace).

Procedure

1. Start tracing.

   By default all trace points are enabled and a default-detail trace is generated. To control which processes generate trace files and what kinds of data they trace, specify the appropriate parameters.

   For example:

   Starting a default detail trace

   - Using the **TRCMQM** command:

     ```
     TRCMQM SET(*ON)
     ```

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC)
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM
     ```

   Starting a default detail trace of queue manager QMA

   - Using the **TRCMQM** command:

     ```
     TRCMQM SET(*ON) MQMNAME(QMA)
     ```

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-m' 'QMA')
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -m QMA
     ```

   Starting a default detail trace of queue manager QMB with no message data included

   - Using the **TRCMQM** command:

     ```
     TRCMQM SET(*ON) MQMNAME(QMB) DATASIZE(*NONE)
     ```

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-m' 'QMB' '-d' '0')
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -m QMB -d 0
     ```

Starting an API trace of jobs named AMQSPUT with all data

- Using the **TRCMQM** command:

```
TRCMQM SET(*ON) TRCTYPE(*API) JOB(AMQSPUT) DATASIZE(*ALL)
```

- Using the command line:

```
CALL PGM(QMQM/STRMQTRC) PARM('-t' 'api' '-p' 'AMQSPUT' '-d' 'all')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -t api -p AMQSPUT -d all
```

Starting a high detail early trace with files that wrap at 16MB

- Using the **TRCMQM** command:

```
TRCMQM SET(*ON) TRCEARLY(*YES) TRCLEVEL(*DETAIL) MAXSTG(16)
```

- Using the command line:

```
CALL PGM(QMQM/STRMQTRC) PARM('-e' '-t' 'all' '-t' 'detail' '-l' '16')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -e -t all -t detail -l 16
```

2. Reproduce the problem while tracing is active, making every attempt to keep the trace as short as possible.

   To check trace status, use one of the following options:

   - Use the **SET** parameter of the **TRCMQM** command:

```
TRCMQM SET(*STS)
WRKSPLF SELECT(QMQM)
```

     Use option 5 to view the **TRCMQM** spool file, which contains the trace status.

   - On the command line, use the **-s** parameter of the **QMQM/STRMQTRC** program:

```
CALL PGM(QMQM/STRMQTRC) PARM('-s')
```

   - In the Qshell, use the **-s** parameter of the **STRMQTRC** command:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -s
```

3. Stop tracing with the **TRCMQM** command.

   As soon as the problem has occurred, stop tracing immediately. If you wait to stop the trace, the data that IBM Support needs might be lost or overwritten.

   For example:

   Stopping all traces

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-a')
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -a
```

     There is no equivalent in the **TRCMQM** command.

Stopping a default detail trace

- Using the **TRCMQM** command:

```
TRCMQM SET(*OFF)
```

- Using the command line:

```
CALL PGM(QMQM/ENDMQTRC)
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM
```

Using **endmqtrc** with no parameters has the same effect as **endmqtrc -e**. You cannot specify the **-e** parameter with the **-m** parameter, the **-i** parameter, or the **-p** parameter.

Stopping a trace of queue manager QMA

- Using the **TRCMQM** command:

```
TRCMQM SET(*OFF) MQMNAME(QMA)
```

- Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-m' 'QMA')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -m QMA
```

Stopping a trace of jobs named AMQSPUT

- Using the **TRCMQM** command:

```
TRCMQM SET(*OFF) JOB(AMQSPUT)
```

- Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-p' 'AMQSPUT')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -p AMQSPUT
```

Stopping an early trace

- Using the **TRCMQM** command:

```
TRCMQM SET(*OFF) TRCEARLY(*YES)
```

- Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-e')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -e
```

4. Optional: Format the binary trace files.

   The trace files generated by IBM MQ are binary, and must be formatted before they can be read. You can either send the binary traces to IBM, or you can format them at the Qshell at a time when your system is not under heavy load. For example:

```
/QSYS.LIB/QMQM.LIB/DSPMQTRC.PGM [-t FormatTemplate] [-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` omits header information from the report.
- `-s` extracts trace header and puts it to stdout.
- `-o` *output_filename* specifies the name of the file into which to write formatted data.

You can also specify `dspmqtrc *` to format all trace.

If you format the traces yourself, delete the binary trace files before sending your files to IBM:

- Using the command line to delete the binary trace files:

```
RMVLNK OBJLNK('/QIBM/UserData/mqm/trace/AMQ*.?.TR?')
```

- Using the Qshell to display a formatted trace file using the **DSPMQTRC** command and then delete the binary trace files:

```
cd /QIBM/UserData/mqm/trace
/QSYS.LIB/QMQM.LIB/DSPMQTRC.PGM AMQ*.TR?
rm -f AMQ*.?.TR?
```

5. Collect the traces and any other troubleshooting information relating to your problem.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. If you are using the **runmqras** command to collect IBM MQ troubleshooting information and you need to include trace data, you must include `trace` in the section names that you specify for the **-section** parameter. If the traces are very large, you might need to use the **-workdirectory** option to package the files in a directory with plenty of free space.

     You must run the **runmqras** command at the Qshell as shown in the following example:

     ```
     /QIBM/ProdData/mqm/bin/runmqras -section trace,defs,cluster -caseno TS001234567 -qmlist
     QMA
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

6. Send the traces and any other troubleshooting information that you have collected for the same problem to IBM.

   For more information, see "Sending troubleshooting information to IBM" on page 404.

7. Clean up the trace directory to reclaim space and also to make sure you are ready to generate new traces in future, if necessary.

   - Using the command line:

     ```
     RMVLNK OBJLNK('/QIBM/UserData/mqm/trace/AMQ*.*')
     ```

   - Using the Qshell:

     ```
     cd /QIBM/UserData/mqm/trace
     rm -f AMQ*.*
     ```

Related concepts

"FFST: IBM MQ for IBM i" on page 416

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for IBM i systems.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

IBM i Using trace with an IBM MQ client on IBM i

On IBM i, you can control tracing for an IBM MQ MQI client at the IBM i command line by calling the **QMQM/STRMQTRC** and **QMQM/ENDMQTRC** programs, or at the IBM i Qshell using the **STRMQTRC**, **ENDMQTRC** and **DSPMQTRC** commands.

Before you begin

An IBM MQ server installation on IBM i can include the client, or you might choose to install a stand-alone client on IBM i with no IBM MQ server functionality. Since there are no queue managers in the case of a stand-alone client, the following parameters are not supported:

- The **-m** parameter on either the start or end trace command.
- The **-qmlist** parameter on the **runmqras**.

There is no Control Language (CL) command to capture the trace when using a stand-alone IBM MQ client.

To use the Qshell, you type STRQSH at the IBM i command line. You can exit and return to the command line at any time by pressing F3. In order to avoid typing the full path for commands like **STRMQTRC.PGM** and **runmqras**, you can add the IBM MQ directories `/QSYS.LIB/QMQM.LIB` and `/QIBM/ProdData/mqm/bin` to your PATH as shown in the following example:

```
export PATH="$PATH:/QSYS.LIB/QMQM.LIB:/QIBM/ProdData/mqm/bin"
```

The examples in this task give the full path names for all commands, but you do not need to type them out if you update your PATH.

About this task

Traces can capture the root cause of many IBM MQ problems. It is important to limit how long tracing is active in order to avoid affecting performance or generating excessively large trace files and filling up your disk.

On IBM i, you use the following commands to control the trace facility for an IBM MQ client:

Starting a trace

You can start tracing at the IBM i command line by calling the **QMQM/STRMQTRC** program, or at the IBM i Qshell by using the **STRMQTRC** command. You can use the optional parameters to specify the level of tracing you want. For example:

- For selective component tracing, use the **-t** and **-x** parameters to control the amount of trace detail to record. Specify the points that you want to trace by using the **-t** parameter or the points you do not want to trace by using the **-x** parameter. You can run multiple traces with different parameters at the same time, if necessary. If you supply multiple trace types, each must have its own **-t** or **-x** flag. You can include any number of **-t** or **-x** 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** or **-x** flags.
- For selective process tracing, use the **-p** parameter to restrict trace generation to specified named processes.
- If you need to prevent IBM MQ from including any message data in its traces, use the **-d 0** parameter.

For more information, see strmqtrc (start trace).

Ending a trace

You can end tracing at the IBM i command line by calling the **QMQM/ENDMQTRC** program, or at the IBM i Qshell by using the **ENDMQTRC** command. You can end early tracing of all processes by specifying no parameters, or you can end a specific type of trace by including the appropriate parameter. For more information, see endmqtrc (end trace).

Formatting trace

The trace files generated by IBM MQ are binary. You can use the **DSPMQTRC** command at the IBM i Qshell to format trace output before sending it to IBM. For more information, see dspmqtrc (display formatted trace).

Procedure

1. Start tracing at the IBM i command line by calling the **QMQM/STRMQTRC** program, or at the IBM i Qshell by using the **STRMQTRC** command.

   By default all trace points are enabled and a default-detail trace is generated. To control which processes generate trace files and what kinds of data they trace, specify the appropriate parameters. For example:

   Starting a default detail trace

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC)
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM
     ```

   Starting a default detail trace of queue manager QMA

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-m' 'QMA')
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -m QMA
     ```

   Starting a default detail trace of queue manager QMB with no message data included

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-m' 'QMB' '-d' '0')
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -m QMB -d 0
     ```

   Starting an API trace of jobs named AMQSPUT with all data included

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-t' 'api' '-p' 'AMQSPUT' '-d' 'all')
     ```

   - Using the Qshell:

     ```
     /QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -t api -p AMQSPUT -d all
     ```

   Starting a high detail trace early trace with files that wrap at 16MB

   - Using the command line:

     ```
     CALL PGM(QMQM/STRMQTRC) PARM('-e' '-t' 'all' '-t' 'detail' '-l' '16')
     ```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -e -t all -t detail -l 16
```

2. Reproduce the problem while tracing is active, making every attempt to keep the trace as short as possible.

   To check trace status use the **-s** parameter of the **QMQM/STRMQTRC** program or **STRMQTRC** command:

   - Using the command line:

```
CALL PGM(QMQM/STRMQTRC) PARM('-s')
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -s
```

3. Stop tracing with the **QMQM/ENDMQTRC** program or **ENDMQTRC** command.

   As soon as the problem has occurred, stop tracing immediately. If you wait to stop the trace, the data that IBM Support needs might be lost or overwritten. You can stop all traces at once, or you can stop each trace individually. For example:

   Stopping all traces

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-a')
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -a
```

   Stopping a default detail trace

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC)
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM
```

   Using **endmqtrc** with no parameters has the same effect as **endmqtrc -e**. You cannot specify the **-e** parameter with the **-m** parameter, the **-i** parameter, or the **-p** parameter.

   Stopping a trace of queue manager QMA

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-m' 'QMA')
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -m QMA
```

   Stopping a trace of jobs named AMQSPUT

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-p' 'AMQSPUT')
```

   - Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/ENDMQTRC.PGM -p AMQSPUT
```

   Stopping an early trace

   - Using the command line:

```
CALL PGM(QMQM/ENDMQTRC) PARM('-e')
```

- Using the Qshell:

```
/QSYS.LIB/QMQM.LIB/STRMQTRC.PGM -e
```

4. Optional: Format the binary trace files.

   The trace files generated by IBM MQ are binary, and must be formatted before they can be read. You can either send the binary traces to IBM, or you can format them at the Qshell at a time when your system is not under heavy load. If you format the traces yourself, delete the binary trace files before sending your files to IBM:

   - Using the command line to delete the binary trace files:

     ```
     RMVLNK OBJLNK('/QIBM/UserData/mqm/trace/AMQ*.?.TR?')
     ```

   - Using the Qshell to display a formatted trace file using the **DSPMQTRC** command and then delete the binary trace files:

     ```
     cd /QIBM/UserData/mqm/trace
     /QSYS.LIB/QMQM.LIB/DSPMQTRC.PGM AMQ*.TR?
     rm -f AMQ*.?.TR?
     ```

5. Collect the traces and any other troubleshooting information relating to your problem.

   You can do this either automatically or manually:

   - Collect the data automatically by using the **runmqras** command as described in "Collecting troubleshooting information automatically with runmqras" on page 311. If you are using the **runmqras** command to collect IBM MQ troubleshooting information and you need to include trace data, you must include `trace` in the section names that you specify for the **-section** parameter. If the traces are very large, you might need to use the **-workdirectory** option to package the files in a directory with plenty of free space.

     You must run the **runmqras** command at the Qshell as shown in the following example:

     ```
     /QIBM/ProdData/mqm/bin/runmqras -section trace,defs,cluster -caseno TS001234567 -qmlist
     QMA
     ```

   - Alternatively, collect the data manually as described in "Collecting troubleshooting information manually" on page 315.

6. Send the traces and any other troubleshooting information that you have collected for the same problem to IBM.

   For more information, see "Sending troubleshooting information to IBM" on page 404.

7. Clean up the trace directory to reclaim space and also to make sure you are ready to generate new traces in future, if necessary.

   - Using the command line:

     ```
     RMVLNK OBJLNK('/QIBM/UserData/mqm/trace/AMQ*.*')
     ```

   - Using the Qshell:

     ```
     cd /QIBM/UserData/mqm/trace
     rm -f AMQ*.*
     ```

Related concepts

"FFST: IBM MQ for IBM i" on page 416
Describes the name, location, and contents of the First Failure Support Technology (FFST) files for IBM i systems.

Related tasks

"Collecting troubleshooting information" on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Windows Tracing on Windows

On Windows, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing. You can also use IBM MQ Explorer to start and end tracing.

Before you begin

To format or view a trace file, you must be either the creator of the trace file, or a member of the mqm group.

Before you enable tracing in IBM MQ, ensure that there is plenty of free disk space in the trace directory, otherwise, if this directory gets filled with trace files, it could impact the running of the queue managers due lack of available disk space.

When taking long traces of an IBM MQ queue manager or a C-based IBM MQ client application, you can mitigate any potential disk space issues by using the environment variable **MQTRACEPATH** to point to a directory that has more disk space as shown in the following example:

```
set MQTRACEPATH=D:\mqtrace
```

Notes:

- Before setting **MQTRACEPATH**, make sure that the queue manager is not already running, otherwise setting this environment variable will NOT change the location when traces are generated.
- You must ensure that the directory has the same file permissions and ownership as `/var/mqm/trace`.

For more information, see the IBM Support technote Approaches to mitigate the disk space issue when taking long traces for IBM MQ

About this task

Traces can capture the root cause of many IBM MQ problems. It is important to limit how long tracing is active in order to avoid affecting performance or generating excessively large trace files and filling up your disk.

On Windows, you use the following commands to control the trace facility:

strmqtrc
On Windows systems, you enable or modify tracing using the **strmqtrc** control command. The command has optional parameters that specify the level of tracing you want. For example:

- For selective component tracing, use the **-t** and **-x** parameters to control the amount of trace detail to record. Specify the points that you want to trace by using the **-t** parameter or the points you do not want to trace by using the **-x** parameter. You can run multiple traces with different parameters at the same time, if necessary.
- For selective process tracing, use the **-p** parameter to restrict trace generation to specified named processes.
- To prevent IBM MQ from including any message data in its traces, use the **-d 0** parameter.

For more information, see strmqtrc (start trace).

endmqtrc
To stop tracing, you use the **endmqtrc** control command. The **endmqtrc** command ends only the trace that is described by its parameters. Using **endmqtrc** with no parameters ends early tracing of all processes. For more information, see endmqtrc (end trace).

On Windows systems, you can alternatively use IBM MQ Explorer to start and stop tracing as follows:

1. Start the IBM 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.

The output files are created in the *MQ_DATA_PATH*/trace 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.

SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1.

The trace files generated by IBM MQ on Windows are readable text files (except for AMQ.SSL.TRC and AMQ.SSL.TRC.1, if present). On Windows, trace files do not need to be formatted before you send them to IBM Support.

Procedure

1. If you have multiple installations of IBM MQ on a single Windows system, make sure that your environment is set for the installation that you want to trace.

   For example, if you are tracing a queue manager or application associated with Installation1, then you must use the **setmqenv** command to switch to Installation1 before running any of the commands to start, stop, or format traces, otherwise, your trace will capture no useful information.

   Check the /etc/opt/mqm/mqinst.ini file to determine the location of the IBM MQ installations on your system. Use the **setmqenv** command to set your environment for the installation you want to trace.

   For example:

   Setting your environment for Installation1

   ```
   "C:\Program Files\IBM\MQ\bin\setmqenv" -n Installation1
   ```

   Setting your environment for the installation with queue manager QMA

   ```
   "C:\Program Files\IBM\MQ\bin\setmqenv" -m QMA
   ```

   The traces are written to the trace subdirectory under the path given by the **MQ_DATA_PATH** environment variable. After running **setmqenv**, you can find the location of the trace directory by running:

   ```
   echo "%MQ_DATA_PATH%\trace"
   ```

2. Start tracing with the **strmqtrc** command.

   By default all trace points are enabled and a default-detail trace is generated. The parameters passed to **strmqtrc** control which processes generate trace files and what kinds of data they trace. For example:

   Tracing only data flowing over communications networks

   ```
   strmqtrc -x all -t comms
   ```

Starting a default detail trace of queue manager QMA

```
strmqtrc -m QMA
```

Starting a default detail trace of queue manager QMB with no message data included

```
strmqtrc -m QMB -d 0
```

Starting an API trace of processes named amqsput with all message data included

```
strmqtrc -t api -p amqsput -d all
```

Starting a high detail trace of everything with files that wrap at 100MB

```
strmqtrc -e -t all -t detail -l 100
```

3. Reproduce the problem while tracing is active, making every attempt to keep the trace as short as possible.

   To check trace status use the **-s** parameter of the **strmqtrc** command:

```
strmqtrc -s
```

4. Stop tracing with the **endmqtrc** command.

   As soon as the problem has occurred, stop tracing immediately. If you wait to stop the trace, the data that IBM Support needs might be lost or overwritten. You can stop all traces at once, or you can stop each trace individually. For example:

   Stopping all traces in an installation

```
endmqtrc -a
```

   Stopping a trace of queue manager QMA

```
endmqtrc -m QMA
```

   Stopping a trace of processes named amqsput

```
endmqtrc -p amqsput
```

5. Collect the traces and any other troubleshooting information relating to your problem.

   If you are using the **runmqras** command to collect IBM MQ troubleshooting information and you need to include trace data, you must include `trace` in the section names that you specify for the **-section** parameter. If the traces are very large, you might need to use the **-workdirectory** option to package the files in a directory with plenty of free space. For more information, see “Collecting troubleshooting information automatically with runmqras” on page 311.

