Purpose of this document:

Provide comprehensive information for the chatbot detailing the exact structure of each issue and examples of all cases.

TODOs:

- More links for Hugo to give to chatbot
- Merge with class diagram
- Definition for put1s, puts, gets,..

Issue (Abstract Class)

Purpose: Represents a generic issue within the system pertaining to MQ objects.

Attributes:

- **issueCode**: A string representing the unique code identifier for the issue.
- **startTimeStamp**: A string storing the time at which the issue started.
- **generalDesc**: A description of the issue in general terms.
- **technicalDetails**: A map that captures the technical details related to the issue. The keys are strings representing the detail's name, and the values can be of any object type.
- **MQObjectType**: Specifies the type of MQ Object associated with the issue. It can be one of {queue, channel, app, queueManager}.
- **MQObjectName**: The name of the MQ object that is related to this issue.

ActivitySpike (Subclass of Issue)

Purpose: Represents a specific type of issue in the system where there's too much activity for a given MQ object (QMGR or queue).

Chronology:

- Statistics messages are generated on a configured time interval, for this error we're using: MQCMD_STATISTICS_MQI and MQCMD_STATISTICS_Q:
    - MQI statistics message data (Use this page to view the structure of an MQI statistics message)
        - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=reference-mqi-statistics-message-data
    - Queue statistics message data (Use this page to view the structure of a queue statistics message)
        - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=reference-queue-statistics-message-data

TechnicalDetails Attribute (*rates are per minute, e.g. connRate = connections*60 / timeInterval):

- archivedRequestRates
    - Rate (per minute) of sum of puts, puts failed, put1s, put1s failed, gets, gets failed for that object over the corresponding time interval.
- archivedTimestamps
    - Time interval at which each corresponding entry in other lists of technical details was over.
- archivedConnRates
    - OPTIONAL: if queue manager is the object then this is usually contained, otherwise not.
    - Rate of connections per minute in the time interval.

Too_Much_Activity Queue Manager Example :
{"issueCode":"Too_Much_Activity","startTimeStamp":"2023-08-31T15:49:54.92431251","generalDesc":"","technicalDetails":{"archivedRequestRates":["1722.0"],"archivedTimestamps":[{"14:49:14":[14,49,54]}],"archivedConnRates":["20.0"]},"mqobjectType":"<QMGR>","mqobjectName":"<QMGR>"}

Too_Much_Activity Queue Example :
{"issueCode":"Too_Much_Activity","startTimeStamp":"2023-08-31T15:49:54.951452045","generalDesc":"","technicalDetails":{"archivedRequestRates":["1650.0"],"archivedTimestamps":[{"14:49:14":[14,49,54]}]},"mqobjectType":"<QUEUE>","mqobjectName":"DEV.QUEUE.2"}

ErrorSpike (Subclass of Issue)

Purpose: IBM MQ produces errors all the time if there is a large quantity of the same type of error on the same object within a short space of time, this could indicate a misconfigured app or potentially a cyber/ddos attack.

Chronology:

- QMGR events messages are generated for many reasons. Our app looks at 2035 (07F3) (RC2035): MQRC_NOT_AUTHORIZED of type 1 and 2, these are generated during unauthorised connections or MQOPENS/MQPUT1s respectively. Also our app looks at the number of 2085 (0825) (RC2085): MQRC_UNKNOWN_OBJECT_NAME errors in the queue manager. The information about these errors is available below:

  IBM Documentation (https://www.ibm.com/docs/en/ibm-mq/9.3?topic=types-queue-manager-events & https://www.ibm.com/docs/en/ibm-mq/9.3?topic=descriptions-unknown-object-name):

Too_Many_2035s IBM Hyperlink - https://www.ibm.com/docs/en/SSFKSJ_9.3.0/reference/q041060_.html:

2035 (07F3) (RC2035): MQRC_NOT_AUTHORIZED

General explanation

Explanation

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

- On an MQCONN or MQCONNX call, the user is not authorized to connect to the queue manager. This can be for one of the following reasons:
  - For locally bound applications, the application user ID has not been granted authority to connect to the queue manager.
  - An invalid user ID or password was specified in the MQCSP structure on an MQCONNX call.
  - The queue manager is configured to require applications to provide authentication credentials in an MQCSP structure when connecting, but the application did not provide any credentials.

  On z/OS®, for CICS® applications, MQRC_CONNECTION_NOT_AUTHORIZED is issued instead.

