The MQ Vienna Listserver: Lessons from the Past

Lee Wheaton
(Im secessit)

September 25, 2017 13:00
September 26, 2017 15:50

The MQ Vienna Listserver: Lessons from the Past

- Disclaimer
  - The web sites listed in this presentation belong to their respective owners, and as such, all rights are reserved by these owners. These web sites are provided for informational purposes only and are provided “AS IS” without warranty of any kind. It should not be construed that the author of this presentation or MQTC is endorsing or recommending these websites or the companies and products listed therein. Any visits you make to these web sites are done strictly at your own risk.
  - As of September 15, 2017, these web links were still valid URL’s.

The MQ Vienna Listserver: Lessons from the Past

“Those who cannot remember the past are condemned to repeat it.”

George Santayana

Spanish philosopher, poet & novelist.

"Those that fail to learn from history, are doomed to repeat it."

Winston Churchill

What is the MQ Vienna Listserver?

- The MQ Vienna Listserver is an E-mail collaboration system hosted at meduniwien.ac.at (i.e. Medical University of Vienna). It facilitates the exchange of ideas among IBM MQ users and serves as a forum for Q&A, education, MQ announcements and solution’s. Since its inception, the MQ listserver has amassed a treasure trove of valuable information over the years and is an ongoing time capsule on the evolution of IBM MQ.

- The MQ Vienna Listserver has monthly archives going back to February 1997 with a variety of MQ content from MQ users around the world. Prior to 1997, some MQ discussions took place on the CICS Vienna Listserver.

- The Listserver uses LISTSERV which is an email list management software solution developed by L-Soft. LISTSERV allows the management of email newsletters, discussion groups, email communities and opt-in email marketing campaigns.

  Source: L-Soft LISTSERV FAQs

MQ Vienna Listserver Content for this Presentation

- In this session, the presenter will use and elaborate on select content from the listserver archives that is still relevant today.
- Selected content may comprise one or more related email threads which have been summarized for this presentation. In addition, the author of this presentation has updated and augmented this feedback where applicable. This selected content is only a fraction of the total Listserver content.
- For privacy reasons, the names of the participants in a given thread (as well as any related or sensitive information) will not be provided.
- Above all, I do want to recognize the contributors to the Listserver over the years as your collective expertise made this presentation possible and helped so many people in need of answers.
- The PRESENTATION FORMAT will consist of a question followed by summarized feedback from the Listserver membership.

Questions & Feedback

from the

MQ Vienna Listserver Community

Questions selected from the MQ Vienna Listserver

Q01: I’m a newbie, where can I learn more about MQ?
Q02: How do I backup and recover my Queue Manager?
Q03: Why do I need a Dead Letter Queue (DLQ)?
Q04: What is a CCSID or I have a conversion problem?
Q05: How do I trigger CICS transactions to service my queue?
Q06: What is a MQ API wrapper? a Message Broker? a Integration Broker?
Q07: Use local or tempdyn queues for response messages?
Q08: How do MQ channels work with a firewall?
Q09: Why is CHIN traffic coming to a grinding halt?
Q10: Why won’t my z/OS CHIN shutdown without a cancel?
Q11: What is meant by saying MQ is async by nature?
Q12: How does message priority work?
Q13: Where can I look for lost messages?
Q14: Can messages be prioritized on a transmit queue?
Q15: Any value using an Alias Queue versus a Local Queue?
Q16: What is a queue manager trigger interval?
Q17: Can I rebuild messages from the MQ logs?
Q18: How to handle a poisoned message and triggering?
Q19: How to setup Unix mqm GID/UID for shared storage and Active Directory?
Q20: What is an MQ eye catcher and what is it used for?

Q01: I'm a newbie, where can I learn more about MQ?

- Read Mark Taylor's Websphere MQ Primer: IBM Redbook: Websphere MQ Primer
- Read IBM MQ Redbooks IBM MQ Redbooks
- Take IBM MQ training classes: IBM MQ Training Paths
  IBM MQ Sys Admin Certification
- Read the IBM MQ Manuals: IBM MQ Manuals/Documentation
- Read articles via IBM developerworks: IBM developerWorks
- Subscribe to the MQ Vienna Listserver: IBM MQ "Vienna" Listserver
- Join IBM MQ User Groups: IBM Middleware User Community
  NY/NJ IBM MQ User Group
- Read the blogs of IBM MQ experts: IBM MQ Blogs via MQGem Software
- Attend the annual September MQ Technical Conference & other venues
  - ✓ MQTC is the world's largest IBM MQ technical conference at a nominal cost.
  - ✓ Read session handouts from previous MQTC conferences

Q02A: How do I backup & recover my Queue Manager?

- To answer this question, it requires you to answer these questions:
  - ➢ Do you consider MQ to be a service or an application to be recovered?
    - Are you using circular or linear logging? Are messages being replicated?
    - Are the messages to be recovered persistent and/or non-persistent?
  - ➢ Will the recovered Frontend or Backend application databases/files be in sync? Will the FE or BE application be doing point in time recovery or replay msgs?
  - ➢ Which system or application is the official system of record after recovery?
  - ➢ What operating systems do your queue managers reside on?
  - ➢ Is the queue manager standalone, multi-instance, POWERHA, clustered or using shared queues (z/OS)?
    - Does the queue manager need to be up 24x7 or can it be taken offline?
  - ➢ What onsite/DR recovery scenarios do I need to anticipate and support?
    - Is recovery attended or unattended and what are the FTE requirements?
    - What single points of failure do I have and how do I optimize my RTO?

- Due to the variety of answers to the questions above, there is no one size fits all answer to our Q02 question. However, there are some best practices that can serve as the basis for your backup and recovery plan.