6. Send the traces and any other troubleshooting information that you have collected for the same problem to IBM.

   For more information, see “Sending troubleshooting information to IBM” on page 404.

7. Clean up the trace directory

   Save a copy of the trace files as long as you are working with IBM Support, but keep them somewhere other than the trace directory. Clean up the trace directory to reclaim space and also to make sure you are ready to generate new traces in future, if necessary. For example:

```
C:\> CD "%MQ_DATA_PATH%\trace"
C:\ProgramData\IBM\MQ\trace> DEL AMQ*.TRC AMQ*.TRS
```

Related concepts

“FFST: IBM MQ for Windows” on page 418

Describes the name, location, and contents of the First Failure Support Technology (FFST) files for Windows systems.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Windows Example trace data for Windows

An extract from an IBM MQ for Windows trace file.

```
TimeStamp         PID.TID    Ident         Data
============================================================
10:55:33.033870   4996.1           :       ---{  zutLookupInitialize
10:55:33.033877   4996.1           :       ----{  xcsCreateThreadMutexSem
10:55:33.033889   4996.1           :       hmtx: 000001DD32A9E0A0, created: TRUE
10:55:33.033896   4996.1           :       ----}  xcsCreateThreadMutexSem (rc=OK)
10:55:33.033903   4996.1           :       ----{  xcsGetMemFn
10:55:33.033911   4996.1           :       Data: 0x000001dd 0x32ab1b30
10:55:33.033923   4996.1           :       component:33 function:431 length:496 options:0
cbmindex:-1 *pointer:000001DD32AB1B30
10:55:33.033932   4996.1           :       ----}  xcsGetMemFn (rc=OK)
10:55:33.033985   4996.1           :       ---}  zutLookupInitialize (rc=OK)
10:55:33.034004   4996.1           :       ---{  xcsGetEnvironmentInteger
10:55:33.034012   4996.1           :       ----{  xcsGetEnvironmentString
10:55:33.034027   4996.1           :       xcsGetEnvironmentString[AMQ_BACKWARDS_TIME_LIMIT] =
NULL
10:55:33.034034   4996.1           :       ----}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034065   4996.1           :       ---}! xcsGetEnvironmentInteger
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034073   4996.1           :       ---{  xcsReleaseThreadMutexSem
10:55:33.034078   4996.1           :       hmtx: 000001DD32A9DE90
10:55:33.034086   4996.1           :       ---}  xcsReleaseThreadMutexSem (rc=OK)
10:55:33.034089   4996.1           :       ---{  xcsGetEnvironmentString
10:55:33.034099   4996.1           :       xcsGetEnvironmentString[AMQ_REUSE_SHARED_THREAD] =
NULL
10:55:33.034106   4996.1           :       ---}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034114   4996.1           :       ---{  xcsGetEnvironmentInteger
10:55:33.034118   4996.1           :       ----{  xcsGetEnvironmentString
10:55:33.034124   4996.1           :       xcsGetEnvironmentString[AMQ_AFFINITY_MASK] = NULL
10:55:33.034131   4996.1           :       ----}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034138   4996.1           :       ---}! xcsGetEnvironmentInteger
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034146   4996.1           :       ---{  xcsGetEnvironmentString
10:55:33.034153   4996.1           :       xcsGetEnvironmentString[AMQ_FFSTINFO] = NULL
10:55:33.034160   4996.1           :       ---}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034168   4996.1           :       ---{  xcsGetEnvironmentString
10:55:33.034176   4996.1           :       xcsGetEnvironmentString[AMQ_CHECK_SEM_OBJECTS] = NULL
10:55:33.034183   4996.1           :       ---}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034191   4996.1           :       ---{  xcsGetEnvironmentString
10:55:33.034199   4996.1           :
xcsGetEnvironmentString[AMQ_OVERRIDE_CONVERSION_TABLE] = NULL
10:55:33.034207   4996.1           :       ---}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034215   4996.1           :       ---{  xcsGetEnvironmentString
10:55:33.034223   4996.1           :       xcsGetEnvironmentString[AMQ_OVERRIDE_CCSID_TABLE] =
NULL
10:55:33.034230   4996.1           :       ---}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034237   4996.1           :       ---{  xcsGetEnvironmentInteger
10:55:33.034241   4996.1           :       ----{  xcsGetEnvironmentString
10:55:33.034248   4996.1           :       xcsGetEnvironmentString[AMQ_CBM_REUSE_FACTOR] = NULL
10:55:33.034255   4996.1           :       ----}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034262   4996.1           :       ---}! xcsGetEnvironmentInteger
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034270   4996.1           :       ---{  xcsGetEnvironmentInteger
10:55:33.034274   4996.1           :       ----{  xcsGetEnvironmentString
10:55:33.034282   4996.1           :       xcsGetEnvironmentString[AMQ_CBM_MAX_CACHEABLE_SIZE]
= NULL
10:55:33.034289   4996.1           :       ----}! xcsGetEnvironmentString
```

```
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034296   4996.1          :       ---}! xcsGetEnvironmentInteger
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034304   4996.1          :       ---{  xcsGetEnvironmentInteger
10:55:33.034308   4996.1          :       ----{  xcsGetEnvironmentString
10:55:33.034314   4996.1          :       xcsGetEnvironmentString[AMQ_CBM_LEN] = NULL
10:55:33.034322   4996.1          :       ----}! xcsGetEnvironmentString
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034330   4996.1          :       ---}! xcsGetEnvironmentInteger
(rc=xecE_E_ENV_VAR_NOT_FOUND)
10:55:33.034337   4996.1          :       --}  InitProcessInitialisation (rc=OK)
```

z/OS Tracing 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.

About this task

The trace facilities available with IBM MQ for z/OS are:

- The channel initiator trace
- The CICS adapter trace
- The generalized trace facility (GTF) trace.

  If trace data is not produced, check the following:

  – Was the GTF started correctly, specifying event identifiers (EIDs) 5E9, 5EA, and 5EE on the USRP option?

  – Was the **START TRACE**(GLOBAL) command entered correctly, and were the relevant classes specified?

  See "Using the GTF with IBM MQ global trace" on page 448 for more information.
- The IBM internal trace used by the support center
- The line trace
- System TLS (formerly SSL) trace
- The user parameter (or API) trace
- z/OS traces

For more information, see "Other types of trace on z/OS" on page 454.

See the sub topics listed in "Collecting troubleshooting information on z/OS" on page 361 for more information on what you need to send to IBM Support when you have a specific problem that requires resolving.

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 IBM Support. There is also information about the other trace facilities that you can use with IBM MQ.

Related tasks

"Collecting troubleshooting information" on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

z/OS Using the GTF with IBM MQ global trace

On z/OS, 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 IBM MQ global trace in conjunction with the z/OS generalized trace facility (GTF).

Before you begin

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.

Procedure

- To start and stop the GTF
  a) Start the GTF at the console by entering a **START GTF** command.

  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 IBM Support 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. The following example shows a sample startup for the GTF, specifying the four EIDs, and a jobname. The lines shown in **bold** 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 Starting GTF.

```
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)
- yy is an identifier for this occurrence of GTF trace
- xxxx is the name of the queue manager
- zzzzzzzz is a batch job or CICS region name

Up to 5 job names can be listed.

When using GTF, specify the primary job name (`CHINIT`, `CICS`, or `batch`) in addition to the queue manager name (`xxxxMSTR`).

b) Stop the GTF at the console.

When you enter the stop command for the GTF, include the additional identifier (*yy*) that you used at startup, as shown in the following example:

```
STOP yy
```

- To use IBM MQ global trace

a) To start writing IBM MQ records to the GTF, use the **START TRACE** command, specifying trace type **GLOBAL**.

You must also specify **dest(GTF)** as shown in the following example:

```
/cpf start trace(GLOBAL)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.

Attention: Note the following:

- You can use more than one of the classes that are allowed for the type of trace started. You specify a range of classes as m:n, for example, CLASS(1:3).
- If you do not specify a class, the default is to start class 1.
- However, if you use the **START TRACE**(STAT) command with no class, the default is to start class 1 and 2.

b) After the trace has started, use the following MQSC commands to display information about or alter the properties of the trace:

- Use **DISPLAY TRACE** to display a list of active traces.
- Use **ALTER TRACE** to change the trace events being traced for a particular active queue manager trace. **ALTER TRACE** stops the specified trace, and restarts it with the altered parameters.

c) To stop tracing, use the **STOP TRACE** command.

For example:

```
/cpf STOP TRACE(GLOBAL) DEST(GTF) CLASS(*) RMID(*)
```

In this command, `/cpf` refers to the command prefix for the IBM MQ subsystem.

z/OS *Formatting and identifying the control block information on z/OS*

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 that is collected by the global trace, use either the batch job that is shown in Figure 61 on page 450 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 61. 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 start of each control block. These messages 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 tasks

"Using the GTF with IBM MQ global trace" on page 448

On z/OS, 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 IBM MQ global trace in conjunction with the z/OS generalized trace facility (GTF).

z/OS *Interpreting the trace information on z/OS*

The GTFTRACE produced by IBM MQ can be examined to determine possible errors with invalid addresses, invalid control blocks, and invalid data.

Start the GTFTRACE subcommand to format generalized trace facility (GTF) records contained in a dump or in a trace data set. For more information on GTF, see “Using the GTF with IBM MQ global trace” on page 448.

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 26 on page 451 illustrates which control blocks are traced for different MQI calls.

Table 26. 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)

Table 26. Control blocks traced for IBM MQ MQI calls (continued)		
MQI call	**Entry**	**Exit**
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
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

Identifier	Description
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

z/OS Using the channel initiator trace

The channel initiator trace can help to determine why a problem is happening in the CHIN address space, whether the cause of the problem is in IBM MQ, the operating system, or the network.

About this task

See Figure 46 on page 278 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.

Procedure

1. To run the trace, use 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.
2. To display this trace information, enter the IPCS command:

   ```
   LIST 1000. DSPNAME(CSQXTRDS)
   ```
3. To format the trace, use the command:

   ```
   CTRACE COMP(CSQX<i>ssnm</i>)
   ```

   where *ssnm* is the subsystem name.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

z/OS Using 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.

About this task

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:

Procedure

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. For more information, see "Snap dumps on z/OS" on page 294.

   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.

   Notes:

   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.

4. To obtain another dump of line trace data, stop and restart the current trace.

   a. 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
   ```

   b. To stop the trace, issue the following command:

   ```
   +ssid STOP TRACE(CHINIT) TNO(01)
   ```

   c. You can then enter another **START TRACE** command with a **DISPLAY CHSTATUS SAVED** command to gather more line trace data to CSQSNAP.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

z/OS Other types of trace on z/OS

There are other trace facilities available for problem determination. Use this topic to investigate CICS adapter trace, SSL trace, and z/OS trace.

It can be helpful to use the following trace facilities with IBM MQ.

- The CICS adapter trace

- System SSL trace
- z/OS z/OS traces

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

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 455.

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 *z/OS MVS Diagnosis: Tools and Service Aids* manual.

z/OS *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 27. 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.

Table 27. CICS adapter trace entries (continued)

Name	Description	Trace sequence	Trace data
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		
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.

Table 27. CICS adapter trace entries (continued)

Name	Description	Trace sequence	Trace data
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.
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.

Table 27. CICS adapter trace entries (continued)			
Name	**Description**	**Trace sequence**	**Trace data**
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.

z/OS Enabling internal trace for the AMSM address space

Trace for the AMSM address space can be enabled using the _AMS_MSG_LEVEL variable, which is passed into the AMSM address space through the ENVARS DD card.

A sample data set for the ENVARS DD card is in `thlqual.SCSQPROC(CSQ40ENV)`.

Trace is written to the SYSOUT of the AMSM address space.

The _AMS_MSG_LEVEL variable specifies the subcomponent and message level that is to be logged. An asterisk indicates all subcomponents to be logged; currently there is only one subcomponent.

The severity levels are:

- S - severe messages only
- E - error and severe messages only
- W - warning, error, and severe messages only
- I - informational, warning, error, and severe messages. This is the default value
- D - debug mode, all messages with additional debug diagnostics
- V - verbose mode, all of the preceding, plus buffer dumps

Attention: You should only enable debug or verbose mode on the advice of an IBM service representative.

For example, to enable the default for _AMS_MSG_LEVEL, issue the following:

```
_AMS_MSG_LEVEL=*.I
```

To enable verbose mode, issue the following:

```
_AMS_MSG_LEVEL=*.V
```

You can also start the internal trace for the AMS address space using the z/OS MODIFY (F) command.

For example, to start debug level trace, issue the following command:

```
/F xxxxAMSM,LOG *.D
```

To stop debug level trace, and revert to the default information level trace, issue the following command:

```
/F xxxxAMSM,LOG *.I
```

Related tasks

“Troubleshooting AMS problems” on page 69
Troubleshooting information to help you identify and resolve problems relating to Advanced Message Security (AMS).

z/OS Using GSKit trace for problems related to certificates and keys when using AMS on z/OS

Use this topic to understand how to turn on and turn off IBM Global Security Kit (GSKit) tracing when using AMS on z/OS.

Introduction

In the JCL for the AMS (ssidAMSM) address space, and in sample JCL hlq.SCSQPROC(CSQ40CFG) that runs program *CSQOUTIL*, there is an ENVARS DD card that can be used to set environment variables. A sample AMS environment variables file called hlq.SCSQPROC(CSQ40ENV) is provided which includes details of how to turn on and turn off GSKit trace.

If you set GSK trace environment variables in the ENVARS DD card in the JCL for the AMS address space, variables are set from the point that the AMS address space is started (that is, as part of queue manager start-up if AMS has been configured). Variables either turn on, or turn off, tracing of all gsk_* calls issued by the AMS address space.

If you set GSK trace environment variables in the ENVARS DD card in sample JCL hlq.SCSQPROC(CSQ40CFG), variables are set for the duration of the CSQ40CFG job. Variables either turn on, or turn off, tracing of all gsk_* calls issued during the processing of AMS commands that, for example, define and display AMS policies.

Turning on GSKit trace

Turn on GSKit trace for the AMS address space

To turn on GSKit trace for the AMS address space, carry out the following procedure:

1. Create JCL job hlq.SCSQPROC(CSQ40ENV) with:

```
GSK_TRACE_FILE=/u/<username>/AMStrace/gsktrace/gskssl.%.trc
GSK_TRACE=0xff
```

2. Specify member hlq.SCSQPROC(CSQ40ENV) on the ENVARS DD card in the ssidAMSM JCL for the AMS address space. For example:

```
//ENVARS   DD DSN=hlq.SCSQPROC(CSQ40ENV),DISP=SHR
```

3. Start the queue manager (ssidMSTR) and channel initiator (ssidCHIN) address spaces. Remember that the AMS (ssidAMSM) address space is started as part of starting the queue manager.

   You can see the environment variable settings in the job log for the AMS address space. For example:

```
-4.09.18 STC13921  CSQ06091 !MQ07 CSQ0DSRV IBM MQ AMS for z/OS starting V9.2.3, level GA
-4.09.18 STC13921  CSQ06191 !MQ07 CSQ0DSRV AMSPROD=ADVANCEDVUE, recording product usage for MQ z/OS Adv
VUE product id 5555AV9
-4.09.18 STC13921  CSQ06331 !MQ07 CSQ0DSRV AMS environment variables values:
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV _CEE_ENVFILE_S=DD:ENVVARS
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV _AMS_MSG_LEVEL=*.V
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV _AMS_MSG_FOLDING=NO
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV _AMS_INIT_THREADS=20
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV _AMS_MAX_THREADS=100
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV TZ=ESTESDT
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV GSK_TRACE_FILE=/u/<username>/AMStrace/gsktrace/gskssl.%.trc
-4.09.18 STC13921  CSQ06341 !MQ07 CSQ0DSRV GSK_TRACE=0xff
```

```
-4.09.21 STC13921  CSQ06531 !MQ07 CSQ0DLCL CRL checking disabled
-4.09.21 STC13921  CSQ06021 !MQ07 CSQ0DCNS AMS initialization complete
```

The gsk_* calls issued by the AMS address space to protect or unprotect IBM MQ messages at put and get time respectively, are traced. A trace file is created when the AMS address space is started, to trace all gsk_* calls subsequently performed by the address space. The use of the % character in the name of the trace file ensures that trace files are named by Unix Systems Services (USS) process identifiers.

4. Issue the following command to list the trace files produced:

```
/u/<username>/AMStrace/gsktrace:>ls
```

For example, you see files like:

```
gskssl.84017302.trc
```

5. To format and view the trace file, issue the following command in USS:

```
/u/<username>/AMStrace/gsktrace:>gsktrace gskssl.84017302.trc
```

which produces output similar to the following:

```
07/01/2022-10:36:41 Thd-0 INFO gsk_svc_init(): System SSL Version 4, Release 4, Service level OA60573
07/01/2022-10:36:41 Thd-0 INFO gsk_svc_init(): LE runtime level 0x42040000, 31-bit addressing mode
07/01/2022-10:36:41 Thd-0 INFO gsk_svc_init(): STDOUT handle=-1, STDERR handle=-1, TRACE handle=0
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): Using variant character table for code set IBM-1047
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): Using local code page IBM-1047
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): Using ISO8859-1 for TELETEX string
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): 64-bit encryption enabled
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): 128-bit encryption enabled
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): 168-bit encryption enabled
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): 256-bit encryption enabled
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Crypto assist supports strong encryption
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): FIPS mode level 1101
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): SHA-1 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): SHA-224 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): SHA-256 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): SHA-384 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): SHA-512 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): DES crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): DES3 crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): AES 128-bit crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): AES 256-bit crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): AES-GCM crypto assist is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Cryptographic accelerator is not available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Cryptographic coprocessor is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Public key hardware support is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Max RSA key sizes in hardware - signature 4096, encryption
4096, verification 4096
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): Maximum RSA token size 3500
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): ECC clear key support is available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): ECC secure key support is available. Maximum key size 521
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): TKDS is available for the storage of persistent PKCS #11
objects
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): ICSF Secure key PKCS #11 support is not available
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): ICSF FIPS compatibility mode
07/01/2022-10:36:41 Thd-0 INFO crypto_init(): ICSF FMID is HCR77D1
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): Job name CSQ40CFG, Process 05020096
07/01/2022-10:36:41 Thd-0 INFO gsk_dll_init_once(): GSKSRVR communication area at 00000000
07/01/2022-10:36:41 Thd-0 ENTRY gsk_dn_to_name(): ---> DN: CN=USER,O=IBM,C=UK
07/01/2022-10:36:41 Thd-0 EXIT gsk_dn_to_name(): <--- Exit status 0x00000000 (0)
07/01/2022-10:36:46 Thd-0 ENTRY gsk_dn_to_name(): ---> DN: CN=USER1,O=IBM,C=UK
07/01/2022-10:36:46 Thd-0 EXIT gsk_dn_to_name(): <--- Exit status 0x00000000 (0)
07/01/2022-10:36:46 Thd-0 ENTRY gsk_dn_to_name(): ---> DN: CN=USER,O=IBM,C=UK
07/01/2022-10:36:46 Thd-0 EXIT gsk_dn_to_name(): <--- Exit status 0x00000000 (0)
```

Turn on GSKit trace for JCL `hlq.SCSQPROC(CSQ40CFG)`

To turn on GSKit trace for JCL CSQ40CFG, carry out the following procedure:

1. Create JCL job `hlq.SCSQPROC(CSQ40CFG)` with:

```
GSK_TRACE_FILE=/u/<username>/AMStrace/gsktrace/gskssl.%.trc
GSK_TRACE=0xff
```

2. Set the ENVARS DD card as in the following example. On the **setmqspl** and the **dspmqspl** commands, specify parameters specific to your environment:

```
//CSQ40CFG JOB (ACCOUNT),'DEFAULT JOBCARD',CLASS=C,
//         MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
//* Job to define and display an AMS policy on a queue. The policy
//* name is the same as the queue name.
//* Make sure column numbers are not included as otherwise they can
//* interfere with the data in SYSIN.
/*JOBPARM SYSAFF=MVnn
//CSQ40CFG EXEC PGM=CSQ0UTIL,
//         PARM='ENVAR("_CEE_ENVFILE_S=DD:ENVARS") /'
//STEPLIB  DD DSN=hlq.SCSQANLE,DISP=SHR
//         DD DSN=hlq.SCSQAUTH,DISP=SHR
//ENVARS   DD DSN=hlq.SCSQPROC(CSQ40ENV),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
setmqspl -m MQ01 -p BANK.RQ
         -r CN=USERID,O=IBM,C=UK -e AES256
dspmqspl -m MQ01 -p BANK.RQ
/*
```

3. Submit the CSQ40CFG job.

   The CSQ40CFG job does not give any indication of whether GSKit trace has been enabled or not. However, you can check if trace is enabled or not by looking at the settings in the environment variables file specified for the job, or by checking if a trace file was created for the process under which the CSQ40CFG job ran.

Turning off GSKit trace

Turn off GSKit trace for the AMS address space

To turn off GSKit trace for the AMS address space, carry out the following procedure:

1. Stop the queue manager (`ssid`MSTR) address space. This stops both the channel initiator and the AMS address spaces.
2. Modify the `hlq.SCSQPROC(CSQ40ENV)` as follows:

   ```
   GSK_TRACE_FILE=/u/<username>/AMStrace/gsktrace/gskssl.%.trc
   GSK_TRACE=0x00
   ```

3. Restart the queue manager (`ssid`MSTR) and channel initiator (`ssid`CHIN) address spaces. Remember that the AMS (`ssid`AMSM) address space is started as part of starting the queue manager.
4. Check the environment variable settings in the job log for the AMS address space to ensure that GSKit trace has been turned off.

Turn off GSKit for JCL `hlq.SCSQPROC(CSQ40CFG)`

To turn off GSKit trace for JCL `hlq.SCSQPROC(CSQ40CFG)`, carry out the following procedure:

1. Modify the `hlq.SCSQPROC(CSQ40CFG)` file as follows:

   ```
   GSK_TRACE_FILE=/u/<username>/AMStrace/gsktrace/gskssl.%.trc
   GSK_TRACE=0x00
   ```

2. Submit the `hlq.SCSQPROC(CSQ40CFG)` job and check that no trace file is produced.

Notes:

- In the environment files, coding `GSK_TRACE=0xff` turns trace on, and coding `GSK_TRACE=0x00` turns trace off.
- Include the `%` character in the trace file name to ensure that trace file names produced for different USS processes, that issue `gsk_*` calls, include the process identifier, and hence are kept separate.

Related information

Generating IBM MQ GTF trace on IBM z/OS

z/OS Examples of trace output for z/OS

Use these examples to help interpret trace output on z/OS.

Figure 62 on page 463 shows an example of a trace taken on entry to an MQPUT1 call. The following items have been produced:

- Queue request parameter list
- Object descriptor (MQOD)
- Message descriptor (MQMD)
- Put message options (MQPMO)
- The first 256 bytes of message data

Compare this to Figure 63 on page 464, which illustrates the same control blocks on exit from IBM MQ.

```
USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQ user parameter trace
PUTONE
    Thread... 004C2B10  Userid... CICSUSER  pObjDesc. 106B2010
    pMsgDesc. 106B20B8  pPMO..... 106B2200
    BufferL.. 00000064  pBuffer.. 106A0578  RSV1..... 00000000
    RSV2..... 00000000  RSV3..... 116BC830
    C9E8C1E8  C5C3C9C3  AA8E8583  76270484  | IYAYECIC..ec...d |
    D4D8E3E3  0000048C  00000000  00000000  | MQTT............ |
    00000000  1910C7C2  C9C2D4C9  E8C14BC9  | ......GBIBMIYA.I |
    C7C3E2F2  F0F48E85  83762979  00010000  | GCS204.ec..`.... |

               GMT-01/30/05 14:42:08.412320    LOC-01/30/05 14:42:08.412320

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQ user parameter trace
 +0000  D6C44040  00000001  00000000  C2404040  | OD  ........B    |
 +0010  40404040  40404040  40404040  40404040  |                  |
 ...
 +00A0  00000000  00000000                      | ........         |

               GMT-01/30/05 14:42:08.412345    LOC-01/30/05 14:42:08.412345

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQ user parameter trace
 +0000  D4C44040  00000001  00000000  00000008  | MD  ............ |
 ...
 +0130  40404040  40404040  40404040  40404040  |                  |
 +0140  40404040                                |                  |

               GMT-01/30/05 14:42:08.412370    LOC-01/30/05 14:42:08.412370

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQ user parameter trace
 +0000  D7D4D640  00000001  00000000  FFFFFFFF  | PMO ............ |
 ...
 +0070  40404040  40404040  40404040  40404040  |                  |

               GMT-01/30/05 14:42:08.412393    LOC-01/30/05 14:42:08.412393

USRD9 5E9 ASCB 00F87E80           JOBN ECIC330
CSQW072I ENTRY: MQ user parameter trace
 +0000  C1C1C1C1  C1C1C1C1  C1404040  40404040  | AAAAAAAAA        |
 ...
 +0060  40404040                                |                  |

               GMT-01/30/05 14:42:08.412625    LOC-01/30/05 14:42:08.412625
```

Figure 62. Example trace data from an entry trace of an MQPUT1 request

```
USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
PUTONE
    Thread... 004C2B10  Userid... CICSUSER  pObjDesc. 106B2010
    pMsgDesc. 106B20B8  pPMO..... 106B2200
    BufferL.. 00000064  pBuffer.. 106A0578  RSV1..... 00000000
    RSV2..... 00000000  RSV3..... 116BC830
    CompCode. 00000002  Reason... 000007FB
    C9E8C1E8  C5C3C9C3  AA8E8583  76270484  | IYAYECIC..ec...d |
    D4D8E3E3  0000048C  00000000  00000000  | MQTT............ |
    00000000  1910C7C2  C9C2D4C9  E8C14BC9  | ......GBIBMIYA.I |
    C7C3E2F2  F0F48E85  83762979  00010000  | GCS204.ec..`.... |
MQRC_OBJECT_TYPE_ERROR

               GMT-01/30/05 14:42:08.412678    LOC-01/30/05 14:42:08.412678

USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
 +0000  D6C44040  00000001  00000000  C2404040  | OD  ........B    |
 ...
 +00A0  00000000  00000000                      | ........         |

               GMT-01/30/05 14:42:08.412789    LOC-01/30/05 14:42:08.412789

USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
 +0000  D4C44040  00000001  00000000  00000008  | MD  ............ |
 ...
 +0140  40404040                                | |

               GMT-01/30/05 14:42:08.412814    LOC-01/30/05 14:42:08.412814

USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
 +0000  D7D4D640  00000001  00000000  FFFFFFFF  | PMO ............ |
 ...
 +0070  40404040  40404040  40404040  40404040  |                  |

               GMT-01/30/05 14:42:08.412836    LOC-01/30/05 14:42:08.412836

USRD9 5EA ASCB 00F87E80            JOBN ECIC330
CSQW073I EXIT:  MQ user parameter trace
 +0000  C1C1C1C1  C1C1C1C1  C1404040  40404040  | AAAAAAAAA        |
 ...
 +0060  40404040                                | |

               GMT-01/30/05 14:42:08.412858    LOC-01/30/05 14:42:08.412858
```

Figure 63. Example trace data from an exit trace of an MQPUT1 request

Tracing the Advanced Message Queuing Protocol (AMQP) Service

The trace facility provided by the Advanced Message Queuing Protocol (AMQP) Service is provided to help IBM Support to diagnose customer issues that are related to the service.

About this task

There are two ways to control trace for the IBM MQ AMQP service:

- By using the **strmqtrc** and **endmqtrc** commands to start and stop trace.

  You can restrict trace output to the AMQP service only by specifying the **-t** parameter with the amqp option.

  Enabling trace using the **strmqtrc** command without the **-t** parameter with the amqp option generates trace information for the entire queue manager where the IBM MQ AMQP service is running. This information includes the IBM MQ AMQP service itself, and the underlying Java Message Queuing Interface (JMQI) that the service uses to communicate with other queue manager components.
- By running the **controlAMQPChannel** command. Turning on trace by using the **controlAMQPChannel** command traces only the IBM MQ AMQP service.

If you are unsure which option to use, contact IBM Support for advice on the best way to collect trace for the issue that you are seeing.

Procedure

1. Method one
   a) Open a command prompt and navigate to the directory:
   MQ_INSTALLATION_PATH\bin
   b) Run the **strmqtrc** command to enable trace:
   To restrict trace output to the AMQP service only, run the following command:

   ```
   strmqtrc -m qmgr_name -t amqp
   ```

   where *qmgr_name* is the name of the queue manager where the IBM MQ AMQP service is running, and **-t** *amqp* restricts trace output to the AMQP service only.
   Running the **strmqtrc** command without the **-t** parameter and the amqp option generates trace information for the entire queue manager where the IBM MQ AMQP service is running.
   c) Reproduce the issue.
   d) Stop trace, by running the **endmqtrc** command:

   ```
   endmqtrc -m qmgr_name
   ```

2. Method two.
   a) Bring up a command prompt and navigate to the directory:
   MQ_INSTALLATION_PATH\bin
   b) Run the following command to enable trace:
   - Windows

     ```
     controlAMQPChannel -qmgr=qmgr_name -mode=starttrace
     ```

   - Linux AIX

     ```
     ./controlAMQPChannel.sh -qmgr=qmgr_name -mode=starttrace
     ```

   where *qmgr_name* is the name of the queue manager where the AMQP Service is running.
   c) Reproduce the issue.
   d) When the issue occurs, stop trace by running the following command:
   - Windows

     ```
     controlAMQPChannel -qmgr=qmgr_name -mode=stoptrace
     ```

   - Linux AIX

     ```
     ./controlAMQPChannel.sh -qmgr=qmgr_name -mode=stoptrace [clientid=ClientIdentifier]
     ```

   where *qmgr_name* is the name of the queue manager where the AMQP Service is running.

Results

To view the trace output, go to the following directory:

- Windows *MQ_DATA_PATH*\trace.
- Linux AIX /var/mqm/trace.

The trace files are named as follows:

- The trace files that contain the information from the AMQP service are called amqpRunMQXRService_*PPPPP*.*N*.trc, where *PPPPP* is the process identifier for the AMQP service and *N* is a number.
- The trace files that contain the information from the **controlAMQPChannel** command are called amqpControlMQXRChannel_*PPPPP*.*N*.trc, where PPPPP is the process identifier for the AMQP service and N is a number.

Trace information that is generated by the JMQI is written to a trace file called amqp_*PPPPP*.trc, where *PPPPP* is the process identifier for the AMQP Service.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Troubleshooting AMQP problems" on page 64
Troubleshooting information to help you solve a problem with running AMQP applications.

ALW Additional diagnostics using the controlAMQPChannel command

Using the **controlAMQPChannel** command to provide additional diagnostic information about the AMQP service.

Procedure

Run the following command to provide useful diagnostic information from the MQXR service:

```
<MQ_INSTALLATION_PATH>\amqp\bin\controlAMQPChannel -qmgr=<QMGR_NAME> -mode=diagnostics
-diagnosticstype=<number>
```

The diagnostic information generated depends on the value of the **-diagnosticstype**=*<number>* parameter:

-diagnosticstype= *0*
Thread dump written to the console

-diagnosticstype= *1*
FDC with some internal service statistics

-diagnosticstype= *2*
FDC with internal statistics, plus information about the clients that are currently connected

-diagnosticstype= *3*
Heap dump

-diagnosticstype= *4*
Javacore

-diagnosticstype= *5*
Full system dump

-diagnosticstype= *6*
Detailed information about a specific client. Note that you must also supply the **-clientid** parameter for that client as well.

Tracing the IBM MQ Console

The trace facilities in the IBM MQ Console are provided to help IBM staff to diagnose customer problems. Various properties control the behavior of these facilities.

Before you begin

Include the following files and directories when you gather diagnostic information for IBM Support:

- The mqweb.xml file.

- The contents of the directory that contains the mqweb server definition:
  - ALW *MQ_DATA_PATH*/web/installations/*installationName*
  - z/OS The directory that was specified when the **crtmqweb** script ran to create the mqweb server definition. By default, this directory is /var/mqm/web/installation1.

About this task

The IBM MQ Console consists of:

- JavaScript, which runs inside a browser.
- Back end code, written in Java, which runs inside the mqweb server.

You can trace the JavaScript running inside a browser, as well as the back end code and JavaScript simultaneously.

The browser trace is output only from the browser that it is enabled in. After you log out of the IBM MQ Console, trace is automatically disabled.

Procedure

- To trace the IBM MQ Console back end code that runs in the mqweb server:
  a) z/OS
  On z/OS, ensure that the WLP_USER_DIR environment variable is set. If necessary, you can set the variable by entering the following command:

  ```
  export WLP_USER_DIR=WLP_user_directory
  ```

  Where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example, export WLP_USER_DIR=/var/mqm/web/installation1
  b) Enter the following command on the command line:

  ```
  setmqweb properties -k traceSpec -v
  "*=info:com.ibm.mq*=all:com.ibm.mq.rest*=all:js.mq*=all"
  ```

  c) To start tracing in the IBM MQ web server:
    a. Start the mqweb server.
    b. Bring up a browser, and log into the IBM MQ Console.
    c. Modify the URI that is being used to access the IBM MQ Console, so that it includes the string

    ```
    ?trace=true
    ```

    as shown in the following example:

    Original URI: https://localhost:9443/ibmmq/console/#/
    New URI: https://localhost:9443/ibmmq/console/?trace=true#/

  d) To turn trace off, modify the URI and remove the string:

  ```
  ?trace=true
  ```

- To trace the IBM MQ Console JavaScript running inside a browser:
  a) z/OS
  On z/OS, ensure that the WLP_USER_DIR environment variable is set. If necessary, you can set the variable by entering the following command:

  ```
  export WLP_USER_DIR=WLP_user_directory
  ```

Where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example, `export WLP_USER_DIR=/var/mqm/web/installation1`

b) Enter the following command on the command line:

```
setmqweb properties -k traceSpec -v "*=info:js.mq*=all"
```

c) To start tracing in the IBM MQ web server:

a. Start the mqweb server.

b. Bring up a browser, and log into the IBM MQ Console.

c. Modify the URI that is being used to access the IBM MQ Console, so that it includes the string

```
?trace=true
```

as shown in the following example:

Original URI: `https://localhost:9443/ibmmq/console/#/`
New URI: `https://localhost:9443/ibmmq/console/?trace=true#/`

d) To turn trace off, modify the URI and remove the string:

```
?trace=true
```

- To trace the IBM MQ Console back end code and JavaScript running inside a browser:

a) z/OS

On z/OS, ensure that the WLP_USER_DIR environment variable is set. If necessary, you can set the variable by entering the following command:

```
export WLP_USER_DIR=WLP_user_directory
```

Where *WLP_user_directory* is the name of the directory that is passed to crtmqweb. For example, `export WLP_USER_DIR=/var/mqm/web/installation1`

b) Enter the following command on the command line:

```
setmqweb properties -k traceSpec -v
"*=info:com.ibm.mq*=all:com.ibm.mq.rest*=all:js.mq*=all"
```

c) To start tracing in the IBM MQ web server:

a. Start the mqweb server.

b. Bring up a browser, and log into the IBM MQ Console.

c. Modify the URI that is being used to access the IBM MQ Console, so that it includes the string

```
?trace=true
```

as shown in the following example:

Original URI: `https://localhost:9443/ibmmq/console/#/`
New URI: `https://localhost:9443/ibmmq/console/?trace=true#/`

d) To turn trace off, modify the URI and remove the string:

```
?trace=true
```

Results

Trace is output to a set of files. The directory where the trace files are created is:

- ALW *MQ_DATA_PATH*/web/installations/*installationName*/servers/mqweb/logs

- z/OS The subdirectory `/servers/mqweb/logs` beneath the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition. By default, this directory is `/var/mqm/web/installation1`.

The active file is called `trace.log`. Historical trace is kept in files that are called `trace_`*`timestamp`*`.log`. The size of these trace files, and the number of historical files that are kept can be configured by setting the `maxTraceFileSize` and `maxTraceFiles` variables. By default, the maximum trace file size is 20 MB, and the maximum number of trace files is 2. For more information, see Configuring logging.

Example

An example of the trace information generated by the IBM MQ Console JavaScript code running inside a browser is shown in the following text:

```
[18/08/20 14:34:13:880 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i.componentDidMount"
[18/08/20 14:34:13:880 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i.componentWillUnmount"
[18/08/20 14:34:13:880 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i._getTopicsData"
[18/08/20 14:34:13:880 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i._showCreateTopic"
[18/08/20 14:34:13:880 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i._showDeleteTopic"
[18/08/20 14:34:13:881 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i._showTopicDetails"
[18/08/20 14:34:13:881 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher
patchFn DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching ->
i._showCreateSubscription"
[18/08/20 14:34:13:881 BST] 00000050 trace-patcher 1 js.mq.mq.lib.service.trace-patcher patchFn
DATA [18/08/20 14:34:10:472 BST] User:[mqadmin] (@static) "Patching -> i.render"
```

Related concepts

Administration using the IBM MQ Console

Related tasks

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Tracing errors in IBM MQ Internet Pass-Thru

IBM MQ Internet Pass-Thru (MQIPT) provides a detailed execution trace facility, which is controlled by the **Trace** property.

About this task

Trace files are written to the *mqipt_home*`\errors` directory (where *mqipt_home* is the MQIPT home directory, which contains `mqipt.conf`). Each trace file produced has a name with the following format:

```
AMQyyyymmddnnnnnnnnn.n.TRC.v
```

where the suffix *v* is an integer that denotes the position of the trace file in the rotating set of files. When the size of the current trace file reaches the maximum trace file size, it is closed and renamed, and a new trace file is opened. The current trace file has the suffix `0`. Older trace files have successively larger integer suffixes.

Unexpected fatal errors are written as FFST records in an error log file located in the *mqipt_home*`\errors` directory. The FFST files have the following format:

```
AMQyyyymmddnnnnnnnnn.n.FDC
```

Procedure

To enable trace, add the **Trace** configuration property to the appropriate section in the `mqipt.conf` file.

The **Trace** property can be either be specified in the `[route]` section of each route that you want to trace, or specified in the `[global]` section. The value of the **Trace** property in the `[global]` section is inherited by all routes that do not specify a **Trace** property. For more information, see Trace.

The amount of data in network transmissions received and sent by a route that is traced can be configured using the **TraceUserData** property. For more information, see TraceUserData.

From IBM MQ 9.4.0, the maximum size of the trace files, and the maximum number of trace files that are kept, can be configured using the **TraceFileCount** and **TraceFileSize** properties in the `[global]` section of the `mqipt.conf` file. For more information, see TraceFileCount and TraceFileSize.

Related tasks

"Troubleshooting IBM MQ Internet Pass-Thru problems" on page 92
There are a number of steps you can follow to help determine the nature of any problems you might encounter when using IBM MQ Internet Pass-Thru (MQIPT).

"Collecting information for MQIPT problems on Multiplatforms" on page 354
If you need to report a problem with MQIPT to IBM, send relevant information that will help to resolve the problem more quickly.

Tracing errors in mqiptAdmin

The **mqiptAdmin** command has an execution trace facility that can diagnose errors that occur when an administrative command is issued to MQIPT from the command line.

Procedure

1. To enable trace for the **mqiptAdmin** command, set the following environment variable before you run the command:
   - Linux AIX On AIX and Linux systems, issue the following command:

     ```
     export MQIPT_JVM_OPTIONS="-Dcom.ibm.mq.ipt.trace.override.path=trace_directory
     -Dcom.ibm.mq.ipt.trace.override.level=5"
     ```

   - Windows On Windows systems, issue the following command:

     ```
     set MQIPT_JVM_OPTIONS=-Dcom.ibm.mq.ipt.trace.override.path=trace_directory
     -Dcom.ibm.mq.ipt.trace.override.level=5
     ```

   Replace *trace_directory* with the name of the directory that trace files are written to.

   When the **mqiptAdmin** command runs, a trace file is created in the specified directory. The trace file name has the following format:

   ```
   AMQyyyymmddnnnnnnnnn.0.TRC.0
   ```

2. To disable trace for **mqiptAdmin**, unset the environment variable.

Tracing user-defined security exits

To help diagnose problems in a user-defined security exit, you can enable a trace facility, similar to that used by MQIPT.

Enable tracing by setting the route **Trace** property to a value in the range 1 - 5. See the entry for **Trace** in MQIPT route properties.

There will probably be more than one instance of the security exit running at the same time so individual entries in the trace file can be identified by using the thread identifier.

The tracing functions are initialized by MQIPT when the security exit is started; all you have to do is choose what information you want to trace. There are many tracing examples in the sample user exits. See Security exits.

The minimum requirements for tracing are an `entry` call, an `exit` call, and the data that you want to trace. For example:

```
/**
* This method is called to initialize the exit (for example, for
* loading validation information) and place itself in a ready
* state to validate connection requests.
*/
public int init(IPTTrace t) {
final String strMethod = "CustomExit.init";

// Trace entry into this method
t.entry(strMethod);

// Trace useful information
t.data(strMethod, "Starting exit - MQIPT version " + getVersion());

// Perform initialization and load any data
t.data(strMethod, "Ready for work");

// Trace exit from this method
t.exit(strMethod);

return 0;
}
```

Tracing IBM MQ .NET applications

There are several different ways to enable trace for IBM MQ .NET applications. You normally need to use the trace facility only at the request of IBM Support.

About this task

If you are using IBM MQ classes for .NET Framework, you can turn on tracing and configure the trace settings in the following ways:

- For a full installation of IBM MQ, by using the **strmqtrc** command to enable trace for IBM MQ classes for .NET Framework. For more information, see “Tracing IBM MQ classes for .NET Framework applications using commands” on page 475.
- By using an application configuration file with a name that consists of the name of the executable program to which the file relates, with the suffix `.config`. For example, the application configuration file for text.exe would have the name `text.exe.config`. For more information, see “Tracing IBM MQ classes for .NET Framework clients using an application configuration file” on page 474.
- By using the **MQDOTNET_TRACE_ON** environment variable. For more information, see “Tracing IBM MQ .NET applications using environment variables” on page 472.
- By using the `mqclient.ini` file and setting the appropriate properties of the Trace stanza. You can also enable and disable tracing dynamically with the `mqclient.ini` file. For more information, see “Tracing IBM MQ .NET applications with mqclient.ini” on page 473.

Using an application configuration file is not supported for .NET (.NET Standard and .NET 6 libraries). If you are using IBM MQ classes for .NET Standard (.NET Standard and .NET 6 libraries), you can configure trace in the following ways:

- By using the **MQDOTNET_TRACE_ON** environment variable. For more information, see “Tracing IBM MQ .NET applications using environment variables” on page 472.
- From IBM MQ 9.3.3, by using the `mqclient.ini` file and setting the appropriate properties of the Trace stanza. You can also enable and disable tracing dynamically with the `mqclient.ini` file. For more information, see “Tracing IBM MQ .NET applications with mqclient.ini” on page 473.

Note:

For the IBM MQ .NETclient libraries built against .NET Framework, the order of precedence is as follows:

1. App.Config
2. Environment variable
3. mqclient.ini

For the IBM MQ .NET client libraries built against .NET Standard and .NET 6, the order of precedence is as follows:

1. Environment variable
2. mqclient.ini

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

"Tracing on Windows" on page 443
On Windows, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing. You can also use IBM MQ Explorer to start and end tracing.

"Tracing on AIX and Linux" on page 427
On AIX and Linux, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file. On AIX, you can use AIX system trace in addition to using the **strmqtrc** and **endmqtrc** commands.

"Tracing XMS .NET applications" on page 515
If you are using IBM MQ classes for XMS .NET Framework, you can configure trace from an application configuration file as well as from the XMS environment variables. If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. You can select the components that you want to trace. Trace is normally used under the guidance of IBM Support.

Tracing IBM MQ .NET applications using environment variables

The environment variable **MQDOTNET_TRACE_ON** is used to enable trace for redistributable clients.

About this task

The **MQDOTNET_TRACE_ON** environment variable checks whether the IBM MQ trace directory is available or not. If the trace directory is available, the trace file is generated in the trace directory. However, if IBM MQ is not installed, the trace file is copied to the current working directory.

Other environment variables including **MQERRORPATH**, **MQLOGLEVEL**, **MQSERVER**, and so on, that are used for IBM MQ classes for .NET Framework, can be used for IBM MQ classes for .NET Standard and .NET 6 and work in the same way.

Procedure

- To enable tracing, set the environment variable to specify the level of tracing that you want.

  Values equal to and less than 0 do not enable trace, 1 enables default trace and values greater than 1 enable detail tracing.

Related concepts

Installing IBM MQ classes for .NET

Related reference

Environment variables descriptions

Tracing IBM MQ .NET applications with mqclient.ini

From IBM MQ 9.4.0, you can use the `mqclient.ini` file to enable traces for the IBM MQ .NET client libraries.

About this task

From IBM MQ 9.4.0, the `mqclient.ini` file includes a Trace stanza. To enable trace, you must set the appropriate properties of the Trace stanza. If a property is set to an invalid value, then the properties are ignored.

You can also enable and disable tracing dynamically. When the application is running, if the `mqclient.ini` file is modified, created, or deleted, the IBM MQ .NET client reads the properties of the trace section again and then enables or disables the trace, so that restart of the application is not needed.

Note: There are several different ways to enable trace.

For the IBM MQ .NETclient libraries built against .NET Framework, the order of precedence is as follows:

1. App.Config
2. Environment variable
3. mqclient.ini

For the IBM MQ .NET client libraries built against .NET Standard and .NET 6, the order of precedence is as follows:

1. Environment variable
2. mqclient.ini

Procedure

- To start trace, specify the following attributes of the Trace stanza:
  a) Set **`MQDotnetTraceLevel`** to the value corresponding to the level of tracing you want.
     If you want to start tracing with full details (recommended), set this attribute to 2. If you want a less detailed trace, set the attribute to 1.
  b) Set **`MQDotnetTracePath`** to the file path of the folder where you want the trace files to be created.
     The current directory of the application is used if the path is left blank or if the **`MQDotnetTracePath`** attribute is not defined.
  c) Set **`MQDotnetErrorPath`** to the file path of the folder where you want the error log files to be created.
     The current directory of the application is used if the path is left blank or if the **`MQDonetErrorPath`** attribute is not defined.
- To stop trace, set **`MQDotnetTraceLevel`** to `0`.
  This is the default value for this attribute.

Related concepts

Trace stanza of the client configuration file

Related tasks

“Tracing IBM MQ classes for .NET Framework applications using commands” on page 475

For a full client installation of IBM MQ you can use the **strmqtrc** command to enable trace for IBM MQ classes for .NET Framework.

Tracing IBM MQ classes for .NET Framework clients using an application configuration file

If you are using IBM MQ classes for .NET Framework, you can generate trace for the IBM MQ .NET redistributable managed client and stand-alone IBM MQ .NET client by using an application configuration file.

About this task

These steps for generating trace apply to the IBM MQ .NET redistributable managed client as well as the stand-alone IBM MQ .NET client and are for IBM MQ classes for .NET Framework only.

You must use the application configuration file and an IBM MQ specific trace configuration file because, since there is no full IBM MQ client installation, the standard tools that are used for starting and stopping trace, **strmqtrc** and **endmqtrc**, are not available.

The application configuration file is not supported in IBM MQ classes for .NET (.NET Standard and .NET 6 libraries). To enable trace for IBM MQ classes for .NET (.NET Standard and .NET 6 libraries), you use the **MQDOTNET_TRACE_ON** environment variable. A value of -1 enables default level tracing. Any other integer value, greater than 0, enables detailed tracing. Setting this environment variable to any other value does not enable trace.

Procedure

- To start and stop trace, use the application configuration file and an IBM MQ specific trace configuration file.

  Application configuration file (`app.config` or `web.config`)

  Applications need to define the **MQTRACECONFIGFILEPATH** property under the `<appSettings>` section of the application configuration file, that is, the `app.config` or `web.config` file. (The actual name of the application configuration file depends on the name of your application.) The value of the **MQTRACECONFIGFILEPATH** property specifies the path for the location of the IBM MQ specific trace configuration file, `mqtrace.config`, as shown in the following example:

  ```
  <appSettings>
  <add key="MQTRACECONFIGFILEPATH" value="C:\MQTRACECONFIG" />
  </appSettings>
  ```

  Tracing is disabled if the `mqtrace.config` file is not found in the path that is specified application configuration file. However, First Failure Support Technology (FFST) and error logs are created in the application's directory, if the application has authority to write to the current directory.

  IBM MQ specific trace configuration file (`mqtrace.config`)

  The `mqtrace.config` file is an XML file that defines properties for starting and stopping trace, the path to the trace files, and the path to the error logs. The following table describes these properties.

  Table 28. Properties defined in the `mqtrace.config` file

Attribute	Description
MQTRACELEVEL	0: Stops tracing - this is the default value. 1: Starts tracing with lesser details. 2: Starts tracing with full details - recommended.

Table 28. Properties defined in the `mqtrace.config` file (continued)	
Attribute	**Description**
MQTRACEPATH	Points to a folder where the trace files will be created. The current directory of the application is used if the path is blank or the **MQTRACEPATH** attribute is not defined.
MQERRORPATH	Points to a folder where error log files will be created. The current directory of the application is used if the path is blank or the **MQERRORPATH** attribute is not defined.

The following example shows a sample `mqtrace.config` file:

```
<?xml version="1.0" encoding="utf-8"?>
<traceSettings>
  <MQTRACELEVEL>2</MQTRACELEVEL>
  <MQTRACEPATH>C:\MQTRACEPATH</MQTRACEPATH>
  <MQERRORPATH>C:\MQERRORLOGPATH</MQERRORPATH>
</traceSettings>
```

Tracing can be started and stopped dynamically when an application is running by altering the value of the **MQTRACELEVEL** attribute in the `mqtrace.config` file.

The running application must have create and write permissions for the folder specified by the **MQTRACELEVEL** attribute for generating trace files. Applications that are running in an Microsoft Azure PaaS environment must also ensure similar access permissions since web applications that use an IBM MQ .NET assembly running in Microsoft Azure PaaS might not have create and write permissions. The generation of the trace, first failure data capture (FDC), and error logs fails if the application does not have the required create and write permissions for the specified folder.

- To capture tracing events and messages relating to the SSLStream class, add a configuration section for system diagnostics to the application configuration file for your application.

  For more information, see Configuring trace for SSLStream.

Related concepts

IBM MQ components and features

Redistributable clients

.NET application runtime - Windows only

Related tasks

Installing IBM MQ classes for .NET

Using the WMQDotnetXAMonitor application

Tracing IBM MQ classes for .NET Framework applications using commands

For a full client installation of IBM MQ you can use the **strmqtrc** command to enable trace for IBM MQ classes for .NET Framework.

About this task

You normally need to use the trace facility only at the request of IBM Support.

Procedure

1. To start tracing, use the **strmqtrc** command.

   For more information, see strmqtrc (start trace).

   Note: The **-i** and **-p** parameters of the **strmqtrc** command, which allow you to specify process and thread identifiers, and named processes, have no effect IBM MQ .NET.

2. To end tracing, use the **endmqtrc** command.

   For more information, see endmqtrc (end trace).

Related tasks

"Tracing IBM MQ .NET applications with mqclient.ini" on page 473
From IBM MQ 9.4.0, you can use the `mqclient.ini` file to enable traces for the IBM MQ .NET client libraries.

Tracing JMS/Jakarta Messaging and Java applications

The trace facilities for JMS/Jakarta Messaging and Java applications are provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

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.

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 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 477.

  From IBM MQ 10.0.0, you can direct trace data directly to stdout (standard output stream) or stderr (standard error stream) by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value. If you are running applications in short-lived containers, this allows you to capture trace information through container logging mechanisms, which persist beyond the container's lifetime. For more information, see Java Standard Environment Trace stanza for JMS trace.
- 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 478.
- The IBM MQ classes for JMS trace mechanism can be configured to turn trace on dynamically when it finds a specific file on the file system and stop tracing when that file no longer exists. 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. For more information, see "Collecting an IBM MQ classes for JMS trace dynamically" on page 480.

If you are unsure which option to use, contact IBM Support 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 420.

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 in shown 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 ...
```

Related concepts

“FFST: IBM MQ classes for JMS” on page 420

Describes the name, location, and contents of the First Failure Support Technology (FFST) files that are generated by the IBM MQ classes for JMS.