- On an MQCONNX call, the length of the user ID or password is greater than the maximum length permitted. The maximum length of the user ID depends on the platform. For more information, see User IDs.
- On an attempt to connect to the queue manager in an IBM® MQ classes for Java, or an IBM MQ classes for JMS application that is using client transport. This can be for one of the following reasons:
    - The application is using compatibility mode to authenticate with the queue manager, and the length of the user ID is greater than the maximum length permitted of 12 characters.
    - The authentication mode that is used by the application has changed after upgrading the IBM MQ classes for Java or IBM MQ classes for JMS to IBM MQ 9.3.0. For more information about the authentication mode used by Java clients, see Connection authentication with the Java client.
- On an MQOPEN or MQPUT1 call, the user is not authorized to open the object for the option(s) specified.
    - On z/OS, if the object being opened is a model queue, this reason also arises if the user is not authorized to create a dynamic queue with the required name.
- The presence of an Advanced Message Security security policy.

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

Completion code

MQCC_FAILED

Programmer response

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

If the application specifies a user ID and password when it connects, ensure that the credentials are valid.

Ask your system programmer or security administrator to check the queue manager logs for messages that provide additional information.

This reason code is also used to identify the corresponding event message. The event message that is produced depends on the operation that failed, and is one of the following messages:

- MQCONN or MQCONNX Not Authorized (type 1).
    - On an MQCONN or system connection call, the user is not authorized to connect to the queue manager. ReasonQualifier identifies the nature of the error.
    - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=descriptions-not-authorized-type-1
- MQOPEN or MQPUT1 Not Authorized (type 2).
    - On an MQOPEN or MQPUT1 call, the user is not authorized to open the object for the options specified.
    - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=descriptions-not-authorized-type-2

- MQCLOSE Not Authorized (type 3).
- Command Not Authorized (type 4).
- MQSUB Not Authorized (type 5).
- MQSUB destination Not Authorized (type 6).

Specific problems generating RC2035

JMSWMQ2013 invalid security authentication

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

MQRC_NOT_AUTHORIZED on a queue or channel

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

MQRC_NOT_AUTHORIZED (AMQ4036 on a client) as an administrator

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

MQS_REPORT_NOAUTH

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

MQSAUTHERRORS

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

Applications connecting to IBM MQ from WebSphere Application Server

See 2035 MQRC_NOT_AUTHORIZED when connecting to IBM MQ from WebSphere® Application Server for information about troubleshooting MQRC 2035 (MQRC_NOT_AUTHORIZED) errors in an application that is connecting to IBM MQ from WebSphere Application Server.

Too_Many_2085s IBM Hyperlink - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=codes-2085-0825-rc2085-mqrc-unknown-object-name:

2085 (0825) (RC2085): MQRC_UNKNOWN_OBJECT_NAME

Last Updated: 2023-03-27

Explanation

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

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

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

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

This reason code can occur when running the REFRESH CLUSTER command. See Application issues seen when running REFRESH CLUSTER

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

Completion code

MQCC_FAILED

Programmer response

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

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

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.

- Time series data for this issue type is appended into the technical details area as lists.

TechnicalDetails Attribute (*rates are per minute, e.g. connRate = connections*60 / timeInterval):

- List: archivedTimestamps
  - Time interval at which each corresponding entry in other lists of technical details was over.
- List: archivedRequestRates
  - Rate per minute of sum of puts, puts failed, put1s, put1s failed, gets, gets failed for that app over the corresponding time interval.
- String: appName
  - MQCACF_APPL_NAME
- String: connName
  - MQCACH_CONNECTION_NAME
- String: channelName
  - MQCACH_CHANNEL_NAME
- String: CSPUserId
  - MQCACF_CSP_USER_IDENTIFIER
- String: userId
  - MQCACF_USER_IDENTIFIER

Too_Many_2035s Queue Manager Example :

{"issueCode":"Too_Many_2035s","startTimeStamp":"2023-08-30T13:10:49.371294088","generalDesc":"","technicalDetails":{"archivedRequestRates":["27.0","30.0"],"archivedTimestamps":["2023-08-30T13:10:49.371763205","2023-08-30T13:11:09.369852999"],"appName":"\\Python\\Python311\\python.exe","connName":"20.117.73.51","channelName":"DEV.APP.SVRCONN","CSPUserId":"jimmy","userId":"mquser"},"mqobjectType":"<QMGR>","mqobjectName":"<QMGR>"}

Too_Many_2035s Queue Example :