Q02B: How do I backup & recover my Queue Manager?

- IBM MQ Backup (or Snapshot) Options
  - ➢ Full Backup of the Queue Manager
    A full backup requires the queue manager to be down so no inflight activity takes place during the backup. It is the best recovery option because it does not require logs to be applied in the queue managers recovery/startup. However, you do have to incur an outage on the queue manager (which may be mitigated by other queue managers standing in for it).
  - ➢ 'Fuzzy' Backup of the Queue Manager
    A Fuzzy backup is taken while the queue manager is up and while inflight activity is taking place. A Fuzzy backup is a snapshot of the queue manager in an inconsistent state. Logs will need to be applied to the recovered queue manager on startup to establish a consistent state. If a required log is missing or corrupted on the queue manager startup, then you will need to go back to a prior Fuzzy backup and apply its associated logs OR use a Full Backup copy.
  - ➢ Backup of the Queue Manager object definitions
    If you view MQ as a service, then an additional option is to use the DMPMQCFG (Distributed) or MAKEDEF (z/OS) command to backup your MQ object and OAM definitions. Then create a new queue manager and use RUNSMQSC/CSQUTIL to restore the objects from the backup. Scripts can be used to expedite things.
    See IBM Knowledgecenter: Backing up and restoring IBM® MQ queue manager data

Q02C: How do I backup & recover my Queue Manager?

- Backup Utilities
  - Distributed or z/OS full disk volume/server backups
  - On Unix, copy the /opt/mqm, /var/mqm and /usr/mqm directories to disk
  - On Windows, backup the Windows Registry and copy the directories noted in this link: IBM Knowledgecenter: Directory structure on Windows systems (Please validate that this still works before using it)
  - On z/OS, copy the BSDS, Pagesets and Logs using ADRDSSU utility (or IDCAMS) See IBM Knowledgecenter: How to back up and recover page sets
    Also see Colin Paice@IBM developerWorks: Backing up and restoring WMQ page sets
    IBM Knowledgecenter: Alternative site recovery
    IBM Knowledgecenter: Shared queue recovery
- Convert Fuzzy to Full Backup Procedure
  If you have a 24x7 queue manager, then take a Fuzzy backup. Restore the Fuzzy backup to another server/LPAR. Start up the queue manager, do some validation and then shut it back down. Then take a full backup. (Some potential risk here. Use carefully.)
- More References to MQ Backup and Recovery
  - IBM Knowledgecenter: Recovering damaged objects
  - IBM Knowledgecenter: Recovering after failure
  - IBM Knowledgecenter: Tips for backup and recovery
  - T.Rob@stackoverflow: Which way is better when recovering Qmgr?
  - mqseries.net: MQ/Queue Manager Backup and Recovery

Q03A: Why do I need a Dead Letter Queue (DLQ)?

- Per IBM, a dead-letter queue (DLQ) is a queue to which a queue manager or application sends messages that cannot be delivered to their correct destination.
  - A DLQ is an electronic version of the Post Office Dead Letter Office.
  - See IBM: Dead-Letter Queues
- An undeliverable message can be due to an incorrect mailing address (queue name), an incorrect CCSID that can’t be converted by a sender channel, security, a destination queue that is full, a message size that is too big and other reasons.
- As an undeliverable message can’t be fully processed and may impact other good messages (e.g. transmit queue, loop), they are sent to the DLQ if one exists. A DLH eyecatcher and control block is added to the beginning of the message block before it is put on the DLQ. The DLH control block contains diagnostic information particularly the reason code for why this message was put on the DLQ. (The receiving application will typically get back a 2033 reason code on their MQGET).
- DLQ messages can be reviewed for what ails them, and then some of them can be resubmitted for processing with a DLQ Handler (RUNMQDLQ) after a fix is in place.
- While developers can put messages on the DLQ, it is recommended that only the MQ system be permitted to put messages on the DLQ especially if a DLQ handler is being used or for security reasons.

Q03B: Why do I need a Dead Letter Queue (DLQ)?

- If a DLQ is not defined to the destination queue manager
  - and a non-persistent message npmclass(fast) cannot be delivered to its destination queue, then that message is DISCARDED!
  - and a non-persistent message npmclass(normal) cannot be delivered to its destination queue, then the sender channel can go indoubt.
  - and a persistent message cannot be delivered to its (remote) destination queue, then that persistent message causes the sender channel to stop and prevents other good messages from processing. Because the bad message is first in line in the transmit queue, starting the channel again will fail.
- If you have a requirement for processing messages in the same exact order as they were sent, a DLQ can impact this ordering should one of the messages end up on it. If you can't have such messages conform to the MQ rules, then specify USEDLQ(NO) on the Sender channel definition to bypass DLQ processing. However, note the issues above and be prepared to address undeliverable messages.
- References

  Angel Rivera@us.ibm.com: Handling Undelivered Messages
  Richard Hamilton@ibm.com: Define a DLQ for each queue manager
  Morag Hughson@imwuc: Little Gem #7: USEDLQ
  protocol7: Websphere MQ Message Spoofing using the DLQ
  IBM Redbook: Secure Messaging Scenarios-Defining a DLQ
  IBM Knowledgecenter: What happens when a message cannot be delivered?

Q04A: What is a CCSID or I have a conversion problem?