Related tasks

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Java Standard Environment Trace stanza for JMS trace

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

The Java system property **com.ibm.msg.client.commonservices.trace.status** controls whether tracing is enabled or disabled. By default, trace information is written to a trace file in the application's current working directory.

V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, you can direct trace data directly to stdout (standard output stream) or stderr (standard error stream) by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value. If you are running applications in short-lived containers, this allows you to capture trace information through container logging mechanisms, which persist beyond the container's lifetime. For more information, see Java Standard Environment Trace stanza for JMS trace.

To collect a trace by using a Java system property, complete the following steps.

Procedure

- V 10.0.0 V 10.0.0

  Run the application that is going to be traced by using one of the commands shown in the following examples, depending on where you want to direct the trace output:

  – To direct the trace output to stdout (standard output stream):

  ```
  java -Dcom.ibm.msg.client.commonservices.trace.outputName=stdout \
  -Dcom.ibm.msg.client.commonservices.trace.status=ON \
  -jar myapp.jar
  ```

  – To direct trace output to stderr (standard error stream):

  ```
  java -Dcom.ibm.msg.client.commonservices.trace.outputName=stderr \
  -Dcom.ibm.msg.client.commonservices.trace.status=ON \
  -jar myapp.jar
  ```

  – To direct trace output to a directory and file name:

  ```
  java -Dcom.ibm.msg.client.commonservices.trace.outputName=/tmp/mqtrace.trc \
  -Dcom.ibm.msg.client.commonservices.trace.status=ON \
  -jar myapp.jar
  ```

  If **com.ibm.msg.client.commonservices.trace.OutputName** is set as shown in these examples, when the application starts, the IBM MQ classes for JMS start writing trace information to the specified output location. If **com.ibm.msg.client.commonservices.trace.status** is not set, 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 on the environment that the application is running in:

  – If the application has loaded the IBM MQ classes for Jakarta Messaging from the relocatable JAR file `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) or the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar` (JMS 2.0), trace is written to a file called `mqjavaclient_`*`%PID%`*`.cl`*`%u`*`.trc`.

  – 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`.

  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 specified location when it is stopped.

  Note: 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 478). 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.

   V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, you can direct trace data to stdout (standard output stream), or stderr (standard error stream), or to a trace file by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value:

   stdout (or System.out)
   Directs trace output to standard output stream (case insensitive). 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=stdout
   ```

   stderr (or System.err)
   Directs trace output to standard error stream (case insensitive). 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=stderr
   ```

   <file>
   Any other value is treated as a file path and trace is directed to that file. 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
   ```

   For more information, see Java Standard Environment Trace stanza for JMS trace.
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.

   If you have set the **com.ibm.msg.client.commonservices.trace.outputName** property as described in Step “3” on page 479, trace information is written to the location specified by this property. If the property is not set, 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:

   - If the application has loaded the IBM MQ classes for Jakarta Messaging from the relocatable JAR file `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) or the IBM MQ classes for JMS from the relocatable JAR file `com.ibm.mq.allclient.jar` (JMS 2.0), trace is written to a file called `mqjavaclient_%PID%.cl%u.trc`.

- 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`.

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.

V 10.0.0 V 10.0.0 To change the location to where the trace 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** (see Step "3" on page 479).

Collecting an IBM MQ classes for JMS trace dynamically

The IBM MQ classes for JMS trace mechanism can be configured to turn trace on dynamically when it finds a specific file on the file system and stop tracing when that file no longer exists. 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

To collect a trace dynamically, complete the following steps.

Procedure

1. Start the application that needs to be traced using the following command:

   ```
   java -Dcom.ibm.msg.client.commonservices.trace.OnOffFile=filename application_name
   ```

   where *filename* is the fully qualified path to the file that the IBM MQ classes for JMS will look for to determine if trace needs to be enabled or disabled.

   For example, to configure the IBM MQ classes for JMS to start tracing an application called `JMSConsumer` when a file called `C:\Trace\enableTrace.txt` is detected, start the application with the following command:

   ```
   java -Dcom.ibm.msg.client.commonservices.trace.OnOffFile=C:\Trace\enableTrace.txt JMSConsumer
   ```

   Note: By default, if

   ```
   -Dcom.ibm.msg.client.commonservices.trace.OnOffFile=filename
   ```

   is not specified, the library looks for a file named `enableMQJavaTrace` under the application's root folder.
2. When trace needs to be enabled, do this in one of the following ways:
   - Create the file specified by *filename* in Step "1" on page 480.
   - If **-Dcom.ibm.msg.client.commonservices.trace.OnOffFile** is not set, create a file named `enableMQJavaTrace` in the application's root folder.

   The IBM MQ classes for JMS writes the following message to the standard output stream and the IBM MQ classes for JMS log file (`mqjms.log`) to show that trace has been turned on:

   ```
   The IBM MQ messaging client trace mechanism is tracing to 'trace-filename'
   ```

   where `trace-filename` is the fully qualified path for the trace file.
3. After you reproduce the issue, turn trace off by removing or renaming the control file in one of the following ways:
   - Delete or rename the file specified by `filename` in Step "1" on page 480.
   - If **-Dcom.ibm.msg.client.commonservices.trace.OnOffFile** is not set, delete or rename `enableMQJavaTrace` in the application's root folder.

   The IBM MQ classes for JMS detect that the file is no longer available, turn off trace and write the following message to the standard output stream and the IBM MQ classes for JMS log file (`mqjms.log`):

```
The IBM MQ messaging client trace mechanism has stopped tracing to 'trace-filename'.
```

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 482.
- 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 483.

  V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, you can direct trace data directly to stdout (standard output stream) or stderr (standard error stream) by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value. If you are running applications in short-lived containers, this allows you to capture trace information through container logging mechanisms, which persist beyond the container's lifetime. For more information, see Java Standard Environment Trace stanza for Java trace.
- The IBM MQ classes for Java trace mechanism can be configured to turn trace on dynamically when it finds a specific file on the file system and stop tracing when that file no longer exists. 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. For more information, see "Collecting an IBM MQ classes for Java trace dynamically" on page 484.

If you are unsure which option to use, contact IBM Support 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 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 ...
```

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

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

The Java system property **com.ibm.msg.client.commonservices.trace.status** controls whether tracing is enabled or disabled. By default, trace information is written to a trace file in the application's current working directory.

V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, you can direct trace data directly to stdout (standard output stream) or stderr (standard error stream) by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value. If you are running applications in short-lived containers, this allows you to capture trace information through container logging mechanisms, which persist beyond the container's lifetime. For more information, see Java Standard Environment Trace stanza for Java trace.

To collect a trace by using a Java system property, complete the following steps.

Procedure

- V 10.0.0 V 10.0.0
  Run the application that is going to be traced by using one of the commands shown in the following examples, depending on where you want to direct the trace output:
  - To direct the trace output to stdout (standard output stream):

    ```
    java -Dcom.ibm.msg.client.commonservices.trace.outputName=stdout \
    -Dcom.ibm.msg.client.commonservices.trace.status=ON \
    -jar myapp.jar
    ```

  - To direct trace output to stderr (standard error stream):

    ```
    java -Dcom.ibm.msg.client.commonservices.trace.outputName=stderr \
    -Dcom.ibm.msg.client.commonservices.trace.status=ON \
    -jar myapp.jar
    ```

  - To direct trace output to a directory and file name:

    ```
    java -Dcom.ibm.msg.client.commonservices.trace.outputName=/tmp/mqtrace.trc \
    -Dcom.ibm.msg.client.commonservices.trace.status=ON \
    -jar myapp.jar
    ```

  If **com.ibm.msg.client.commonservices.trace.OutputName** is set as shown in these examples, when the application starts, the IBM MQ classes for JMS start writing trace information

to the specified output location. If **com.ibm.msg.client.commonservices.trace.status** is not set, 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 on the environment that the application is running in:

- 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`.
- If the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) or `com.ibm.mq.allclient.jar` (JMS 2.0), 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 specified location when it is stopped.

Note: 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 Java trace by using the IBM MQ classes for Java configuration file" on page 483). 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 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.

   V 10.0.0 V 10.0.0 From IBM MQ 10.0.0, you can direct trace data to stdout (standard output stream), or stderr (standard error stream), or to a trace file by setting the **com.ibm.msg.client.commonservices.trace.outputName** property to the appropriate value:

   stdout (or System.out)
   Directs trace output to standard output stream (case insensitive). 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=stdout
   ```

   stderr (or System.err)
   Directs trace output to standard error stream (case insensitive). 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=stderr
   ```

   <file>
   Any other value is treated as a file path and trace is directed to that file. 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
```

For more information, see Java Standard Environment Trace stanza for Java trace.

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.

If you have set the **com.ibm.msg.client.commonservices.trace.outputName** property as described in Step "3" on page 483, trace information is written to the location specified by this property. If the property is not set, 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 on the environment that the application is running in:

- 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`.
- If the application has loaded the IBM MQ classes for Java from the relocatable JAR file `com.ibm.mq.jakarta.client.jar` (Jakarta Messaging 3.0) or `com.ibm.mq.allclient.jar` (JMS 2.0), 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.

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.

Collecting an IBM MQ classes for Java trace dynamically

The IBM MQ classes for Java trace mechanism can be configured to turn trace on dynamically when it finds a specific file on the file system and stop tracing when that file no longer exists. 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

To collect a trace dynamically, complete the following steps.

Procedure

1. Start the application that needs to be traced using the following command:

```
java -Dcom.ibm.msg.client.commonservices.trace.OnOffFile=filename application_name
```

where `filename` is the fully qualified path to the file that the IBM MQ classes for Java will look for to determine if trace needs to be enabled or disabled.

For example, to configure the IBM MQ classes for Java to start tracing an application called `JMSConsumer` when a file called `C:\Trace\enableTrace.txt` is detected, start the application with the following command:

```
java -Dcom.ibm.msg.client.commonservices.trace.OnOffFile=C:\Trace\enableTrace.txt
MQJavaPutter
```

Note: V 9.4.4 By default, if

```
-Dcom.ibm.msg.client.commonservices.trace.OnOffFile=filename
```

is not specified, the library looks for a file named `enableMQJavaTrace` under the application's root folder.

2. When trace needs to be enabled, do this in one of the following ways:
   - Create the file specified by *filename* in step 1.
   - V 9.4.4 If `-Dcom.ibm.msg.client.commonservices.trace.OnOffFile` is not set, create a file named `enableMQJavaTrace` in the application's root folder.

   The IBM MQ classes for Java writes the following message to the standard output stream to show that trace has been turned on:

   ```
   The IBM MQ messaging client trace mechanism is tracing to 'trace-filename'
   ```

   where `trace-filename` is the fully qualified path for the trace file.
3. After you reproduce the issue, turn trace off by removing or renaming the control file in one of the following ways:
   - Delete or rename the file specified by `filename` in step 1.
   - V 9.4.4 If `-Dcom.ibm.msg.client.commonservices.trace.OnOffFile` is not set, delete or rename `enableMQJavaTrace` in the application's root folder.

   The IBM MQ classes for Java detect that the file is no longer available, turn off trace and write the following message to the standard output stream:

   ```
   The IBM MQ messaging client trace mechanism has stopped tracing to 'trace-filename'.
   ```

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.

About this task

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 29 on page 485 lists the properties of the ResourceAdapter object that are associated with diagnostic tracing.

Table 29. 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.

Table 29. Properties of the ResourceAdapter object that are associated with diagnostic tracing (continued)			
Name of property	**Type**	**Default value**	**Description**
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 30 on page 486 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 30 on page 486 describes the levels of detail for diagnostic tracing.

Table 30. 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, you can enable diagnostic tracing on AIX and Linux systems for example, by setting properties.

Procedure

- To enable diagnostic tracing on AIX 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
```

Hints and tips:

- You do not need to define all the properties of the ResourceAdapter object. Any properties that remain 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.
- When using WebSphere Application Server 9.0 with the IBM MQ 9.0 resource adapter, as the Java EE Dependency Injection is a common Java EE paradigm, the standard trace string should be updated to include `com.ibm.ws.cdi.jms*=all`. This means that the full string is:

  ```
  *=info:jmsApi=all:Messaging=all:com.ibm.mq.*=all:JMSApi=all:com.ibm.ws.cdi.jms*=all
  ```

For more information about using trace with WebSphere Application Server, see the technote Enabling Java Message Service (JMS) trace for WebSphere Application Server.

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 488.

For instructions on locating trace information and FFDC files, see “Java trace and FFDC files” on page 489.

Related concepts

“Tracing runmqakm” on page 514
How to request **runmqakm** trace.

Related tasks

“Tracing on Windows” on page 443

On Windows, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing. You can also use IBM MQ Explorer to start and end tracing.

"Tracing on AIX and Linux" on page 427
On AIX and Linux, you can use the **strmqtrc** and **endmqtrc** commands to start and end tracing, and **dspmqtrc** to display a trace file. On AIX, you can use AIX system trace in addition to using the **strmqtrc** and **endmqtrc** commands.

"Using trace with IBM MQ server on IBM i" on page 434
If you have an IBM MQ server installation on IBM i, you can use the **TRCMQM** command to start and stop tracing and specify the type of trace that you require. Alternatively, you can control tracing from the IBM i command line by calling the **QMQM/STRMQTRC** and **QMQM/ENDMQTRC** programs, or in the IBM i Qshell using the **STRMQTRC**, **ENDMQTRC** and **DSPMQTRC** commands.

"Tracing on z/OS" on page 447
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.

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 AIX 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 64 on page 489. Lines beginning with the number sign (#) are treated as comments.

```
#
# Diagnostics for MQ Explorer are enabled
#
Diagnostics.wmqexplorer
#
# 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 64. Sample com.ibm.mq.commonservices properties file

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 IBM MQ Explorer, it is written to file with a name of the format AMQYYYYMMDDHHmmssmmm.TRC.n. Each time IBM 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 IBM 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 IBM 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.

Multi Tracing Managed File Transfer resources on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

Related tasks

"Collecting troubleshooting information" on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Multi Tracing Managed File Transfer agents on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with an agent, use one of the following options, depending on whether it is possible for you to stop the agent for a short period of time.

If you are unsure which option to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop an agent for a short period of time, collect a trace of the agent from startup.

  For more information, see "Collecting a Managed File Transfer agent trace from startup" on page 490.
- If it is not possible for you to stop an agent, then collect a trace dynamically using the **fteSetAgentTraceLevel** command.

  For more information, see "Collecting a Managed File Transfer agent trace dynamically" on page 491.

Multi *Collecting a Managed File Transfer agent trace from startup*

Where it is possible for you to stop an agent for a short period of time, you should collect Managed File Transfer agent trace from startup.

Before you begin

You need to set various properties in the `agent.properties` file for the agent that needs to be traced.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

   The `agent.properties` file can be found in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/agents/`*agent_name* directory.
2. Edit the files and add entries for the following properties:

   - **trace**=*trace specification*

     The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
   - **traceFiles**=*number of trace files to use*
   - **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see The MFT `agent.properties` file.

3. Stop the agent that needs to be traced, using the **fteStopAgent** command.
4. Start the agent, by running the **fteStartAgent** command.
5. Reproduce the issue.
6. Stop the agent.
7. Edit the `agent.properties` file for the agent, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step "2" on page 490.

   This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the agent.

Multi

Collecting a Managed File Transfer agent trace dynamically

The **fteSetAgentTraceLevel** command allows trace to be collected from a running agent. This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from an agent using the **fteSetAgentTraceLevel** command, complete the following steps.

Procedure

1. Turn on trace for the agent by running the following command:

   ```
   fteSetAgentTraceLevel -traceAgent classes=level agent_name
   ```

   Note: The `-traceAgent` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.

   An example command might look like this for an agent named AGENT1:

```
fteSetAgentTraceLevel -traceAgent com.ibm.wmqfte=all AGENT1
```

2. Reproduce the issue.
3. Turn off trace for the agent by running the following command:

   ```
   fteSetAgentTraceLevel -traceAgent =off agent_name
   ```

If an agent is busy, the trace files might wrap quickly and overwrite the information needed to investigate the issue. If this is so, schedule some time to stop the agent then proceed as detailed in the following steps. If you cannot stop the agent for a short period of time, contact IBM Support and discuss alternative trace specifications to use, to reduce the amount of trace data that is being generated.

4. Locate the `agent.properties` file for the agent that needs to be traced.

   The `agent.properties` file can be found in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/agents/`*agent_name* directory.
5. Edit the file and add entries for the following properties:

   ```
   traceFiles=number_of_trace_files_to_use
   traceSize=size_of_each_trace_file_in_MB
   ```

   The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

   The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

   You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

   ```
   traceFiles=5
   traceSize=200
   ```

   This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

   For more information on these agent properties, see The MFT `agent.properties` file.
6. Stop the agent, by running the **fteStopAgent** command.
7. Start the agent, by running the **fteStartAgent** command.
8. Enable trace for the agent, by running the following command:

   ```
   fteSetAgentTraceLevel -traceAgent classes=level agent_name
   ```

   Unless otherwise specified by your IBM Support representative, set the **-traceAgent** property to the value `com.ibm.wmqfte=all`.
9. Reproduce the issue.
10. Turn trace off on the agent by running the following command:

    ```
    fteSetAgentTraceLevel -traceAgent =off agent_name
    ```

Results

The resultant trace files are written to the *MQ_DATA_PATH*`/mqft/logs/`*coordination_qmgr_name*`/agents/`*agent_name*`/logs/trace`*%PID%* directory, where %PID% is the process identifier for the agent.

Multi Tracing Managed File Transfer commands on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. You can use this facility to trace commands.

About this task

Attention: Tracing a command only collects information about the processing done by the command. It does not trace any activity that an agent might perform while processing that command.

Procedure

1. Bring up a command prompt, and navigate to the `MQ_INSTALLATION_PATH\bin` directory.

2. Run the command:

Linux AIX