{"issueCode":"Too_Many_2035s","startTimeStamp":"2023-08-30T15:52:55.94274773","generalDesc":"","technicalDetails":{"archivedTimestamps":["2023-08-30T15:52:55.943007455","2023-08-30T15:53:15.942087577","2023-08-30T15:53:35.941973427","2023-08-30T15:53:55.941262234"],"appName":"IBM MQ REST API","archivedRequestRates":["33.0","30.0","30.0","15.0"],"userId":"app"},"mqobjectType":"<Q>","mqobjectName":"INACCESSIBLE.QUEUE"}

Too_Many_2085s Queue Manager Example :

{"issueCode":"Too_Many_2085s","startTimeStamp":"2023-08-30T16:06:34.172473662","generalDesc":"","technicalDetails":{"archivedTimestamps":["2023-08-30T16:06:34.172555697","2023-08-30T16:06:54.169020208","2023-08-30T16:07:14.168558346"],"appName":"\\Python\\Python311\\python.exe","QName":"NON-EXISTENT-QUEUE","connName":"20.117.73.51","archivedRequestRates":["24.0","39.0","36.0"],"channelName":"DEV.APP.SVRCONN"},"mqobjectType":"<QMGR>","mqobjectName":"<QMGR>"}

ConnectionPatternIssue (Subclass of Issue)

Purpose: Represents a specific type of issue associated with a misconfigured connection pattern in the system. If an application has the following pattern:

Connection pattern 1: CONN-PUT-PUT-PUT-DISCON.

This is efficient when compared to:

Connection pattern 2: CONN-PUT-DISCON-CONN-PUT-DISCON-CONN-PUT-DISCON.

Chronology:

- MQCMD_ACCOUNTING_MQI messages are generated when an app connects and disconnects from a queue manager. The application will analyse this data and produce this issue according to the following rules:
- Misconfigured_Connection_Pattern IBM Hyperlink - https://www.ibm.com/docs/en/ibm-mq/8.0?topic=collection-accounting-message-generation:

> Accounting messages are generated when an application disconnects from the queue manager. Intermediate accounting messages are also written for long running IBM® MQ applications.
>
> Accounting messages are generated in either of the following ways when an application disconnects:
>
> - The application issues an MQDISC call
> - The queue manager recognises that the application has terminated
>
> Intermediate accounting messages are written for long running IBM MQ applications when the interval since the connection was established or since the last intermediate accounting message that was written exceeds the configured interval. The queue manager attribute, ACCTINT, specifies the time, in seconds, after which intermediate accounting messages can be automatically written. Accounting messages are generated only when the application interacts with the queue manager, so applications that remain connected to the queue manager for long periods without executing MQI requests do not generate accounting messages until the execution of the first MQI request following the completion of the accounting interval.

The default accounting interval is 1800 seconds (30 minutes). For example, to change the accounting interval to 900 seconds (15 minutes) use the following MQSC command:

- ALTER QMGR ACCTINT(900)

-
- If an app connects more than a set threshold per minute then this issue is created.
- A ratio of connections to queue requests (puts+gets+...) must be below a certain threshold or this issue is produced- i.e. if connections are too many per minute compared to actual operations then this issue is produced. Obviously some apps can have connection pattern 2 but over a long period - this is fine. Therefore the ratio of CONNS to requests is only going to create an issue if connections per minute are high enough.

TechnicalDetails Attribute (note: not using per minute rate in this error, instead count over time interval):

- List: archivedLogTimes
  - Time interval at which each corresponding entry in other lists of technical details was over.
- List: archivedRequestCount
  - sum of puts, puts failed, put1s, put1s failed, gets, gets failed for that app over the corresponding time interval.
- List: archivedconns
  - Sum of connections from that app to queue manager during the corresponding time interval.
- List: archiveduserRatio
  - Ratio of connections to requests (puts and gets) during corresponding time intervals.

Missconfigured_Connection_Pattern Example:

{"issueCode":"Misconfigured_Connection_Pattern","startTimeStamp":"2023-08-30T14:47:59.64515021","generalDesc":"Ratio of MQCONNS to GETS/PUTS is above 0.8 too high in the configured interval of 10.","technicalDetails":{"archivedconns":["50","50"],"archivedlogTimes":[{"14:47:49.645349832":[14,47,59,645349832]},{"14:48:39.644501251":[14,48,49,644501251]}],"archiveduserRatio":["1.0","1.0"],"archivedRequestCount":["50","50"]},"mqobjectType":"<APPLICATION>","mqobjectName":"app"}

QueueServiceHighIssue (Subclass of Issue)

Purpose: Represents a specific type of issue associated with a high service event in a queue. This issue is raised when there's an unusually high activity in a specific queue that might need attention.