- Per IBM, a coded character set identifier (aka Code Page) is a 16-bit number that includes a specific set of encoding scheme identifiers, character set identifiers, code page identifiers, and other information that uniquely identifies the coded graphic-character representation.
- A code page will assign each character and non-character byte to a hexadecimal value ranging from ‘00’ to ‘FF’; or ‘0000’ to ‘FFFF’ in Double Byte Character Set (DBCS) code pages (e.g. Unicode).
- MQ or the O/S has CCSID conversion tables which are used to convert characters in one code page to characters in another code page when either convert on Send or a MQGet with the MQGMO_CONVERT option is specified.
  See IBM Knowledgecenter: Data conversion
  IBM Knowledgecenter: Code page conversion
- CCSID Samples/Examples:
  - ➢ 437-Windows ASCII
  - ➢ 819-Unix/AIX Western European ASCII
  - ➢ 500-z/OS EBCDIC
  - ➢ 1208-Unicode UTF-8 (Default for MQ JMS MQHRF2)
  - ➢ See Characters, Symbols and the Unicode Miracle - Computerphile

Q04B: What is a CCSID or I have a conversion problem?

- When you create a queue manager, it normally is assigned the CCSID of the OS unless you alter it. e.g. A mainframe queue manager will have a CCSID of 500. The Queue Manager CCSID specifies a value that is appropriate for the O/S platform it resides on, and is used when MQCCSI_Q_MGR is specified in MQMD.
- In the message descriptor for a message, the MQMD_CODEDCHARSETID field should contain the CCSID value associated with the data in the message (which should match the MQCCSI_Q_MGR value of the originating platform). Except Unicode
- On a put message, there is a MQMD_FORMAT field value provided by the developer that is used to tell the queue manager the nature of the message data. e.g.
  - MQFMT_NONE - the message data can't be converted i.e. binary/special characters
  - **MQFMT_STRING** - the message data can be converted i.e. Best one for most data
  - MQFMT_RF_HEADER - the message data can be converted i.e. MQRFH header
  - MQFMT_RF_HEADER_2 – the message can be converted i.e. MQRFH2 header
- For MQFMT_NONE, the only way to convert the data is to use a message exit or a conversion routine outside of MQ.
  NOTE-the MQMD-CORRELID and MQMD-MSGID fields are never converted regardless of the specified MQMD_FORMAT value.
- To complete the conversion config for a valid MQMD_FORMAT, you either need to specify convert Yes on the Sender channel or do a MQGET with MQGMO_CONVERT

Q04C: What is a CCSID or I have a conversion problem?

- The generally accepted best practice is to never use convert on Sender, but rather, always specify the MQGET with the MQGMO_CONVERT option on the receiving side. If a message is malformed and you use convert on Send, such a message will generate an error and typically end up back on the sender channel's transmit queue only to be retried over and over.
- If your messages are never malformed and convertible, then I offer up this teaser to you:
  - ➢ Assume a client application is using a request-reply messaging pattern to come into a distributed gateway Queue manager (consolidator) which passes the request to the mainframe and then passes the response back to the client.
    Message conversion is needed between the gateway Queue Manager and the mainframe Queue Manager.
  - ➢ Knowing that MQ z/OS sub-cap costs are based on the processor time used, but not so on the distributed side, it would seem reasonable to have all MQ conversions take place on the distributed gateway queue manager to cut costs
  - ➢ Accordingly, you have the gateway do a convert on send, keep the mainframe sender channel at convert No and the distributed application doing a MQGET with the MQGMO_CONVERT option
  - ➢ Do you continue with the current best practice or go with this new paradigm?

Q05A: How do I trigger CICS txns to service my queue?

- When it comes to servicing queues, it's important to right size and balance the number of CICS servicing tasks, across one or more associated CICS regions, to minimize queue depths and optimize response times. Ideally, these CICS servicing tasks (e.g. message brokers) should be dynamic in nature: increasing in number as traffic increases and conversely going away when no longer needed. See The Reactive Manifesto
- Don't use triggering. Instead, at startup time, kick off dedicated tasks in 1-M CICS regions to listen on and service their respective queues. The Queues need to permit shared access. These tasks also need to be right sized to minimize queue buildups and avoid increased response times. Downside is having idle tasks running that unnecessarily tie up CICS processing slots (that count against max tasks) and use MQ resources.
- Specify Trigger on Every on your local queue. For every message arriving on the queue, a trigger message will be generated that in turn will kickoff a CICS transaction to service the queue. Under heavy load, I have seen Trigger on Every freeze a CICS production region. Also note that MQ z/OS sub-cap cost$ are based on the processor time used.
- Specify Trigger on First on your local queue. A trigger message will be generated when an empty queue receives its first message. Subsequent messages on the queue do not generate a trigger message (until the queue is empty again). For queues with low traffic volumes, this option may suffice, but for higher traffic volumes it won't suffice unless your triggered CICS transaction spawns more iterations of itself. Trigger on First can be the basis for just in time processing.

Q05B: How do I trigger CICS txns to service my queue?

- Specify Trigger on Depth on your local queue. When the number of messages on the queue equals the specified Depth count, a trigger message will be generated that in turn will kickoff a CICS transaction to service the queue. However, when the trigger message is generated, the local queue's trigger flag will be turned off. You will need a way to reactivate the trigger flag as needed. Like Trigger on First, one CICS transaction may not be enough to service the queue.
- Write your own homegrown 'CKTI' transaction to trigger CICS transactions to service the queues. (It's a way to assign a different userid on the started CICS task but fraught with other design/performance considerations).
- Have two or more CICS regions attached to the same queue manager and listening on the same exact trigger initiation queue. To do so, this trigger initiation queue must be set up to permit shared access. The CKTI trigger monitor, in each of the CICS regions, opens up its assigned trigger initiation queue with MQOO-INPUT-AS-Q-DEF and gets the trigger messages to initiate the appropriate CICS transaction in their region. This configuration may result in CICS-MQ traffic being skewed in one or more associated CICS regions.
- Have a CICS TOR region listen in on the trigger initiation queue and the CKTI trigger monitor will kick off the CICS transaction (specified in the MQ process definition for the queue). The TOR can utilize workload management to dynamically route these transactions to one or more CICS AOR regions for servicing. Each of these AOR regions must be attached to the queue manager. One TOR can also be a single point of failure.