```
./command_name -trace classes=level -tracePath directory_path command_arguments
```

Windows

```
command_name -trace classes=level -tracePath directory_path command_arguments
```

where

- *command_name* is the name of the command to be traced.
- *classes=level* is the trace level to use, and which classes to enable trace for. Unless otherwise specified by IBM Support, set this to `com.ibm.wmqfte=all`.
- *directory_path* is the directory where the trace files will be written to.
- *command_arguments* are the arguments that need to be passed to the command, for example, the name of the agent for the **ftePingAgent** command.

Results

The resultant trace files are written to the directory specified by the **-tracePath** parameter.

The trace files are called `trace`*%PID%*`.txt.`*number*, where:

- *%PID%* is the process identifier for the command.
- *number* is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of `0`.

  However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of `0`, the next oldest trace file has a sequence number of `1`, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the **-tracePath** parameter, the trace output is written to standard error.

Example

In this example, the **fteListAgents** command is traced, and the trace is written to the `C:\trace` directory:

```
fteListAgents -trace com.ibm.wmqfte=all -tracePath C:\trace
```

In this example, the **fteCreateTransfer** command is traced, and the trace is written to the `/tmp` directory:

```
fteCreateTransfer -trace com.ibm.wmqfte=all -tracePath /tmp -t text -sa AGENT1
-da AGENT2 -df /import/transferredfile.txt /export/originalfile.txt
```

The trace file written to `/tmp` only contains information about the processing performed by the **fteCreateTransfer** command, such as, how the command builds the transfer request message that is sent to the agent, and how long it waits for the agent to send back an acknowledgment indicating that it has received the request. The trace file does not contain any information about the transfer itself.

Multi Tracing Managed File Transfer standalone loggers on Multiplatforms

The trace facility in Managed File Transfer is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with a logger, use one of the following options, depending on whether it is possible for you to stop the logger for a short period of time.

If you are unsure which option to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop a logger for a short period of time, collect a trace of the logger from startup.

  See “Collecting a Managed File Transfer standalone logger trace from startup” on page 494.
- If it is not possible for you to stop a logger, then collect a trace dynamically using the **fteSetLoggerTraceLevel** command.

  See “Collecting a Managed File Transfer standalone logger trace dynamically” on page 495.

Multi *Collecting a Managed File Transfer standalone logger trace from startup*

Where it is possible for you to stop a logger for a short period of time, you should collect Managed File Transfer logger trace from startup.

Before you begin

You need to set various properties in the `logger.properties` file for the logger that needs to be traced.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

   The `logger.properties` file can be found in the *MQ_DATA_PATH*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.
2. Edit the file and add entries for the following properties:

   - **trace**=*trace specification*

     The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
   - **traceFiles**=*number of trace files to use*
   - **traceSize**=*size of each trace file, in MB*

     The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see The MFT `logger.properties` file.

3. Stop the logger that needs to be traced, using the **fteStopLogger** command.
4. Start the logger, by running the **fteStartLogger** command.
5. Reproduce the issue.
6. Stop the logger.
7. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step "2" on page 494.

   This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the *MQ_DATA_PATH*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the logger.

Multi *Collecting a Managed File Transfer standalone logger trace dynamically*

The **fteSetLoggerTraceLevel** command allows trace to be collected from a running logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a Managed File Transfer logger using the **fteSetLoggerTraceLevel** command, complete the following steps.

Procedure

1. Turn trace on for the logger, by running the following command:

   ```
   fteSetLoggerTraceLevel -traceLogger classes=level logger_name
   ```

   The `-traceLogger` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
2. Reproduce the issue.
3. Turn trace off for the logger, by running the following command:

   ```
   fteSetLoggerTraceLevel -traceLogger =off logger_name
   ```

4. If a logger is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

   If you can stop the logger for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

   a) Schedule some time to stop the logger.

   b) Locate the `logger.properties` file for the logger that needs to be traced.

      The `logger.properties` file can be found in the *MQ_DATA_PATH*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory.

   c) Edit the file and add entries for the following properties:

      traceFiles=*number_of_trace_files_to_use*
      traceSize=*size_of_each_trace_file_in_MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

```
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see The MFT `logger.properties` file.

d) Stop the logger, by running the **fteStopLogger** command.

e) Start the logger, by running the **fteStartLogger** command.

f) Turn trace on for the logger, by running the following command. Unless otherwise specified by your IBM Support representative, set the **-traceLogger** property to the value `com.ibm.wmqfte=all`.

```
fteSetLoggerTraceLevel -traceLogger classes=level logger_name
```

g) Reproduce the issue.

h) Turn trace off for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

Results

The resultant trace files are written to the *MQ_DATA_PATH*`/mqft/logs/`*coordination_qmgr_name*`/loggers/`*logger_name*`/logs/trace`*%PID%* directory, where %PID% is the process identifier for the logger.

z/OS Tracing Managed File Transfer for z/OS resources

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. You can trace various different resources.

Procedure

- To trace a Managed File Transfer agent on z/OS, see “Tracing Managed File Transfer for z/OS agents” on page 497.
- To trace a command, see “Tracing Managed File Transfer for z/OS commands” on page 503.
- To trace a Managed File Transfer stand-alone database logger on z/OS see “Tracing Managed File Transfer for z/OS standalone database loggers” on page 505.

Related tasks

“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

z/OS Tracing Managed File Transfer for z/OS agents

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with an agent, use one of the following options.

If you are unsure which option to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop an agent for a short period of time, collect a trace of the agent from startup.

  For more information, see "Collecting a Managed File Transfer for z/OS agent trace from startup" on page 497.
- If it is not possible for you to stop an agent, then collect a trace dynamically using the **fteSetAgentTraceLevel** command.

  For more information, see "Collecting a Managed File Transfer for z/OS agent trace dynamically" on page 499.

z/OS *Collecting a Managed File Transfer for z/OS agent trace from startup*

Where it is possible for you to stop an agent for a short period of time, you should collect IBM MQ Managed File Transfer agent trace from startup.

About this task

The way to collect the trace depends on whether the agent is being administered using z/OS UNIX System Services (z/OS UNIX) or JCL.

If you are unsure which of the following options to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using z/OS UNIX, see "Collecting an agent trace from startup using z/OS UNIX" on page 497.
- If you are using JCL, see "Collecting an agent trace from startup using JCL" on page 498.

z/OS *Collecting an agent trace from startup using z/OS UNIX*

To collect a trace of a Managed File Transfer for z/OS agent that is being administered using z/OS UNIX System Services (z/OS UNIX) from startup, you need to set various properties need in the `agent.properties` file for that agent before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

The `agent.properties` file can be found in the *BFG_DATA*`/mqft/config/`*`coordination_qmgr_name`*`/agents/`*`agent_name`* directory.

2. Edit the files and add entries for the following properties:

   - **`trace`**=*trace specification*

     The **`trace`** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.

   - **`traceFiles`**=*number of trace files to use*
   - **`traceSize`**=*size of each trace file, in MB*

     The **`traceFiles`** and **`traceSize`** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

   ```
   trace=com.ibm.wmqfte=all
   traceFiles=5
   traceSize=200
   ```

   This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

   For more information on these agent properties, see The MFT `agent.properties` file.

3. Stop the agent that needs to be traced, using the **`fteStopAgent`** command.
4. Start the agent, by running the **`fteStartAgent`** command.
5. Reproduce the issue.
6. Stop the agent.
7. Edit the `agent.properties` file for the agent, and remove the entries for the **`trace`**, **`traceFiles`**, and **`traceSize`** properties that you added in step "2" on page 498.

   This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the *BFG_DATA*`/mqft/logs/`*`coordination_qmgr_name`*`/agents/`*`agent_name`*`/logs/trace`*`%PID%`* directory, where %PID% is the process identifier for the agent.

z/OS *Collecting an agent trace from startup using JCL*

To collect a trace of a Managed File Transfer for z/OS agent that is being administered using JCL from startup, you need to set various properties need in the `agent.properties` file for that agent before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `agent.properties` file for the agent that needs to be traced.

   The `agent.properties` file can be found in the *BFG_DATA*`/mqft/config/`*`coordination_qmgr_name`*`/agents/`*`agent_name`* directory.

2. Edit the files and add entries for the following properties:

   - **`trace`**=*trace specification*

     The **`trace`** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.

- **traceFiles**=*number of trace files to use*
- **traceSize**=*size of each trace file, in MB*

  The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `agent.properties` file:

```
trace=com.ibm.wmqfte=all
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see The MFT `agent.properties` file.

3. Locate the data set containing the JCL for the agent that needs to be traced.
4. Submit the BFGAGSP member within the data set to stop the agent.
5. Restart the agent, by submitting the BFGAGST member in the data set .
6. Reproduce the issue.
7. Submit the BFGAGSP member in the data set to stop the agent again.
8. Edit the `agent.properties` file for the agent, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step "2" on page 498.

   This ensures that trace is not enabled the next time you restart the agent.

Results

The resultant trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the agent.

z/OS *Collecting a Managed File Transfer for z/OS agent trace dynamically*

Where it is not possible for you to stop an agent for a short period of time, you should collect Managed File Transfer for z/OS agent trace dynamically.

About this task

The way to collect the trace depends on whether the agent is being administered using z/OS UNIX System Services (z/OS UNIX) or JCL.

If you are unsure which of the following options to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
  - z/OS UNIX, see "Collecting an agent trace dynamically using z/OS UNIX" on page 500.
  - JCL, see "Collecting an agent trace dynamically using JCL" on page 501.

z/OS *Collecting an agent trace dynamically using z/OS UNIX*

Under z/OS UNIX System Services (z/OS UNIX), you can use the **fteSetAgentTraceLevel** command to collect trace from a running agent. This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from a Managed File Transfer for z/OS agent using the **fteSetAgentTraceLevel** command, complete the following steps.

Procedure

1. Turn trace on for the agent, by running the following command:

   ```
   fteSetAgentTraceLevel -traceAgent classes=level agent_name
   ```

   The `-traceAgent` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
2. Reproduce the issue.
3. Turn trace off for the agent, by running the following command:

   ```
   fteSetAgentTraceLevel -traceAgent =off agent_name
   ```
4. If an agent is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

   If you can stop the agent for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

   a) Schedule some time to stop the agent.

   b) Locate the `agent.properties` file for the agent that needs to be traced.

   The `agent.properties` file can be found in the *BFG_DATA*`/mqft/config/`*coordination_qmgr_name*`/agents/`*agent_name* directory.

   c) Edit the file and add entries for the following properties:

   traceFiles=*number_of_trace_files_to_use*
   traceSize=*size_of_each_trace_file_in_MB*

   The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

   The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

   You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

   ```
   traceFiles=5
   traceSize=200
   ```

   This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

   For more information on these agent properties, see The MFT `agent.properties` file.

   d) Stop the agent, by running the **fteStopAgent** command.

e) Start the agent, by running the **fteStartAgent** command.

f) Turn trace on for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent trace_specification agent_name
```

g) Reproduce the issue.

h) Turn trace off for the agent, by running the following command:

```
fteSetAgentTraceLevel -traceAgent =off agent_name
```

Results

The trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the agent.

z/OS *Collecting an agent trace dynamically using JCL*

You can use the BFGAGTC member within the data set containing the JCL, for the agent that needs to be traced, to collect trace from a running Managed File Transfer for z/OS agent This can be very useful if IBM Support need to see a trace from an agent that cannot be stopped.

About this task

To collect a trace from an agent using the BFGAGTC member, complete the following steps.

Procedure

1. Locate the data set containing the JCL for the agent that needs to be traced.
2. Edit the BFGAGTC member within the data set, and locate the line that contains the text:

   ```
   -traceAgent
   ```

   The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

   ```
   com.ibm.wmqfte=all
   ```

   Unless otherwise specified by IBM Support, leave this value as is.
3. Submit the BFGAGTC member.
4. Reproduce the issue.
5. Edit the BFGAGTC member again, and set the **-traceAgent** parameter to *=off*, as shown:

   ```
   -traceAgent =off +
   ```
6. Submit the BFGAGTC member again, to turn trace off.
7. If an agent is busy, then it is possible that the trace files will wrap quickly and overwrite the information needed to investigate the issue.

   In this situation there are two options:

   a) The first option is to:

      i) Schedule some time to stop the agent.

      ii) Locate the `agent.properties` file for the agent that needs to be traced. The `agent.properties` file can be found in the *BFG_DATA*/mqft/config/*coordination_qmgr_name*/agents/*agent_name* directory.

      iii) Edit the files and add entries for the following properties:

      - **traceFiles**=*number of trace files to use*
      - **traceSize**=*size of each trace file, in MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `agent.properties` file:

```
traceFiles=5
traceSize=200
```

This results in the agent writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these agent properties, see The MFT `agent.properties` file.

iv) Locate the data set containing the JCL for the agent that needs to be traced.

v) Submit the BFGAGSP member within the data set to stop the agent.

vi) Restart the agent, by submitting the BFGAGST member in the data set.

vii) Edit the BFGAGTC member within the data set, and locate the line that contains the text:

```
-traceAgent
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by IBM Support, leave this value as is.

viii) When it is time to enable trace, submit the BFGAGTC member.

ix) Reproduce the issue.

x) Edit the BFGAGTC member again, and set the **-traceAgent** parameter to *=off*, as shown:

```
-traceAgent =off +
```

xi) Submit the BFGAGTC member again, to turn trace off.

b) The second option is to contact your IBM Support representative, if it is not possible to stop the agent for a short period of time.

You can then discuss alternative trace specifications to use, in order to reduce the amount of trace data that is being generated.

Results

The resultant trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/agents/*agent_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the agent.

z/OS Tracing Managed File Transfer for z/OS commands

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. You can use this facility to trace commands.

About this task

Attention: Tracing a command only collects information about the processing done by the command. It does not trace any activity that an agent might perform while processing that command.

The way to collect the trace depends on whether the command is being run using either z/OS UNIX System Services (z/OS UNIX) or JCL.

If you are unsure which option to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using z/OS UNIX, see "Collecting an agent trace from startup using z/OS UNIX" on page 497.
- If you are using JCL, see "Collecting an agent trace from startup using JCL" on page 498.

z/OS *Collecting a trace of a command using z/OS UNIX*

To collect a trace of a Managed File Transfer for z/OS command using z/OS UNIX System Services (z/OS UNIX) carry out the following procedure.

Procedure

1. Bring up a command prompt, and navigate to the *BFG_PROD*/bin directory.
2. Run the command:

   ```
   ./command_name -trace classes=level -tracePath directory_path command_arguments
   ```

   where

   - *command_name* is the name of the command to be traced.
   - *classes=level* is the trace level to use, and which classes to enable trace for. Unless otherwise specified by IBM Support, set this to `com.ibm.wmqfte=all`.
   - *directory_path* is the directory where the trace files will be written to.
   - *command_arguments* are the arguments that need to be passed to the command, for example, the name of the agent for the **ftePingAgent** command.

Results

The resultant trace files are written to the directory specified by the **-tracePath** parameter.

The trace files are called `trace`*%PID%*`.txt.`*number*, where:

- *%PID%* is the process identifier for the command.
- *number* is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of `0`.

  However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of `0`, the next oldest trace file has a sequence number of 1, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the **-tracePath** parameter, the trace output is written to standard error.

Example

In this example, the **fteListAgents** command is traced, and the trace is written to the /u/fteuser directory:

```
./fteListAgents -trace com.ibm.wmqfte=all -tracePath /u/fteuser
```

In this example, the **fteCreateTransfer** command is traced, and the trace is written to the /tmp directory:

```
./fteCreateTransfer -trace com.ibm.wmqfte=all -tracePath /tmp -t text -sa AGENT1
-da AGENT2 -df /tmp/IEEUJV.txt "//'SYS1.SAMPLIB(IEEUJV)'"
```

The trace file written to /tmp only contains information about the processing performed by the **fteCreateTransfer** command, such as, how the command builds the transfer request message that is sent to the agent, and how long it waits for the agent to send back an acknowledgment indicating that it has received the request. The trace file does not contain any information about the transfer itself.

z/OS *Collecting a trace of a command using JCL*

To collect a trace of a Managed File Transfer for z/OS command that is being submitted using JCL you need to complete the following steps.

Procedure

1. Locate the data set containing the JCL for the command that needs to be traced.
2. Within the data set, locate the member for that command.
3. Edit the member, and locate the line that contains the name of the command that needs to be traced. Modify this line so that it includes the text after the command name and before the + sign:

   ```
   -trace classes=level -tracePath directory_path
   ```

   where:

   - *classes=level* is the trace level to use, and which classes to enable trace for. Unless otherwise specified by IBM Support, set this to com.ibm.wmqfte=all.
   - *directory_path* is the z/OS UNIX System Services directory where the trace files will be written to.
4. Submit the member.
5. After the issue has been reproduced, edit the member again and remove the text:

   ```
   -trace classes=level -tracePath directory_path
   ```

   that you added in Step “3” on page 504.

Results

The resultant trace files are written to the directory specified by the **-tracePath** parameter.

The trace files are called trace*%PID%*.txt.*number*, where:

- *%PID%* is the process identifier for the command.
- *number* is a sequence number for the trace file. Typically, the trace information generated by a command is contained within a single trace file that has a sequence number of 0.

  However, it is possible that a command will generate a lot of trace information. In this situation, the trace will be written to multiple files. The current trace file has a sequence number of 0, the next oldest trace file has a sequence number of 1, and so on.

Trace output for commands are written to a maximum of five wrapping trace files. The maximum size of each trace file is 20MB.

Note: If the user running the command does not have permission to write to the directory specified by the **-tracePath** parameter, the trace output is written to standard error.

Example

In this example, the member BFGMNLI has been modified to trace the **fteListMonitors** command:

```
//*********************************************************************
//* <copyright
//* notice="lm-source"
//* pids="5655-MF9"
//* years="2013,2016"
//* crc="3927276320" >
//* Licensed Materials - Property of IBM
//*
//* 5655-MF9
//*
//* (C) Copyright IBM Corp. 2013, 2026. All Rights Reserved.
//* </copyright>
//*********************************************************************
//* fteListMonitors
//*********************************************************************
//BFGCMD   EXEC PGM=IKJEFT01,REGION=0M
//SYSEXEC  DD DSN=++LIBRARY++,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//STDOUT   DD SYSOUT=*
//STDERR   DD SYSOUT=*
//SYSTSIN  DD *
 %BFGCMD CMD=fteListMonitors -trace com.ibm.wmqfte=all -tracePath /u/trace +
    -v -p QM1
/*
//
```

When the member is submitted, the **fteListMonitors** command writes trace to the z/OS UNIX directory `/u/trace`.

z/OS Tracing Managed File Transfer for z/OS standalone database loggers

The trace facility in Managed File Transfer for z/OS is provided to help IBM Support diagnose your problems and issues. Various commands and properties control the behavior of this facility.

About this task

If you are asked to provide trace output to investigate an issue with a standalone database logger, use one of the following options.

If you are unsure which option to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If it is possible for you to stop a logger for a short period of time, collect a trace of the logger from startup.

  For more information, see “Collecting a Managed File Transfer for z/OS standalone database logger trace from startup” on page 506.
- If it is not possible for you to stop a logger, then collect a trace dynamically using the **fteSetLoggerTraceLevel** command.

  For more information, see “Collecting a Managed File Transfer for z/OS standalone database logger trace dynamically” on page 508.

z/OS *Collecting a Managed File Transfer for z/OS standalone database logger trace from startup*

Where it is possible for you to stop a logger for a short period of time, you should collect IBM MQ Managed File Transfer logger trace from startup.

About this task

The way to collect the trace depends on whether the logger is being administered using z/OS UNIX System Services (z/OS UNIX) or JCL.

If you are unsure which of the following options to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
  - z/OS UNIX, see "Collecting a standalone database logger trace from startup using z/OS UNIX" on page 506.
  - JCL, see "Collecting a standalone database logger trace from startup using JCL" on page 507.

z/OS *Collecting a standalone database logger trace from startup using z/OS UNIX*

To collect a trace of a Managed File Transfer for z/OS logger that is being administered using z/OS UNIX System Services (z/OS UNIX) from startup, you need to set various properties in the `logger.properties` file for that logger before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

   The `logger.properties` file can be found in the *BFG_DATA*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.
2. Edit the file and add entries for the following properties:
   - **`trace`**=*trace specification*

     The **`trace`** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
   - **`traceFiles`**=*number of trace files to use*
   - **`traceSize`**=*size of each trace file, in MB*

     The **`traceFiles`** and **`traceSize`** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

   ```
   trace=com.ibm.wmqfte=all
   traceFiles=5
   traceSize=200
   ```

   This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

   For more information on these logger properties, see The MFT `logger.properties` file.
3. Stop the logger that needs to be traced, using the **`fteStopLogger`** command.

4. Start the logger, by running the **fteStartLogger** command.
5. Reproduce the issue.
6. Stop the logger.
7. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step "2" on page 506.

   This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the logger.

z/OS *Collecting a standalone database logger trace from startup using JCL*

To collect a trace of a Managed File Transfer for z/OS logger that is being administered using JCL from startup, you need to set various properties in the `logger.properties` file for that logger before it is started.

About this task

To collect a trace from startup, complete the following steps.

Procedure

1. Locate the `logger.properties` file for the logger that needs to be traced.

   The `logger.properties` file can be found in the *BFG_DATA*/mqft/config/*coordination_qmgr_name*/loggers/*logger_name* directory.
2. Edit the file and add entries for the following properties:

   - **trace**=*trace specification*

     The **trace** property determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
   - **traceFiles**=*number of trace files to use*
   - **traceSize**=*size of each trace file, in MB*

     The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected. You should set these properties to large values, to collect as much trace data as possible.

   For example, to collect 1GB of wrapping trace using the trace specification `com.ibm.wmqfte=all`, add the following lines to the `logger.properties` file:

   ```
   trace=com.ibm.wmqfte=all
   traceFiles=5
   traceSize=200
   ```

   This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

   For more information on these logger properties, see The MFT `logger.properties` file.
3. Locate the data set containing the JCL for the logger that needs to be traced.
4. Submit the BFGLGSP member within the data set to stop the logger.
5. Restart the logger, by submitting the BFGLGST member in the data set.
6. Reproduce the issue.
7. Submit the BFGLGSP member in the data set to stop the logger again.
8. Edit the `logger.properties` file for the logger, and remove the entries for the **trace**, **traceFiles**, and **traceSize** properties that you added in step "2" on page 507.

   This ensures that trace is not enabled the next time you restart the logger.

Results

The resultant trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the logger.

z/OS *Collecting a Managed File Transfer for z/OS standalone database logger trace dynamically*

Where it is not possible for you to stop a logger for a short period of time, you should collect Managed File Transfer for z/OS logger trace dynamically.

About this task

The way to collect the trace depends on whether the logger is being administered using z/OS UNIX System Services (z/OS UNIX) or JCL.

If you are unsure which of the following options to use, contact IBM Support and they will advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- If you are using:
  - z/OS UNIX, see "Collecting a standalone database logger trace dynamically using z/OS UNIX" on page 508.
  - JCL, see "Collecting a standalone database logger trace dynamically using JCL" on page 509.

z/OS *Collecting a standalone database logger trace dynamically using z/OS UNIX*

Under z/OS UNIX System Services (z/OS UNIX), you can use the **fteSetLoggerTraceLevel** command to collect trace from a running logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a Managed File Transfer for z/OS logger using the **fteSetLoggerTraceLevel** command, complete the following steps.

Procedure

1. Turn trace on for the logger, by running the following command:

   ```
   fteSetLoggerTraceLevel -traceLogger classes=level logger_name
   ```

   The `-traceLogger` parameter determines the internal classes and packages that are to be traced. Unless otherwise specified by IBM Support, set this property to the value `com.ibm.wmqfte=all`.
2. Reproduce the issue.
3. Turn trace off for the logger, by running the following command:

   ```
   fteSetLoggerTraceLevel -traceLogger =off logger_name
   ```

4. If a logger is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

   If you can stop the logger for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

   a) Schedule some time to stop the logger.

   b) Locate the `logger.properties` file for the logger that needs to be traced.

The `logger.properties` file can be found in the *BFG_DATA*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

c) Edit the file and add entries for the following properties:

traceFiles=*number_of_trace_files_to_use*
traceSize=*size_of_each_trace_file_in_MB*

The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

You should set these properties to large values, to collect as much trace data as possible.

For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

```
traceFiles=5
traceSize=200
```

This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

For more information on these logger properties, see The MFT `logger.properties` file.

d) Stop the logger, by running the **fteStopLogger** command.

e) Start the logger, by running the **fteStartLogger** command.

f) Turn trace on for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger trace_specification logger_name
```

g) Reproduce the issue.

h) Turn trace off for the logger, by running the following command:

```
fteSetLoggerTraceLevel -traceLogger =off logger_name
```

Results

The resultant trace files are written to the *BFG_DATA*`/mqft/logs/`*coordination_qmgr_name*`/loggers/`*logger_name*`/logs/trace`*%PID%* directory, where %PID% is the process identifier for the logger.

z/OS *Collecting a standalone database logger trace dynamically using JCL*

You can use the BFGLGTC member within the dataset containing the JCL, for the logger that needs to be traced, to collect trace from a running Managed File Transfer for z/OS logger. This can be very useful if IBM Support need to see a trace from a logger that cannot be stopped.

About this task

To collect a trace from a logger using the BFGLGTC member, complete the following steps.

Procedure

1. Locate the dataset containing the JCL for the logger that needs to be traced.
2. Edit the BFGLGTC member within the dataset, and locate the line that contains the text:

```
-traceLogger
```

The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by IBM Support, leave this value as is.

3. Submit the BFGLGTC member.
4. Reproduce the issue.
5. Edit the BFGLGTC member again, and set the **-traceLogger** parameter to *=off*, as shown:

```
-traceLogger =off +
```

6. Submit the BFGLGTC member again, to turn trace off.
7. If a logger is busy, then the trace files might wrap quickly and overwrite the information needed to investigate the issue.

   If you can stop the logger for a short period of time, complete the following steps to reduce the amount of trace data that is collected. Otherwise, contact IBM Support and discuss alternative trace specifications to reduce the amount of trace data that is collected.

   a) Schedule some time to stop the logger.

   b) Locate the `logger.properties` file for the logger that needs to be traced.

      The `logger.properties` file can be found in the *BFG_DATA*`/mqft/config/`*coordination_qmgr_name*`/loggers/`*logger_name* directory.

   c) Edit the file and add entries for the following properties:

      traceFiles=*number_of_trace_files_to_use*
      traceSize=*size_of_each_trace_file_in_MB*

      The **traceFiles** and **traceSize** properties are used to control the amount of trace data that is collected.

      The default value of the **traceFiles** property is 5, and the **traceSize** property has the default value of 20MB. This means that if you turn on trace dynamically, and you have not set the properties, the agent writes trace information to 5 wrapping trace files, each with a maximum size of 20MB.

      You should set these properties to large values, to collect as much trace data as possible.

      For example, to collect 1GB of wrapping trace, add the following lines to the `logger.properties` file:

      ```
      traceFiles=5
      traceSize=200
      ```

      This results in the logger writing trace data to a maximum of 5 files, where each file has a size of 200MB.

      For more information on these logger properties, see The MFT `logger.properties` file.

   d) Locate the data set containing the JCL for the logger that needs to be traced.

   e) Submit the BFGLGSP member within the data set to stop the logger.

   f) Restart the logger, by submitting the BFGLGST member in the data set.

   g) Edit the BFGLGTC member within the data set, and locate the line that contains the following text:

      ```
      -traceLogger
      ```

      The text following this contains the list of internal classes and packages that are to be traced. By default, this list is set to:

```
com.ibm.wmqfte=all
```

Unless otherwise specified by IBM Support, leave this value as is.

h) When it is time to enable trace, submit the BFGLGTC member.

i) Reproduce the issue.

j) Edit the BFGLGTC member again, and set the **-traceLogger** parameter to *=off* by running the following command:

```
-traceLogger =off +
```

k) Submit the BFGLGTC member again, to turn trace off.

Results

The trace files are written to the *BFG_DATA*/mqft/logs/*coordination_qmgr_name*/loggers/*logger_name*/logs/trace*%PID%* directory, where %PID% is the process identifier for the logger.

ALW 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.

  You can also generate trace information for selected areas of interest.
- 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 IBM Support and they will be able to advise you on the best way to collect trace for the issue that you are seeing.

Procedure

- 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.
  Run the following command:

  ```
  strmqtrc -m qmgr_name -t mqxr
  ```

  where *qmgr_name* is the name of the queue manager where the IBM MQ MQXR service is running, and **-t** *mqxr* restricts trace output to the MQXR service only.
  c) Reproduce the issue.
  d) Stop trace, by running the command:

  ```
  endmqtrc -m qmgr_name
  ```

- 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 AIX

```
./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 AIX

```
./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 AIX /var/mqm/trace.

The trace files containing the information from the MQXR service are called mqxr_*N*.trc, where *N* is a number.

The trace files are named as follows:

- The trace files containing the information from the MQXR service are called mqxrRunMQXRService_*PPPPP*.*N*.trc, where *PPPPP* is the process identifier for the MQXR service and *N* is a number.
- The trace files containing the information from the **controlMQXRChannel** command are called mqxrControlMQXRChannel_*PPPPP*.*N*.trc, where *PPPPP* is the process identifier for the MQXR service and *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 tasks

“Troubleshooting IBM MQ Telemetry problems” on page 178
Look for a troubleshooting task to help you solve a problem with running IBM MQ Telemetry applications.

Related reference

strmqtrc

Tracing the REST API

The trace facilities in the REST API are provided to help IBM Support representatives to diagnose problems. Various properties control the behavior of these facilities.

Before you begin

Include the following files and directories when you gather diagnostic information for IBM Support:

- The `mqweb.xml` file.
- The contents of the directory that contains the mqweb server definition:
  - ALW *MQ_DATA_PATH*/web/installations/*installationName*
  - z/OS The directory that was specified when the **crtmqweb** script ran to create the mqweb server definition. By default, this directory is `/var/mqm/web/installation1`.

About this task

The REST API consists of two functional areas, each with their own trace mechanisms:

- Trace for the REST API code that runs in the mqweb server.
- Messaging trace for the REST API code that runs in the mqweb server.

For information about enabling trace for the IBM MQ Console, see "Tracing the IBM MQ Console" on page 466

Procedure

1. Enabling trace for the REST API code that runs in the mqweb server
   a) Enter the following command on the command line:

   ```
   setmqweb properties -k traceSpec -v
   "*=info:com.ibm.mq*=all:com.ibm.mq.rest*=all:js.mq*=all"
   ```

   If the mqweb server is running, trace is immediately enabled.

   Trace is output to a set of files. The directory where the trace files are created is:

   - ALW *MQ_DATA_PATH*/web/installations/*installationName*/servers/mqweb/logs
   - z/OS The subdirectory `/servers/mqweb/logs` beneath the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition. By default, this directory is `/var/mqm/web/installation1`.

   The active file is called `trace.log`. Historical trace is kept in files that are called `trace_`*timestamp*`.log`. The size of these trace files, and the number of historical files that are kept can be configured by setting the `maxTraceFileSize` and `maxTraceFiles` variables. By default, the maximum trace file size is 20 MB, and the maximum number of trace files is 2. For more information, see Configuring logging.
2. Enabling messaging trace for the REST API code that runs in the mqweb server
   a) Create a file called `jmstrace.config` in one of the following directories:

   ALW *MQ_DATA_PATH*/web/installations/*installationName*/servers/mqweb

   z/OS *WLP_user_directory*/servers/mqweb

   Where *WLP_user_directory* is the directory that was specified when the **crtmqweb** script ran to create the mqweb server definition.

b) Add the following lines to the `jmstrace.config` file:

```
com.ibm.msg.client.commonservices.trace.outputName=PATH/logs/jmstrace.txt
com.ibm.msg.client.commonservices.trace.limit=104857600
com.ibm.msg.client.commonservices.trace.count=10
com.ibm.msg.client.commonservices.trace.status=ON
```

Where *PATH* specifies the full path to the directory where you want the `jmstrace.txt` file to be written.

These lines set the maximum trace file size to 100 MB, and set the maximum number of trace files to 10. Ensure that you have disk space available for these files.

c) In the same directory as the `jmstrace.config` file, open or create the `jvm.options` file.

d) Add the following lines to the `jvm.options` file:

```
-Dcom.ibm.msg.client.commonservices.trace.startup=TRUE
-Dcom.ibm.msg.client.config.location=CONFIG_PATH/jmstrace.config
```

Where *CONFIG_PATH* specifies the full path to the directory where the `jmstrace.config` file is located, as a URL. For example, `file:c:/ProgramData/IBM/MQ/web/installations/Installation2/servers/mqweb/`.

e) Restart the mqweb server by using the following commands on the command line:

```
endmqweb
strmqweb
```

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Tracing `runmqakm`

How to request **runmqakm** trace.

`runmqakm` trace

To request **runmqakm** trace, 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 by using FTP, it must be transferred in binary transfer mode.

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Tracing the WCF custom channel for IBM MQ

You can use IBM MQ trace to collect detailed information about what various parts of the IBM MQ code is doing. When using Windows Communication Foundation (WCF), a separate trace output is generated for the Microsoft Windows Communication Foundation (WCF) custom channel trace integrated with the Microsoft WCF infrastructure trace.

About this task

Fully enabling trace for the WCF custom channel produces two output files:

1. The WCF custom channel trace integrated with the Microsoft WCF infrastructure trace.
2. The WCF custom channel trace integrated with XMS .NET.

By having two trace outputs, problems can be tracked at each interface using the appropriate tools, for example:

- WCF problem determination using suitable Microsoft tooling.
- IBM MQ MQI client issues using the XMS trace format.

To simplify trace enablement, the .NET TraceSource and XMS .NET trace stack are both controlled using a single interface.

There are two options for configuring WCF trace for the Non-SOAP/Non-JMS interface. You can either configure trace programmatically or through an environment variable.

Procedure

To enable WCF trace for the Non-SOAP/Non-JMS interface, choose one of the following options:

- Configure trace through an environment variable by setting **WMQ_TRACE_ON** as the environment variable.
- Configure trace programmatically by adding the following section of code to the `<system.diagnostics><sources>` section in the `app.config` file

```
<source name="IBM.WMQ.WCF" switchValue="Verbose, ActivityTracing"
xmsTraceSpecification="*=all=enabled"
xmsTraceFileSize="2000000" xmsTraceFileNumber="4"
xmsTraceFormat="advanced">
</source>
```

Related concepts

“FFST: WCF XMS First Failure Support Technology” on page 425
You can collect detailed information about what various parts of the IBM MQ code is doing by using IBM MQ trace. XMS FFST has its own configuration and output files for the WCF custom channel.

Related tasks

“Troubleshooting WCF custom channel for IBM MQ problems” on page 250
Troubleshooting information to help you solve problems with running Microsoft Windows Communication Foundation (WCF) custom channel for IBM MQ applications.

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Developing Microsoft Windows Communication Foundation applications with IBM MQ

Tracing XMS .NET applications

If you are using IBM MQ classes for XMS .NET Framework, you can configure trace from an application configuration file as well as from the XMS environment variables. If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. You can select the components that you want to trace. Trace is normally used under the guidance of IBM Support.

About this task

Tracing for XMS .NET is based on the standard .NET trace infrastructure.