Chronology:

QueueService interval events IBM Hyperlink - https://www.ibm.com/docs/en/ibm-mq/9.2?topic=events-queue-service-interval:

Queue service interval events

Queue service interval events indicate whether an operation was performed on a queue within a user-defined time interval called the *service interval*. Depending on your installation, you can use queue service interval events to monitor whether messages are being taken off queues quickly enough.

Queue service interval events are not supported on shared queues.

The following types of queue service interval events can occur, where the term *get operation* refers to an `MQGET` call or an activity that removes a messages from a queue, such as using the `CLEAR QLOCAL` command:

Queue Service Interval OK
Indicates that after one of the following operations:

- An MQPUT call
- A get operation that leaves a non-empty queue

a get operation was performed within a user-defined time period, known as the *service interval*.
Only a get operation can cause the Queue Service Interval OK event message.
Queue Service Interval OK events are sometimes described as OK events.

Queue Service Interval High
Indicates that after one of the following operations:

- An MQPUT call
- A get operation that leaves a non-empty queue

a get operation was not performed within a user-defined service interval.

Either a get operation or an MQPUT call can cause the Queue Service Interval High event message. Queue Service Interval High events are sometimes described as High events.

To enable both Queue Service Interval OK and Queue Service Interval High events, set the `QServiceIntervalEvent` control attribute to High. Queue Service Interval OK events are automatically enabled when a Queue Service Interval High event is generated. You do not need to enable Queue Service Interval OK events independently.

OK and High events are mutually exclusive, so if one is enabled the other is disabled. However, both events can be simultaneously disabled.

Figure 1 shows a graph of queue depth against time. At time P1, an application issues an MQPUT, to put a message on the queue. At time G1, another application issues an MQGET to remove the message from the queue.

The possible outcomes of queue service interval events are as follows:

- If the elapsed time between the put and the get is less than or equal to the service interval:
  - A *Queue Service Interval OK* event is generated at time G1, if queue service interval events are enabled
- If the elapsed time between the put and get is greater than the service interval:
  - A *Queue Service Interval High* event is generated at time G1, if queue service interval events are enabled.

The algorithm for starting the service timer and generating events is described in Rules for queue service interval events.

- The service timer
  Queue service interval events use an internal timer, called the *service timer*, which is controlled by the queue manager. The service timer is used only if a queue service interval event is enabled.
- Rules for queue service interval events
  Formal rules control when the service timer is set and queue service interval events are generated.

- Enabling queue service interval events
  To configure a queue for queue service interval events you set the appropriate queue manager and queue attributes.

- MQRC_Q_SERVICE_INTERVAL_HIGH IBM Hyperlink: https://www.ibm.com/docs/en/ibm-mq/9.3?topic=descriptions-queue-service-interval-high

2226 (08B2) (RC2226): MQRC_Q_SERVICE_INTERVAL_HIGH

Explanation

- No successful gets or puts have been detected within an interval that is greater than the limit specified in the **`QServiceInterval`** attribute.

Completion code

- MQCC_WARNING

Programmer response

- None. This reason code is only used to identify the corresponding event message Queue Service Interval High.

-

- MQRC_Q_SERVICE_INTERVAL_OK IBM Hyperlink:
  - https://www.ibm.com/docs/en/ibm-mq/9.3?topic=descriptions-queue-service-interval-ok

2227 (08B3) (RC2227): MQRC_Q_SERVICE_INTERVAL_OK

Explanation

- A successful get has been detected within an interval that is less than or equal to the limit specified in the **`QServiceInterval`** attribute.

Completion code

- MQCC_WARNING

- Programmer response
  - None. This reason code is only used to identify the corresponding event message Queue Service Interval OK.
  -

- https://www.ibm.com/docs/en/ibm-mq/9.3?topic=events-queue-service-interval

TechnicalDetails Attribute:

- int: isActiveIssue
  - When a queue generates a service high event, the issue is active. When the service ok event is detected then this flips to an inactive issue.

Queue_Service_High example (before queue service ok event detected):

{"issueCode":"Queue_Service_High","startTimeStamp":"2023-08-30T20:25:45.669868427","generalDesc":"Queue: DEV.QUEUE.2 produced a service high event at: 2023-08-30T20:25:45.669868427","technicalDetails":{"isActiveIssue":"1"},"mqobjectType":"<QUEUE>","mqobjectName":"DEV.QUEUE.2"}

Queue_Service_High example (after queue service ok event detected):

{"issueCode":"Queue_Service_High","startTimeStamp":"2023-08-30T20:25:45.669868427","generalDesc":"Queue: DEV.QUEUE.2 produced a service high event at: 2023-08-30T20:25:45.669868427; Queue interval is now ok as of: 2023-08-30T20:26:15.701040124.","technicalDetails":{"isActiveIssue":"1"},"mqobjectType":"<QUEUE>","mqobjectName":"DEV.QUEUE.2"}

Layout of Timestamp:


```
{
        "14:47:49.645349832": [
          14,
          47,
          59,
          645349832
        ]
```

}

=>
"14:47:49 - 14:47:59"

Queue Depth Issues (Subclasses of Issue)

Purpose:

This section describes the types of issues that may arise when a specific MQ queue is becoming overused (THRESHOLD_EXCEEDED) or entirely full (QUEUE_FULL). The aim is to promptly notify users of potential problems so that corrective measures can be taken to avoid

system disruption.

THRESHOLD_EXCEEDED and QUEUE_FULL IBM Hyperlink - https://www.ibm.com/docs/en/ibm-mq/9.1?topic=events-queue-depth:

Queue depth events

Queue depth events are related to the queue depth, that is, the number of messages on the queue.

In IBM® MQ applications, queues must not become full. If they do, applications can no longer put messages on the queue that they specify. Although the message is not lost if this occurs, a full queue can cause considerable inconvenience. The number of messages can build up on a queue if the messages are being put onto the queue faster than the applications that process them can take them off.

The solution to this problem depends on the particular circumstances, but might involve:

- Diverting some messages to another queue.
- Starting new applications to take more messages off the queue.
- Stopping nonessential message traffic.
- Increasing the queue depth to overcome a transient maximum.

Advance warning that problems might be on their way makes it easier to take preventive action. For this purpose, IBM MQ provides the following queue depth events:

Queue Depth High events

Indicate that the queue depth has increased to a predefined threshold called the Queue Depth High limit.

Queue Full events

Indicate that the queue has reached its maximum depth, that is, the queue is full.

A Queue Full Event is generated when an application attempts to put a message on a queue that has reached its maximum depth. Queue Depth High events give advance warning that a queue is filling up. This means that having received this event, the system administrator needs to take some preventive action. You can configure the queue manager such that, if the preventive action is successful and the queue depth drops to a safer level, the queue manager generates a Queue Depth Low event.

Chronology:

There are two primary triggers for these issues:

- THRESHOLD_EXCEEDED:
    - Every queue has an associated threshold value that determines the level of utilization before an alert is raised.
    - By default, the threshold is set to 80%, meaning an alert will be triggered when the queue utilization exceeds this limit.
    - However, users have the flexibility to set custom threshold values through the frontend, allowing them to decide when they should be alerted based on the expected usage patterns of each queue.
    - The issue is activated when the ratio (queue.current_depth / queue.max_number_of_messages) exceeds the specified threshold.
- QUEUE_FULL:
    - This issue is more critical than the former.
    - It is triggered when the current depth of the queue matches its maximum capacity, i.e., queue.current_depth == queue.max_number_of_messages. At this point, the queue can't accept any more messages, leading to potential message losses or backlogs unless promptly addressed.

Technical Details Attribute:

- maxThreshold: This attribute in the alert indicates the threshold limit which, when exceeded, triggers the alert. The value is a fraction (e.g., 0.8 for 80%), representing the ratio of the queue.current_depth to queue.max_number_of_messages. It helps users understand the level at which they have set alerts and assists in taking corrective measures, especially if multiple queues have different custom thresholds.

QUEUE_FULL Issue:
Copy code{ "issueCode": "QUEUE_FULL", "startTimeStamp": "2023-09-04T12:30:45", "generalDesc": "The queue is 100% full. Immediate action required!", "technicalDetails": { "maxThreshold": 0.8 }, "mqobjectType": "<QUEUE>", "mqobjectName": "DEV.QUEUE.1", "objectDetails": "QueueObjectRepresentationForDEV.QUEUE.1" }

THRESHOLD_EXCEEDED Issue:
Copy code{ "issueCode": "THRESHOLD_EXCEEDED", "startTimeStamp": "2023-09-04T14:25:30", "generalDesc": "The queue has exceeded the 80.0% threshold limit. Please take necessary actions to avoid potential issues.", "technicalDetails": { "maxThreshold": 0.8 }, "mqobjectType": "<QUEUE>", "mqobjectName": "DEV.QUEUE.2", "objectDetails": "QueueObjectRepresentationForDEV.QUEUE.2" }