Q05C: How do I trigger CICS txns to service my queue?

- A CKTI task in a CICS-MQ AOR region can start a CICS transaction (specified in the trigger message it got) that is defined to that region’s PCT table as a remote transid. Assuming the underlying program has no affinities, the CICSPlex SM WLM will then dynamically route the transaction to whatever CICS region is the best candidate at the time.
- Haven’t tried this yet. Specify an alias queue as the InitQ on the local queue definition (with triggering enabled). Then see IBM: How to generate duplicate messages into other queues . The idea here is to have a trigger message replicated to two or more Trigger initiation queues that two or more CICS-MQ AOR regions are listening on. This config provides more servicing tasks and protects against the loss of a CICS region.
- You can have more than one instance of CKTI running in the same CICS region, but they can not be listening to the same initiation queue. This config can help speed up servicing trigger messages especially in high traffic conditions.
  See IBM Knowledgecenter: Task initiator or trigger monitor (CKTI)

Q06A: What is a MQ API wrapper?

- An MQ API wrapper is essentially a callable subroutine/component that encapsulates and performs MQ calls on behalf of the calling program (i.e. client). The calling program uses a shared API control block to pass the appropriate processing parameters and message data to the subroutine. The subroutine passes back the responses it receives to the calling program through the same API control block.
- Imagine 10 programmers doing their own MQ programming and error handling as well as reinventing the MQ wheel over and over again. Imagine new MQ releases and product features that would have to be propagated/tested across your MQ development environment. What about issues with tightly coupling the MQ code with the business logic?
- What if you could standardize all your MQ code and error handling into a callable subroutine that is highly reliable and easy to use (as well as reduce the app coding effort)? In short, you have an MQ API Wrapper.
- An MQ API Wrapper can be homegrown, obtained commercially or obtained for free as an open source adapter.
- The benefits of a proven MQ API Wrapper:
  - Insulates developers from having to know how to code MQ
  - Supports Rapid Application Development (RAD) by using a Plug-and-Play approach
  - Separates MQ code from the business logic
  - Reduces production and development support calls
  - Facilitates quicker MQ upgrades and adoption of new MQ or security features
  - Partially insulates your business applications from MQ upgrades. They still have to recertify against the upgrade.
  - Supports SOA component reusability
  - Lowers development costs

Q06B: What is an MQ Message Broker?

- A Message Broker is an intermediary program/software, layered on top of the MQ infrastructure, that
  - typically resides between MQ and the servicing layer (whereas the API Wrapper resides between the Client application and MQ)
  - performs all sanctioned MQ calls on behalf of its constituency e.g. Get's Requests/Put's Responses
  - transforms a message from the sender's protocol/format into the protocol/format of the receiver
  - routes messages to their servicing layer and places any results on the outbound response queues
  - supports transaction coordination
  - supports PUB-SUB
  - handles events and errors
  - insulates developers from having to know how to code MQ
  - is a loosely coupled function or service (not a specific product)
- The definition and specifications of a Message Broker have changed over time (see next slide). A Message Broker may also be known as a Message Adapter.
- MQ is passive, in that, it does not initiate messages or service them on its own. Like the network, it takes an application to put or get application messages (on a queue). A Message Broker can replace the need for hundreds of individual applications servicing the queues and isolate MQ code from the business logic.
- A Message Broker needs to be written in a IBM MQ supported programming language and be able to interface with its downstream applications.
- See IBM: MQ: Introduction to MQ, MOM, and Brokers

Q06C: What is an Integration Broker? Is it an ESB?

- Per IBM, an integration broker is a component that integrates data among heterogeneous applications. An integration broker typically provides various services that can route data, as well as a repository of rules that govern the integration process, connectivity to various applications, and administrative capabilities that facilitate integration.
- Per Gartner, an integration broker is a third-party intermediary that facilitates interactions between applications. IBs minimally provide message transformation and routing services. They mostly communicate program to program; they integrate previously independent applications at the application-logic level of the software design.
- An Integration Broker is a superset of a Message Broker, in that, it supports and integrates multiple brokers (or adapters) for Messaging, Database Services, Web Services, Java/JMS Services, XML etc.
- With the rise of Service-Oriented Architecture, the evolution of Brokers and API Wrappers for the various services in the ESB stack, there was a need for an Integration Broker to bring these services together in a cohesive manner to enable service orchestration for SOA.
- Per whatis.com, an ESB can be thought of as a mechanism that manages access to applications and services (especially legacy versions) to present a single, simple, and consistent interface to end-users via ... client-side front ends".
- Is an Integration Broker a Bus? Depending on your viewpoint, it is or it isn’t. Is it a Enterprise Application Integration (EAI) solution? Here are some articles to help you decide.
  - ✓ See Tibco: Integration Broker or Enterprise Service Bus?
  - ✓ See stackoverflow.com: Difference between a Message Broker and an ESB
  - ✓ See MuleSoft: Service Orchestration and SOA

Q07A: Use local or dynamic queues for response msgs?

- For a request-reply messaging pattern, I want the reply message to come back to the original requester and nobody else. It would be a disaster if the wrong person got a response intended for another person. Should I use a local queue or a dynamic queue for response messages?
- Question: does your reply need to be persistent?
  - ➢ A local queue or a permanent dynamic queue can contain both persistent and nonpersistent messages. A temporary dynamic queue can only contain non-persistent messages. Further, a tempdyn queue is unrecoverable & can't use CF.
- IBM recommends avoiding temporary dynamic queues on the mainframe due to higher cpu costs and longer elapsed time. (Distributed MQ is not charged by cpu)
- To route the reply back to the original requester, there are at least several options:
  - ➢ Note-Group ID and MsgToken options won't be covered here.
  - ➢ If you've selected a Dynamic Queue for your reply queue, your application will 1st open up this Dynamic Queue to obtain the unique system generated queue name which will then be used to set the MQMD_REPLYTOQ before issuing a MQPUT. Your application will then do a MQGet w/ wait on the system generated queue name while the backend servicing application processes the message and then sends it back to you by using the MQMD_REPLYTOQ value.

Q07B: Use local or dynamic queues for response msgs?

- To route the reply back to the original requester, there are several options (continued):
  - If you've selected a Local Queue for your reply queue, your application will 1st open up this known Local Queue and then set the MQMD_REPLYTOQ. To ensure that your reply comes back just to you, as it will be on a shared queue, you will also need to set the MQMD_CORRELID and/or MQMD_MSGID fields to a unique value of your choice (or set MQMD_CORRELID to MQCI_NONE and/or MQMD_MSGID to MQMI_NONE for a system generated ID(s)) before you do the MQPUT. Before you do your MQGET w/ wait, you will need to set the MQMD_CORRELID and/or the MQMD_MSGID as well as specifying MQMO_MATCH_MSG_ID or MQMO_MATCH_CORREL_ID.
- Dynamic reply Queues with their generated token id's and limited shelf life may have security value in some cases as only the requesting program knows the name of the Dynamic Queue unlike a known static Local Queue.
- PermDyn queues can build up and need to be manually deleted if the developer does not purge them. Also, how do you handle orphaned reply msgs on LocalQ?
- See IBM Knowledgecenter: Getting a specific message using MsgId and CorrelId
  IBM Knowledgecenter: Dynamic and Model queues
  IBM Support: Performance issues getting messages from a queue with a large CURDEPTH

Q08A: How do MQ channels work with a firewall?

- Per IBM, a firewall is a network configuration, typically both hardware and software, that prevents unauthorized traffic into and out of a secure network.
  - ➢ A firewall can act as a proxy server by intercepting requests intended for another server & then acting on the client's behalf to obtain the requested service (& hiding that service).
- MQ messages coming in to a queue manager on your company's internal network from a queue manager or MQ client on an external network are typically routed through a DMZ on the sending side and a DMZ on the receiving side. (An end-to-end VPN may also be used).
- A demilitarized zone (DMZ) typically consists of a host server or device that is sandwiched in between two firewalls. The 1st Firewall receives in external untrusted traffic and validates the source credentials. If valid, the MQ messages are allowed to pass to a host presence residing in the DMZ. The host's job is to act as a relay station to pass your messages into the 2nd Firewall which will again check the credentials and pass your MQ traffic on to your queue manager residing in the internal network. NOTE-Placing queue managers in the DMZ is not recommended (persistent msgs).
- Possible DMZ Hosts:

  1) MS81 Internet Pass-thru on a server
  2) IBM Datapower appliance
  3) Web server that uses the MQ client

     Consider using OS file security to lockdown access to your MQ client software

Q08B: How do MQ channels work with a firewall?

- Firewall Rules are part of the validation process
  By default, Firewalls will prevent external traffic from coming into your internal network unless you ‘punch holes into the firewall’. Hole punching (i.e. rules based) is a way to connect two entities behind firewalls and/or behind routers that use network address translation (NATing).
- Firewall Admins will need MQ Admin and Telecom input on the IP addresses and ports involved so they can write their Firewall Rules to enable connectivity. e.g

Source IP	Source Port	DMZ IP	DMZ Port	Destination IP	Dest Port	Direction Flow
172.10.18.3	4000	192.14.6.1	2020	10.10.18.24	1414	External-Internal
10.10.18.24	*	192.14.6.1	2021	172.10.18.4	4001	Internal-External
180.9.9.9	*	*	*			**Deny**

- The above table is also an example of packet filtering. Per IBM, a packet filter is a filter that blocks traffic based on a specific IP address , [protocol (TCP)] or type of application (such as email, FTP, or Web) which is specified by port number.
- The MQ listener ports (e.g. 1414) are fixed ports. Outbound ports are Ephemeral.
- **BTW, Port 1414 is a well known port. Consider using a different port (iana.org)**

Q08C: How do MQ channels work with a firewall?

- References

  Paul Sehorne@IBM Dallas via Capitalware: MQSeries and Firewalls

  IBM Support: WebSphere MQ firewall security port selection

  IBM Support: Troubleshooting MQ channels

  IBM Knowledgecenter: Introduction to IBM MQ Internet Pass-Thru

  Robin Wiley@MQTC 2016: MQ-DataPower Connectivity Deep Dive

  Leif Davidsen's Blog: When is a wall a great wall? When it’s a firewall?

  “IBM *does not recommend deploying a queue manager in the DMZ*”
  Source: IBM: IBM MQ Appliance

Q09: Why is CHIN traffic coming to a grinding halt?

- Case Study: We are seeing the channel initiator address space getting swapped out and message processing coming to a grinding halt until the address space is swapped back in. How can we resolve this issue?
- Make the CSQxCHIN (and CSQxMSTR) address space non-swappable
- Set the execution priority of the MSTR and CHIN regions to match the higher priorities of your CICS regions.
- The TCP/IP and VTAM priorities should be higher than your CICS and MQ priorities. (Or you could see your MQ channels bounce up and down).

Q10: Why won't my z/OS CHIN shutdown w/o a cancel?

- When a MQ z/OS queue manager is shutdown, the MSTR region comes down but why does the CHIN continue running or hang? When the MQ z/OS queue manager is brought back up, there is a new and second instance of the CHIN running and the original CHIN has to be manually cancelled carefully.
- This issue has had various permutations in MQ versions over the years. Possible catalysts for this issue based on listserver feedback:
  - ➢ Distributed Client applications are not using '*Fail if Quiescing*' on their MQ calls
  - ➢ Distributed Client applications are not issuing a MQDISC and leaving orphan MQ connections
  - ➢ Distributed Client applications terminating their MQ connection but immediately trying to establish new MQ connections to the z/OS queue manager (i.e. connection loop).
  - ➢ One site found their MQ z/OS listener was still running after the shutdown was issued
  - ➢ One or more MQ z/OS channels are in a abnormal state preventing CHIN shutdown
- Possible Solutions:
  - ➢ If CHIN hangs after STOP QMGR MODE(QUIESCE), try STOP QMGR MODE(FORCE) or STOP CHINIT.
  - ➢ Contact IBM about any APAR/PTF's for this issue
  - ➢ Resolve one or more of the catalysts noted above

Q11A: What is meant by saying MQ is async by nature?

- "Most MQI calls are *SYNCHRONOUS",* bi-directional (SVRCONN) and tightly coupled with a queue manager. (The frontend application does not talk directly to the backend application).
  Source: Morag Hughson@SHARE Anaheim 2011: Introduction to WebSphere MQ Clients
- Placing a MQ queue manager between two applications allows their communications to be asynchronous and loosely coupled.
  Source: IBM: MQ Asynchronous and Synchronous Communication
- A synchronous communication is bi-directional and usually blocks the calling program from doing any other work until the response comes back from the backend application.
- An asynchronous communication is unidirectional and doesn't have to wait for a response to come back (i.e. non-blocking) which allows the calling program to continue on with other work. e.g. The MQ client put's a fire-and-forget message to a queue manager intermediary and continues on with it's work while the backend application get's the message from the queue (manager) and processes it.

Q11B: What is meant by saying MQ is async by nature?

- A “Pseudo-Synchronous” communication uses a synchronous client connection (e.g. SVRCONN) over asynchronous infrastructure to simulate a direct synchronous communication between the client and a backend application. This mode supports a request-reply messaging pattern.

  e.g. The MQ client application can MQPUT a message and then do a MQGET w/ wait for the response (i.e. blocking) or MQPUT the message, do other work and then MQGET the response later (i.e. non-blocking) within the same unit of work. In parallel, the backend application MQGET’s the request from the queue manager, processes it, and MQPUT’s the response back to the local queue that the frontend application is listening on.

- References
  - Neil Johnston@SHARE Pittsburgh 2014: MQ: Beyond the Basics
  - IBM Knowledgecenter: WebSphere MQ synchronous communication
  - IBM Knowledgecenter: WebSphere MQ asynchronous communication

Q12: How does message priority work?

- When a local queue is defined with MSGDLVSQ(PRIORITY), messages put on this queue are handled as follows:
  1) Messages will be split up into a Persistent list and a Non-persistent list within that queue (i.e. sub-queues).
  2) Each list is built dynamically and can contain up to 10 priority levels (0-9).
  3) Each established priority level contains a linked list of message pointers to messages assigned that priority. While persistent and non-persistent Messages with the same priority will be assigned to different lists, at run-time, the lists for each priority level will be merged together at run-time. (FIFO order)
  4) When a persistent message is MQPUT onto a queue with MQMD_PRIORITY = 1, it is assigned to the persistent list for that queue and it's MQPUT timestamp is used to determine its place on the link list for priority 1.
  5) When a unqualified MQGET is done against a queue with priority, the entries in the combined run-time link list for the top priority (9) are processed from beginning to end followed by the next priority level (8). Assuming no priority messages exist for 2-7, then the priority 1 messages will be serviced.

  See IBM Knowledgecenter: Message Priorities
  IBM Knowledgecenter: Logical and physical ordering
  IBM: Getting a particular message

Q13A: Where can I look for lost messages?

- ‘Lost’ messages are a serious matter. Even an incorrect perception that a message was lost can have a number of negative implications. People can lose confidence in their MOM software or incur loss of reputation & financial penalties.
- I break down ‘Lost’ messages into two categories: operational and historical. Operational as in the here and now (real-time) and Historical as in the past.
- Consider a case where customer support or a developer says they ‘lost’ a message three days ago (even though it was successfully delivered).
- Consider the case of Insider/Outsider Hackers/Fraud and MQ audit data is needed to facilitate the investigation and determine the extent of the intrusion. *What do you need in place to provide a Historical audit trail for these cases?*
- Proactive Action Items for addressing ‘Lost’ messages:
  - ➢ Develop and execute a plan to address ‘Lost’ messages
  - ➢ Document all of your application messaging pathways and what components they use
  - ➢ Understand the MQ pathways or stopping points an application message will take
  - ➢ Review typical ‘Lost’ message scenarios (see page Q13C)
  - ➢ Document your triage or forensic instruments/utilities and procedures
  - ➢ Do you need to purchase more forensic tools or other solutions?

Q13B: Where can I look for lost messages?

- The best way to prove a message was delivered successfully is to have an audit trail solution (e.g. Capitalware's MQ Auditor, Homegrown) in place. Ideally, such a solution should track the date/time a message was put or got and by whom as well as contain the associated message data. Connection info is also helpful.
- MQ provides exit points that you can code to that allow you to replicate inbound and outbound messages to an audit log.
- When you lose your car keys, you typically will retrace your steps to find them. So to with MQ. Hence the importance of documenting your MQ and application pathways to trace a given messages steps.
- Leverage your documentation in conjunction with your Admin tools:

➢ OS Event Logs	Any Qdepth buildup	Was Msg persistent?	Recovery Log
➢ MQ Error Logs	Any Channel issues	Was syncpoint cntl used?	Appl MQRC?
➢ Audit Logs	Qmgr healthy?	Msg Expiry used?	Any DLQ msg

- Utilize Forensic tools or utilities

➢ Run MQ, Application and/or OS trace	MQ Explorer (Browse Q)	Third Party Tools
➢ Run network sniffer trace	Notrig LQ & Dump Msgs	Server OS Monitors
➢ Can problem be recreated in test?	Stop Chl. Dump XMITQ	CKQC, CEMT I TRAN

  - ➢ Display Qstatus IPPROCS OPPROCS LGETDATE LGETTIME LPUTDATE LPUTTIME
  - ➢ See IBM Support: Instructions to find persistent Messages on the MQ recovery log

Q13C: Where can I look for lost messages?

- Typical 'Lost' message scenarios with tips on resolution
  - Mark Taylor@MQTC 2016: Where's my message?
  - Barry Lamkin@MQTC 2016: The top issues in IBM MQ and IIB
  - Barry Lamkin@MQTC 2016: What happened to my Transaction?
  - Lyn Elkins@MQTC 2016: Common Problems and PD for MQ V8 on z/OS
  - Tameka Woody@IBM developerWorks: Where are my WebSphere MQ messages?!?
  - IBM Knowledgecenter: Problems with missing messages when using distributed queuing
  - IBM Knowledgecenter: Improving performance of non-persistent messages
  - IBM developerWorks: Why are MQ expired messages discarded or disappearing from a queue without a MQGET?
  - stackoverflow: Websphere and MQ as JMS provider: Lost messages
  - stackoverflow: How do I stop losing messages on MQ

Q14: Can messages be prioritized on a transmit queue?

- If the transmit queue is defined with MSGDLVSQ(PRIORITY), then yes, messages put on that transmit queue can be prioritized for servicing.
- The Sender Channel’s message channel agent (MCA) does an MQGET on the transmission queue and will send the messages over the network in the priority order it gets them.
- If a priority=0 message is created first, and being sent by the channel when a priority=5 message is created, the priority=0 message will arrive first.
- A sending channel MCA will not see any messages that are written under syncpoint until the LUW is committed. Thus a priority 1 message will be sent before a yet-to-be-committed priority 5 message even if the priority 5 message was MQPUT first.
- If the transmit queue has a mixture of small and large sized messages, you may want to create a second transmit queue-sender channel combination to get better throughput. The first combo to process small messages and the second to process the larger messages. If a small, high priority, interactive message arrives on the xmitq, it must wait until a larger message is moved across the channel.

Q15A: Any value using an Alias Queue vs a Local Queue?

- An Alias Queue is a logical construct in that it’s definition is stored as data but it does not have a physical queue to contain messages like a Local Queue.
- On an MQPUT to an Alias Queue, certain Alias Queue attributes will override the corresponding base queue attributes.
- An Alias Queue allows an application to put a message with a different priority/persistence than it’s base queue has and eliminates hard coding priorities/persistence in your program if the MQPUT MQMD specifies MQPRI_PRIORITY_AS_Q_DEF and MQPER_PERSISTENCE_AS_Q_DEF .
- Using Alias Queues, you can permit one set of applications to only put to the base local queue and another set of applications to only get from the base local queue.
- Having multiple Alias Queues for the same base queue allows you to split out or segregate messages in the common base queue into other local queues at a later point in time (without impacting the associated applications). eg AQ by Warehouse
- If an application program hard codes the local queue name and that local queue becomes corrupted, then either you need to recover the local queue (assuming you’re using linear logging) or the program will need to be reassembled to use a different local queue name. Meanwhile, you have an outage situation.

Q15B: Any value using an Alias Queue vs a Local Queue?

- If an application program hard codes the alias queue name and the underlying base queue becomes corrupted, then the MQ Admin can change the base queue name on the alias queue definition. You have a smaller outage (and you still retain the corrupted local queue for triaging).
- Consider eliminating hard coded Alias Queue or Local Queue names by using (external) parameters to the application. However, some consider a hard coded AQ to be OK as it is an internal representation of an external object and the property of the application.
- Using an Alias Queue obscures the name of the base queue name from the requesting application, and enables security access to just the Queue Alias (and not the Local Queue) thus limiting what queue attributes a developer can utilize.
- A Queue Alias allows you to duplicate messages:
  See IBM: How to generate duplicate messages into other queues

Q16: What is a queue manager trigger interval?

- A queue manager definition contains a trigger interval attribute, TRIGINT, which is set in milliseconds. The default value is 999999999 milliseconds or 11.5+ days.
- The TRIGINT attribute applies to every local queue in the queue manager with a trigger type (TRIGTYPE) setting of FIRST. Based on the queue manager TRIGINT setting, it is possible to generate a trigger message even when the local queue is not empty. This method of processing is referred to as backstop triggering.
- In short, if no trigger on first message has been generated for a non-empty local queue within the TRIGINT time interval, then the queue manager will kick off one trigger message to the initiation queue (and the timeclock starts all over again).
- A queue manager trigger interval can act as a auto-safety net should a normal Trigger on First fail for some reason (e.g. intermittent application or system issue).
- Setting a non-empty local queue manually to NOTRIG and then TRIG will also generate a trigger message.
- Your MQ monitor may have the capability to NOTRIG and TRIG a local queue when CURDEPTH is greater than zero and IPPROCS equals zero.

Q17: Can I rebuild messages from the MQ logs?

- The MQ logs are used primarily for transaction recovery.
  At this time, I am unaware of any official published document showing the record layout for a MQ log. Such a document would be subject to change with later MQ releases. See IBM Knowledgecenter: MQ z/OS log files
- (Committed) Persistent messages are always logged and survive a restart of the queue manager. See IBM Knowledgecenter: IBM MQ Message Persistence
- Non-persistent messages are not logged. They do not survive a restart of the queue manager unless it is a normal/controlled restart and queue attribute NPMCLASS(HIGH) is specified, or stored in a z/OS coupling facility (CF).
- Circular logs wrap around & overlay previous log entries. Linear logs keep entries.
- The MQ z/OS log print utility (CSQ1LOGP) has an Extract function that may allow persistent messages to be replayed. See IBM Knowledgecenter: MQ CSQ1LOGP Utility
- Distributed MQ has the DMPMQLOG utility
  See IBM Knowledgecenter: dmpmqlog & IBM Knowledgecenter: Dumping the MQ Log
- 3rd party vendors may have solutions to rebuild MQ log messages & replay them.
- In short, extracting from the MQ log is ugly, does not contain non-persistent messages and someone could bypass a MQ log audit with non-persist messages.

Q18: How to handle a poisoned message and triggering?

- Per IBM, a poisoned message is an incorrectly formatted message that the receiving application cannot process. The message can be repeatedly delivered to the input queue and repeatedly backed out by the application (i.e. a loop).
- Case Study: A triggered CICS txn does a mqget under syncpoint from a local queue. The application abends causing a back out of the message. The message is now back on the local queue which causes another trigger message to be fired off. This process continually repeats itself. Static CICS txns can also be impacted. NOTE-if the mqget is not done under syncpoint then the message will be lost.
- Suggested remedies:
  - ➢ Turn off triggering for the queue or disable the CICS transid
  - ➢ Enable **HardenGetBackout** on the local queue and set the **BackoutThreshold** and **BackoutRequeueQName (**by providing the name of the back out queue). The receiving application then needs to interrogate the MQMD backout count after the MQGET and determine if it needs to do a MQINQ call on the local queue to get the **BackoutThreshold** and **BackoutRequeueQName** values. If the message is considered poisoned, then the receiving application MQPUT's the message to the backout request queue for later disposition and then proceeds to MQGET the next message in the queue
  - ➢ See IBM Knowledgecenter: Handling poison messages in IBM MQ classes for JMS
  - ➢ Test Exception Handling for poisoned messages in the development environment

Q19: How to setup mqm GID/UID for shared storage & AD?

- On Unix systems, when MQ is installed, a "mqm" group id (GID) and a "mqm" user id (UID) are created. The mqm uid is a member of the mqm group. These ID's are stored as integer numbers and are unique. You can either let the MQ install create the mqm id's or you can predefine them manually before you do the MQ install.
- The Unix files created from an MQ install are owned by the mqm UID
  e.g. /opt/mqm, /usr/mqm and /var/mqm directories and files
- When local or SAN storage is shared for a Unix multi-instance queue manager or for PowerHA, both the active node and the passive node must have the "mqm" GID and UID in sync. i.e. Their integer numbers for mqm must be the same value. **match /etc/passwd?**
- It's easier and safer to predefine these mqm id's and sync them up before installing MQ instead of altering the id's (and their associated files) after MQ has been installed.
- If an Active Directory solution is managing your Unix server security for one identity centralization, then you will typically need to create mqm GID/UID on each Unix server that matches the mqm GID/UID defined to Active Directory before installing MQ. Based on experience, place your MQ Admin UID's in AD and the local mqm group.
- If you need to change the local mqm GID/UID values after MQ installed then safer to uninstall and reinstall MQ after the new mqm values have been set. A more riskier approach is to stop MQ, change the mqm GID/UID and the associated mqm file ownerships and hope MQ comes up again. Carefully consider the use of these web links: rivera@us.ibm.com: MIQ
  IBM developerWorks: How do you change the group id for MQ?
  IBM developerWorks: Changing UIDs and GIDs

Q20: What is an MQ eye catcher and what is it used for?

- Per IBM, an eye catcher is a string value included in program code or data to make it easy to find specific program sections or types of information. At the beginning of every MQ data control block in a MQ trace, is a four character structure identifier or string value (e.g. “DLH “, “MQ “). Each structure identifier is at the beginning of their respective record layout/control block.
  See IBM Knowledgecenter: Data types used in the MQI and z/OS SCSQCOBC PDS
  (Note the ‘Initial values and language declarations’ section for each structure)
- If you are going through a MQ Trace or FFST, an eye catcher will help you jump faster to the section you want. You will then need the record layout for that eye catcher to find out such things as the reason a message was put on the DLQ, the ReplyToQ name in the MQMD, the MQPUT/MQGET parameters used (PMO/GMO), the CCSID, etc.
- MQ Explorer can be used to browse and format individual messages in the queue into their respective record layouts.
- Binary/Integer values are stored differently internally depending on whether your platform supports the Little Endian or Big Endian format. See Wikipedia: Endianness
  e.g. A MQ reason code 2033 equals x’07F1’. z/OS stores the reason code in Big Endian format or x’07F1’. A Windows server stores the reason code in Little Endian as x’F107’

Summary

The MQSeries Vienna Listserver is one of the first MQ forums established to facilitate the exchange of ideas, answer questions and provide solutions among members of the IBM MQ(series) community. Since its inception in 1997, the MQSeries listserver has amassed a treasure trove of valuable information over the years.

In this session, we reviewed 20 common questions from the Listserver, a small subset of their total content, and the summarized/updated feedback that was provided by the Listserver membership and the presenter. Most of these questions have been repeatedly asked (and answered) at various times over the years.

I hope you enjoyed our trip down memory lane.

“Those who do not learn from the past are condemned to repeat it.”

Questions & Answers

The MQ Vienna Listserver: Lessons from the Past

Thank you for attending this session!