All tracing except for error tracing is disabled by default.

If you are using IBM MQ classes for XMS .NET Framework, you can turn on tracing and configure the trace settings in the following ways:

- By using an application configuration file with a name that consists of the name of the executable program to which the file relates, with the suffix `.config`. For example, the application configuration file for text.exe would have the name `text.exe.config`. Using an application configuration file is the preferred way of enabling trace for XMS .NET applications. For further details, see "Tracing XMS .NET applications using an application configuration file" on page 517.
- By using XMS environment variables as for XMS C or C++ applications. For more information, see "Tracing XMS .NET applications using XMS environment variables" on page 519.
- By using the `mqclient.ini` file and setting the appropriate properties of the Trace stanza. You can also enable and disable tracing dynamically with the `mqclient.ini` file. For more information, see "Tracing XMS .NET applications with mqclient.ini" on page 521.

Using an application configuration file is not supported for IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries). If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you can configure trace in the following ways:

- From the XMS environment variables. For more information, see "Tracing XMS .NET applications using XMS environment variables" on page 519.
- From IBM MQ 9.3.3, by using the `mqclient.ini` file and setting the appropriate properties of the Trace stanza. You can also enable and disable tracing dynamically with the `mqclient.ini` file. For more information, see "Tracing XMS .NET applications with mqclient.ini" on page 521.

Note:

For the IBM MQ .NETclient libraries built against .NET Framework, the order of precedence is as follows:

1. App.Config
2. Environment variable
3. mqclient.ini

For the IBM MQ .NET client libraries built against .NET Standard and .NET 6, the order of precedence is as follows:

1. Environment variable
2. mqclient.ini

The active trace file has a name of the format `xms_trace`*`PID`*`.log` where *PID* represents the process ID of the application. The size of the active trace file is by default limited to 20 MB. When this limit is reached, the file is renamed and archived. Archived files have names of the format `xms_trace`*`PID_YY.MM.DD_HH.MM.SS`*`.log`.

By default, the number of trace files that are retained is four, that is, one active file and three archived files. These four files are used as a rolling buffer until the application stops, with the oldest file being removed and replaced by the newest file. You can change the number of trace files by specifying a different number in the application configuration file. However, there must be at least two files (one active file and one archived file).

Two trace file formats are available:

- Basic format trace files are human readable, in a WebSphere Application Server format. This format is the default trace file format. The basic format is not compatible with trace analyzer tools.
- Advanced format trace files are compatible with trace analyzer tools. You must specify that you want to produce trace files in advanced format in the application configuration file.

Trace entries contain the following information:

- The date and time that the trace was logged
- The class name
- The trace type
- The trace message

The following example shows an extract from some trace:

```
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Allocate    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    >    Initialize    Entry
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Initialize    Exit
[09/11/2005 14:33:46:914276]    00000004    IBM.XMS.Comms.IoRequest    <    Allocate    Exit
```

In the previous example, the format is:

```
[Date Time:Microsecs]         Thread-id    Classname        Trace-type    Methodname    Entry
or Exit
```

where `Trace-type` is:

> for Entry
< for Exit
d for Debug information

Related concepts

"FFDC configuration for XMS .NET applications" on page 426
For the .NET implementation of XMS, one FFDC file is produced for each FFDC.

Related tasks

"Collecting troubleshooting information" on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

Installing IBM MQ classes for XMS .NET

Downloading IBM MQ classes for XMS .NET Standard from the NuGet repository

Tracing XMS .NET applications using an application configuration file

If you are using IBM MQ classes for XMS .NET Framework, you can configure trace for XMS .NET applications with an application configuration file. The trace section of this file includes parameters that define what is to be traced, the trace file location and maximum allowed size, the number of trace files used, and the trace file format.

About this task

Using an application file to configure trace is supported for IBM MQ classes for XMS .NET Framework.

Using an application configuration file is not supported for IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries). If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. For more information, see "Tracing XMS .NET applications using XMS environment variables" on page 519. From IBM MQ 9.3.3, you can also use the `mqclient.ini` file. For more information, see "Tracing XMS .NET applications with mqclient.ini" on page 521.

Procedure

- To turn on trace using the application configuration file, place the file in the same directory as the executable file for your application.

  Trace can be enabled both by component and trace type. It is also possible to turn on trace for an entire trace group. You can turn on trace for components in a hierarchy either individually or collectively. The types of trace available include:

  - Debug trace
  - Exception trace
  - Warnings, informational messages, and error messages
  - Method entry and exit trace

The following example shows the trace settings defined in the Trace section of an application configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <configSections>
        <sectionGroup name="IBM.XMS">
            <section name="Trace"
                     type="System.Configuration.SingleTagSectionHandler" />
        </sectionGroup>
    </configSections>

    <IBM.XMS>
        <Trace traceSpecification="*=all=enabled" traceFilePath=""
               traceFileSize="20000000" traceFileNumber="3"
               traceFormat="advanced" />
    </IBM.XMS>
 </configuration>
```

Table 31 on page 518 describes the parameter settings in more detail.

Table 31. Application configuration file trace parameter settings

Parameter	Description
traceSpecification=*ComponentName*=*type*=*state*	*ComponentName* is the name of the class that you want to trace. You can use a * wildcard character in this name. For example, `*=all=enabled` specifies that you want to trace all classes, and `IBM.XMS.impl.*=all=enabled` specifies that you require API trace only. *type* can be any of the following trace types: – all – debug – event – EntryExit *state* can be either enabled or disabled. You can string multiple trace elements together by using a ':' (colon) delimiter.
traceFilePath="*filename*"	If you do not specify a traceFilePath, or if the traceFilePath is present but contains an empty string, the trace file is placed in the current directory. To store the trace file in a named directory, specify the directory name in the traceFilePath, for example: `traceFilePath="c:\somepath"`
traceFileSize="*size*"	The maximum allowed size of the trace file. When a file reaches this size, it is archived and renamed. The default maximum is 20 MB, which is specified as `traceFileSize="20000000"`.
traceFileNumber="*number*"	The number of trace files that are to be retained. The default is 4 (one active file and 3 archive files). The minimum number allowed is 2.

Table 31. Application configuration file trace parameter settings (continued)	
Parameter	**Description**
`traceFormat="`*`format`*`"`	The default trace format is basic. Trace files are produced in this format if you specify `traceFormat="basic"`, or if you do not specify a traceFormat, or if the traceFormat is present but contains an empty string. If you require trace that is compatible with trace analyzer tools, you must specify `traceFormat="advanced"`.

The trace settings in the application configuration file are dynamic, and are reread every time the file is saved or replaced. If errors are found in the file after it is edited, the trace file settings revert to their default values.

Related tasks

Tracing XMS .NET applications using XMS environment variables
You can turn on trace using XMS environment variables such as **XMS_TRACE_ON**.

Tracing XMS .NET applications with mqclient.ini
From IBM MQ 9.4.0, you can use the `mqclient.ini` file to enable traces for the XMS .NET client libraries.

Tracing XMS .NET applications using XMS environment variables

You can turn on trace using XMS environment variables such as **XMS_TRACE_ON**.

About this task

If you are using IBM MQ classes for XMS .NET Framework, you can turn on trace using XMS environment variables an alternative to using an application configuration file. The environment variables are only used if there is no trace specification in the application configuration file.

If you are using IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries), you must configure trace from the XMS environment variables. Using an application configuration file is not supported for IBM MQ classes for XMS .NET (.NET Standard and .NET 6 libraries).

Procedure

- To configure trace for an XMS .NET application, set the following environment variables before running the application:

Table 32. Environment variable settings for XMS .NET trace			
Environment variables	**Default**	**Settings**	**Meaning**
XMS_TRACE_ON	Not applicable	Not applicable: the value of this variable is ignored.	If XMS_TRACE_ON is set, all trace is enabled by default.
XMS_TRACE_FILE_NUMBER	4 (one active file and 3 archive files).	The minimum number allowed is 2.	The number of trace files that are to be retained.

Table 32. Environment variable settings for XMS .NET trace (continued)

Environment variables	Default	Settings	Meaning
XMS_TRACE_FILE_PATH	Current working directory	/dirpath/	The directory path that trace and FFDC records are written to. XMS creates FFDC and trace files in the current working directory, unless you specify an alternative location. You can specify an alternative location by setting the environment variable XMS_TRACE_FILE_PATH to the fully qualified path name of the directory where you want XMS to create the FFDC and trace files. You must set this environment variable before you start the application that you want to trace. You must make sure that the user identifier under which the application runs has the authority to write to the directory where XMS creates the FFDC and trace files.
XMS_TRACE_FILE_SIZE	20 MB.	File size is specified in bytes. For example, `XMS_TRACE_FILE_SIZE ="20000000"`.	The maximum allowed size of the trace file. When a file reaches this size, it is archived and renamed.
XMS_TRACE_FORMAT	BASIC	BASIC, ADVANCED	Specifies the required trace format, which can be either BASIC or ADVANCED. The default format is BASIC. The ADVANCED format is compatible with trace analyzer tools.
XMS_TRACE_SPECIFICATION	Not applicable	See “Tracing XMS .NET applications using an application configuration file” on page 517 (IBM MQ classes for XMS .NET Framework only)	Overrides the trace specification, which follows the format specified in “Tracing XMS .NET applications using an application configuration file” on page 517 (IBM MQ classes for XMS .NET Framework only).

Related tasks

Tracing XMS .NET applications using an application configuration file
If you are using IBM MQ classes for XMS .NET Framework, you can configure trace for XMS .NET applications with an application configuration file. The trace section of this file includes parameters that define what is to be traced, the trace file location and maximum allowed size, the number of trace files used, and the trace file format.

Tracing XMS .NET applications with mqclient.ini
From IBM MQ 9.4.0, you can use the `mqclient.ini` file to enable traces for the XMS .NET client libraries.

Related reference

Environment variables descriptions

Tracing XMS .NET applications with mqclient.ini

From IBM MQ 9.4.0, you can use the `mqclient.ini` file to enable traces for the XMS .NET client libraries.

About this task

From IBM MQ 9.4.0, the `mqclient.ini` file includes a Trace stanza. To enable trace, you must set the appropriate properties of the Trace stanza. If a property is set to an invalid value, then the properties are ignored.

You can also enable and disable tracing dynamically. When the application is running, if the `mqclient.ini` file is modified, created, or deleted, the XMS .NET client reads the properties of the trace section again and then enables or disables the trace, so that restart of the application is not needed.

Note: There are several different ways to enable trace.

For the XMS .NET client libraries built against .NET Framework, the order of precedence is as follows:

1. App.Config
2. Environment variable
3. mqclient.ini

For the XMS .NET client libraries built against .NET Standard and .NET 6, the order of precedence is as follows:

1. Environment variable
2. mqclient.ini

Procedure

- To start trace, specify the following attributes of the Trace stanza:
  a) Set **`XMSDotnetTraceLevel`** to the value corresponding to the level of tracing you want.
  If you want to start tracing with an advanced format, set this attribute to 2. If you want to start tracing with a basic format, set the attribute to 1.
  b) Set **`XMSDotnetTraceFilePath`** to the file path of the folder where you want the trace files to be created.
  For example: `XMSDotnetTraceFilePath="c:\somepath"`. The current directory of the application is used if the path is left blank or if the **`XMSDotnetTraceFilePath`** attribute is not defined.
  c) Specify the trace elements that you want to include with **`XMSDotnetTraceSpecification`**.
  You can string multiple trace elements together by using a ':' (colon) delimiter.

- *ComponentName* is the name of the class that you want to trace. You can use a * wildcard character in this name. For example, `*=all=enabled` specifies that you want to trace all classes, and `IBM.XMS.impl.*=all=enabled` specifies that you require API trace only.
- *type* can be any of the following trace types: all, debug, event, EntryExit.
- *state* can be either enabled or disabled.

d) Set the trace file size with **XMSDotnetTraceFileSize**.
The default maximum is 20 MB, which is specified as `XMSDotnetTraceFileSize=20`.

e) Set the number of trace files that are to be retained with **XMSDotnetTraceFileNumber**.
The default is 4 (one active file and three archive files). The minimum number allowed is 2.

- To stop trace, set **XMSDotnetTraceLevel** to `0`.
This is the default value for this attribute.

Related concepts
Trace stanza of the client configuration file
Related tasks
Tracing XMS .NET applications using an application configuration file
If you are using IBM MQ classes for XMS .NET Framework, you can configure trace for XMS .NET applications with an application configuration file. The trace section of this file includes parameters that define what is to be traced, the trace file location and maximum allowed size, the number of trace files used, and the trace file format.

Tracing XMS .NET applications using XMS environment variables
You can turn on trace using XMS environment variables such as **XMS_TRACE_ON**.

Enabling dynamic tracing of LDAP client library code

You can switch LDAP client trace on and off without also stopping or starting the queue manager.

About this task

To enable this behavior, you must 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.

Related tasks
“Collecting troubleshooting information” on page 310

When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

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 IBM Support.

Procedure

See the following links for instructions on how to recover from different types of failures:

- “Disk drive failures” on page 524
- “Damaged queue manager object” on page 525
- “Damaged single object” on page 525
- “Automatic media recovery failure” on page 525

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 526
- z/OS “Active log problems” on page 527
- z/OS “Archive log problems” on page 532
- z/OS “BSDS problems” on page 535
- z/OS “Page set problems” on page 541
- z/OS “Coupling facility and Db2 problems” on page 543
- z/OS “Problems with long-running units of work” on page 546
- z/OS “IMS-related problems” on page 546
- z/OS “Hardware problems” on page 548

Related tasks

“Collecting troubleshooting information” on page 310
When you open a case with IBM, you can include additional troubleshooting information (MustGather data) that you have collected to help with investigating the problem that you are having with IBM MQ.

“IBM MQ troubleshooting and support” on page 5

If you are having problems with your queue manager network or IBM MQ applications, you can use the information in this section to help you diagnose and either solve the problem yourself or obtain help to resolve it.

"Making initial checks" on page 6
There are some initial checks that you can make that may provide answers to common problems that you might have.

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

If a queue manager object is itself damaged, the queue manager performs a preemptive shut down.

There are two ways of recovering in these circumstances, depending on the type of logging you use:

- **For linear logging**, restart the queue manager. Media recovery of the damaged queue manager object is automatic.
- **For circular logging**, restore the last backup of the queue manager data and log, and restart the queue manager.

Damaged single object

If a single object is reported as damaged during normal operation, for linear logging and replicated logging you can re-create the object from its media image. However, for circular logging you cannot re-create a single object.

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 shut down in order to get a clean backup of the queue files.

For information on recovery from a media image, see Recovering damaged objects.

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.

z/OS 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 33. 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 526
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 527

Table 33. Example recovery procedures (continued)		
Problem category	**Problem**	**Where to look next**
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 532
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 535
Page set problems	• Page set full. • A page set has an I/O error.	"Page set problems" on page 541
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 543
Unit of work problems	A long-running unit of work is encountered.	"Problems with long-running units of work" on page 546
IMS problems	• An IMS application terminates abnormally. • The IMS adapter cannot connect to IBM MQ. • IMS not operational.	"IMS-related problems" on page 546
Hardware problems	Media recovery procedures	"Hardware problems" on page 548

z/OS 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.

z/OS 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 527
- "Active log stopped" on page 527
- "One or both copies of the active log data set are damaged" on page 528
- "Write I/O errors on an active log data set" on page 529
- "I/O errors occur while reading the active log" on page 529
- "Active log is becoming full" on page 531
- 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 531

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.

If the offload process has stalled because some resource is not available or for some other reason, it might be possible to resolve the problem by canceling the currently running offload task using the ARCHIVE LOG CANCEL OFFLOAD command.

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 "Troubleshooting IBM MQ for z/OS problems" on page 255.

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.

z/OS 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 532
- "Offload task terminated abnormally" on page 533
- "Insufficient DASD space to complete offload processing" on page 533
- "Read I/O errors on the archive data set while IBM MQ is restarting" on page 534

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).

     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.

z/OS 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 535
- "Log content does not agree with the BSDS information" on page 536
- "Both copies of the BSDS are damaged" on page 536
- "Unequal time stamps" on page 537
- "Out of synchronization" on page 537
- "I/O error" on page 538
- "Log range problems" on page 539

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 earlier 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 earlier 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 two input copies of the BSDSs have different time stamps, or contain a record that is inconsistent. 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, message 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.

If the error code for message CSQJ107E or CSQJ108E is unexpected, a diagnostic dump might be taken.

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 re-create 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 IBM Support.

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 IBM Support 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.

z/OS 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.

An LRSN uses the six 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 LRSN 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 queue sharing group 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.
```

Notes:

1. Message CSQE132I is also generated as the result of auto recovery being invoked. For example, "CSQE153I: Auto recovery for structure ABCD has been scheduled" where **RECAUTO** has been set to YES.
2. As part of the System Programmer Response, message CSQE112E directs you to check for the RBA range referenced in message CSQE130I. However, there are certain instances where message CSQE130I is not produced; for example, if no backup has ever been taken before, or if the backup is ignored because of the value of its LRSN.

z/OS 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 542 describes what happens if a page set is damaged.
- “Page set full” on page 542 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.

z/OS 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 543
- "A Db2 system fails" on page 544
- "A Db2 data-sharing group fails" on page 544
- "Db2 and the coupling facility fail" on page 545

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.

z/OS 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.

z/OS 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 546
- "IMS application problem" on page 547
- "IMS is not operational" on page 547

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.

z/OS 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

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department 49XA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

© Copyright IBM Corp. 2007, 2026

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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 IBM 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

IBM, the IBM logo, ibm.com®, are trademarks of IBM Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information"www.ibm.com/legal/copytrade.shtml. Other product and service names might be trademarks of IBM or other companies.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

This product includes software developed by the Eclipse Project (https://www.eclipse.org/).

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Part Number:

(1P) P/N: