

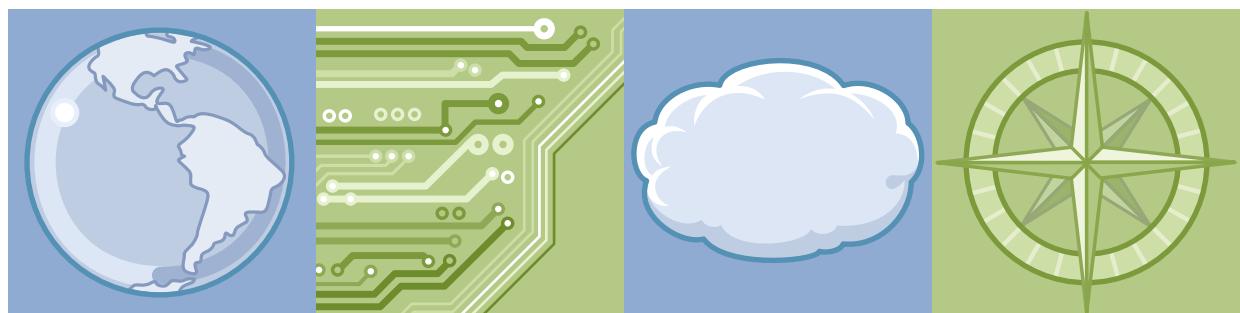


IBM Training

Student Notebook

IBM MQ V8 System Administration for z/OS

Course code WM302 ERC 1.1



WebSphere Education

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

CICS®	DB™	DB2®
developerWorks®	Express®	IMS™
MQSeries®	MVS™	RACF®
Redbooks®	RETAIN®	RMF™
VTAM®	WebSphere®	z/OS®

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

July 2016 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Contents

Trademarks	xv
.Course description.....	xvii
Agenda	xix
Unit 1. IBM MQ review	1-1
Unit objectives	1-2
Business drivers for enterprises today	1-3
Traditional connectivity and message-oriented middleware	1-5
IBM MQ and extended family	1-6
IBM MQ components and baseline	1-7
Queue manager	1-8
Queue manager terminology checkpoint	1-9
Messages	1-10
Messages: Partial MQMD and data	1-11
Messaging styles	1-12
Queues	1-13
Some local queues are designated for a special purpose (1 of 2)	1-14
Some local queues are designated for a special purpose (2 of 2)	1-15
QLOCAL partial attributes and defaults	1-16
QREMOTE complete display less date and time created fields	1-17
Terminology checkpoint	1-18
Channels	1-19
Sender-receiver channel pair	1-20
Putting it all together	1-21
Cumulative checkpoint with distributed platforms	1-22
IBM MQ distributed object relationships	1-23
Message Queue Interface (MQI)	1-24
Headers and defaults	1-25
Message expiration	1-26
Report messages	1-27
JMS support	1-28
Transactions: Terminology	1-29
Transactions: Local units of work	1-30
Transactions: Global units of work	1-31
Unit summary	1-32
Checkpoint questions (1 of 2)	1-33
Checkpoint questions (2 of 2)	1-34
Checkpoint answers (1 of 2)	1-35
Checkpoint answers (2 of 2)	1-36
Unit 2. IBM MQ architecture, installation, and configuration.....	2-1
Unit objectives	2-2
IBM MQ for z/OS V8.0	2-3
Queue manager on z/OS	2-4
Channel initiator: CHIN	2-5
z/OS IBM MQ building blocks	2-6

Page sets	2-7
Buffers and buffer pools	2-8
Queues, buffer pools and page sets: Message path view	2-9
Buffer pools above the bar	2-10
Buffer pools above the bar considerations	2-11
Buffer pool and page set usage display	2-12
Logging	2-13
Relative byte address: Height of a stack of currency	2-15
IBM MQ MSTR region partial restart messages	2-16
Object definitions: CSQ4MSTR in SCSQPROC data set	2-17
Parameters: CSQ4ZPRM in SCSQPROC data set	2-18
Channel initiator parameters to review	2-19
z/OS IBM MQ building blocks recap	2-20
IBM MQ administrative tasks	2-21
Installation and configuration	2-23
Collect and review most recent documentation	2-24
Preparing to install IBM MQ for z/OS V8	2-25
IBM MQ V8 hardware and software requirements	2-26
Other requirements that depending on implemented functions	2-27
Complete SMP/E install	2-28
Partial libraries and directories after installation	2-29
APF authorization requirements	2-30
Early code: update LPA	2-31
Update program properties table	2-32
Define the IBM MQ subsystem to z/OS	2-33
Queue manager and channel initiator configuration	2-34
CSQ4PAGE: Create the page data sets	2-35
CSQ4BSDS 1 of 2: Create the bootstrap and log data sets	2-37
CSQ4BSDS 2 of 2: Create the bootstrap and log data sets	2-39
CSQ4ZPRM: Create start parameter module	2-40
CSQ4MSTR: Create queue manager started task	2-41
CSQ4MSTR	2-42
Initialization and restart overview	2-44
CSQ4INP1 initialization commands: Define buffer pools and page sets	2-45
CSQINP2: Initialization command members	2-47
CSQ4CHIN: Create channel initiator started task	2-48
Queue manager start and verification	2-49
Set up the operations and control panels	2-50
Channel initiator start and verification	2-51
IBM MQ V8 Channel initiator error	2-52
Include the IBM MQ dump formatting member for IPCS	2-53
Batch and TSO application issues	2-54
Implementing eight-byte RBA	2-56
Implementing 8-byte RBA steps (1 of 3)	2-57
Implementing 8-byte RBA steps (2 of 3)	2-58
Implementing 8-byte RBA steps (3 of 3)	2-59
Unit summary	2-60
Checkpoint questions (1 of 2)	2-61
Checkpoint questions (2 of 2)	2-62
Checkpoint answers (1 of 2)	2-63
Checkpoint answers (2 of 2)	2-64

Exercise 1	2-65
Exercise objectives	2-66
Unit 3. IBM MQ for z/OS administrative interface options	3-1
Unit objectives	3-2
Queues revisited	3-3
QLOCAL: Local queues	3-4
Selected queue attributes and considerations	3-5
MQMD: Partial message descriptor attributes from MQPUT	3-7
Application and definition attributes: Exclusive and SHARED	3-8
QREMOTE	3-9
QALIAS	3-10
Model and dynamic queues	3-12
Queue name resolution	3-14
Triggering	3-15
Triggering: Process scenario	3-16
IBM MQ administrative tasks revisited	3-17
IBM MQ for z/OS administrative capabilities	3-18
MQSC commands: Most common components	3-19
IBM MQ z/OS ISPF panels	3-20
IBM MQ panels: Prompt for object type	3-21
Queue manager attributes with IBM MQ ISPF panels	3-22
Queue manager partial attributes using the MQSC command	3-23
Define a local queue with ISPF panels	3-24
Define a local queue with CPF and MQSC commands	3-25
Display a local queue with ISPF panels	3-26
Display a local queue with CPF and MQSC commands	3-27
Filters with ISPF (1 of 2)	3-28
Filters with ISPF (2 of 2)	3-29
Filters with MQSC	3-30
DISPLAY filtering: MQSC examples	3-31
Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (1 of 2)	3-32
Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (2 of 2)	3-33
Display QSTATUS handles with ISPF	3-34
MQSC queue status display default and handle options	3-35
ISPF displaying connections	3-36
ISPF displaying connection handles	3-37
Display page set usage with ISPF command option	3-38
Display page set usage with MQSC ISPF command option	3-39
ISPF manage a local queue and equivalent MQSC commands	3-40
IBM MQ Explorer	3-41
Programmable command formats: PCFs	3-42
CSQUTIL parameters and options	3-43
CSQUTIL command option: Defining IBM MQ objects	3-44
CSQUTIL command option: Backing up object definitions in an active queue manager ..	3-45
CSQUTIL SDEFS option: Backing up object definitions in a stopped queue manager ..	3-46
Message handler sample: ISPF option H in class system	3-47
Dead-letter queue header with a message handler sample	3-48
IBM MQ sample put program and parameters	3-49
IBM MQ sample get program and parameters	3-50
Unit summary	3-51

Checkpoint questions (1 of 2)	3-52
Checkpoint questions (2 of 2)	3-53
Checkpoint answers (1 of 2)	3-54
Checkpoint answers (2 of 2)	3-55
Exercise 2	3-56
Exercise objectives	3-57

Unit 4. Distributed queuing **4-1**

Unit objectives	4-2
Distributed queuing components	4-3
Transmission queues and queue name resolution	4-4
Transmission header	4-5
Case 1: Transmission queues and queue name resolution with queue manager name in MQOPEN	4-6
Case 2: Message transfer with a QREMOTE definition	4-7
Case 3: Multi-hopping	4-8
Queue name resolution and queue manager alias scenarios	4-9
What happens when the destination queue is not found	4-10
Channel initiators and listeners	4-11
Message channel agent	4-12
Channel categories and types	4-13
MCA actions per distributed channel pair combinations	4-14
Connecting queue managers	4-16
Definitions that are required to send messages from MQ00 to MQ0A	4-17
Automatically starting z/OS channels: Triggering	4-18
Two-way definitions MQ00 to MQ03 and MQ03 to MQ00	4-19
Callback requester sender definitions MQ0A MQ00	4-20
Channel attributes: Sender channel partial list	4-21
Data conversion: Choose one of two options	4-23
COMPMSG: Data compression	4-24
Channel exits	4-25
Before considering a channel exit	4-27
SSL support for TCP/IP channels	4-28
Administering channels: ISPF or MQSC	4-30
Channel states	4-31
Message sequence numbers and logical unit of work identifiers	4-33
Message sequence number error	4-35
Current and active channels	4-36
Current and saved status data	4-37
Message channel agent (MCA) operation and the synchronization queue	4-39
Resolve an in-doubt channel manually	4-42
Cumulative checkpoint	4-44
Definitions required to send messages from MQ00 to MQ0A	4-46
What happens when the destination queue is not found	4-47
What if the dead-letter queue is not available?	4-49
Objects are the correct type but the channel fails	4-50
Considerations when stopping a channel	4-51
Unit summary	4-52
Checkpoint questions (1 of 2)	4-53
Checkpoint questions (2 of 2)	4-54
Checkpoint answers (1 of 2)	4-55

Checkpoint answers (2 of 2)	4-56
Exercise 3	4-57
Exercise objectives	4-58
Unit 5. IBM MQ clients	5-1
Unit objectives	5-2
IBM MQ clients	5-3
IBM MQ client connectivity options and search order	5-5
MQI channel objects	5-6
MQI environment variables in IBM MQ client side	5-9
Testing client to server connectivity with MQCONNX	5-10
Connect with CCDT and put a message (1 of 2)	5-11
Connect with CCDT and put a message (2 of 2)	5-12
CCDT and queue manager groups	5-14
Use MQSERVER environment variable	5-15
IBM MQ client connection capabilities summary	5-16
IBM MQ client security considerations	5-17
WebSphere MQ Base and Extended Transactional Client	5-18
Unit summary	5-19
Checkpoint questions (1 of 2)	5-20
Checkpoint questions (2 of 2)	5-21
Checkpoint answers (1 of 2)	5-22
Checkpoint answers (2 of 2)	5-23
Exercise 4	5-24
Exercise objectives	5-25
Exercise instructions	5-26
Unit 6. IBM MQ cluster basics	6-1
Unit objectives	6-2
IBM MQ clusters	6-3
Before you start	6-5
IBM MQ cluster components	6-7
Queue manager cluster definition (1 of 2)	6-9
Queue manager cluster definition (2 of 2)	6-10
MQ0A definition check 1 of 2: What happens in the cluster	6-11
MQ0A definition check 2 of 2: What happens in the cluster	6-13
MQ00 definition check 1 of 2: What happens in the cluster	6-14
MQ00 definition check 2 of 2: What happens in the cluster	6-15
MQ03 definition check: What happens in the cluster	6-16
MQ03 definition check: Log messages	6-17
Cluster queues	6-19
Cluster definition review	6-20
Cumulative checkpoint	6-21
IBM MQ cluster commands	6-23
The case against use of REFRESH CLUSTER: System console after cluster correction	6-24
DIS QCLUSTER(CASH*) DEFBIND CLUSQMGR	6-25
Cluster administrative options	6-26
Workload balancing	6-27
Topology with a cluster gateway	6-30
Definitions to MQ00 gateway from an external queue manager	6-31
Cluster transmission queue options	6-32

Highly available WebSphere MQ topologies	6-33
Best practices	6-34
Unit summary	6-35
Checkpoint questions (1 of 2)	6-36
Checkpoint questions (2 of 2)	6-37
Checkpoint answers (1 of 2)	6-38
Checkpoint answers (2 of 2)	6-39
Exercise 5	6-40
Exercise objectives	6-41
Unit 7. Publish/subscribe basics	7-1
Unit objectives	7-2
Point-to-point and publish/subscribe	7-3
Distributed publish/subscribe patterns	7-4
Publish/subscribe basic components	7-5
Current publish/subscribe functionality: Some history	7-6
Publish/subscribe terminology baseline (1 of 2)	7-8
Publish/subscribe terminology baseline (2 of 2)	7-9
Publish/subscribe functionality	7-10
Topic tree, topic strings, topic nodes, and topic objects	7-12
Topic objects	7-13
Topic object attributes: DISPLAY TOPIC view	7-14
Topic object attributes: DISPLAY TPSTATUS view	7-15
Topic alias	7-16
Topic administration options	7-17
Topic status with IBM MQ Explorer	7-18
Topic administration: Topic scavenger	7-19
Subscriptions: The three aspects (1 of 2)	7-20
Subscriptions: The three aspects (2 of 2)	7-21
Subscription commands: DIS SUB(VEGET*) ALL	7-22
Subscription commands: DISPLAY PUBSUB ALL	7-24
Subscription commands: DISPLAY SUB(*) TOPICSTR DEST	7-25
Subscription related command: DISPLAY QLOCAL	7-26
Subscription commands: SBSTATUS for an active subscription	7-27
Subscription related commands: DIS CONN	7-28
Publications	7-29
Retained publications	7-31
Publish/subscribe testing in IBM MQ Explorer	7-32
Publish/subscribe lifecycle descriptions	7-33
Publish/subscribe lifecycles: Managed nondurable subscriber	7-34
Distributed publish/subscribe	7-36
Publish/subscribe implicit structure: Proxy subscriptions	7-37
Publish/subscribe hierarchies	7-38
Hierarchy configuration view	7-39
Topic attributes that influence all distributed publish/subscribe configurations	7-40
Publish/subscribe clusters: Direct and topic host routing	7-41
Clustered publish/subscribe proxy subscriptions	7-42
PSCLUS queue manager attribute	7-43
Publish/subscribe cluster administration: Direct routing	7-44
Clustered publish/subscribe views	7-45
DISPLAY TCLUSTER view	7-46

Changing cluster topic from direct to topic host routing	7-47
Distributed publish/subscribe topologies considerations	7-48
Highly available subscriptions: A different approach	7-49
Before you start implementing publish/subscribe (1 of 2)	7-50
Before you start implementing publish/subscribe (2 of 2)	7-51
Unit summary	7-52
Checkpoint questions (1 of 2)	7-53
Checkpoint questions (2 of 2)	7-54
Checkpoint answers (1 of 2)	7-55
Checkpoint answers (2 of 2)	7-56
Exercise 6	7-57
Exercise objectives	7-58
Unit 8. Queue sharing groups	8-1
Unit objectives	8-2
What are shared queues?	8-3
Differences between nonshared and shared local queues	8-4
Queue-sharing group components	8-5
Queue-sharing groups and CF list structures	8-7
Coupling facility	8-8
Large message support and CF structure use history	8-9
Coupling facility capabilities and the CFLEVEL attribute	8-10
The IBM MQ CFSTRUCT object	8-11
Enabling queue-sharing groups checklist (1 of 2)	8-12
Enabling queue-sharing groups checklist (2 of 2)	8-13
SMDS offload	8-14
CF structure extension with SCM	8-15
What is the best option to mitigate CF space constraints?	8-16
MQSC commands to manage shared queue environment	8-17
DQM support for shared queues: Inbound	8-18
DQM support for shared queues: Outbound	8-20
Intragroup queuing	8-21
How do shared queues compare with clusters?	8-23
Incorporating a queue-sharing group into an existing cluster	8-24
Unit summary	8-25
Checkpoint questions (1 of 2)	8-26
Checkpoint questions (2 of 2)	8-27
Checkpoint answers (1 of 2)	8-28
Checkpoint answers (2 of 2)	8-29
Unit 9. Using IBM MQ events and the dead-letter queue utility	9-1
Unit objectives	9-2
Approaches to monitoring IBM WebSphere MQ	9-3
Event queues: Overview	9-4
Queue manager event	9-5
Performance events in ISPF panels	9-6
Depth event sample	9-7
Service interval events	9-8
Command events	9-9
Configuration events	9-10
Channel events	9-11

Activity reporting and route tracking	9-12
When messages go to the dead-letter queue	9-13
Dead-letter queue handler utility	9-14
Dead-letter queue handler patterns and actions	9-15
Unit summary	9-17
Checkpoint questions (1 of 2)	9-18
Checkpoint questions (2 of 2)	9-19
Checkpoint answers (1 of 2)	9-20
Checkpoint answers (2 of 2)	9-21
Exercise 7	9-22
Exercise objectives	9-23

Unit 10. Security considerations..... 10-1

Unit objectives	10-2
Terminology baseline: Security areas	10-3
Access control mechanisms	10-4
Security considerations for IBM MQ	10-5
Link-level and application-level security	10-6
Access required for IBM MQ z/OS libraries	10-7
External Security Manager: RACF terminology	10-8
Authorization: IBM MQ for z/OS resource classes	10-9
Authorization: MQADMIN class switch profiles	10-11
Authorization: Connection security profiles	10-12
Enabling IBM MQ for z/OS security	10-13
Authorization: RESLEVEL profile	10-14
Authorization: Access levels for resource checks	10-15
Resolving queue names on MQOPEN	10-16
Authorization: Alias queue considerations	10-18
Authorization: Model queues and dynamic queues	10-19
Authorization: Controlling access to remote queues	10-20
Overview of context data	10-21
How context data is used	10-22
Authorization: Context options and permissions required	10-23
Authorization: Use of alternate user security	10-24
Authorization: RACF access for an alternate user ID	10-25
Authorization: Checking IBM MQ commands	10-26
Authority to issue commands	10-27
Access to command resources	10-28
Queue-sharing group level security	10-29
Determine the level of security	10-30
Securing transmission queues	10-31
Securing the remote queue access	10-32
User IDs to be checked	10-34
Security controls	10-36
Authentication: Security exits	10-37
Security inquiry	10-38
Link-level security: Connection authentication	10-39
CONNAUTH settings: Defaults	10-41
CHCKCLNT and CHCKLOCL values (1 of 3)	10-42
CHCKCLNT and CHCKLOCL values (2 of 3)	10-43
CHCKCLNT and CHCKLOCL values (3 of 3)	10-44

Client-side security exit: Distributed client side	10-45
Link-level security: Identification and authentication	10-46
Channel blocking points	10-47
Replacing and removing CHLAUTH rules	10-48
CHLAUTH types	10-49
CHLAUTH rules using IP addresses and host names	10-50
CHLAUTH rule precedence	10-51
CHLAUTH initial settings	10-52
Getting started with CHLAUTH (1 of 5)	10-53
Getting started with CHLAUTH (2 of 5)	10-54
Getting started with CHLAUTH (3 of 5)	10-55
Getting started with CHLAUTH (4 of 5)	10-56
Getting started with CHLAUTH (5 of 5)	10-57
Use of MQ Explorer for CHLAUTH (1 of 2)	10-58
Use of MQ Explorer for CHLAUTH (2 of 2)	10-59
SSL/TLS: Confidentiality, data integrity, and authentication	10-60
Dead-letter queue considerations	10-61
Confidentiality and data integrity: Application-level security	10-62
Auditing	10-63
IBM MQ security summary	10-64
Unit summary	10-65
Checkpoint questions (1 of 2)	10-66
Checkpoint questions (2 of 2)	10-67
Checkpoint answers (1 of 2)	10-68
Checkpoint answers (2 of 2)	10-69
Exercise 8	10-70
Exercise objectives	10-71
Unit 11. Problem determination	11-1
Unit objectives	11-2
Initial problem identification	11-3
Identifying programming errors	11-4
IBM MQ for z/OS diagnostics	11-5
Investigating missing or unexpected output	11-7
Activity reports and route tracking	11-9
Scheduled abend codes	11-10
Formatting system dumps using IPCS	11-12
Analyzing dump data using IPCS	11-13
Displaying error diagnostics	11-14
Formatting and displaying entire dump	11-15
Using the IBM WebSphere MQ for z/OS trace	11-16
Channel problems	11-18
DQM queue and trigger problems	11-19
In-doubt resolution with DQM: ISPF and MQSC	11-21
Overall performance considerations	11-22
Queue manager statistics and accounting data	11-23
Channel initiator statistics and accounting data	11-24
Unit summary	11-25
Checkpoint questions (1 of 2)	11-26
Checkpoint questions (2 of 2)	11-27
Checkpoint answers (1 of 2)	11-28

Checkpoint answers (2 of 2)	11-29
Unit 12. IBM MQ Managed File Transfer	12-1
Unit objectives	12-2
Introducing IBM MQ Managed File Transfer	12-3
Main components and product options	12-4
Sample WebSphere MQ managed file transfer architecture	12-5
Protocol bridge	12-7
Required IBM MQ connectivity paths (1 of 2)	12-8
Required IBM MQ connectivity paths (2 of 2)	12-9
Steps to configure an agent on z/OS	12-10
Copy the IBM MQFT libraries	12-11
Partial list of jobs that are generated by BFGCUSTM	12-12
BFGCUSTM (1 of 2)	12-13
BFGCUSTM (2 of 2)	12-14
Partial output of BFGCUSTM job	12-15
Set up coordination: BFGCFCR	12-16
Set up coordination results	12-18
Set up commands: BFGCMCR	12-20
Set up commands results	12-21
Create agent: BFGAGCR	12-22
Create agent results	12-23
Start agent: BFGAGST	12-24
Display agent and status fteListAgent: BFGAGLI	12-25
Agent logs (1 of 2)	12-26
Agent logs (2 of 2)	12-27
Create a transfer using fteCreateTransfer: BFGTRCRS	12-28
Adding an existing configuration to IBM MQ Explorer (1 of 2)	12-29
Adding an existing configuration to IBM MQ Explorer (2 of 2)	12-30
Displaying the results of the configuration	12-31
Create a transfer with IBM MQ Explorer (1 of 2)	12-33
Create a transfer with IBM MQ Explorer (2 of 2)	12-34
Checking results (1 of 2)	12-35
Checking results (2 of 2)	12-36
Starting a transfer by placing a specially formatted message in the agents command queue	12-37
IBM MQ Managed File Transfer use of publish/subscribe	12-38
Auditing loggers	12-39
Locating FFDCs	12-40
BFGAGSH fteShowAgentDetails output (1 of 2)	12-41
BFGAGSH fteShowAgentDetails output (2 of 2)	12-42
Unit summary	12-43
Checkpoint questions (1 of 2)	12-44
Checkpoint questions (2 of 2)	12-45
Checkpoint answers (1 of 2)	12-46
Checkpoint answers (2 of 2)	12-47
Exercise 9	12-48
Exercise objectives	12-49
IBM MQ Managed File Transfer exercise reminders	12-50

Unit 13. IBM MQ for z/OS backup, recovery, and related file tasks	13-1
Unit objectives	13-2
Unit of recovery (UR)	13-3
Interaction of buffers, logs, and page sets	13-5
Logging overview	13-7
Active and archive logs	13-8
Consistency with other resource managers	13-9
Two-phase commit and in-doubt unit of recovery	13-10
Log shunt	13-12
IBM MQ use of the bootstrap data set	13-13
Contents of the BSDS (1 of 2)	13-14
Contents of the BSDS (2 of 2)	13-15
Log control ZPARM macro CSQ6LOGP	13-16
Dynamic addition of active log data sets	13-18
Archive control ZPARM macro CSQ6ARVP	13-19
ARCHIVE LOG command	13-20
SUSPEND and RESUME QMGR LOG commands	13-21
List and change the BSDS contents	13-22
MQ for z/OS termination	13-23
Restart activities	13-25
System restart messages	13-27
Caring for in-doubt threads (CICS sample)	13-28
Manually resolving IMS units of recovery	13-30
Log print utility CSQ1LOGP	13-32
Principle of media recovery	13-34
PSID recovery is part of MQ restart	13-36
Taking page set backups	13-37
Useful commands for PSID backup handling	13-38
Display usage for PAGESET command output	13-40
Display usage for DATASET command output	13-41
BSDS recovery considerations	13-42
When logs are lost or damaged	13-44
CSQUTIL RESETPAGE command	13-45
CSQJU003 CRESTART command	13-46
Queue manager cold start	13-47
Dynamic page set handling	13-48
CSQUTIL COPYPAGE command	13-50
CSQUTIL queue and page set management	13-51
Implementing above the bar buffer pools: BUFFPOOL	13-52
Unit summary	13-53
Checkpoint questions (1 of 2)	13-54
Checkpoint questions (2 of 2)	13-55
Checkpoint answers (1 of 2)	13-56
Checkpoint answers (2 of 2)	13-57
Exercise 10	13-58
Exercise objectives	13-59
Unit 14. Support for CICS, IMS, and HTTP applications	14-1
Unit objectives	14-2
CICS WebSphere MQ adapter	14-3
Starting and stopping the adapter	14-5

CICS IBM MQ adapter CICS resources	14-7
CICS IBM MQ bridges	14-8
IBM MQ CICS bridge	14-9
CICS bridge request message (MQCIH) contents	14-11
Preparing CICS IBM MQ applications	14-12
IBM MQ IMS OTMA bridge	14-13
IBM MQ IMS OTMA bridge implementation checklist	14-14
Define bridge to IMS	14-15
Define bridge to IBM MQ	14-16
IMS /DIS OTMA command	14-17
Coding messages short format: No IIH header	14-18
Coding bridge messages using IIH header (1 of 2)	14-19
Coding bridge messages using IIH header (2 of 2)	14-20
Other bridge considerations	14-21
Transaction pipes: Tpipes	14-22
Attribute summary	14-23
Tpipes	14-24
IMS exits	14-25
OTMA exits	14-26
IBM MQ IMS OTMA bridge security	14-27
IBM MQ and IMS transaction expiration support	14-28
Common bridge problems	14-30
Comparison: IBM MQ-IMS adapter to IBM MQ IMS OTMA bridge	14-31
Setting up the IMS adapter	14-32
MQ-IMS adapter	14-33
Defining IBM MQ to IMS	14-34
Defining LITs to the adapter	14-35
IBM MQ for z/OS IMS program stub	14-36
Operating the IMS adapter	14-37
IBM MQ supplied IMS trigger monitor	14-38
IBM MQ bridge for HTTP	14-39
IBM MQ bridge for HTTP components	14-40
Unit summary	14-41
Checkpoint questions	14-42
Checkpoint answers	14-43
Unit 15. Course summary	15-1
Unit objectives	15-2
Course learning objectives (1 of 2)	15-3
Course learning objectives (2 of 2)	15-4
To learn more on the subject (1 of 2)	15-5
To learn more on the subject (2 of 2)	15-6
Unit summary	15-7
Appendix A. List of abbreviations.....	A-1
Appendix B. Resource guide.....	B-1

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

CICS®	DB™	DB2®
developerWorks®	Express®	IMS™
MQSeries®	MVS™	RACF®
Redbooks®	RETAIN®	RMF™
VTAM®	WebSphere®	z/OS®

Adobe is either a registered trademark or a trademark of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware and the VMware “boxes” logo and design, Virtual SMP and VMotion are registered trademarks or trademarks (the “Marks”) of VMware, Inc. in the United States and/or other jurisdictions.

Other product and service names might be trademarks of IBM or other companies.

Course description

IBM MQ V8 System Administration for z/OS

Duration: 4 days

Purpose

This course provides the skills that are necessary to configure and manage an IBM MQ V8 queue manager on z/OS.

Audience

This course is designed for technical support personnel who implement, operate, and perform day-to-day administration of IBM MQ V8 on z/OS.

Prerequisites

Before taking this course, you should have:

- Basic knowledge of IBM MQ V8 concepts, obtained through experience or by successfully completing course WM102, *Technical Introduction to IBM MQ*.
- Working proficiency with the z/OS platform, obtained through experience or by successfully completing course ES10G, *Fundamental System Skills in z/OS*.

Knowledge of TCP/IP is also helpful.

Objectives

After completing this course, you should be able to:

- Describe message-oriented middleware and the capabilities it must provide
- Identify the key components of IBM MQ for z/OS
- Summarize the responsibilities of the IBM MQ administrator
- Configure IBM MQ V8 for z/OS
- Enable IBM MQ for z/OS eight-byte RBA and buffers above 2 GB
- Demonstrate how to create and change queues and place and retrieve messages from a queue
- Define and demonstrate how to set up and work with distributed queuing
- Differentiate between an IBM MQ queue manager and an IBM MQ client
- Describe and demonstrate how to set up an IBM MQ cluster

- Contrast point-to-point and publish/subscribe messaging styles
- Describe shared queues and queue sharing groups
- Summarize IBM MQ for z/OS recovery and restart activities
- Demonstrate how to use IBM MQ events for monitoring
- Summarize performance considerations
- Describe security considerations for IBM MQ for z/OS
- Describe and implement connection authentication and channel authentication
- Identify correct problem determination techniques for IBM MQ for z/OS
- Summarize basic use and configuration of IBM MQ Managed File Transfer
- Describe IBM MQ support for CICS and IMS interfaces

Curriculum relationship

- The prerequisite for this course is WM102, Technical Introduction to IBM MQ V8.
- This course is a prerequisite for WM312, IBM MQ V8 Advanced System Administration for z/OS

Agenda

Day 1

- Course introduction
- Unit 1. IBM MQ review
- Unit 2. IBM MQ architecture, installation, and configuration
- Exercise 1. Configuring an IBM MQ for z/OS queue manager
- Unit 3. IBM MQ for z/OS administrative interface options
- Exercise 2. Working with queues

Day 2

- Unit 4. Distributed queuing
- Exercise 3. Working with channels
- Unit 5. IBM MQ clients
- Exercise 4. Working with IBM MQ clients
- Unit 6. IBM MQ cluster basics
- Exercise 5. Working with IBM MQ clusters

Day 3

- Unit 7. Publish/subscribe basics
- Exercise 6. Publish/subscribe basics
- Unit 8. Queue sharing groups
- Unit 9. Using IBM MQ events and the dead-letter queue utility
- Exercise 7. Working with IBM MQ events
- Unit 10. Security considerations
- Exercise 8. Security

Day 4

- Unit 11. Problem determination
- Unit 12. IBM MQ Managed File Transfer
- Exercise 9. IBM MQ Managed File Transfer configuration for z/OS
- Unit 13. IBM MQ for z/OS backup, recovery, and related file tasks
- Exercise 10. Working with file handling utilities
- Unit 14. Support for CICS, IMS, and HTTP applications
- Unit 15. Course summary

Unit 1. IBM MQ review

What this unit is about

This unit reviews basic IBM MQ concepts.

What you should be able to do

After completing this unit, you should be able to:

- Summarize how message-oriented middleware is critical to an enterprise
- Describe the role of a queue manager
- Describe messages and their contents
- Describe a queue
- Distinguish between queues that hold messages and queues that do not hold messages
- Explain some of the queues that are used for special purposes
- Describe the role of a channel
- Describe the calls that help place and retrieve messages from queues
- Identify key queue and message attributes and how to set them
- Summarize JMS support
- Describe transactions

Unit objectives

- Summarize how message-oriented middleware is critical to an enterprise
- Describe the role of a queue manager
- Describe messages and their contents
- Describe a queue
- Distinguish between queues that hold messages and queues that do not hold messages
- Explain some of the queues that are used for special purposes
- Describe the role of a channel
- Describe the calls that help place and retrieve messages from queues
- Identify key queue and message attributes and how to set them
- Summarize JMS support
- Describe transactions

© Copyright IBM Corporation 2015

Figure 1-1. Unit objectives

WM3021.1

Notes:

Business drivers for enterprises today

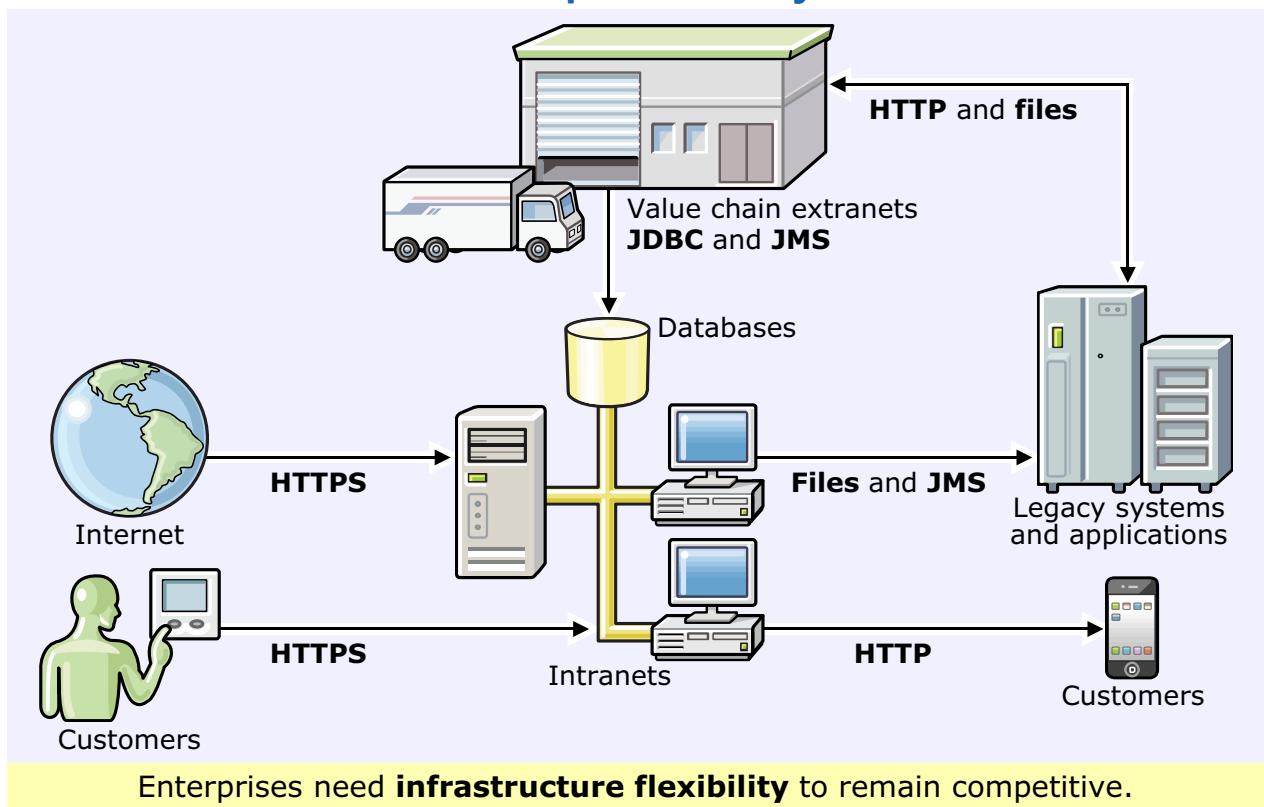


Figure 1-2. Business drivers for enterprises today

WM3021.1

Notes:

Business requirements are changed in response to “the Internet of Things.” What is meant by “the Internet of Things”?

Where in the past there were a few connections across users and applications, today there are many. Connectivity requests are increasing exponentially.

The amount of data that is exchanged is also increasing and organizations must be ready to accept high amounts of information from Business Partners.

Old “batch” processes, if not disappearing, are diminishing, resulting in fewer change windows for the enterprise.

Applications must be resilient and capable of handling failure, yet current users keep expecting the same quick response, if not better, than when they used to have a direct connection to the data.

Users also need to get information now rather than wait for a batch process.

Now also means accessible by a mobile device on demand.

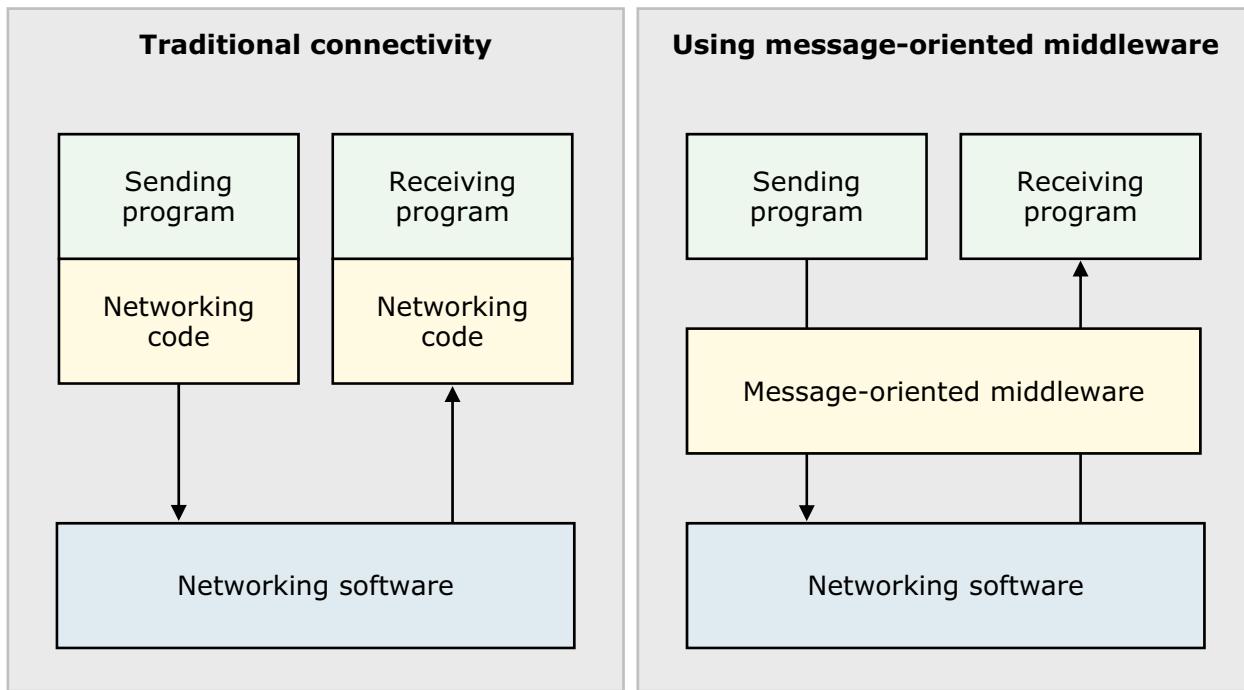
Extending existing processes would get prohibitively expensive; applications need to become flexible so that they can be used as needed.

What does flexible mean?

Data must be available on demand, in different ways such as in mobile applications.

Customers and Business Partners have higher expectations, and the inability to meet these expectations translates to loss of business.

Traditional connectivity and message-oriented middleware



© Copyright IBM Corporation 2015

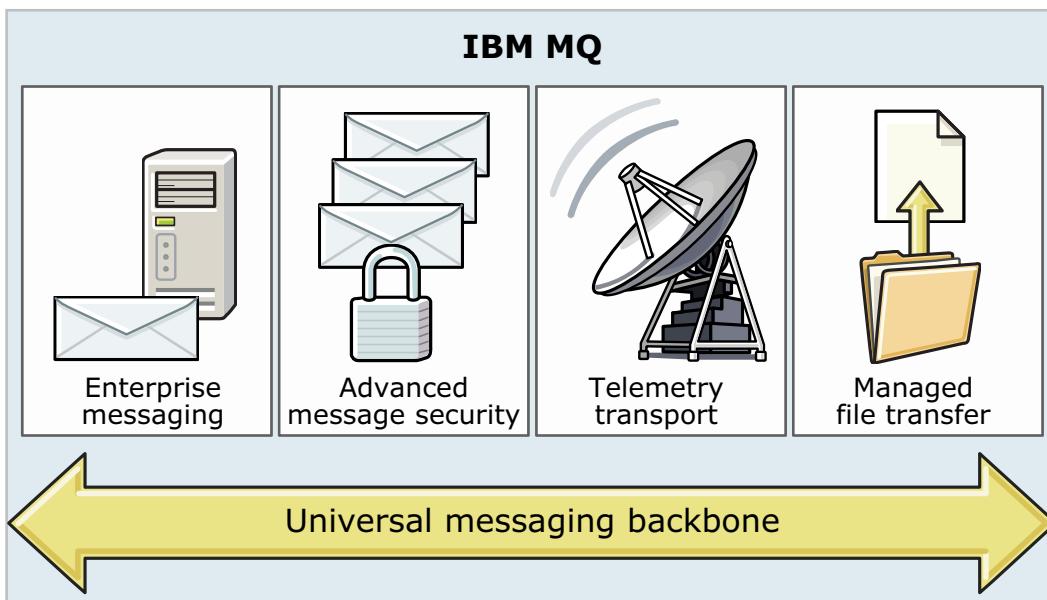
Figure 1-3. Traditional connectivity and message-oriented middleware

WM3021.1

Notes:

When message-oriented middleware is used, all the work of connecting, polling, handling failure, and implementing change is removed from the application and handed off to the messaging middleware, providing flexibility to react to business conditions.

IBM MQ and extended family



© Copyright IBM Corporation 2015

Figure 1-4. IBM MQ and extended family

WM3021.1

Notes:

IBM MQ offers enterprise class messaging-oriented middleware. In this course, you also look at implementation basics for IBM Managed File Transfer, along with options for JMS and HTTP interfaces.

Telemetry transport is another key area that IBM MQ supports; in particular, mobile applications, although they are not covered in this class.

IBM MQ components and baseline

- The first part of this unit focuses on the basic IBM MQ components
 - Queue manager
 - Messages
 - Queues
 - Channels
 - Message queue interface (MQI)
- IBM MQ can be installed with IBM MQ server software, with IBM MQ client software, or with both
 - IBM MQ server and IBM MQ client are two distinct installed products
 - Unless a topic is identified as applying to IBM MQ client, the information pertains to installations of IBM MQ server software

© Copyright IBM Corporation 2015

Figure 1-5. IBM MQ components and baseline

WM3021.1

Notes:

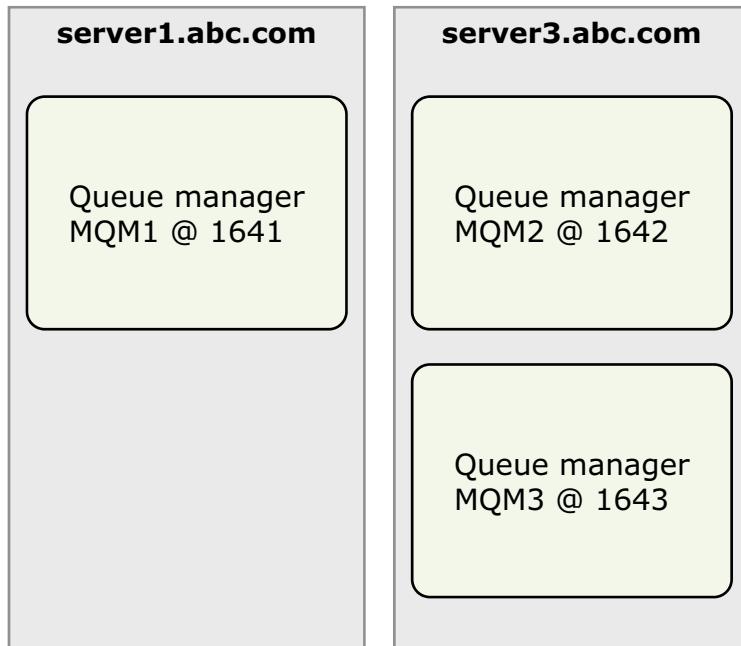
This unit discusses the basic IBM MQ concepts:

- The queue manager, or messaging server, owns IBM MQ resources and controls processes such as defining objects, trigger starting a channel or process, creating event messages, expiring messages, and message distribution by using clustering.
- Messages are the unit of data that the queue manager exchanges.
- Queues are where messages are kept.
- Channels move messages from one queue manager to another queue manager, sometimes jumping across other queue managers.
- The Message Queue Interface, or MQI, is the application interface that is used to put and get messages in IBM MQ.

Do not confuse the MQI with IBM MQ client channels, which are also referred to as MQI channels. In this unit, you refer to IBM MQ servers only. IBM MQ clients are discussed in a separate unit.

Queue manager

- System program that owns the resources it services
- Provides services that are needed for messaging
- A server can host more than one queue manager
- Queue managers that share a server need different TCP/IP port numbers



© Copyright IBM Corporation 2015

Figure 1-6. Queue manager

WM3021.1

Notes:

In this class, you configure an IBM MQ for z/OS queue manager.

There can be more than one queue manager in the same z/OS or distributed server. Each queue manager must listen on a different port.

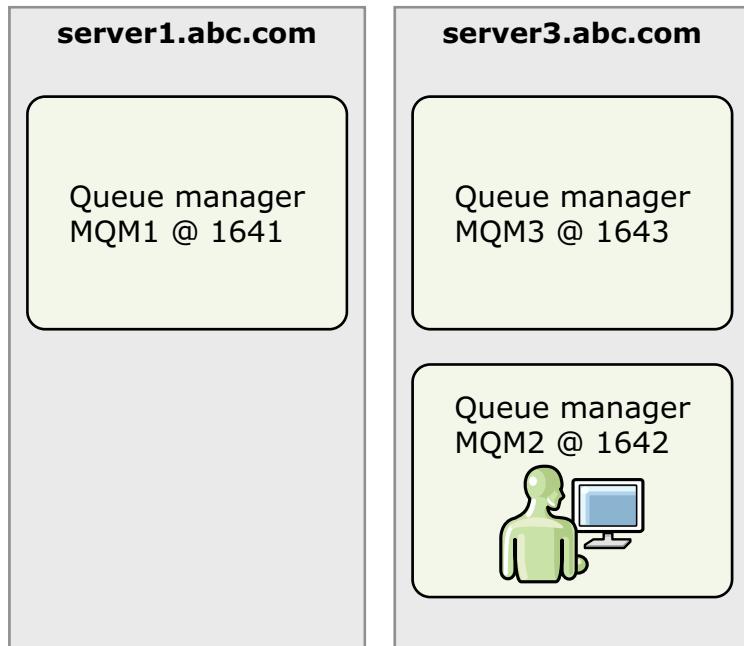
You create queue managers in the same server, so when the channel initiator is configured, each of your queue managers has a specific assigned port number. In the next unit, you look at what components compose an IBM MQ z/OS queue manager.

Queue manager terminology checkpoint

- **Local queue manager** is the currently referenced queue manager
- All other queue managers are referred to as **remote queue managers**

Example:

- If working with MQM2, then MQM2 is the local queue manager, and MQM1 and MQM3 are remote queue managers
- Objects and resources that are defined in MQM2 are said to be “locally owned”



© Copyright IBM Corporation 2015

Figure 1-7. Queue manager terminology checkpoint

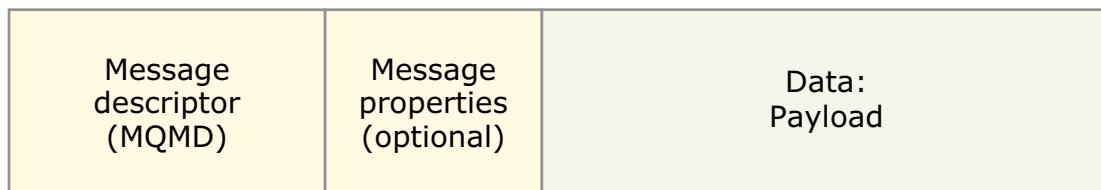
WM3021.1

Notes:

When you work with IBM MQ, you refer to the queue manager you are working with at the time as the “local” queue manager, and all other queue managers as “remote” queue managers.

Messages

- Messages contain the business data or payload
- Messages are a distinct string of bytes exchanged between two programs
- All messages contain an IBM MQ message descriptor (MQMD) with control information
- Optionally, applications can add message properties to a message



© Copyright IBM Corporation 2015

Figure 1-8. Messages

WM3021.1

Notes:

Messages contain the message descriptor, or MQMD structure, and the message data or payload. Messages might contain other headers.



Messages: Partial MQMD and data

- IBM MQ structures and objects are set with numerous defaulted fields or attributes
 - It is critical to recognize when a defaulted attribute should be changed

```
StrucId : 'MD' Version : 2
Report : 0 MsgType : 8
Expiry : -1 Feedback : 0 ...
...
Priority : 0 Persistence : 0
MsgId : X'414D5120574D313032202020202020A811235320002202'
CorrelId : X'000000000000000000000000000000000000000000000000000000000000000'
BackoutCount : 0
ReplyToQ : ''
ReplyToQMgr : 'MQM03
** Identity Context
UserIdentifier : 'ibmusr9
...
*****
Message *****
length - 9 of 9 bytes
00000000: 6161 6162 6262 6363 63                                'aaabbbccc

```

© Copyright IBM Corporation 2015

Figure 1-9. Messages: Partial MQMD and data

WM3021.1

Notes:

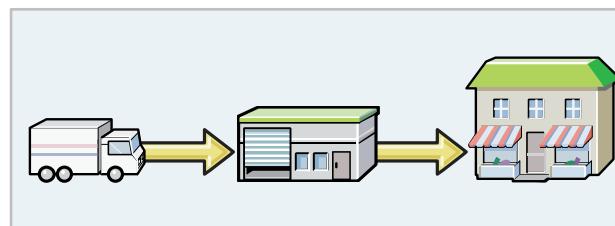
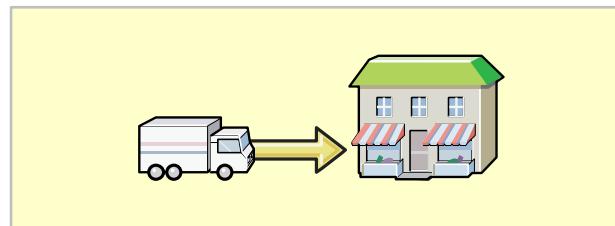
The IBM MQ message descriptor, or MQMD, contains several fields. These fields have default values. The slide shows a partial view of an MQMD taken from a distributed IBM MQ server.

Notice fields such as expiry and persistence. Many fields in the MQMD and other headers can override attributes that are defined in queues. You look at these attributes in more detail in a later unit.

Messaging styles

- Point-to-point application
 - Sends messages to a predefined destination
 - The application does not need to know where the destination is because IBM MQ locates the target by using object definitions

- Publish/subscribe application
 - Publishes messages to an interim destination according to a topic
 - Interested recipients subscribe to the topic
 - No explicit connection between publishing and subscribing applications



Unless publish/subscribe is specifically mentioned, subsequent topics apply to the point-to-point messaging style.

© Copyright IBM Corporation 2015

Figure 1-10. Messaging styles

WM3021.1

Notes:

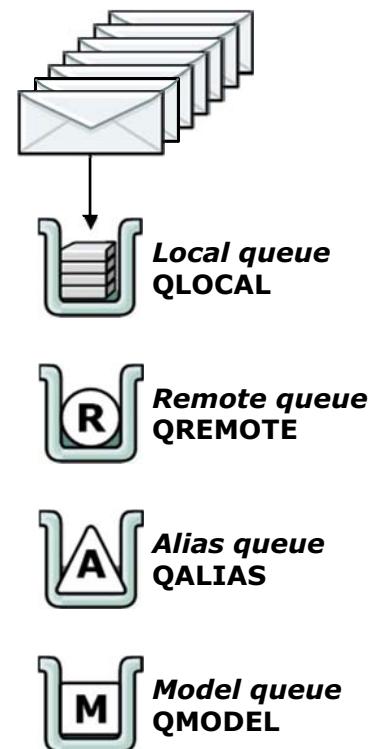
There are two messaging styles, point-to-point and publish/subscribe.

With point-to-point messaging, you know where the message is going. The publish/subscribe messaging style is further decoupled because the producers of messages send or publish messages to an interim location. Here the consumers or subscribers obtain, or subscribe to, a topic or topic string that is linked to the messages.

You look at publish/subscribe in a later unit. The topics on the rest of this unit apply to point-to-point messaging.

Queues

- A queue is a defined destination for messages
- Four types of queues can be created: QLOCAL, QREMOTE, QALIAS, QMODEL
- Some local queues are designated for special purposes in a queue manager
- Remote queues or QREMOTE are pointers to local queues in a remote queue manager
- An alias queue or QALIAS is a pointer to a local queue or a locally owned remote queue
- A model queue or QMODEL is a template to create a dynamic local queue



Only local queues that are defined as a QLOCAL queue type can hold messages.

Figure 1-11. Queues

© Copyright IBM Corporation 2015

WM3021.1

Notes:

Queues are where applications put and send messages.

When you start working with IBM MQ, you might spend some time with developers or testers to determine “where is my message?”

A basic concept is that only local queues that are defined as QLOCAL can hold messages. Keeping this concept in mind is helpful when looking for a message.

Other queue types are pointers to other queues:

- Remote queues point to a transmit queue, and a remote queue manager and target queue. QREMOTE is the queue type only. These queues are “local QREMOTES”.
- Alias queues, or QALIAS, point to another local queue or a topic. You revisit these queues in a later unit.

Model queues, or QMODEL, can be thought of as a template to create queues, but also do not hold messages.

You also hear about shared queues. These queues belong to a queue-sharing group and might also be QLOCALs. A unit of this class is dedicated to shared queues.

Some local queues are designated for a special purpose (1 of 2)

- Transmit or transmission queue 
 - Local queue with its usage attribute set to XMITQ in the queue definition
 - If an application puts a message to a remote queue, it goes to a transmit queue
 - Channels use transmit queues to send messages to the remote queue manager
 - Transmit queues have a special MQXQH header
 - Applications should not write to a transmission queue, only the queue manager

- Initiation queues 
 - Identified as an initiation queue in a definition of another local queue
 - Associated with triggering

© Copyright IBM Corporation 2015

Figure 1-12. Some local queues are designated for a special purpose (1 of 2)

WM3021.1

Notes:

The basic queue types were discussed in the last slide. Several other queues are named after their purpose, but these queues are also local queues, or QLOCALs.

Transmission queues and initiation queues are QLOCALs.

Some local queues are designated for a special purpose (2 of 2)

- Dead-letter queue 
- Local queue that is identified to the queue manager as its queue to hold undeliverable messages
- Usually the SYSTEM.DEAD.LETTER queue is designated as dead-letter queue; however, a different local queue can be defined such as BILLING.DLQ
- The dead-letter queue has an MQDLH header that contains the reason that a message was placed in the dead-letter queue and other pertinent information

- Queues starting with “SYSTEM” 
- Most of the SYSTEM.* queues are local queues that are used by the queue manager
- SYSTEM.* queues can be browsed, but unless otherwise documented, should not be used to put messages

© Copyright IBM Corporation 2015

Figure 1-13. Some local queues are designated for a special purpose (2 of 2)

WM3021.1

Notes:

You have the dead-letter queue, which can be assigned a different name; it is also a QLOCAL.

There is a series of queues with the prefixed SYSTEM.*, unless the SYSTEM.* queue is a default template for a remote or alias queue, and these SYSTEM.* queues are also special-purpose QLOCALs.

QLOCAL partial attributes and defaults

- When defining a QLOCAL, the only required parameter is the queue name
- When an attribute is not specified in the definition, it is given a default value
- Some of the attributes of a local queue definition get populated by the queue manager:
 - QUEUE shows queue name
 - Create date (CRDATE)
 - Count of existing messages in the queue, or current depth (CURDEPTH)
- The persistence attribute (DEFPSIST) defaults to nonpersistent
 - The persistence attribute determines whether a message survives restarts of the queue
 - Standards on how and when to use persistence should be established
- A transmission queue is a QLOCAL with its USAGE attribute set to XMITQ

```

QUEUE (BILL_IN)
TYPE (QLOCAL)
CRDATE (2014-02-26)
CURDEPTH (1)
DEFBIND (OPEN)
DEFPSIST (NO)
INITQ ( )
NOTRIGGER
PROCESS ( )
QDEPTHHI (80)
TRIGDATA ( )
TRIGDPTH (1)
TRIGTYPE (FIRST)
USAGE (NORMAL)

```

© Copyright IBM Corporation 2015

Figure 1-14. QLOCAL partial attributes and defaults

WM3021.1

Notes:

Defining a queue local is simple. To define a queue local without any special requirements, you issue a DEFINE QLOCAL followed by the queue name, and the queue is created; but many attribute values are defaulted. As you do more work with IBM MQ, you learn the rest of the attributes and their importance.



Note

The queue manager sets some of the queue attributes; for instance, CURDEPTH, which displays how many messages are in the queue, or CRDATE, which displays the date that the queue was created. An application might specify attributes that can supersede most of the attributes that are specified or defaulted in a queue definition. If it is imperative to use a specific attribute, it is a good practice to have the application set it rather than default to the attribute in the queue definition.

A message's persistence determines whether the message survives restarts of the queue manager. If the message is critical to the application, for instance, is not a simple query, it should be made persistent.

QREMOTE complete display less date and time created fields

- A message count (CURDEPTH) attribute is not present in a QREMOTE
- The remote queue points to the name of the transmission queue (XMITQ) where messages put to this queue are placed
- The remote queue also points to the destination queue, or name of the local queue in the remote queue manager (RNAME) where messages are transmitted
- The remote queue manager name attribute (RQMNAME) of the remote queue definition contains the name of the queue manager where messages are transmitted

```

QUEUE (REQ.OUT)
TYPE (QREMOTE)
CLUSNL( )
CLUSTER( )
CLWLPRTY(0)
CLWLRank(0)
CUSTOM( )
DEFBIND (OPEN)
DEFPRTY (0)
DEFPSIST (NO)
DEFPRESp (SYNC)
DESCR( )
PUT (ENABLED)
RQMNAME (QM3)
RNAME (SALE . IN)
SCOPE (QMGR)
XMITQ (QM3)

```

© Copyright IBM Corporation 2015

Figure 1-15. QREMOTE complete display less date and time created fields

WM3021.1

Notes:

As you see in the *queue name resolution* topic, it is not imperative to have a QREMOTE definition to send a message to another queue manager. However, if you use a QREMOTE, it does make sending the messages less code dependent and easier.

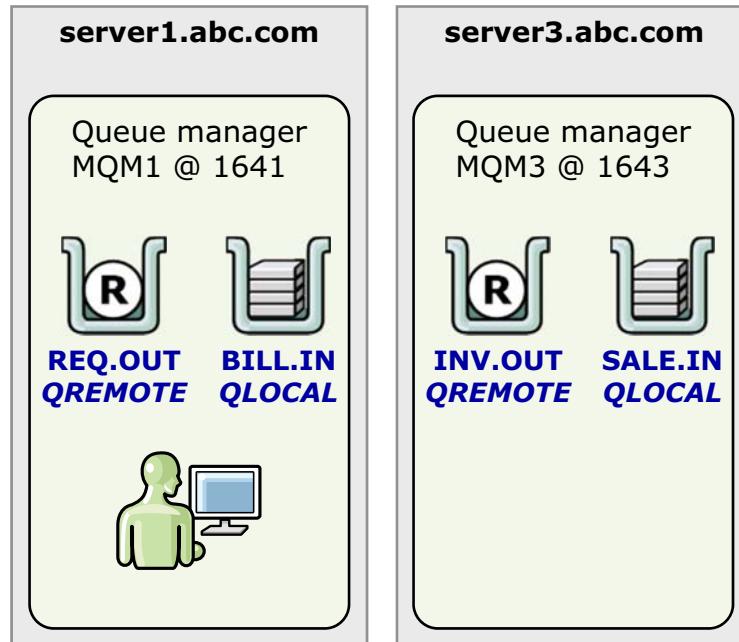
Notice that in the display of the QREMOTE attributes there is no CURDEPTH attribute because the QREMOTE object itself never holds messages. As you see in later slides, messages that are placed in a QREMOTE might be found in:

- The transmit queue that is associated with the QREMOTE. The transmit queue is specified in the QREMOTE XMITQ attribute.
- The dead-letter queue of the local or remote queue manager. Here the “local” dead-letter queue is the dead-letter queue that is in the same queue manager as the QREMOTE definition.
- The target queue; that is, the QLOCAL in the remote queue manager to which the QREMOTE RNAME attribute points.

Terminology checkpoint

Current queue manager is MQM1

- REQ.OUT is a local remote queue, locally owned remote queue, or local QREMOTE
- BILL.IN is a locally owned local queue, or a local QLOCAL
- INV.OUT is a remote QREMOTE, or remotely owned remote queue
- SALE.IN is a remote local queue, or remotely owned local queue
- Locally owned remote queue REQ.OUT points to remotely owned local queue SALE.IN



© Copyright IBM Corporation 2015

Figure 1-16. Terminology checkpoint

WM3021.1

Notes:

With the introduction of queues, it is time to make more distinctions about the terminology used.

Sometimes practitioners use the term “current queue manager” instead of “local queue manager”; they both mean the queue manager currently being worked on.

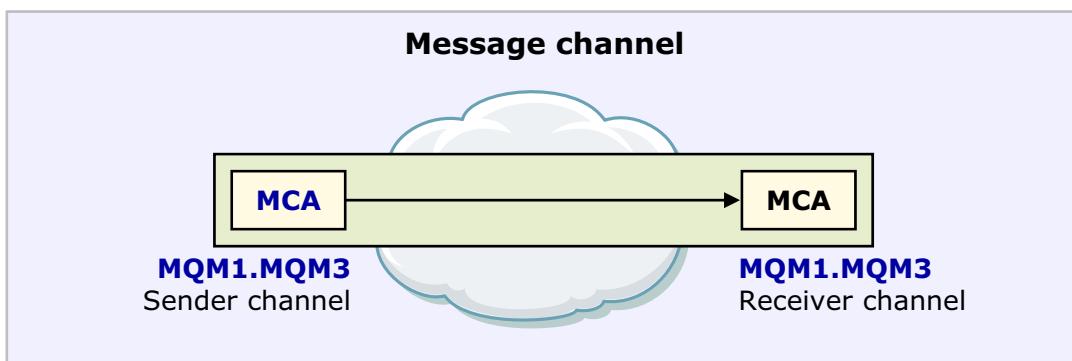
As noted in the queue manager terminology checkpoint, the terms local and remote are relative to the queue manager where work is done.

While becoming familiar with WebSphere MQ, it is helpful to draw a box for each queue manager and draw the respective queues inside each box when referring to queue manager definitions. This diagram helps to facilitate any conversations about the infrastructure.

Channels

- Channels are the combination of MCA's and network
- Usually the sending MCA, the network connection, and the receiving MCA
- Channels come in two main categories:
 - **Message channels:** Distributed or clustered
 - **Client (or MQI) channels**

```
AMQ8414: Display Channel details.
CHANNEL(MQM1.MQM3)
CHLTYPE(SDR)
CONNNAME(server3.abc.com(1643))
TRPTYPE(TCP)
XMITQ(MQM3)
```



© Copyright IBM Corporation 2015

Figure 1-17. Channels

WM3021.1

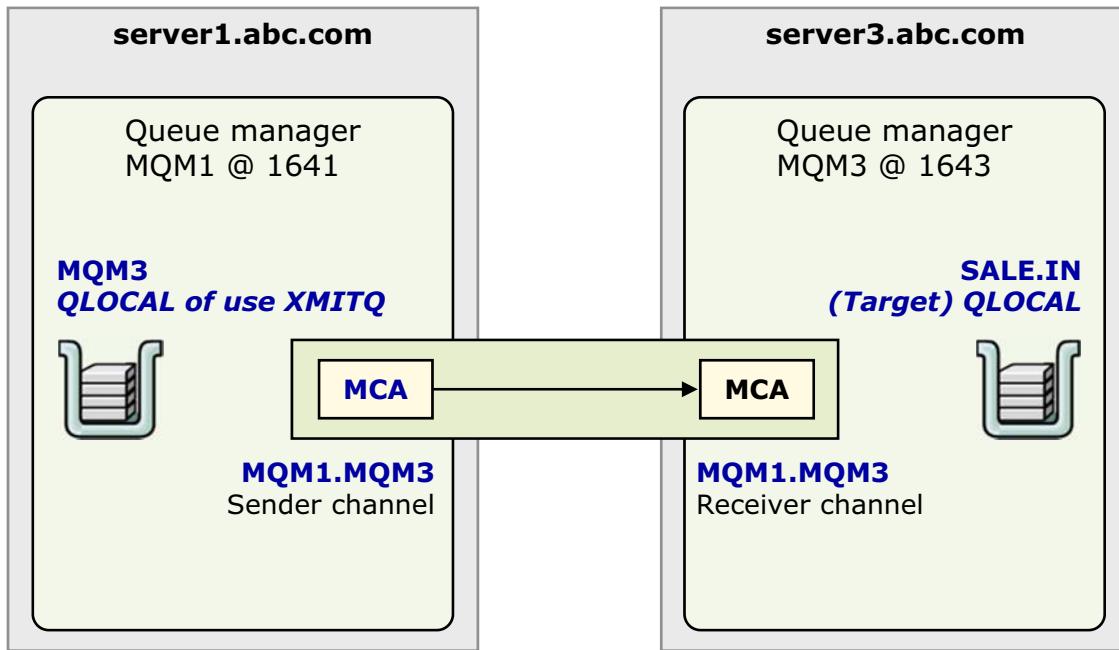
Notes:

Channels are the processes that move messages across queue managers. There are different types of channels, but the most prevalent are sender-receiver channels and client channels.

Clustered channels are discussed in the “IBM MQ cluster basics” unit. The focus now is on distributed message channels.

While there are four types of distributed message channels, the most prevalent types found are sender-receiver channels. The different distributed message channels are detailed in the “Distributed queuing” unit.

Sender-receiver channel pair



© Copyright IBM Corporation 2015

Figure 1-18. Sender-receiver channel pair

WM3021.1

Notes:

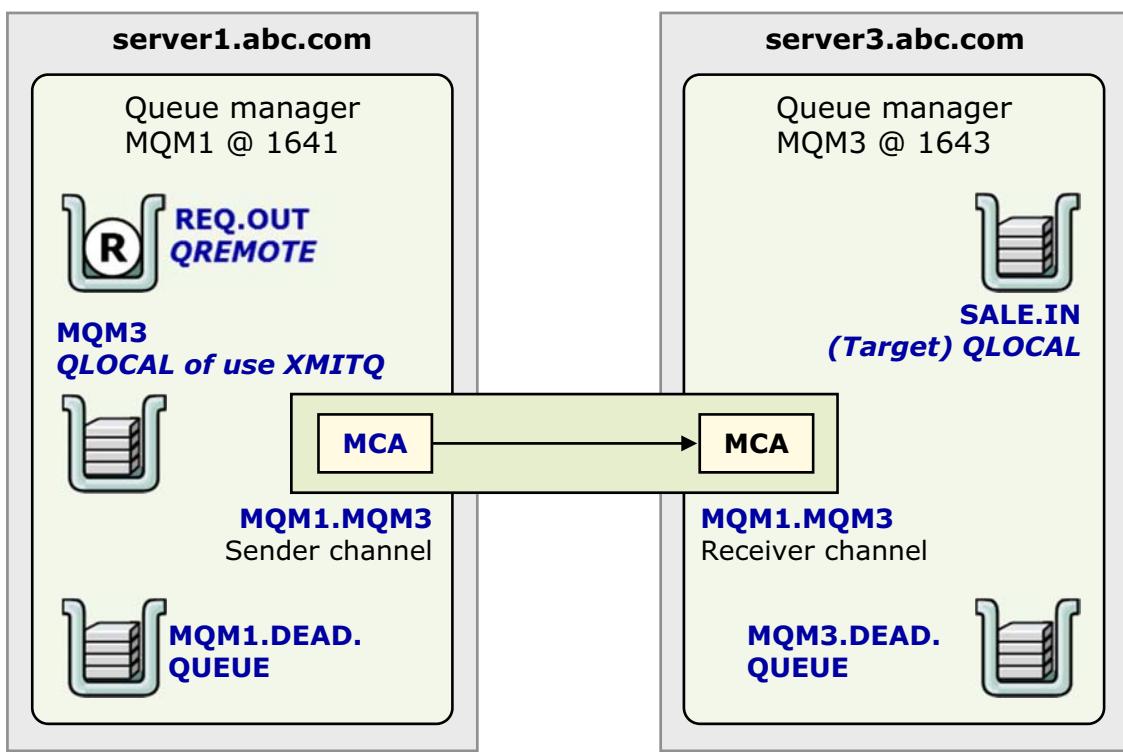
This visual of a sender-receiver channel pair displays some basic concepts. The names of the sender and receiver channels are both MQM1.MQM3. If the sender and receiver names are not identical, the channel does not start. The channels might be called anything, such as TRICK.OR.TREAT, and if both ends of the channel have the same name, they work.

Be careful with the connection name attributes. Omitting the port number results in a connection to port 1414 attempted. 1414 is the IBM MQ default port.

This channel takes messages from transmit queue MQM3 to send it to the remote queue manager of the same name. It is a good practice to name the channels to be the same as the “from” and “to” queue managers, in the correct from-to order.

If a second channel pair must be defined from and to the same queue managers, the name can always be qualified at the end to differentiate. However, the from-to convention should be followed as closely as possible for self-documentation.

Putting it all together



© Copyright IBM Corporation 2015

Figure 1-19. Putting it all together

WM3021.1

Notes:

You defined the queues, so now you get help for finding the message.

This application uses the REQ.OUT QREMOTE in the MQM1 local queue manager to place a message to the SALE.IN remote local queue in the MQM3 remote queue manager. Where can the message be?

QREMOTES, or local remote queues, do not hold messages, so here is the path of the message on its way to the SALE.IN queue:

- When the application uses REQ.OUT for the MQPUT, the message goes to the transmission queue associated with the REQ.OUT remote queue; that is, the MQM3 local transmit queue.
- If the channel is triggered or already running, it picks up the message and takes the message to the MQM3 queue manager.
- If the SALE.IN remote local queue is defined and available, the message is placed in its target queue.

The message might not make it to the target queue for several reasons. The message might end up in the local or remote dead-letter queue. Look for console messages that indicate what happened.

Cumulative checkpoint with distributed platforms

At server1.abc.com

- Define and configure queue manager MQM1
- Set dead-letter queue to SYSTEM.DEAD.LETTER.QUEUE
- Define and start listener to automatically start with queue manager at port 1641
- Define QLOCAL MQM3 to use as a transmission queue
- Define QREMOTE REQ.OUT to:
 - Use QLOCAL MQM3 as transmit queue
 - Point to remote QLOCAL SALE.IN in queue manager MQM3 as target
- Define sender channel MQM1.MQM3 to:
 - Point to server server3.abc.com port 1643
 - Use MQM3 as its transmission queue
 - After RECEIVER is defined, start channel

At server3.abc.com

- Define and configure queue manager MQM3
- Set dead-letter queue to SYSTEM.DEAD.LETTER.QUEUE
- Define and start listener to automatically start with queue manager at port 1643
- Define QLOCAL SALE.IN
- Define receiver channel MQM1.MQM3

© Copyright IBM Corporation 2015

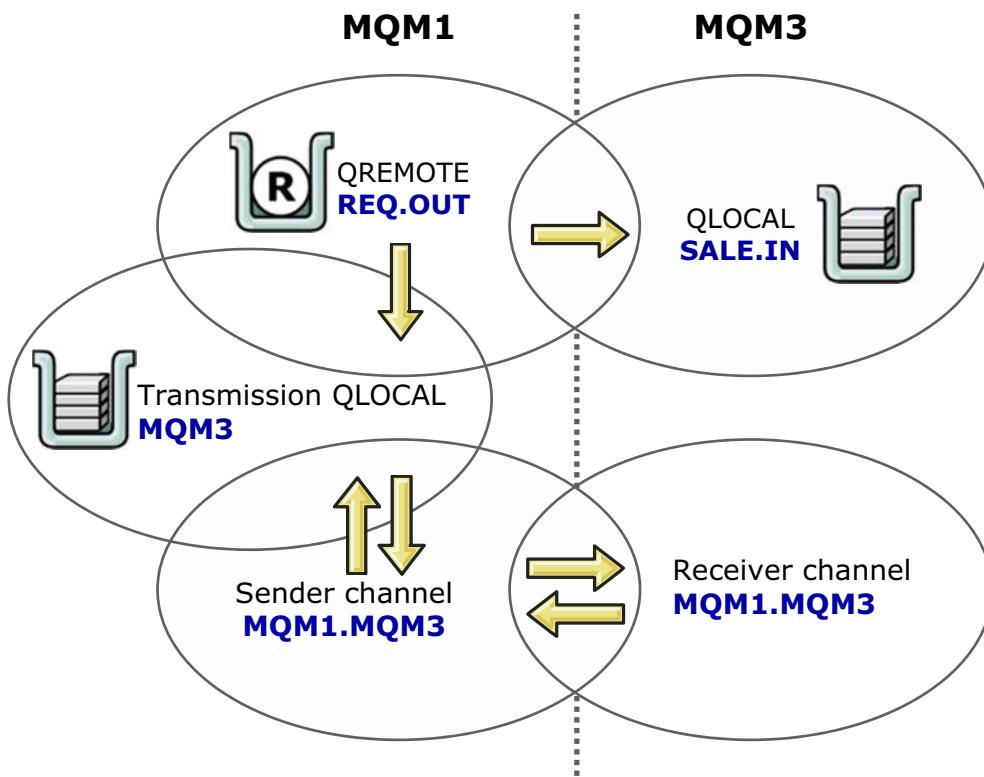
Figure 1-20. Cumulative checkpoint with distributed platforms

WM3021.1

Notes:

This slide summarizes the definitions that are made for the channel pair from MQM1 to MQM3.

IBM MQ distributed object relationships



© Copyright IBM Corporation 2015

Figure 1-21. IBM MQ distributed object relationships

WM3021.1

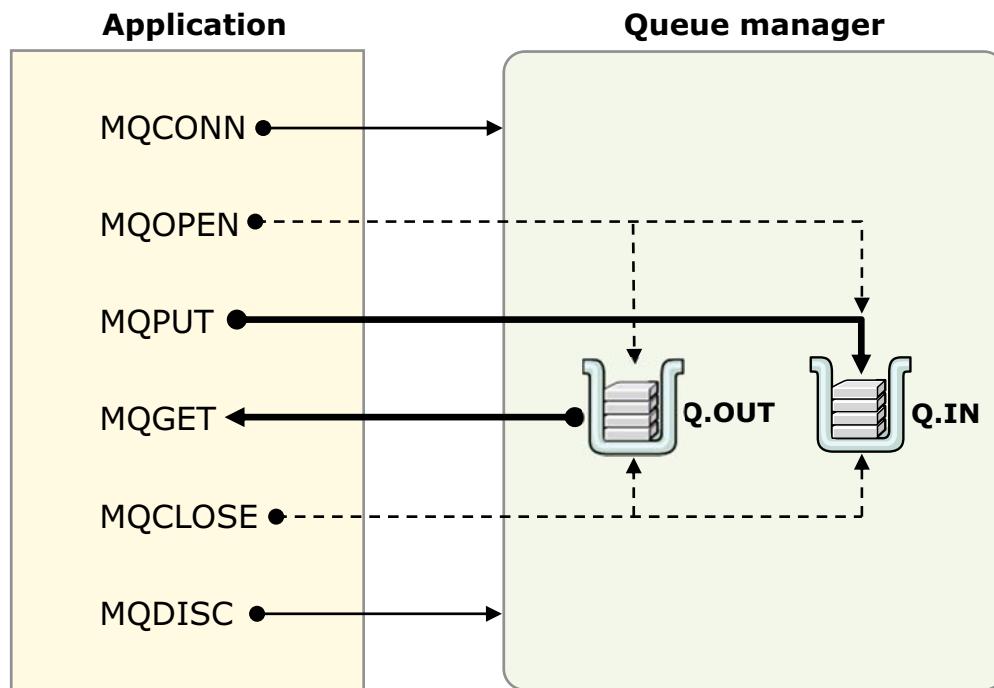
Notes:

The same transmit queue can be used for several remote queues, and for one sender channel. Looking at object relationships, when a message is put in the MQM1 local remote queue REQ.OUT, this message is placed in the MQM3 transmit queue.

If channel MQM1.MQM3 is running, the MCA consumes the message and sends the message to its remote target local queue at MQM3, SALE.IN.

Designating MQM1 as the local queue manager, you can see how the QREMOTE is related to the transmission queue on the local queue manager, and also to the target queue in the remote queue manager. The sender channel is related to the transmission queue in the local queue manager and the receiver channel in the remote queue manager.

Message Queue Interface (MQI)



IBM MQ provides numerous sample programs.

© Copyright IBM Corporation 2015

Figure 1-22. Message Queue Interface (MQI)

WM3021.1

Notes:

So far, queue managers, messages, queues, and channels are explained. But how do messages get placed into queues?

Applications connect to a queue manager and open a queue. If the MQCONNECT and MQOPEN calls are successful, then messages are put to the queue. If many messages are expected to be placed in the queue, there can be a loop of puts to accomplish this purpose. It would be more efficient than connecting and opening the queue for every put.

It is important to always close the queues and disconnect from the queue manager when finished placing messages in the queue.

WebSphere MQ contains numerous code samples and compiled sample applications that can be used to test putting and getting messages from a queue. Some of these sample programs are:

- `amqspput`: Puts messages in a queue.
- `amqsbcg`: Gets messages from the queue in a non-destructive way, returns the formatted MQMD header, and dumps the message data. This sample program was used to provide the screen capture in the following slide.
- `amqsgget`: Does a destructive get of the message data.

Headers and defaults

- IBM MQ has attribute default values for definitions and headers
- The persistence attribute defaults to nonpersistent in queue and MQMD



- Applications should set MQMD value to persistent only when needed
- Attributes set by the application override object definitions

```
/* MQMD Structure - (partial view)
...
    MQLONG    Report;          /* Options for report messages */
    MQLONG    Expiry;          /* Message lifetime */
    MQLONG    Priority;        /* Message priority */
    MQLONG    Persistence;     /* Message persistence */
    MQBYTE24  MsgId;          /* Message identifier */
    MQBYTE24  CorrelId;        /* Correlation identifier */
    MQLONG    BackoutCount;    /* Backout counter */
    MQCHAR48  ReplyToQ;        /* Name of reply queue */
    MQCHAR48  ReplyToQMgr;     /* Name of reply queue manager */
    MQCHAR12  UserIdentifier;  /* User identifier */
```

© Copyright IBM Corporation 2015

Figure 1-23. Headers and defaults

WM3021.1

Notes:

If you compare this display of the MQMD header to the attributes of a local queue definition, you see many fields in common. Fields have defaults in the queue attributes and in the MQMD header.

It is a good practice to establish setting attributes in the application, which is closer to the requirements than on the IBM MQ administrator.

Message expiration

- Expiry is a message property set by an IBM MQ application that limits the time that a message can remain in a queue before it is processed
- When the expiry time set by the application expires, the message is discarded as follows:
 - The next time an MQGET is attempted on the expired message
 - Explicitly by issuing an MQSC command to expire the messages on the queue
 - Periodically by configuring the queue manager to run expiry scans at set intervals
- Setting expiry is a good option for time-sensitive applications where messages become invalid if remaining in the queue longer than expected
- A special report message can be generated when a message expires



© Copyright IBM Corporation 2015

Figure 1-24. Message expiration

WM3021.1

Notes:

Expiry is a queue attribute and also an MQI option that holds the amount of time a message remains valid after it is placed in a queue. Expiry is used in time-sensitive applications where the payload can become obsolete.

When a message expires, it can generate a report message.

Report messages

- Messages that inform an application about events that pertain to the original message
 - Requested by setting the MQMD option when writing an application that puts messages to a queue
 - Delivered to the ReplyToQueue named in the MQMD
- Examples of report messages are:
 - Confirmation of arrival
 - Confirmation of delivery
 - Expiry report
- Consuming report messages should be part of the application design
- Report messages can be configured to contain:
 - The entire original message
 - The first 100 bytes of the original message
 - Just the report and MQMD but none of the original message



© Copyright IBM Corporation 2015

Figure 1-25. Report messages

WM3021.1

Notes:

Coordinating the capture of expiry notification needs to be planned across different WebSphere MQ roles. Unit 9 explains the enabling of events. After they are enabled, if needed, these event messages need to be processed when tracking of expired messages is an application requirement. Like any other local queue, if the messages in the event queues are not consumed, a full queue situation occurs.

JMS support

- IBM MQ V8 supports JMS 2.0
 - Delivery delay option support
 - New queue: SYSTEM.DDELAY.LOCAL.QUEUE
- IBM MQ Resource Adapter available for Java Platform, Enterprise Edition 7 standard application servers
 - Adapter available for all platforms except z/OS
 - If using an adapter, you must have JMS 2.0 on both sides

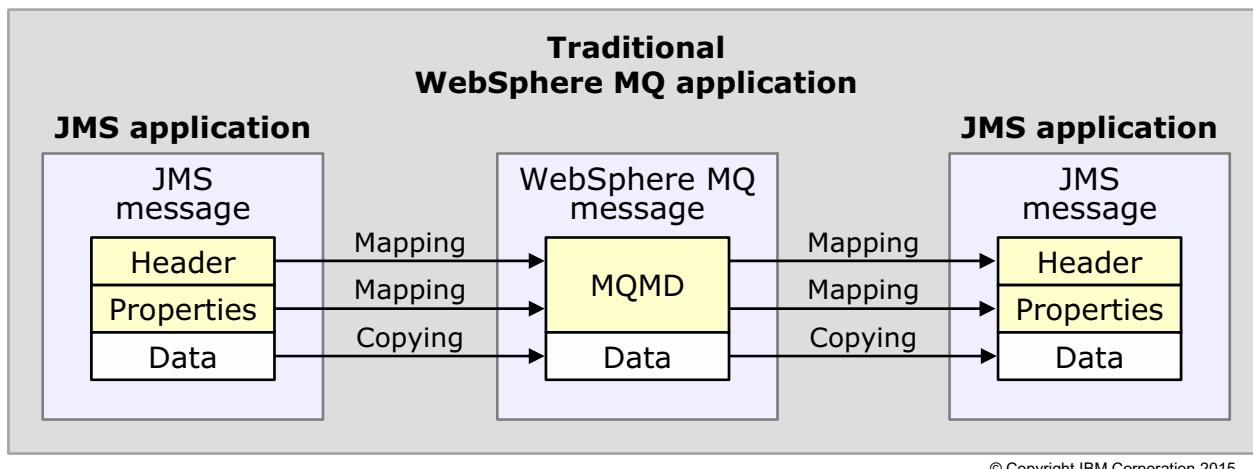


Figure 1-26. JMS support

WM3021.1

Notes:

IBM MQ V8 supports version 2.0 JMS and the Delivery Delay option.

The Delivery Delay option works in a way that is opposite to expiry by allowing messages to be unavailable until after a specified period. This feature is implemented by allowing the message to be placed in a new system local queue that is called the SYSTEM.DDELAY.LOCAL.QUEUE.

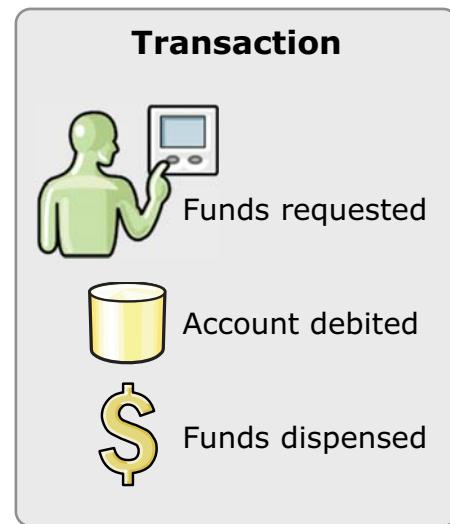
The delay function is available for applications that use the JMS API only, without any MQI code.

JMS 2.0 is available:

- For stand-alone applications directly
- Through a resource adapter that is provided for applications in an application server that supports Java Platform Enterprise Edition 7 standards

Transactions: Terminology

- A **resource manager** is a system that owns and controls its components such as:
 - A database manager owns its tables
 - A queue manager owns its queues
- A **transaction** or **unit of work** is a set of changes that must be completed in their entirety (**committed**) or restored to a previous consistent state (**backed out**)
- A **transaction manager** is a subsystem that coordinates units of work
- An IBM MQ queue manager can:
 - Act as a transaction manager over its own resources
 - Manage updates to DB2 tables
 - Run under a compatible external transaction manager
- An **ATM withdrawal** is a common scenario of a process that needs to be coordinated with a transaction manager



© Copyright IBM Corporation 2015

Figure 1-27. Transactions: Terminology

WM3021.1

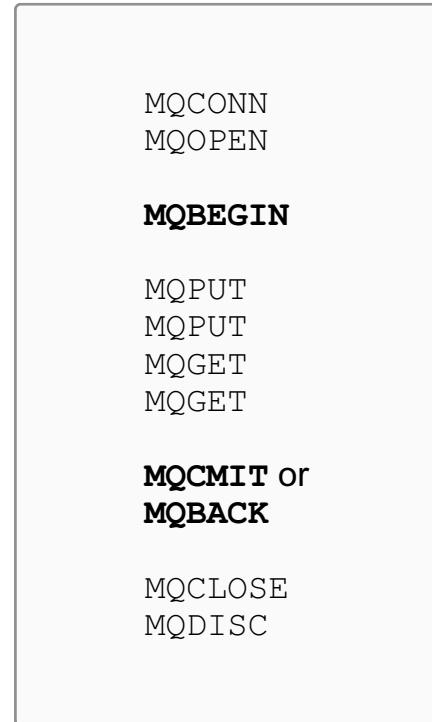
Notes:

This slide baselines transaction-related terminology.



Transactions: Local units of work

- Local units of work occur when all the puts and gets are coordinated by the same queue manager as a **single phase commit** process
- The MQBEGIN and MQCMIT or MQBACK verbs are used to start and complete the unit of work
- The channel process is an example of a local unit of work
 - A channel process defaults to a batch of 50 messages
 - The channel commits after it sends all the messages in the transmission queue or reaches the configured batch size
 - Messages are not visible by the receiving application until the MQCMIT is issued



© Copyright IBM Corporation 2015

Figure 1-28. Transactions: Local units of work

WM3021.1

Notes:

Local units of work involve only the queue manager. Coordination is achieved through the local queue manager.

Applications start a local unit of work with one of the MQI functions, such as MQGET or MQPUT, specifying the appropriate syncpoint option.

When the work is completed, the function MQCMIT is used to commit the work, or MQBACK is used to roll it back.

Transactions: Global units of work

- Global units of work are transactions that need to be coordinated across two or more resource managers by using a **two-phase commit** process
- Global units of work might involve different queue managers and database tables

Using an IBM MQ queue manager as the transaction manager

- Supported on Linux, UNIX, and Windows environments only
- Unit of work that is started with the MQBEGIN verb
- Unit of work that is completed with the MQCMIT or MQBACK verb, for example:

```
MQCONNECT
MQOPEN
MQBEGIN
MQGET
MQPUT
SQL INSERT
MQCMIT or MQBACK
MQCLOSE
MQDISC
```

Using an external transaction manager

- The external transaction manager must be compatible (X/Open XA compliant)
- IBM MQ queue manager is a participant but does not manage the transaction

- An application requests the external transaction manager (TM) to start the unit of work
- TM API controls the transaction
- No MQBEGIN, MQCMIT, or MQBACK
- Examples of transaction managers are CICS and Microsoft Transaction Server
- Requirements for each TM must be confirmed

© Copyright IBM Corporation 2015

Figure 1-29. Transactions: Global units of work

WM3021.1

Notes:

This slide presents global units of work, which involve coordinating transactions over two or more resources.

MQBEGIN is available on distributed platforms. z/OS always has a transaction manager.

Unless running under an external transaction manager, MQCMIT and MQBACK are available in all platforms.

Syncpoint options should always be specified on MQI calls.

- z/OS MQI assumes SYNCPOINT.
- Distributed MQI assumes NO_SYNCPOINT.

Two-phase transactions on z/OS can use RRS, CICS, and IMS for transaction management. On distributed platforms, XA is the standard two-phase interface, and IBM MQ can act as a transaction manager.

Unit summary

- Summarize how message-oriented middleware is critical to an enterprise
- Describe the role of a queue manager
- Describe messages and their contents
- Describe a queue
- Distinguish between queues that hold messages and queues that do not hold messages
- Explain some of the queues that are used for special purposes
- Describe the role of a channel
- Describe the calls that help place and retrieve messages from queues
- Identify key queue and message attributes and how to set them
- Summarize JMS support
- Describe transactions

© Copyright IBM Corporation 2015

Figure 1-30. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. Which of the following queues can hold messages?
 - a. A remote queue
 - b. The dead-letter queue
 - c. A local queue
 - d. A transmission queue

2. Work is being done in queue manager QM1. This queue manager has remote queue BILL.OUT pointing to queue manager's QM2 local queue BILL.OUT.DATA. Which statements accurately describe the environment?
 - a. BILL.OUT.DATA is a remotely owned local queue
 - b. BILL.OUT.DATA is a remotely owned remote queue
 - c. BILL.OUT queue holds messages
 - d. QM2 is the remote queue manager

© Copyright IBM Corporation 2015

Figure 1-31. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

1.

2.

Checkpoint questions (2 of 2)

3. Select the best answer or answers. What port will a channel with the CONNAME attribute coded as ('server123.enterp.com') point to?
 - a. The default IBM MQ port
 - b. The TCP/IP queue manager designated port
 - c. 1414
 - d. The remote queue manager's port

© Copyright IBM Corporation 2015

Figure 1-32. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

Checkpoint answers (1 of 2)

1. Which of the following queues can hold messages?

- a. A remote queue
- b. The dead-letter queue
- c. A local queue
- d. A transmission queue

Answer: b, c, d

2. Work is being done in queue manager QM1. This queue manager has remote queue BILL.OUT pointing to queue manager's QM2 local queue BILL.OUT.DATA. Which statements accurately describe the environment?

- a. BILL.OUT.DATA is a remotely owned local queue
- b. BILL.OUT.DATA is a remotely owned remote queue
- c. BILL.OUT queue holds messages
- d. QM2 is the remote queue manager

Answer: a, d

© Copyright IBM Corporation 2015

Figure 1-33. Checkpoint answers (1 of 2)

WM3021.1

Notes:



Checkpoint answers (2 of 2)

3. Select the best answer or answers. What port will a channel with the CONNAME attribute coded as ('server123.enterp.com') point to?
 - a. The default IBM MQ port
 - b. The TCP/IP queue manager designated port
 - c. 1414
 - d. The remote queue manager's port

Answer: a, c

© Copyright IBM Corporation 2015

Figure 1-34. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Unit 2. IBM MQ architecture, installation, and configuration

What this unit is about

This unit describes the IBM MQ z/OS components and describes installation and configuration details for a queue manager on z/OS.

What you should be able to do

After completing this unit, you should be able to:

- Identify the key components of an IBM MQ z/OS queue manager
- List the responsibilities of the IBM MQ z/OS system administrator
- Identify z/OS system configuration tasks that are required before a queue manager can be configured
- Summarize the installation process for IBM WebSphere MQ for z/OS
- Describe the process, data sets, and JCL used to configure a z/OS queue manager
- Summarize eight-byte relative byte address considerations
- Complete and start a z/OS IBM MQ V8 queue manager

Unit objectives

- Identify the key components of an IBM MQ z/OS queue manager
- List the responsibilities of the IBM MQ z/OS system administrator
- Identify z/OS system configuration tasks that are required before a queue manager can be configured
- Summarize the installation process for IBM WebSphere MQ for z/OS
- Describe the process, data sets, and JCL used to configure a z/OS queue manager
- Summarize eight-byte relative byte address considerations
- Complete and start a z/OS IBM MQ V8 queue manager

© Copyright IBM Corporation 2015

Figure 2-1. Unit objectives

WM3021.1

Notes:

IBM MQ for z/OS V8.0

IBM MQ z/OS and extensions	IBM MQ	MQMFT	MQAMS
IBM MQ for z/OS V8.0	✓		
IBM MQ for z/OS Value Unit Edition (VUE) V8.0	✓		
IBM MQ Managed File Transfer for z/OS V8.0 (MQMFT)		✓	
IBM MQ Advanced Message Security for z/OS V8.0 (MQAMS)			✓
IBM MQ Advanced for z/OS V8.0 (MQADV)		✓	✓

© Copyright IBM Corporation 2015

Figure 2-2. IBM MQ for z/OS V8.0

WM3021.1

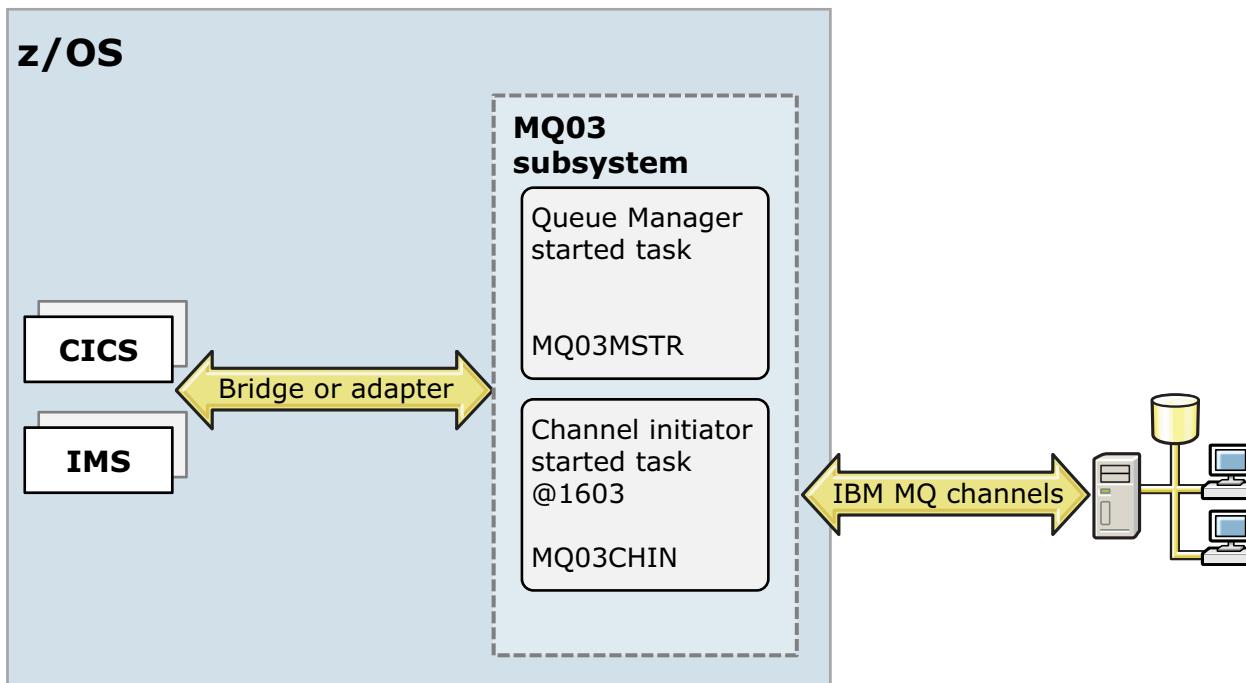
Notes:

In this class, you work with IBM MQ for z/OS V8.0 base product, and with IBM MQ Managed File Transfer for z/OS V8.0.

IBM MQ for z/OS Value Unit Edition V8.0 is a differently priced version of IBM MQ for z/OS V8.0. IBM MQ Advanced for z/OS is a packaging of IBM MQ Managed File Transfer and IBM MQ Advanced Message Security for z/OS.

WebSphere MQ is being renamed IBM MQ. The change was completed too late to change how it was referred to in the product and in IBM MQ Explorer. Throughout this course, you still see mention of WebSphere MQ in some parts of the course material.

Queue manager on z/OS



© Copyright IBM Corporation 2015

Figure 2-3. Queue manager on z/OS

WM3021.1

Notes:

In z/OS, WebSphere MQ runs as a subsystem made up of two started tasks: the queue manager master task and the channel initiator. The queue manager is given a unique four-character name.

The queue manager provides services to applications and manages its resources. The queue manager is also responsible for managing the data sets it owns, such as the logs and message data sets. It also manages the memory buffers, the channel initiator, and the ability to interface with other subsystems such as CICS or IMS.

Channel initiator: CHIN

What takes place in the CHIN started task:

```
CSQX011I MQ03 CSQXGIP Client Attachment available
CSQX151I MQ03 CSQXSSLI 0 SSL server subtasks started, 0 failed
CSQX410I MQ03 CSQXREPO Repository manager started
CSQT975I MQ03 CSQXDPSI Distributed Pub/Sub Controller has start
CSQX022I MQ03 CSQXSUPR Channel initiator initialization complete
CSQT806I MQ03 CSQXFCTL Queued Pub/Sub Daemon started
CSQX004I MQ03 CSQXSPRM Channel initiator is using 30 MB of local
632 storage, 1678 MB are free
CSQX251I MQ03 CSQXSTRL Listener started, TRPTYPE=TCP INDISP=QMGR
CSQX023I MQ03 CSQXLSTT Listener started, 634 port 1603 address *,
```

© Copyright IBM Corporation 2015

Figure 2-4. Channel initiator: CHIN

WM3021.1

Notes:

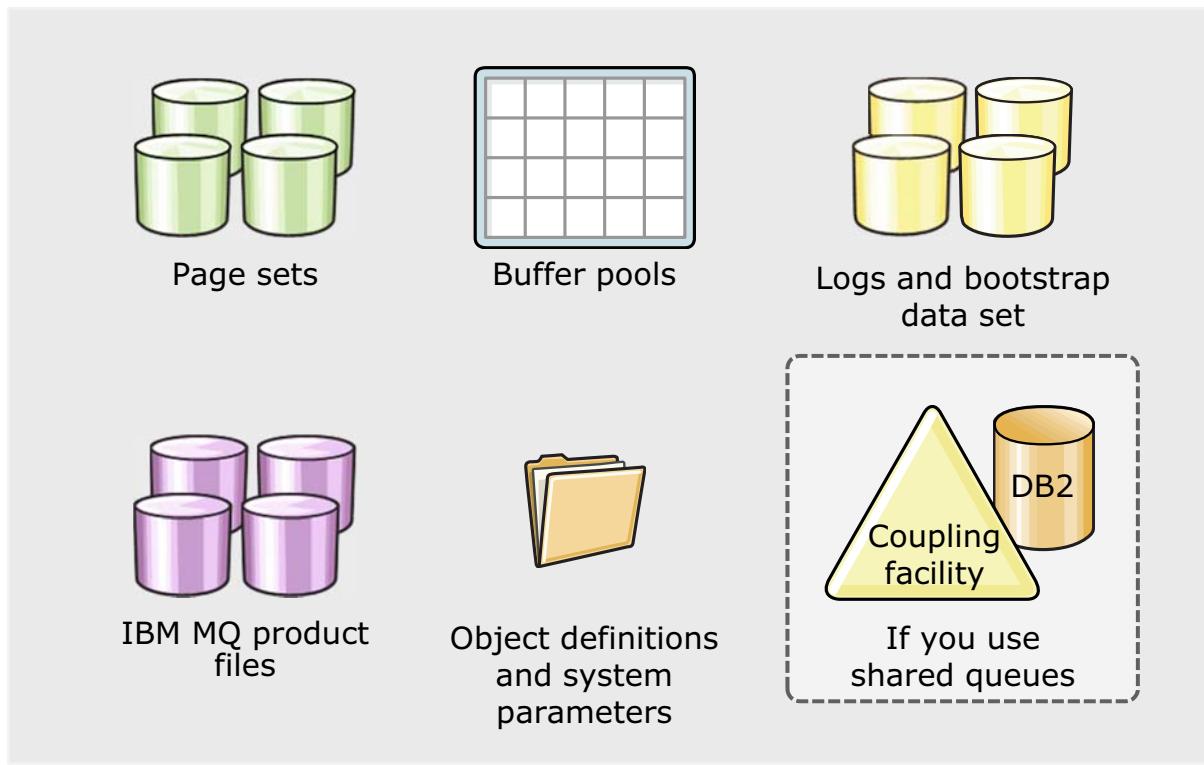
The display shows a partial list of channel initiator messages.

The first item that you see is the handling of the client attachment, followed by SSL.

The channel initiator hosts the IBM MQ cluster repository, and also publish/subscribe. Notice that two different publish/subscribe processes are started; these processes are discussed in the publish/subscribe unit.

The listener is a key component that the channel initiator handles. When the started task comes up, the cluster channels that are associated with the queue manager become active.

z/OS IBM MQ building blocks



© Copyright IBM Corporation 2015

Figure 2-5. z/OS IBM MQ building blocks

WM3021.1

Notes:

You start out with the IBM MQ product files, where the IBM MQ software is located after the SMP/E installation. To create a queue manager, you need to define:

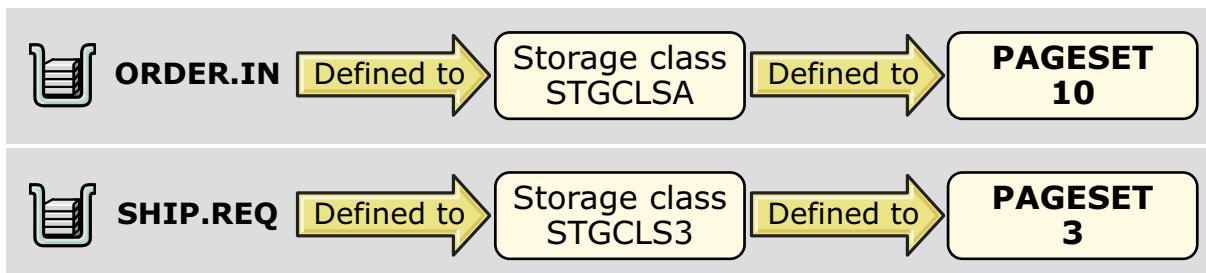
- Buffer pools, where messages are held when they arrive.
- Page sets, where messages are written if not retrieved from the queue soon after arriving at the queue.
- Logs, where persistent messages and transactions are recorded, and the resource that is used for restart and recovery. The bootstrap data set tracks the logs.

The IBM MQ object definitions and properties are an important part of the queue manager definition.

z/OS queue managers are also able to host shared queues and queue manager groups. You discuss shared queues in a later unit.

Page sets

- Specially formatted VSAM linear data sets that hold messages and object definitions
- Queues are defined to storage classes by using the QLOCAL STGCLASS attribute
- Storage classes are defined to page sets by using the Storage class PSID attribute
- Page set 0 is reserved for object definitions and other IBM MQ uses



Define the local queue:

- `DEFINE QLOCAL('ORDER.IN') STGCLASS('STGCLSA') CFSTRUCT(' ')`

Define the storage class:

- `DEFINE STGCLASS('STGCLSA') PSID(10)`

Define the PSID:

- `DEFINE PSID(10) BUFFPOOL(4) EXPAND (USER) <== Note: the page set definition is done in the CSQINP1 input to the MSTR address space`

© Copyright IBM Corporation 2015

Figure 2-6. Page sets

WM3021.1

Notes:

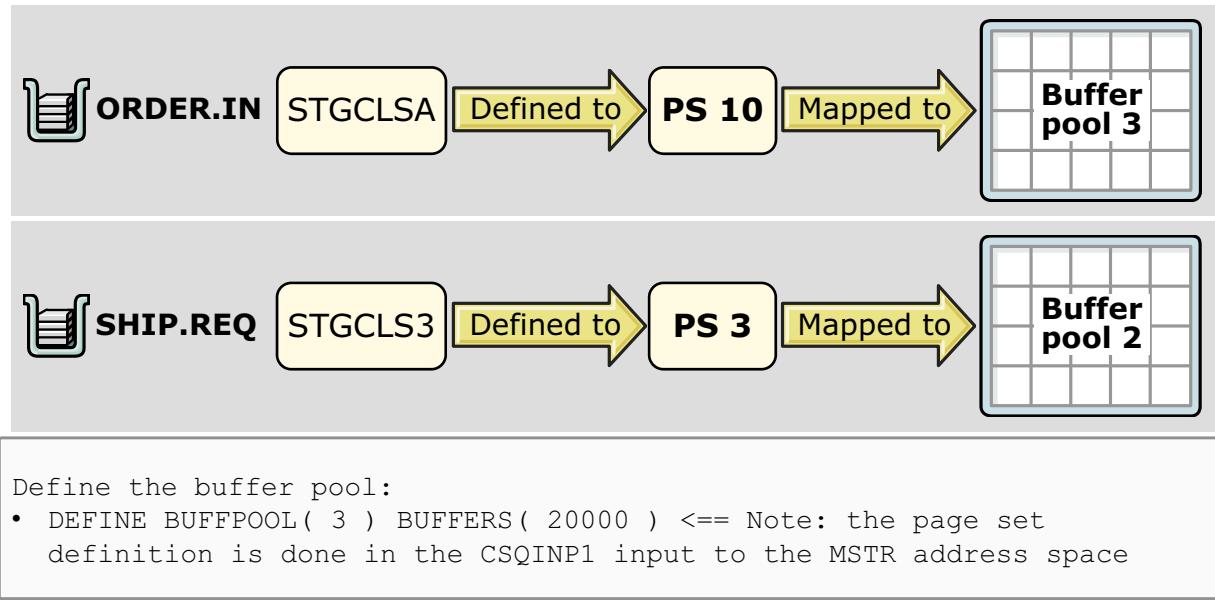
You now look at how the page sets are incorporated into the queue manager. After a page set is defined when the queue manager is created, a storage class is defined to use the specific data set. When a queue in a z/OS queue manager is used, it is assigned to a specific storage class. These storage classes might be associated with short messages, long messages, long-lived messages, or any other category.

When the storage class is defined, it is assigned a specific page set ID, which in turn is associated with a specific buffer pool.

Definition of page sets and storage classes is part of the lab exercise where a queue manager is configured.

Buffers and buffer pools

- Buffer pools are the first landing location, or primary storage areas for incoming messages
- When buffer pools become full, they are backed up with page set storage
- The two stage storage process aims to reduce I/O for messages that are consumed promptly



© Copyright IBM Corporation 2015

Figure 2-7. Buffers and buffer pools

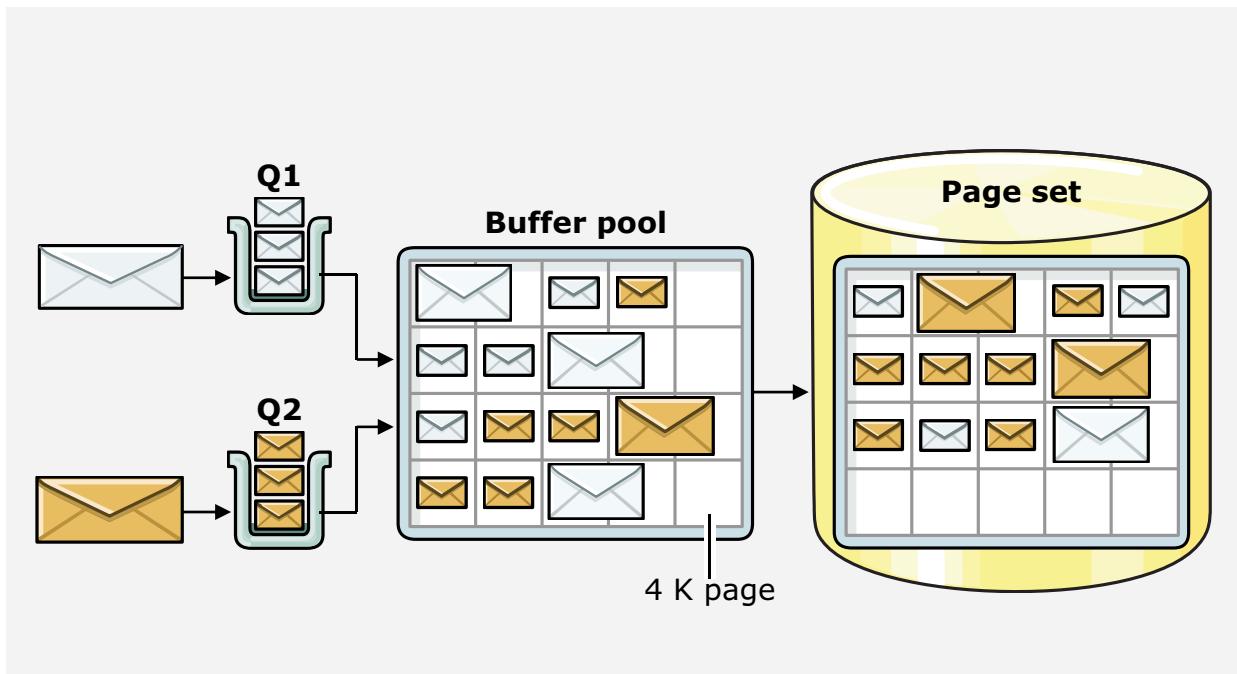
WM3021.1

Notes:

This page has the perspective of the object definitions. It shows the relationship between a local queue and a storage class, between the storage class and the page set, and the page set to the buffer pool. The buffer pool provides the first place where a message “lands” when it arrives at the queue manager.

Well behaved applications should read the message in a timely manner. When the message is removed promptly, depending on the timing, the message might not even be written to the page set; but if the message is persistent, it is logged.

Queues, buffer pools and page sets: Message path view



© Copyright IBM Corporation 2015

Figure 2-8. Queues, buffer pools and page sets: Message path view

WM3021.1

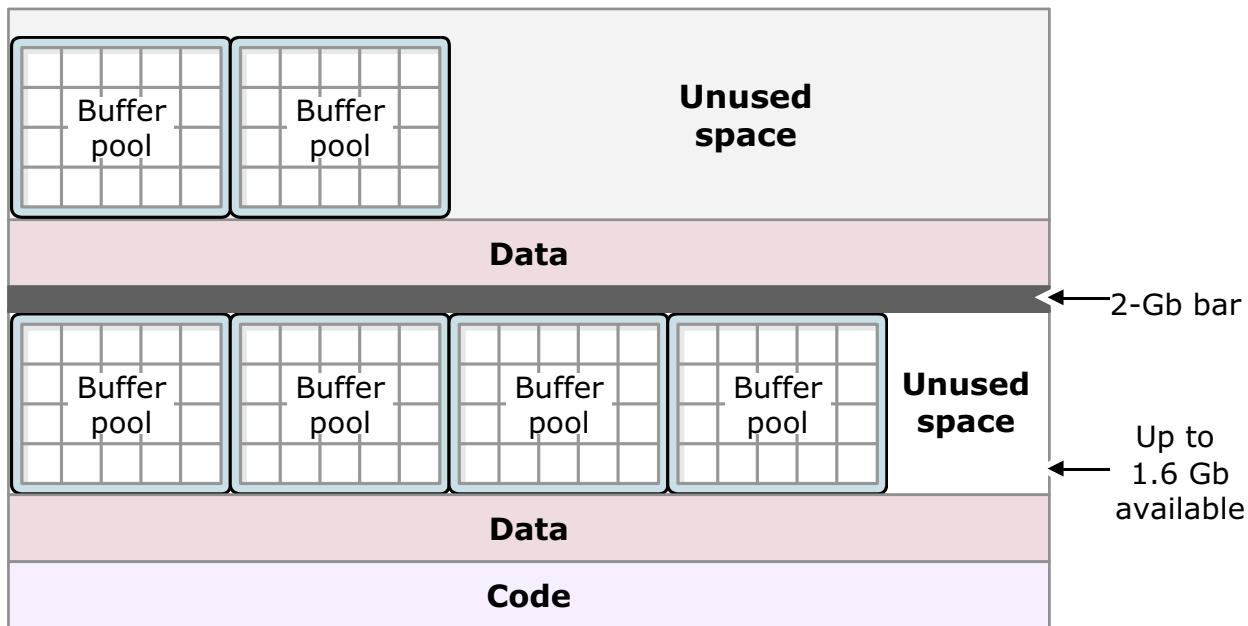
Notes:

You now look at buffer pools and page sets with focus on the path of the messages. A message that arrives at a queue is placed in one of the 4-K buffer pools. Depending on how long it takes before the application does the MQGET, the message might be removed from the buffer pool, or might need to be placed on a page set to await retrieval. So the application that gets the message might be retrieving it from the buffer pool, or from a page set.

Before IBM MQ V8, buffer pools had to be defined in 31-bit storage and shared this space with IBM MQ data and code. That left approximately 1.6 GB of space for buffer pools, for a maximum of 16 buffer pools. Up to 100 page sets might share these buffer pools.

Retrieving messages from buffer pools is much faster than retrieving messages from a page set, so IBM MQ V8 looked at a way to mitigate the buffer pool constraints.

Buffer pools above the bar



© Copyright IBM Corporation 2015

Figure 2-9. Buffer pools above the bar

WM3021.1

Notes:

IBM MQ V8 introduced the capability of 64-bit addressing for buffer pools, significantly improving the 31-bit buffer pool constraints. Instead of the 16 buffer pool limit, now it is possible to have up to 100 buffer pools.

Buffer pools above the bar considerations

- To enable new IBM MQ V8 functions such as storage above the bar, the **OPMODE** attribute of the **CSQ6SYSP** macro needs to be set to **OPMODE (NEWFUNC, 800)**
- New LOCATION buffer pool setting for IBM MQ V8 and later queue managers
 - BELOW: (Default) Buffer pool that is located below the 2 GB bar in 31-bit storage
 - ABOVE: Buffer pool is located above the bar in 64-bit storage
 - Can be changed dynamically
- Up to 100 buffer pools supported

```
DEFINE BUFFPOOL( 4 ) BUFFERS( 20000 ) LOCATION( BELOW )
ALTER BUFFPOOL( 4 ) LOCATION( ABOVE )
```

- SMF 115 subtype accommodates the extended buffer statistics records



Always check for sufficient real storage before allocating buffers above the bar, or other address spaces might be impacted

© Copyright IBM Corporation 2015

Figure 2-10. Buffer pools above the bar considerations

WM3021.1

Notes:

Three things must happen before enabling use of buffer pools above the bar:

1. Check for sufficient real storage to prevent impact to other address spaces.
2. Perform conversion of the BSDS data sets.
3. Enable the new v8 function in the queue manager's parameter module.

When a new queue manager is created with IBM MQ V8 or a queue manager is migrated to IBM MQ V8, the new v8 functions are not enabled by default.

Buffer pool and page set usage display

MQ00 DIS USAGE

CSQI010I MQ00 Page set usage ... 263

Page set	Buffer pool	Total pages	Unused pages	Persistent data pages	NonPersist data pages	Expansion count
- 0	0	322	286	36	0	USER 0
- 1	0	322	301	21	0	USER 0
- 2	1	322	318	4	0	USER 0
- 3	2	322	322	0	0	USER 0
- 4	3	322	292	28	2	USER 0

End of page set report

CSQI065I MQ00 Buffer pool attributes ... 264

Buffer pool	Available buffers	Stealable buffers	Stealable percentage	Page class	Location
- 0	50000	49962	99	4KB	BELOW
- 1	20000	19999	99	4KB	BELOW
- 2	50000	49997	99	4KB	ABOVE
- 3	20000	19984	99	4KB	BELOW

End of buffer pool attributes

CSQI024I MQ00 CSQIDUSE Restart RBA for system as 265

configured=00000000000601DD

© Copyright IBM Corporation 2015

Figure 2-11. Buffer pool and page set usage display

WM3021.1

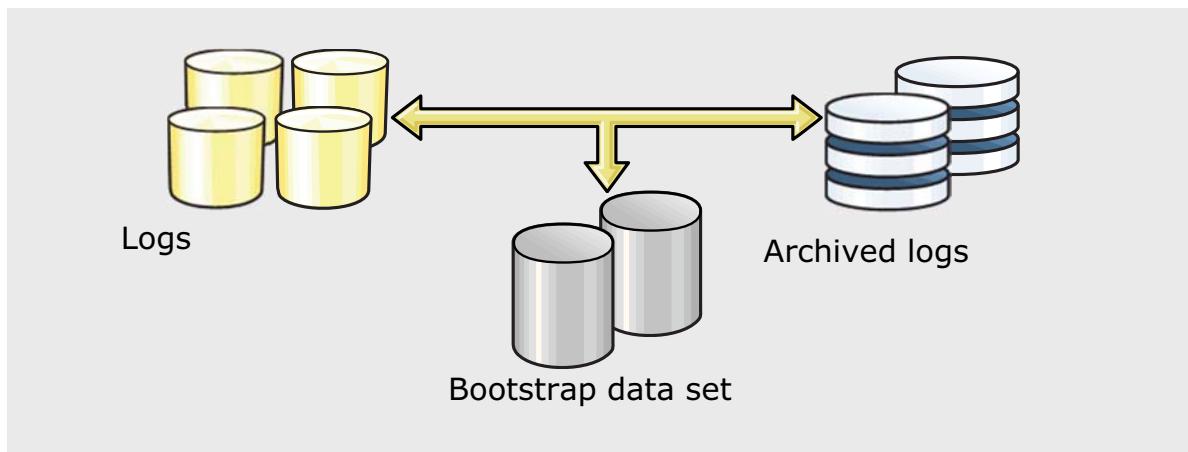
Notes:

This display shows the page and buffer usage that is obtained with the MQSC command DISPLAY USAGE. The same information can be displayed with IBM MQ Explorer or the IBM MQ ISPF display option for SYSTEM object type.

In this display, buffer pool 2 is allocated above the bar.

Logging

- IBM MQ logs queue manager information that is needed for restart and recovery in an active log
- The active log is composed of a series of up to 31 data sets
- The bootstrap data set (BSDS) tracks the log data sets
- Logs are archived to maintain available space



© Copyright IBM Corporation 2015

Figure 2-12. Logging

WM3021.1

Notes:

Logs are a critical part of IBM MQ. If you look at the queue manager master region started task, you can see how the logs are replayed when a queue manager starts.

On z/OS, the log is a set of up to 31 data sets used in a circular manner that saves persistent messages, IBM MQ objects, and transaction information. Logs do not keep statistics or monitoring information. When it is determined that a log data set is not needed for recovery, the log data set is archived to free up space for IBM MQ to continue its work. Without archiving, the logs fill up and IBM MQ stops working due to lack of space. It is critical to ensure that archiving is functional, particularly for production systems. The set of logs available for use is referred to as the active log. Archiving is started in the following situations:

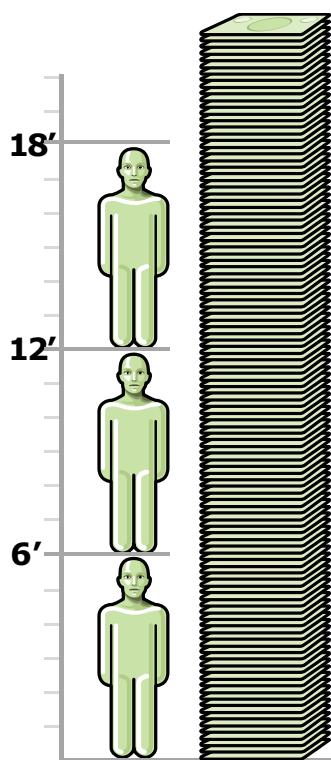
- When an active log data set is filled
- Explicitly when the ARCHIVE LOG command is issued
- If it is detected that a log data set is damaged

The bootstrap data set tracks which logs are active and which logs can be archived.

Log data can be written in single mode, or twice in dual mode to mitigate the risk of a data set getting corrupted.

The log is composed of log records. A relative byte address, or RBA, accesses each byte. Before IBM MQ V8, some customers would exceed the amount of data that might be logged; that is, their RBA wrapped over the 255 TB, or x'FFFFFFFFFFFF' addressing limitation of the 6-byte RBA. When the RBA wrapped, it was necessary to take an extended outage to reset the RBA, sometimes at the risk of losing persistent data.

Relative byte address: Height of a stack of currency



- The starting address of the IBM MQ logs is called the relative byte address, or RBA
- Each byte in the log is accessible by its offset from the RBA
- The RBA in z/OS IBM MQ was expanded from a 6-byte to an 8-byte RBA
- To use the new RBA addressing
 - Set OPMODE(NEWFUNC,800)
 - Convert the existing BSDS to the new format
 - Restart the queue manager

After a queue manager starts with OPMODE set to NEWFUNC,800 and the BSDS is converted to the new format, OPMODE cannot revert to an older level and the BSDS cannot revert to the old format.

© Copyright IBM Corporation 2015

Figure 2-13. Relative byte address: Height of a stack of currency

WM3021.1

Notes:

IBM MQ V8 introduced the capability of using an 8-byte RBA range, increasing the RBA limit by 64 K times. The additional range that is provided by the 8-byte RBA feature would resemble comparing 6-byte RBA to a single currency bill, and the new limit to a stack of bills over 28 feet tall. A customer that exceeded the 6-byte RBA range within 12 or 18 months now takes approximately 5578 years at the same volume to reach the RBA limit.

To handle the 8-byte RBA, the BSDS needs to be converted to the new BSDS format that can handle the 8-byte RBA.

New queue managers and existing queue managers that are migrated to IBM MQ V8 always have a 6-byte RBA. Moving to the 8-byte RBA requires running the conversion utility and reassembling the parameter module to reflect the new function in the OPMODE parameter of the CSQ6SYSP macro.

After a queue manager starts with a BSDS converted to the new 8-byte RBA format, it cannot revert to a 6-byte RBA format; the queue manager must stay at 8-byte RBA.

IBM MQ MSTR region partial restart messages

```

CSQJ127I MQ00 SYSTEM TIME STAMP FOR BSDS=2014-09-18 12:41:05.31
CSQJ001I MQ00 CURRENT COPY 1 ACTIVE LOG DATA SET IS 933
DSNAME=MQ00.LOGCOPY1.DS01, STARTRBA=0000000000000000
ENDRBA=000000000437FFF
CSQJ001I MQ00 CURRENT COPY 2 ACTIVE LOG DATA SET IS 934
DSNAME=MQ00.LOGCOPY2.DS01, STARTRBA=0000000000000000
ENDRBA=000000000437FFF
CSQJ099I MQ00 LOG RECORDING TO COMMENCE WITH 935
STARTRBA=000000000005E000
CSQJ034I MQ00 CSQJW007 END OF LOG RBA RANGE IS FFFFFFFFFFFFFF
CSQP007I MQ00 Page set 0 uses buffer pool 0
...
CSQR001I MQ00 RESTART INITIATED
CSQR003I MQ00 RESTART - PRIOR CHECKPOINT RBA=00000000005C946
CSQR004I MQ00 RESTART - UR COUNTS - 946
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
CSQI049I MQ00 Page set 0 has media recovery 947
RBA=0000000000052A51, checkpoint RBA=0000000000052A51
...

```

© Copyright IBM Corporation 2015

Figure 2-14. IBM MQ MSTR region partial restart messages

WM3021.1

Notes:

This display shows messages that are produced when the queue manager started task becomes active. A few details can be observed.

One of the first things to notice is the importance of the BSDS and the log data sets, and how the BSDS and logs are used to restart the queue manager.

The queue manager also identifies the active log data set.

It is also of interest that this queue manager is converted to use the 8-byte RBA as message CSQJ341I shows the end of the log range as x'FFFFFFFFFFFFFF'.

Object definitions: CSQ4MSTR in SCSQPROC data set

CSQ4INP2 data set is a concatenation of members such as:

• CSQ4INSG	System object definitions: Objects that are prefixed with SYSTEM*
• CSQ4INSA	System object and default rules for channel authentication
• CSQ4INSX	System object definitions
• CSQ4INYC	Clustering definitions: Sample template for objects that are used in clustering
• CSQ4INYD	Distributed queuing definitions: Sample template for distributed objects
• CSQ4INYG	General definitions: Such as dead letter queue, ALTER QMGR, CICS queues
• CSQ4INYR	Storage class definitions that use multiple page sets for the major classes of message
• CSQ4INYS	Storage class definitions that use 1 page set for each class of message
• CSQ4INP2 (member)	Include other commands, not necessarily object definitions, to issue when the queue manager is starting The channel initiator started task can be added here

© Copyright IBM Corporation 2015

Figure 2-15. Object definitions: CSQ4MSTR in SCSQPROC data set

WM3021.1

Notes:

Another component of the z/OS queue manager is object definitions. These PDS members are found in the SCSQPROC data set. The members to include are selected depending on what capabilities are included in the queue manager. Some members are used as is, while other members need to be edited to change variables to the required object names. A subset of these members is used in the first lab exercise.

Parameters: CSQ4ZPRM in SCSQPROC data set

- System parameters are included in the CSQ4ZPRM member that includes three macros to be assembled and linked:

CSQ4ZPRM is an assembly and link job for three macros:

CSQ6LOGP	Logging parameters such as the size of input buffer storage for active and archive data sets, OFFLOAD to turn archiving on or off, and TWOBSDS for using a single or dual BSDS
CSQ6ARVP	Archiving parameters such as the block size of the archive data set, space allocation, retention period, and security settings
CSQ6SYSP	System parameters such as LOGLOAD and tracing options

- A subset of these parameters can be altered when the queue manager is using MQSC commands SET SYSTEM, SET LOG, and SET ARCHIVE to run

© Copyright IBM Corporation 2015

Figure 2-16. Parameters: CSQ4ZPRM in SCSQPROC data set

WM3021.1

Notes:

A critical component of the queue managers is the parameter module, usually referred to as the "ZPARMS."

It is here that the logging and archiving parameters are set.

The CSQ6SYSP macro has other critical queue manager parameters, including the OPMODE setting to enable the above-the-bar buffer pools and 8-byte RBA.

Channel initiator parameters to review

Parameter	Description	Suggested values: But all depends on usage
CHIADAPS	Number of task control blocks (TCBs) used to make IBM MQ MQI calls to the queue manager (MQCONN and others)	<ul style="list-style-type: none"> Development and test systems: <i>Default of 8 normally OK unless significant number of persistent messages</i> Production systems: 30
CHIDISPS	Number of TCBs that are used to make calls to the communications network, such as to start channels	<ul style="list-style-type: none"> Default value if few channels Systems with more than 100 channels: 20 Rule of thumb: <i>One TCB for every 50 current channels, for example 40 if 2000 active channels</i> Limit: 100 for TCP channels
MAXCHL	Used to spread channels across available TCBs	<ul style="list-style-type: none"> Settings vary by the number of channels that are used and the need to match CHIDISPS

© Copyright IBM Corporation 2015

Figure 2-17. Channel initiator parameters to review

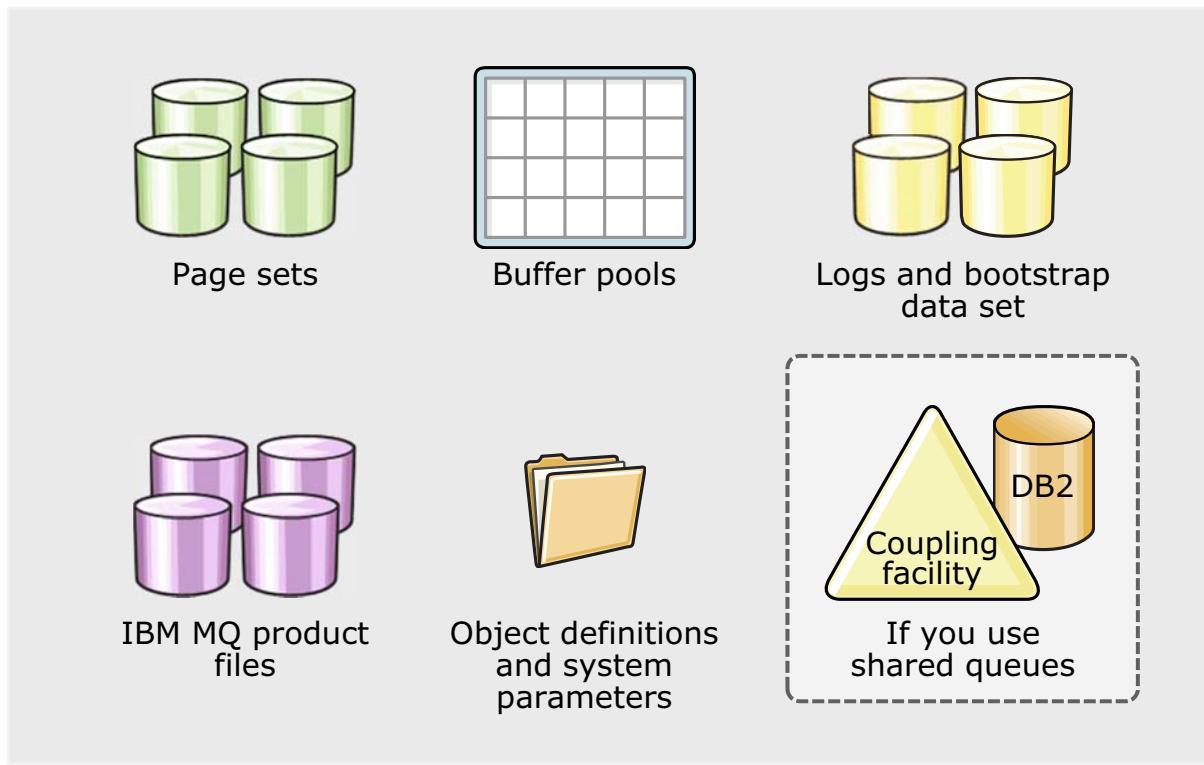
WM3021.1

Notes:

When you create the channel initiator started task in the lab, the defaults are taken. When you set up a channel initiator, particularly for a quality assurance or production environment, it is important to set these parameters according to the volumes and connections that are expected for the specific queue manager.

Given the scope of this course, detailed coverage of performance and tuning topics is deferred to an advanced course. Extensive performance and tuning information for IBM MQ on z/OS can be found in support pack MP16: *Capacity Planning and Tuning Guide for WebSphere MQ for z/OS*.

z/OS IBM MQ building blocks recap



© Copyright IBM Corporation 2015

Figure 2-18. z/OS IBM MQ building blocks recap

WM3021.1

Notes:

You reviewed the basic components that are required to configure a z/OS queue manager. Shared queues are not part of the installation and configuration lab and are covered in a later unit.

As a recap, the components that go into creating a queue manager are:

- The IBM MQ product files
- Page, log, and bootstrap data sets
- Storage classes, and allocate buffers
- Parameter module
- System objects and other definitions

Before creating the queue manager, other tasks need to be completed in the z/OS system.

Before proceeding to the installation and configuration, you review some of the tasks that are required of the IBM MQ administrator.

IBM MQ administrative tasks

- Install, apply maintenance, and configure:
 - IBM WebSphere MQ
 - IBM WebSphere MQ Managed File Transfer
- Configure queue managers
- Create queues, clusters, channels, queue-sharing groups, or any objects that users request
- Administer publish/subscribe topologies
- Administer security and SSL
- Back up data sets and object definitions



- Participate in performance analysis
- Monitor buffers and data set spaces
- Perform problem determination
- Participate in capacity planning
- Participate in infrastructure planning sessions
- Participate in naming standards

© Copyright IBM Corporation 2015

Figure 2-19. IBM MQ administrative tasks

WM3021.1

Notes:

Depending on the environment, an IBM MQ administrator might fulfill different functions. In some shops, the administrator is given system administrator privileges. In most shops, there are dedicated z/OS system administrators.

Regardless of the situation, some tasks always involve the IBM MQ administrator. It is important to understand the environment, the groups IBM MQ interfaces with, and the degree of governance that is built around IBM MQ.

While an IBM MQ administrator is able to maintain and correct problems in the IBM MQ environment, architects and developers must have an adequate understanding of IBM MQ. Often, architects or developers might introduce a problem in the environment that is difficult to rectify, for example:

- An architect can design an application that uses IBM MQ to hold messages indefinitely; this misuse can contribute to full log data sets if there is a major conversion or data migration effort.
- A developer might code an application and overlook proper closure, resulting in a queue handle that stays busy.

- A serious case is designing a long-running transaction that spans across many log data sets and causes log constraints and performance issues.

The IBM MQ administrator can resolve the problems, but the optimal scenario would be to prevent such situations so the time does not need to be spent troubleshooting problems that can be avoided.

For this reason, unless there is adequate governance in place, the IBM MQ administrator should be involved with the users to mitigate or intercept possible problems and provide guidance to the user community.

SSL, for instance, is also a challenge. It is not only the act of configuring and making it work, but who owns the certificates? Who makes sure these certificates are renewed in a timely manner?

Many areas of IBM MQ must be coordinated with other teams. The IBM MQ administrator needs to be at the center of issues that involve the health of the IBM MQ infrastructure.

Installation and configuration

- Collect and review the most recent documentation for IBM MQ version that is installed to confirm requirements and prepare to install
- Complete SMP/E install
- Perform these actions one time for each z/OS system:
 - Identify and update correct z/OS **SYS1.PARMLIB(IEASYSPP)** parameters
 - APF authorize the IBM MQ load libraries
 - Update the z/OS link list and LPA
 - Update the z/OS program properties table
- Define the IBM MQ subsystem to z/OS
- **Perform queue manager and channel initiator configuration**
- Define IBM MQ to a z/OS WLM service class
- If a queue-sharing group is used, perform related tasks
- Implement ESM security controls
- Set up the operations and control panels
- Include the IBM MQ dump formatting member for IPCS

© Copyright IBM Corporation 2015

Figure 2-20. Installation and configuration

WM3021.1

Notes:

Before a queue manager is installed and configured, some tasks need to be completed in the z/OS environment.

In this section, you walk through the steps that were completed in advance in the lab environment for this course. You also walk through the section to be completed in the lab, titled “Perform queue manager and channel initiator configuration.”

There are two sources of reference information for the work in this section:

- Program Directory for WebSphere MQ for z/OS V8.0. This document contains key installation information such as SMP/E details, resulting data sets, requirements, and package names.
- Section “Configuring queue managers on z/OS” is found either in the IBM MQ V8.0 IBM Knowledge Center or under the same section name in the corresponding PDF file: `wmq80.install.configure.pdf`, available at the IBM MQ V8.0 IBM Knowledge Center.

There can be more than one queue manager in the same z/OS server. When configuring multiple queue managers in the same server, some of the steps that are outlined are done one time per server. These steps do not need to be repeated, while other steps are done one time per queue manager.



Collect and review most recent documentation

- Program Directory for WebSphere MQ for z/OS V8
 - Information on components
 - SMP/E instructions
 - Hardware and software requirements
- Check whether there is any IBM MQ for z/OS maintenance to apply
- Configuring queue managers on z/OS documentation from either:
 - WebSphere MQ 8.0.0 IBM Knowledge Center, or
 - Corresponding PDF download titled “Installing IBM WebSphere MQ”
- Capacity Planning and Tuning Guide for WebSphere MQ for z/OS V8.0 support pack MP16

© Copyright IBM Corporation 2015

Figure 2-21. Collect and review most recent documentation

WM3021.1

Notes:

Documentation for IBM MQ is updated frequently. It is likely that new or updated documentation will be available after this course is released. While this course mentions the most recent documentation at the time the course was written, it is important to double check that you always have current documentation for the version of IBM MQ you work with.

It might be that sometimes an older document still applies, as discussed with the information in the IBM MQ performance support packs. Although not listed in this slide, depending on the functions that are used, it might also be helpful to download the IBM MQ performance support packs for the last few IBM MQ versions:

- MP16: Capacity planning and tuning for WebSphere MQ for z/OS
- MP1B: WebSphere MQ for z/OS V7.1: Interpreting accounting and statistics data
- MP1H: WebSphere MQ for z/OS V7.1 Performance Report
- MP1J: WebSphere MQ for z/OS V8.0 Performance Report

Preparing to install IBM MQ for z/OS V8

- Determine what components to install

File system description	Required or optional
IBM WebSphere MQ for z/OS Base	Required
IBM WebSphere MQ for z/OS US English ENU	Required
IBM WebSphere MQ for z/OS Japanese JPN	Optional
IBM WebSphere MQ for z/OS Simplified Chinese CHS	Optional
IBM WebSphere MQ for z/OS Uppercase English ENP	Optional
IBM WebSphere MQ for z/OS French FRA	Optional
IBM WebSphere MQ for z/OS UNIX System Services Components	Optional

- Confirm that hardware and software meet requirements
- Environment planning
 - Use new or existing SMP/E environment
 - What high-level qualifiers are used
 - Space requirements

© Copyright IBM Corporation 2015

Figure 2-22. Preparing to install IBM MQ for z/OS V8

WM3021.1

Notes:

This information was obtained from the program directory.

IBM MQ V8 hardware and software requirements

Program number, product name, and minimum VRM/service level

5694-A01 z/OS V01.13.0 or higher
 5650-ZOS z/OS V02.01.0 or higher

Library type	Total space that is required in 3390 tracks	File system description
Target	3432	Target libraries
Distribution	7185	Distribution libraries
File system	162 Mb	IBM WebSphere MQ for z/OS UNIX System Services Components

© Copyright IBM Corporation 2015

Figure 2-23. IBM MQ V8 hardware and software requirements

WM3021.1

Notes:

Other requirements that depend on implemented functions

Function	Program number, product name, and minimum VRM/service level
JMS applications	5655-W43 IBM 31-bit SDK for z/OS, Java Technology Edition Version 7.0 5655-W44 IBM 64-bit SDK for z/OS, Java Technology Edition Version 7.0
SSL channels	5694-A01 Cryptographic Services System SSL (FMID HCPT3D0)
Some CipherSpecs for SSL channels	5694-A01 Cryptographic Services Security Level 3 (FMID JCPT3D1)
Optional random number generator	5650-ZOS Cryptographic Support for z/OS V1R13 - z/OS V2R1 (FMID HCR77A1)
IMS applications	5635-A04 IMS Version 13.1 or higher 5635-A03 IMS Version 12.1 or higher 5635-A02 IMS Version 11.1 or higher
CICS applications	5655-Y04 CICS Transaction Server for z/OS Version 5.1 or higher 5655-S97 CICS Transaction Server for z/OS Version 4.1 or higher 5655-M15 CICS Transaction Server for z/OS Version 3.2 (with APAR PK66866) or higher
Shared queue access	5615-DB2 DB2 for z/OS 11 or higher shared-queue access 5605-DB2 DB2 for z/OS 10.1 or higher shared-queue access

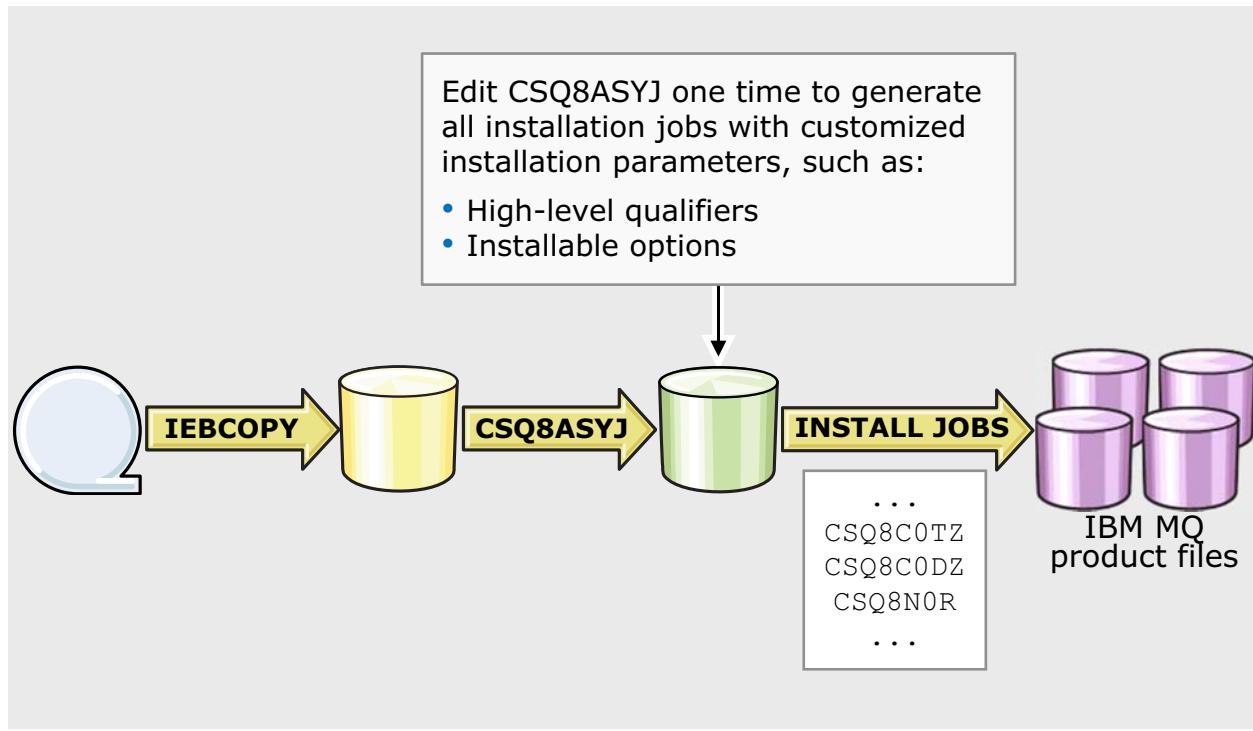
© Copyright IBM Corporation 2015

Figure 2-24. Other requirements that depending on implemented functions

WM3021.1

Notes:

Complete SMP/E install



© Copyright IBM Corporation 2015

Figure 2-25. Complete SMP/E install

WM3021.1

Notes:

As a first step when installing from the product tape, a temporary installation library is created. The library contains a job generator CLIST and the JCL skeletons for the jobs to be created.

The JCL generator is found in job (member) CSQ8AYSJ. CSQ8AYSJ is where you specify your data set names (high-level qualifiers), job card layout, volume names, and other details in order for the JCL jobs to be populated with the customized information.

Ready-to-run JCL is created as output of the CSQ8AYSJ job. These jobs create the SMP/E environment of your IBM WebSphere MQ for z/OS installation and the complete set of product libraries.

These activities are documented in detail in the *IBM WebSphere MQ for z/OS Program Directory*.

Product libraries, together with SMP/E data sets, can be ordered and delivered as a CBDPO package, or as part of a ServerPac.

Partial libraries and directories after installation

Library	Contents
SCSQANLE	IBM WebSphere MQ for z/OS US English ENU product code and utilities
SCSQASMS	Assembly language sample source
SCSQAUTH	Load modules: WebSphere MQ for z/OS product code and utilities
SCSQCICS	Load modules that CICS DFHRPL requires
SCSQMVR1	Load modules that are required for the mover when using TCP/IP with z/OS UNIX System Services sockets or IUCV interface, or LU 6.2
SCSQPROC	Sample JCL

	Directories present only when installation includes IBM MQ for z/OS UNIX System Services components	
Directory	File	Contents
http	WMQHTTP.war	The IBM MQ bridge for HTTP application
http/samples		Samples that use the IBM MQ bridge for HTTP
java/bin		IVPs and utilities
Javadoc	wmqjms_javadoc.jar	IBM MQ classes for JMS Javadoc

© Copyright IBM Corporation 2015

Figure 2-26. Partial libraries and directories after installation

WM3021.1

Notes:

The data sets created can be grouped into four basic areas:

- The basic set is always required and used. Multicultural support is provided through the SCSQANLx library; SCSQANLE is for US English, mixed case.
- The set that ISPF uses. These libraries are useful for administration tasks, but IBM WebSphere MQ does not need them for operation.
- This set is used for application programming and the sample programs (IVPs).

It is advantageous to become familiar with the numerous sample programs that are delivered with the product. Application programmers require these libraries when they develop IBM MQ programs.

- Where multicultural support is applicable, the library name for the default option (US English, mixed case) is shown.

Refer to the Program Directory for WebSphere for IBM MQ z/OS for a complete list of these data sets.

APF authorization requirements

	Library	Contents
APF authorize	SCSQANLE	IBM WebSphere MQ for z/OS US English ENU product code and utilities
APF authorize	SCSQASMS	Assembly language sample source
APF authorize	SCSQAUTH	Load modules: WebSphere MQ for z/OS product code and utilities
APF authorize	SCSQCICS	Load modules that CICS DFHRPL requires
APF authorize	SCSQMVR1	Load modules that are required for the mover
APF authorize	SCSQCOBC	COBOL copybooks: Sample and product copybooks
APF authorize	SCSQLINK	Load modules: Early code (must be in LPA)
APF authorize	SCSQLOAD	Non-APF authorized samples, user exits, IVPs, stubs, utilities, C++ runtime DLLs
APF authorize	SCSQPNLE	IBM WebSphere MQ for z/OS US English ENU panels to be included in ISPPLIB concatenation
APF authorize	SCSQSNLE	IBM WebSphere MQ for z/OS US English ENU load modules that are required for specialized function
	SCSQPROC	Sample JCL

© Copyright IBM Corporation 2015

Figure 2-27. APF authorization requirements

WM3021.1

Notes:

Early code: update LPA

- Early code libraries are:
 - CSQ3INI and CSQ3EPX in the library hlq.SCSQLINK
 - CSQ3ECMX in the library hlq.SCSQSNLx, where x indicates language
- To update LPA
 - Include libraries with most recent version, release, or maintenance level of IBM MQ early code in member LPALSTnn member of SYS1.PARMLIB
 - Optionally use CSQ8UERL sysmod from the SMP/E JCL to copy contents of hlq.SCSQSNLx into hlq.SCSQLINK
 - Schedule IPL
- To update without an IPL refresh z/OS and IBM MQ (V6 and later)
 - If you did not use CSQ8UERL


```
SETPROG LPA,ADD,MODNAME=(CSQ3INI,CSQ3EPX),DSNAME=thqual.SCSQLINK
SETPROG LPA,ADD,MODNAME=(CSQ3ECMX),DSNAME=thqual.SCSQSNLx
```
 - If you used CSQ8UERL


```
SETPROG LPA,ADD,MASK=*,DSNAME=thqual.SCSQLINK
```
 - Stop the queue manager
 - Refresh the early code for the queue manager: REFRESH QMGR TYPE(EARLY)
 - Restart the queue manager

© Copyright IBM Corporation 2015

Figure 2-28. Early code: update LPA

WM3021.1

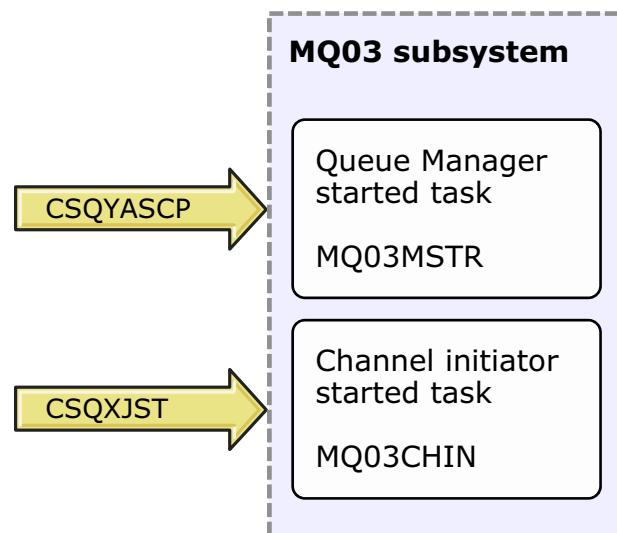
Notes:

Early code libraries need to be set up one time per server. For any queue managers in the server, a message is displayed indicating the early code level:

CSQ3111I MQ0A CSQYSCMD - EARLY PROCESSING PROGRAM IS V8.0.0 LEVEL 007-000

Update program properties table

- CSQYASCP needs to be included the program properties table (PPT)
- If the z/OS version is 1.12 or later CSQYASCP is already included in the PPT
- Use SYS1.PARMLIB (SCHEDxx) to add CSQYASCP to the PPT
- CSQXJST can optionally be added to the PPT



© Copyright IBM Corporation 2015

Figure 2-29. Update program properties table

WM3021.1

Notes:

Define the IBM MQ subsystem to z/OS

- Add manually to SYS1.PARMLIB(IEFSSNxx) member
 - Requires IPL
 - Keyword format (preferred)

```
SUBSYS SUBNAME (MQ00) INITRTN (CSQ3INI)  
INITPARM ('CSQ3EPX, MQ00, M')
```

- Positional format

```
MQ00, CSQ3INI, 'CSQ3EPX, MQ00, M'
```

- To activate before IPL with SETSSI command

```
SETSSI ADD, S=MQ00, I=CSQ3INI, P='CSQ3EPX, MQ00, M'
```

© Copyright IBM Corporation 2015

Figure 2-30. Define the IBM MQ subsystem to z/OS

WM3021.1

Notes:

Queue manager and channel initiator configuration

- Create all the data sets that IBM MQ for z/OS needs for operation
- Create start procedures for queue manager and channel initiator
- Set up standard IBM MQ for z/OS definitions
- Set up required adapters:
 - TSO or batch, RRS
 - CICS
 - IMS
- Set up channel definitions
- Set up security controls
- Set up queue-sharing groups

© Copyright IBM Corporation 2015

Figure 2-31. Queue manager and channel initiator configuration

WM3021.1

Notes:

For this course, you skip setting up the adapters, security controls, and queue-sharing groups.

You continue looking at the steps to be completed in the lab.

CSQ4PAGE: Create the page data sets

```
//CSQ4PAGE JOB
//DEFINE EXEC PGM=IDCAMS, REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
      DELETE '++HLQ++.PSID00' ERASE CLUSTER
      SET MAXCC=0
      DEFINE CLUSTER
          (NAME (++HLQ++.PSID00)           -
           RECORDS (1000 500)             -
           LINEAR                         -
           VOLUMES (++VOL0++)            -
           SHAREOPTIONS (2 3) )          -
      DATA
          (NAME (++HLQ++.PSID00.DATA)  )

/*
//FORM      EXEC     PGM=CSQUTIL,COND=(0,NE)
//STEPLIB  DD  DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
//          DD  DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
//CSQP0000 DD  DISP=OLD,DSN=++HLQ++.PSID00
//SYSPRINT  DD     SYSOUT=*
//SYSIN      DD     *
FORMAT
/*
```

© Copyright IBM Corporation 2015

Figure 2-32. CSQ4PAGE: Create the page data sets

WM3021.1

Notes:

The model job to create the page data sets used by the CSQ4MSTR procedure is in thlqual.SCSQPROC(CSQ4PAGE). CSQ4PAGR is the appropriate job for the CSQ4MSRR procedure that uses nine page data sets.

These page sets are standard VSAM data sets created with IDCAMS and formatted with the IBM WebSphere MQ for z/OS batch utility CSQUTIL.

They are VSAM *linear* data sets. Therefore, the application (the queue manager) sets up its own access method inside the data sets. Do not expect to be able to access them with anything except IBM WebSphere MQ for z/OS programs.

You must determine their size, up to the number of messages that are held on behalf of your applications. A formula for space calculation is provided in the chapter on planning your storage requirements in the *IBM WebSphere MQ for z/OS Concepts and Planning Guide*. IBM WebSphere MQ for z/OS allows for a maximum page set size of 64 GB, if the SMS storage class that is used has the “extended addressability” function that is activated at page set creation time.

- More about how to determine the use of certain page sets and controlling secondary allocation is explained later within this topic.

- The model JCL contains the following notes on space considerations: customize the RECORDS value for each page set to allocate the space that you require. The page sets are defined with secondary space so that they can be automatically expanded without having to stop the queue manager.
- Page data sets cannot be shared between queue managers. It is suggested that you use the queue manager name (subsystem ID) as a part of the data set name for these data sets.
- The supplied model job creates and initializes five page data sets, which are meant to be used specifically:
 - **PSID00** for object definitions only
 - **PSID01** for system-related messages
 - **PSID02** for important long-lived messages
 - **PSID03** for short-lived messages
 - **PSID04** for miscellaneous messages

The version of CSQ4PAGE shown on the slide is how the member is created. The version that you see in the next lab is slightly modified, so that you enter your given number to substitute the pound (#) sign:

```
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE 'TSM0003.MQ##.PSID00' ERASE CLUSTER
DELETE 'TSM0003.MQ##.PSID01' ERASE CLUSTER
```

CSQ4BSDS 1 of 2: Create the bootstrap and log data sets

```
//CSQ4BSDS JOB
//DEFINE EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
    DELETE (++HLQ++.BSDS01)           ERASE CLUSTER
    DELETE (++HLQ++.LOGCOPY1.DS01)   ERASE CLUSTER
// ...
    DEFINE CLUSTER
        (NAME (++HLQ++.BSDS01) VOLUMES (++VOLBSDS1++) SHAREOPTIONS (2 3) )
-
    DATA
        (NAME (++HLQ++.BSDS01.DATA) RECORDS (60 60)      -
         RECORDSIZE(4089 4089) CONTROLINTERVALSIZE(4096)  -
         FREESPACE(0 20) KEYS(4 0) )                      -
    INDEX
        (NAME (++HLQ++.BSDS01.INDEX)          -
         RECORDS (5 5) CONTROLINTERVALSIZE(1024) )       -
    DEFINE CLUSTER
        (NAME (++HLQ++.LOGCOPY.DS01) LINEAR           -
         VOLUMES (++VOLLOG1A++) SHAREOPTIONS (2 3)      -
         RECORDS (180000) )                           -
    DATA
        (NAME (++HLQ++.LOGCOPY1.DS01.DATA) )           -
// ...
```

© Copyright IBM Corporation 2015

Figure 2-33. CSQ4BSDS 1 of 2: Create the bootstrap and log data sets

WM3021.1

Notes:

The JCL on this visual is taken from `thlqual.SCSQPROC(CSQ4BSDS)`. You need to edit this member to supply names, volumes, and sizes for your installation.

Both bootstrap and log data sets are standard VSAM data sets created with IDCAMS. You can use up to 53 single or pairs of log data sets with your queue manager. The minimum number that is required is two, but in practice you should have at least three, to allow time for each log data set to be copied to archive before it is reused. The IBM WebSphere MQ for z/OS utility program `CSQJU003` is used to set up the defined data sets as IBM WebSphere MQ for z/OS logs. Details about the use of the BSDS and the log data sets are provided by the recovery and restart unit.

The model job provides the following extra notices:

- It is suggested that you define at least three logs and put adjacent logs on separate volumes.
- This sample defines four dual logs with alternating volumes and dual BSDSs.

If you use the Storage Management Subsystem (SMS), you cannot use the extended addressability function of SMS as IBM WebSphere MQ does not support it.

This member is also shown as it is in SCSQPROC. The actual lab member is changed to expedite updates, so only the pound (#) needs to be changed:

```
//SYSPRINT DD SYSOUT=*
//SYSIN      DD *
DELETE (TSM0003.MQ##.BSDS01)    ERASE CLUSTER
DELETE (TSM0003.MQ##.BSDS02)    ERASE CLUSTER
DELETE (TSM0003.MQ##.LOGCOPY1.DS01) ERASE CLUSTER
DELETE (TSM0003.MQ##.LOGCOPY1.DS02) ERASE CLUSTER
```

CSQ4BSDS 2 of 2: Create the bootstrap and log data sets

```
// ....  
//CSQTLOG EXEC PGM=CSQJU003  
//STEPLIB      DD DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR  
//              DD DSN=++THLQUAL++.SCSQAUTH,DISP=SHR  
//SYSUT1       DD DISP=OLD,DSN=++HLQ++.BSDS01  
//SYSUT2       DD DISP=OLD,DSN=++HLQ++.BSDS02  
//SYSPRINT     DD SYSOUT=*,DCB=BLKSIZE=629  
//SYSIN        DD *  
NEWLOG DSNAME=++HLQ++.LOGCOPY1.DS01,COPY1
```

© Copyright IBM Corporation 2015

Figure 2-34. CSQ4BSDS 2 of 2: Create the bootstrap and log data sets

WM3021.1

Notes:

This slide shows continuation of the JCL.

CSQ4ZPRM: Create start parameter module

- A sample procedure is supplied as CSQ4ZPRM in data set HLQ.SCSQPROC

Copy CSQ4ZPRM to a member named #####ZPRM where ##### is the subsystem ID defined in SYS1.PARMLIB(IEFSSNxx) for your queue manager

```
SUBSYS SUBNAME(MQ00) INITRTN(CSQ3INI)
INITPARM('CSQ3EPX,MQ00,M')
```

- After required changes, assemble and link the three macros
- Check that the procedure is linked into the expected library
- The name of this parameter module is supplied to the start queue manager command

© Copyright IBM Corporation 2015

Figure 2-35. CSQ4ZPRM: Create start parameter module

WM3021.1

Notes:

When the queue manager starts, you can specify a system parameter module.

The supplied default is called CSQZPARM and is found in +thlqual+.SCSQAUTH. Its operands are as described in the slide.

You can use the macros that are supplied to change how the queue manager uses the logs and archives, specifies buffer storage, and sets up tracing. If you do so, you must either give the system parameter module a different name, or place it in a **JOBLIB** or **STEPLIB** concatenation ahead of the system-supplied module.

Definitions of the individual operands can be found in the *IBM WebSphere MQ for z/OS Configuring queue managers or z/OS* section of the configuration documentation.

If dual bootstrapping is requested through the parameter module (**TWOBSDS=YES**), the JCL must provide the DD statements for both data sets. The same is true for logs that must be registered in the BSDS at startup.

CSQ4MSTR: Create queue manager started task

- A sample procedure is supplied as CSQ4MSTR in data set HLQ.SCSQPROC

Copy CSQ4MSTR to a member named #####MSTR where ##### is the subsystem ID defined in SYS1.PARMLIB(IEFSSNxx) for your queue manager

```
SUBSYS SUBNAME (MQ00) INITRTN (CSQ3INI)
INITPARM ('CSQ3EPX, MQ00, M')
```

Obtain command prefix string from IEFSSNxx member

- Copy renamed MQ00MSTR to the designated proclib
- IBM MQ is started by using console command

```
/MQ00 START QMGR PARM (MQ00ZPRM)
```

Obtain parameter module name from CSQ4ZPRM, in this case MQ00ZPRM

© Copyright IBM Corporation 2015

Figure 2-36. CSQ4MSTR: Create queue manager started task

WM3021.1

Notes:

Each IBM WebSphere MQ for z/OS subsystem requires a start procedure.

- The procedure name is a concatenation of the IBM WebSphere MQ for z/OS subsystem name and the string “MSTR”.
- The procedure must be placed into a procedure library that is defined to JES.
- IBM WebSphere MQ for z/OS provides a start procedure template as `thlqual.SCSQPROC(CSQ4MSTR)`, which must be tailored to agree with the data sets used during installation.

You define a user ID for the IBM WebSphere MQ for z/OS started task to run under. To define the ID, use either the STARTED class or the started procedure table `ICHRIN03`.

- Make sure that this user ID has READ access to the queue manager product libraries and UPDATE authorization to the system data sets.

CSQ4MSTR

```
//PROCSTEP EXEC PGM=CSQYASCP,REGION=0M,MEMLIMIT=2G
//*****
//STEPLIB    DD DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
//          DD DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
//          DD DSN=++LEQUAL++.SCEERUN,DISP=SHR
//          DD DSN=++DB2QUAL++.SDSNLOAD,DISP=SHR
//* BOOTSTRAP DATA SETS
//BSDS1      DD DSN=++HLQ++.BSDS01,DISP=SHR
//BSDS2      DD DSN=++HLQ++.BSDS02,DISP=SHR
//* SYSTEM INITIALIZATION INPUT FILES
//CSQINP1    DD DSN=++THLQUAL++.SCSQPROC(CSQ4INP1),DISP=SHR
//CSQINP2    DD DSN=++THLQUAL++.SCSQPROC(CSQ4INYS),DISP=SHR
//          DD DSN=++THLQUAL++.SCSQPROC(CSQ4INSX),DISP=SHR
//          ...
//          DD DSN=++THLQUAL++.SCSQPROC(CSQ4DISP),DISP=SHR
//CSQINPT   DD DSN=++THLQUAL++.SCSQPROC(CSQ4INST),DISP=SHR
//          ...
//CSQOUT1    DD SYSOUT=++OUTCLASS++
//CSQOUT2    DD SYSOUT=++OUTCLASS++
//*****
//* PAGE SET DATA SETS - you must have page set 00
//CSQP0000   DD DSN=++HLQ++.PSID00,DISP=SHR
//CSQP0001   DD DSN=++HLQ++.PSID01,DISP=SHR
//          ...
//*****
```

© Copyright IBM Corporation 2015

Figure 2-37. CSQ4MSTR

WM3021.1

Notes:

This JCL shows the data sets used by an IBM WebSphere MQ for z/OS queue manager.

There are three main groups of data sets:

- The bootstrap data sets: No log data sets appear in the JCL. They are referred to inside the BSDS and allocated dynamically.
- The system initialization files: They contain IBM WebSphere MQ commands that are processed on start.
- The page set data sets: Remember that the page sets are where the messages are stored if they are not retrieved from the buffer pools.

All these data sets must exist before the procedure can be run.

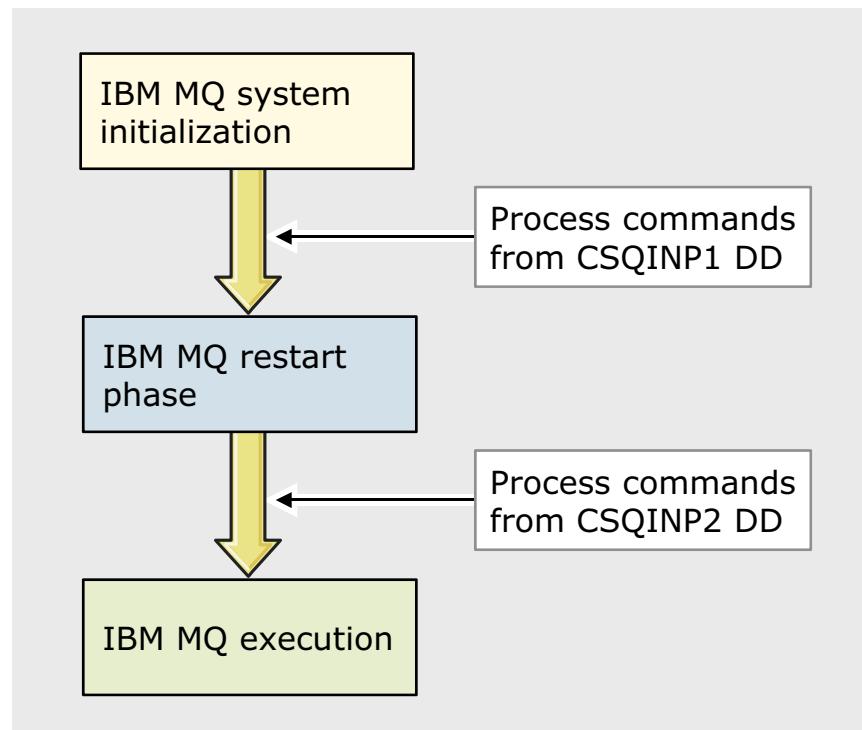
Member **CSQ4MSRR** in the SCSQPROC library provides an alternative start procedure with nine page data sets (PSID00 to PSID08).

**Note**

The CSQINPDT DD statement was added in the IBM MQ V8 queue manager master procedure due to the different publish/subscribe initialization times. This statement is a change for IBM MQ V8.

If publish/subscribe is disabled, the topic environment is removed. If publish/subscribe is re-enabled, the topic environment is re-created by using the CSQINPDT DD.

Initialization and restart overview



© Copyright IBM Corporation 2015

Figure 2-38. Initialization and restart overview

WM3021.1

Notes:

The data sets (usually members of a library) referenced by DD statements **CSQINP1** and **CSQINP2** in the start procedure contain IBM WebSphere MQ commands that are processed at different times during IBM WebSphere MQ for z/OS startup.

The appropriate results are written to **CSQOUT1** and **CSQOUT2**, which might be data sets or JES sysout classes.

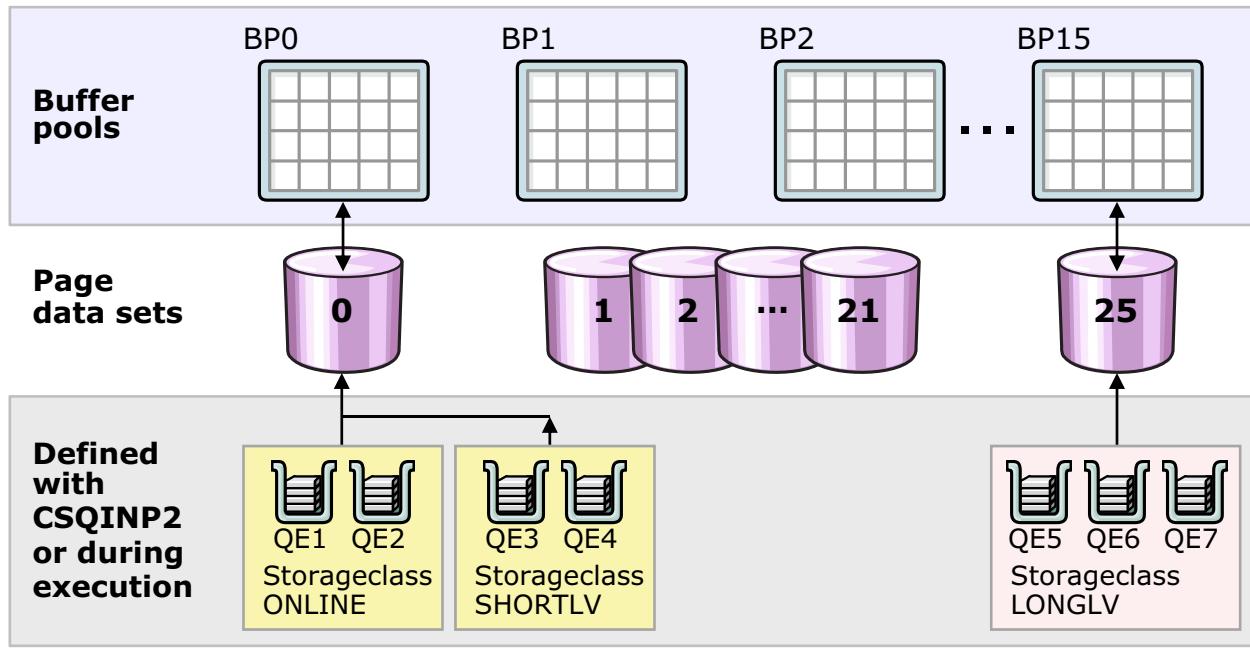
- Use of **CSQINP1** is mandatory.
- Use of **CSQINP2** is optional, except for the first start of IBM WebSphere MQ for z/OS because use of CSQINP2 is the only way to define system objects that are needed for execution.

The **thlqual.SCSQPROC** product library contains templates for this use as well.

CSQ4INP1 initialization commands: Define buffer pools and page sets

For example:

```
DEFINE BUFFPOOL( 0 ) BUFFERS (10000 )
DEFINE PSID ( 01 ) BUFFPOOL( 0 ) EXPAND(USER/SYSTEM/NONE)
```



© Copyright IBM Corporation 2015

Figure 2-39. CSQ4INP1 initialization commands: Define buffer pools and page sets

WM3021.1

Notes:

Here is what member <HLQ>.SCSQPROC(CSQ4INP1) contains:

```
DEFINE BUFFPOOL( 0 ) BUFFERS( 50000 )
DEFINE BUFFPOOL( 1 ) BUFFERS( 20000 )
DEFINE BUFFPOOL( 2 ) BUFFERS( 50000 )
DEFINE BUFFPOOL( 3 ) BUFFERS( 20000 )
Define page sets and associate each page set with a buffer pool.
DEFINE PSID( 00 ) BUFFPOOL( 0 ) EXPAND(USER)
DEFINE PSID( 01 ) BUFFPOOL( 0 ) EXPAND(USER)
DEFINE PSID( 02 ) BUFFPOOL( 1 ) EXPAND(USER)
DEFINE PSID( 03 ) BUFFPOOL( 2 ) EXPAND(USER)
DEFINE PSID( 04 ) BUFFPOOL( 3 ) EXPAND(USER)
```

Buffer pools hold message contents in main storage. They are created at IBM WebSphere MQ for z/OS initialization.

- The maximum number of buffer pools depends on the IBM MQ version and the OPMODE parameter setting.
 - IBM MQ V7.5 and below: Up to 16 buffer pools might be defined. OPMODE is not applicable.
 - IBM MQ V8 and above: Up to 100 buffer pools might be defined when the OPMODE setting is (NEWFUNC,800) and “ABOVE” is specified in the buffer pool definition.
- Buffer pool size is specified as sets of 4-K buffers. It is up to your installation to define how large the buffer pools should be.
- Each of the page sets your start JCL references is “connected” to one buffer pool. Multiple page sets might be allocated to a single buffer pool. This definition is used by IBM WebSphere MQ for z/OS to define what page sets are used for this run. Up to 100 page sets (00 – 99) can be used. PSID00 is always used for holding IBM WebSphere MQ object definitions, and applications should not use it.
- Buffer pools control whether and how page set expansion is to take place:
 - If no secondary extent size is defined, page set expansion fails.
 - Page sets can grow to 75,000 times the original size, and this mechanism works even if a page set is defined with either no or zero secondary extent size.
- To explain the complete chain of definitions and their effect on the use of buffers and page sets, storage class and queues are displayed in the visual.

However, **STGCLASS** and **QUEUE** are recoverable objects that cannot be defined at this CSQINP1 stage.

CSQINP2: Initialization command members

- CSQ4INSG: System objects definitions
- CSQ4INSX: System objects definitions
- CSQ4INSS: Customize and include this member if you are using queue-sharing groups
- CSQ4INSJ: Use Java Message Service (JMS) to customize and include this member if you are using publish/subscribe
- CSQ4INSR: You might need to customize and include this member if you are using WebSphere Application Server or a broker
- CSQ4INYC: Clustering definitions
- CSQ4INYD: Distributed queuing definitions
- CSQ4INYG: General definitions
- CSQ4INYR: Storage class definitions that use multiple page sets for the major classes of message
- CSQ4INYS: Storage class definitions that use one page set for each class of message

© Copyright IBM Corporation 2015

Figure 2-40. CSQINP2: Initialization command members

WM3021.1

Notes:

Data sets referred to by CSQINP2 provide:

- Definitions for recoverable objects, such as queues and storage classes
- Queue manager attributes
- Any IBM WebSphere MQ commands that can be processed at this late point during initialization

The use of CSQINP2 is optional. When IBM WebSphere MQ is started for the first time, most of the objects that are defined in the template members must be defined by using this mechanism because they are required for operating IBM WebSphere MQ for z/OS.

- Most of the objects that these commands define are of the fixed name type, so no tailoring of the provided command collection is needed. These members can be referred to directly by the start JCL.

However, some members are set up in the same way as the sample JCL, with variable names included, as indicated by the visual. These members can be used only as templates, and a tailored version must be configured for real use.

CSQ4CHIN: Create channel initiator started task

- A sample procedure is supplied as CSQ4CHIN in data set HLQ.SCSQPROC

Copy CSQ4CHIN to a member named #####CHIN where ##### is the subsystem ID defined in SYS1.PARMLIB(IEFSSNxx) for your queue manager

```
SUBSYS SUBNAME(MQ00) INITRTN(CSQ3INI)
INITPARM('CSQ3EPX,MQ00,M')
```

Obtain command prefix string from IEFSSNxx member

- Copy renamed MQ00CHIN to the designated proclib
- IBM MQ is started by using console command

/MQ00 START CHINIT

© Copyright IBM Corporation 2015

Figure 2-41. CSQ4CHIN: Create channel initiator started task

WM3021.1

Notes:

The channel initiator is the second started task to support the queue manager. The functions that the channel initiator controls were discussed earlier in this unit.

After the channel initiator successfully starts, it can be set up to automatically start when the MQ##MSTR region starts. The lab documents the automatic startup.

Queue manager start and verification

```
/MQ## START QMGR PARM(MQ##ZPRM)
```

- Identify the parameter module used
- Identify the early code level
- Print all the parameter values from the active ZPARM
- Identify the current log data sets, the place where logging continues, and the addressability limits for the queue manager
- Display the security options used
- Display the number and sizes of buffers used
- Display the active page sets and their association to buffers
- Display restart activities per page set
- Display the results to the following commands that are issued automatically:
 - DISPLAY THREAD TYPE(INDOUBT)
 - DISPLAY SYSTEM
 - DISPLAY LOG
 - DISPLAY ARCHIVE

© Copyright IBM Corporation 2015

Figure 2-42. Queue manager start and verification

WM3021.1

Notes:

By using these messages, you can identify the current settings of the options that control the behavior of the queue manager.



Set up the operations and control panels

- To start the IBM WebSphere MQ for z/OS operation and control panels, ISPF-related IBM WebSphere MQ libraries must be added to a TSO user's data set allocations
- Allocation of IBM WebSphere MQ libraries can be permanent at user logon or dynamic when the panels are used
- IBM WebSphere MQ ISPF panels can be started:
 - Either by running the EXEC provided as HLQ.SCSQEXEC(CSQOREXX) from the TSO command processor panel
 - Or from an ISPF menu option that is set up as: CMD(%CSQOREXX) NEWAPPL(CSQO) PASSLIB

```

IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action . . . . . 1 0. List with filter 4. Manage
                   1. List or Display 5. Perform
                   2. Define like   6. Start
                   3. Alter        7. Stop
                   8. Command

Object type . . . . . QLOCAL +
. . . . .

```

© Copyright IBM Corporation 2015

Figure 2-43. Set up the operations and control panels

WM3021.1

Notes:

Channel initiator start and verification

/MQ## START CHINIT

```
CSQX011I MQ03 CSQXGIP Client Attachment available
CSQX151I MQ03 CSQXSSLI 0 SSL server subtasks started, 0 failed
CSQX410I MQ03 CSQXREPO Repository manager started
CSQX022I MQ03 CSQXSUPR Channel initiator initialization complete
CSQT806I MQ03 CSQXFCTL Queued Pub/Sub Daemon started
CSQX500I MQ03 CSQXRCTL Channel WMZOSCLS.TSM0000 started
CSQX004I MQ03 CSQXSPRM Channel initiator is using 30 MB of local
632 storage, 1678 MB are free
CSQX251I MQ03 CSQXSTRL Listener started, TRPTYPE=TCP INDISP=QMGR
CSQX023I MQ03 CSQXLSTT Listener started, 634
port 1603 address *,
CSQX500I MQ03 CSQXRESP Channel SYSTEM.ADMIN.SVRCONN started
CSQX457I MQ03 CSQXREPO Repository available,
cluster WMZOSCLS,
channel WMZOSCLS.TSM0000,
sender MQ00.C235C8D81EB9EC4A
```

© Copyright IBM Corporation 2015

Figure 2-44. Channel initiator start and verification

WM3021.1

Notes:

IBM MQ V8 Channel initiator error

```
+CSQX213E MQ00 CSQXSUPR Communications error, 054  
function 'CSFPPRF' RC=12 reason=00000000
```

- Error occurs due to client password obfuscation capability new in V8
- Error occurs even if SSL is not enabled
- To correct the error enable ICSF

© Copyright IBM Corporation 2015

Figure 2-45. IBM MQ V8 Channel initiator error

WM3021.1

Notes:

A new error might surface in the IBM MQ V8 z/OS channel initiator started task.

This error occurs because IBM MQ needs the ability to read an incoming password, even if SSL is not enabled. The password is obfuscated at the client side, and needs to be uncovered on the z/OS side. For this purpose z/OS needs to have ICSF started. There was no need to provide access to the cryptographic hardware.

If you see this error in your shop, starting ICSF should correct it.

The password obfuscation is discussed in the security unit.

Include the IBM MQ dump formatting member for IPCS

- Copy the data set HLQ.SCSQPROC(CSQ7IPCS) to SYS1.PARMLIB. You should not need to edit this data set.
- HLQ.SCSQPROC(CSQ7IPCS) can be copied into any library in the IPCSPARM definition if the TSO procedure for IPCS is customized
- Include the library HLQ.SCSQPNLA in the ISPPLIB concatenation.
- To make the dump formatting programs available to a TSO session or IPCS job, also:
 - Include the library HLQ.SCSQAUTH to the STEPLIB concatenation or
 - Activate it using the TSO TSOLIB command; this step must be completed regardless if HLQ.SCSQAUTH is already in the LPA

```
CSQ7IPCS member:
DIALOG NAME(CSQMAIN) PARM('PANEL(CSQ7MAIN') +
  ABSTRACT('WebSphere MQ dump formatter panel interface')
  EXIT EP(CSQWD530) VERB(CSQWDMF) +
  ABSTRACT('WebSphere MQ dump formatter ')
  DATA STRUCTURE(CSQ0) FORMAT(CSQ7CBF)
  DATA STRUCTURE(CSQ1) FORMAT(CSQ7CBF)
```

© Copyright IBM Corporation 2015

Figure 2-46. Include the IBM MQ dump formatting member for IPCS

WM3021.1

Notes:

Batch and TSO application issues

- IBM MQ for z/OS is ready to serve batch and TSO applications after installation and setup
 - Application programs must be link-edited with the MQI Batch and TSO program stub CSQBSTUB
 - IBM MQ libraries **HLQ.SCSQAUTH** and **HLQ.SCSQANLE** must be accessible to batch or TSO programs through the STEPLIB concatenation (or through the z/OS link list)
- A default queue manager can be specified by using the **CSQBDEFV** load module structure
- IBM MQ automatically registers to z/OS Resource Recovery Service (RRS) at startup
- IBM MQ batch and TSO applications can use the RRS API program stubs for unit of work control

© Copyright IBM Corporation 2015

Figure 2-47. Batch and TSO application issues

WM3021.1

Notes:

Batch or TSO programs must be link-edited with the IBM WebSphere MQ-supplied program stub.

Using the IBM WebSphere MQ macro **CSQBDEF**, a load module that is named **CSQBDEFV** must be created and placed into one of the batch jobs **STEPLIB** libraries.

- Resource Recovery Service (RRS) provides unit of work control for batch and TSO-based applications that access different resource managers' data, such as IBM WebSphere MQ and DB2.
- IBM WebSphere MQ for z/OS automatically registers with RRS on startup.
- A batch or TSO application that **GETs** messages from an IBM WebSphere MQ queue and updates a DB2 table or a VSAM data set, for example, can coordinate these data changes by using RRS.
- RRS takes responsibility to ensure that both IBM WebSphere MQ and DB2 changes are committed, or that both are backed out, if the application or one of the resource managers fails during commit processing.

Applications that need to use the RRS service must use the `RRS*` calls and must be linked with an extra program stub named CSQBRSTB.

Therefore, these application programs do not need to be changed; they need to be relinked with this stub to use the RRS.

Implementing 8-byte RBA

- Process to follow for 8-byte RBA varies by scenario
 - Is the queue manager new, never started, or is it a current queue manager?
 - Is it using queue-sharing groups?
- All scenarios require a reassembly of the ZPARM module
OPMODE=(NEWFUNC,800) specified in the CSQ6SYSP macro



After a queue manager successfully starts with a version 2 converted BSDS, no attempt should be made to start it with the old version 1 BSDS

- Queue-sharing groups must convert the BSDS for all queue managers to use 8-byte RBA
 - Refer to the IBM MQ Knowledge Center for details
- The OPMODE parameter consists of two parts
 - The mode, COMPAT or NEWFUNC
 - The release level, same as the command level, called verification level in OPMODE
- After the MODE component of OPMODE is changed to NEWFUNC, you cannot revert to an earlier version level

© Copyright IBM Corporation 2015

Figure 2-48. Implementing eight-byte RBA

WM3021.1

Notes:

Implementing 8-byte RBA steps (1 of 3)

Existing BSDS data sets:

MQ00.BSDS01
MQ00.BSDS01.DATA
MQ00.BSDS01.INDEX
MQ00.BSDS02

1. Create a set of BSDS files with different names:

```
//ALLOC      EXEC PGM=IDCAMS,REGION=4M
...
...
...
DEFINE CLUSTER
        (NAME (MQ00.NEW.BSDS01) ...
        ...
        ...
        DATA
        (NAME (MQ00.NEW.BSDS01.DATA) ... ... ... ...
```

2. Stop the queue manager
3. If not previously done, update the **CSQ6SYSP OPMODE** attribute and reassemble the ZPARMS:

```
//SYSIN      DD *
        CSQ6SYSP
        ...
        ...
        ...
        OPMODE=(NEWFUNC,800),           V8 ENABLED OPMODE    X
                                         X
```

© Copyright IBM Corporation 2015

Figure 2-49. Implementing 8-byte RBA steps (1 of 3)

WM3021.1

Notes:

Implementing 8-byte RBA steps (2 of 3)

4. Run the BSDS conversion utility: JCL at CSQ4BCNV

```
//CONVERT EXEC PGM=CSQJUCNV,REGION=32M,  
//*          PARM= ('INQSG,++QSGNAME++,++DSGNAME++,++DB2SSID++')  
//          PARM= ('NOQSG')  
//STEPLIB  DD DSN=MQ8000.SCSQAUTH,DISP=SHR  
//          DD DSN=MQ8000.SCSQANLE,DISP=SHR  
//*          DD DSN=++DB2QUAL++.SDSNLOAD,DISP=SHR  
//SYSPRINT DD SYSOUT=*  
//SYSUT1   DD DSN=MQ00.BSDS01,DISP=SHR  
//SYSUT2   DD DSN=MQ00.BSDS02,DISP=SHR  
//SYSUT3   DD DSN=MQ00.NEW.BSDS01,DISP=OLD  
//SYSUT4   DD DSN=MQ00.NEW.BSDS02,DISP=OLD  
... ... ... ... ...
```

© Copyright IBM Corporation 2015

Figure 2-50. Implementing 8-byte RBA steps (2 of 3)

WM3021.1

Notes:

Implementing 8-byte RBA steps (3 of 3)

5. Rename the current BSDS to an “OLD” BSDS name:

```
//IDCAMS EXEC PGM=IDCAMS,REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
ALTER 'MQ00.BSDS01' NEWNAME('MQ00.OLD.BSDS01')
ALTER 'MQ00.BSDS01.DATA' NEWNAME('MQ00.OLD.BSDS01.DATA')
ALTER 'MQ00.BSDS01.INDEX' NEWNAME('MQ00.OLD.BSDS01.INDEX')
...
```

6. Rename the “NEW” BSDS to be the current BSDS:

```
...
ALTER 'MQ00.NEW.BSDS01' NEWNAME('MQ00.BSDS01')
ALTER 'MQ00.NEW.BSDS01.DATA' NEWNAME('MQ00.BSDS01.DATA')
ALTER 'MQ00.NEW.BSDS01.INDEX' NEWNAME('MQ00.BSDS01.INDEX')
...
```

7. Start the queue manager with the newly assembled ZPARM module:

/MQ00 START QMGR PARM(MMQ00ZPRM)

8. Check queue manager startup **CSQJ034I RBA RANGE FFFFFFFFFFFFFF**

© Copyright IBM Corporation 2015

Figure 2-51. Implementing 8-byte RBA steps (3 of 3)

WM3021.1

Notes:

Unit summary

- Identify the key components of an IBM MQ z/OS queue manager
- List the responsibilities of the IBM MQ z/OS system administrator
- Identify z/OS system configuration tasks that are required before a queue manager can be configured
- Summarize the installation process for IBM WebSphere MQ for z/OS
- Describe the process, data sets, and JCL used to configure a z/OS queue manager
- Summarize eight-byte relative byte address considerations
- Complete and start a z/OS IBM MQ V8 queue manager

© Copyright IBM Corporation 2015

Figure 2-52. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. True or false: The installer needs ALTER access to the installation libraries.
2. True or false: You specify tailoring details only one time in CQ8SASYJ.
3. The start procedure MQ##MSTR points to:
 - a. The BSDS
 - b. The queues
 - c. The logs
 - d. The page data sets
 - e. The system initialization files
 - f. All of the above

© Copyright IBM Corporation 2015

Figure 2-53. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint questions (2 of 2)

4. To use 64-bit buffer pools, the administrator needs to do which of the following steps?
 - a. Convert the BSDS to the new format and restart the queue manager
 - b. Set the OPMODE parameter to OPMODE(NEWFUNC,800), reassemble the parameter module, and restart the queue manager
 - c. Change OPMODE(NEWFUNC,800), reassemble the parameter module, start the queue manager, and alter or create buffers with the attribute LOCATION(ABOVE)
 - d. Confirm adequate memory availability. Change OPMODE(NEWFUNC,800), reassemble the parameter module, restart the queue manager, and alter or create buffers with the attribute LOCATION(ABOVE)

© Copyright IBM Corporation 2015

Figure 2-54. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

4.

Checkpoint answers (1 of 2)

1. True or false: The installer needs ALTER access to the installation libraries.

Answer: True

2. True or false: You specify tailoring details only one time in CQ8SASYJ.

Answer: True

3. The start procedure MQ##MSTR points to:

- a. The BSDS
- b. The queues
- c. The logs
- d. The page data sets
- e. The system initialization files
- f. All of the above

Answer: a, d, e

© Copyright IBM Corporation 2015

Figure 2-55. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

4. To use 64-bit buffer pools, the administrator needs to do the following steps:
 - a. Convert the BSDS to the new format and restart the queue manager
 - b. Set the OPMODE parameter to OPMODE(NEWFUNC,800), reassemble the parameter module, and restart the queue manager
 - c. Change OPMODE(NEWFUNC,800), reassemble the parameter module, start the queue manager, and alter or create buffers with the attribute LOCATION(ABOVE)
 - d. Confirm adequate memory availability. Change OPMODE(NEWFUNC,800), reassemble the parameter module, restart the queue manager, and alter or create buffers with the attribute LOCATION(ABOVE)

Answer: The best answer is d, although c also works

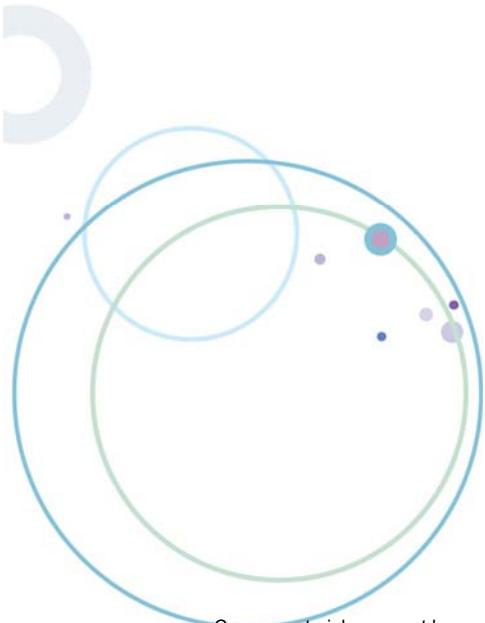
© Copyright IBM Corporation 2015

Figure 2-56. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 1



Configuring an IBM MQ for z/OS queue manager

© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 2-57. Exercise 1

WM3021.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Describe the z/OS setup for a queue manager
- Start and stop a queue manager master and channel initiator task
- Describe and use the input command files
- Explain some of the messages that a z/OS queue manager issues at startup
- Implement 8-byte RBA; change the OPMODE parameter and convert BSDS to version 2 format

© Copyright IBM Corporation 2015

Figure 2-58. Exercise objectives

WM3021.1

Notes:

Unit 3. IBM MQ for z/OS administrative interface options

What this unit is about

This unit examines the facilities that are available to administer IBM MQ on z/OS and demonstrates these capabilities with queue definitions.

What you should be able to do

After completing this unit, you should be able to:

- Explain the differences between the various types of queues
- Contrast the effect of selected queue definition attributes and development options on the handling of messages
- Demonstrate use of the Interactive System Productivity Facility (ISPF) panel interface to change queue manager and queue attributes
- Describe how to issue IBM MQ script commands (MQSC) by using the IBM MQ for z/OS CSQUTIL or the command line to create queues, including local, model, and alias queues
- Describe and use the display and monitoring functions that allow inquiry on queue-related application activities
- Demonstrate how to use supplied programs to put and retrieve messages in queues

Unit objectives

- Explain the differences between the various types of queues
- Contrast the effect of selected queue definition attributes and development options on the handling of messages
- Demonstrate use of the Interactive System Productivity Facility (ISPF) panel interface to change queue manager and queue attributes
- Describe how to issue IBM MQ script commands (MQSC) by using the IBM MQ for z/OS CSQUTIL or the command line to create queues, including local, model, and alias queues
- Describe and use the display and monitoring functions that allow inquiry on queue-related application activities
- Demonstrate how to use supplied programs to put and retrieve messages in queues

© Copyright IBM Corporation 2015

Figure 3-1. Unit objectives

WM3021.1

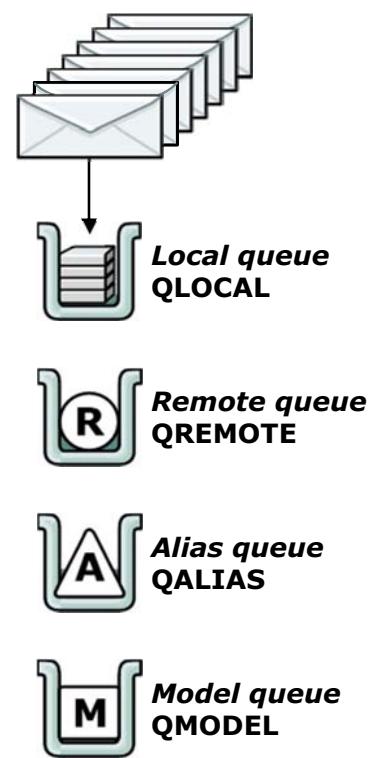
Notes:

Queues revisited

- A queue is a defined destination for messages
- Four types of queues can be created: QLOCAL, QREMOTE, QALIAS, QMODEL
- Some local queues are designated for special purposes in a queue manager
- Remote queues or QREMOTE are pointers to local queues in a remote queue manager
- An alias queue or QALIAS is a pointer to a local queue or a locally owned remote queue
- A model queue or QMODEL is a template to create a dynamic local queue



Only local queues that are defined as a QLOCAL queue type hold messages



© Copyright IBM Corporation 2015

Figure 3-2. Queues revisited

WM3021.1

Notes:

To get started with administrative options, you revisit queue use and definitions.

QLOCAL: Local queues

- If defined with only the queue name as input, takes defaults from SYSTEM.DEFAULT.LOCAL.QUEUE
- Some attributes in a queue display are populated with a queue manager, such as CRDATE and CURDEPTH
- Standards must be set for certain attributes such as DEFPSIST and DEFBIND
- Monitoring attributes need other actions to be taken in the queue manager
 - MONQ requires the queue manager to also have its MONQ attribute set
 - For event detection such as QDEPTHHI, events must be configured



Partial selection of QLOCAL attributes is listed

```

QUEUE (ORDER.IN)
TYPE (QLOCAL)
CRDATE (2014-08-26)
CURDEPTH (1)
MONQ (QMGR)
DEFBIND (OPEN)
DEFPSIST (YES)
OPPROCS (0)
IPPROCS (0)
DEFTYPE (PREDEFINED)
QDEPTHHI (80)
TRIGDATA ( )
TRIGDPTH (1)
TRIGTYPE (FIRST)
USAGE (NORMAL)

```

© Copyright IBM Corporation 2015

Figure 3-3. QLOCAL: Local queues

WM3021.1

Notes:

Queues have numerous attributes with established defaults if not specified in the definition. Some of the attributes are:

- *Persistence* and *priority* are attributes of single *messages* that can be determined from the originating application in the MQMD of the messages. The queue definition provides the default for these attributes if **PERSISTENCE_AS_Q_DEFINED** is used by the application program.
- Applications might access messages by specifying a *message ID* or a *correlation ID*, or both, in the MQMD on **MQGET**. Using the message ID and correlation ID is common for request/reply applications where it is important to relate a response to a previous request message. If the message ID or correlation ID is not used in the MQGET, then the queue is read in first-in first-out order, or FIFO.
- *Maximum queue depth* can be coded to prevent a page data set from filling up, which would severely affect other applications.
- *Maximum message length* should be coded if it is known, to prevent erroneous messages from being put on the queue.

Selected queue attributes and considerations

- CURDEPTH displays the number of messages currently on the queue
- MONQ determines whether statistics are collected for the queue
 - QMGR, OFF, LOW, MEDIUM, HIGH
 - Queue manager MONQ attribute must be enabled
 - DIS QSTATUS provides two formats helpful for troubleshooting or determining who is holding a queue handle
 - Test or production queue manager
- Persistence
 - Do these messages really need to be persistent?
 - Application MQPUT option will take precedence over queue definition attribute
- DEFBIND: Is the cluster workload expected to be balanced?
- SHARE or NOSHARE: Determines whether one or more application instances can get messages from the queue
- STGCLASS: Beware of inadvertent use of the IMS storage class for non-IMS queues



Attributes that are specified in queue definitions can be influenced by options in application code or queue manager properties.

© Copyright IBM Corporation 2015

Figure 3-4. Selected queue attributes and considerations

WM3021.1

Notes:

Looking at other attributes:

- CURDEPTH is one of the most frequently used attributes as it shows the number of messages in a queue, a good attribute to know when trying to determine “where is the message?”
- To display individual queue statistics such as the date and time that a message was last put or taken from a queue, it is necessary to check two attributes:
 - The queue manager MONQ attribute: If OFF, no information is provided.
 - The queue attribute by the same name.
- It is necessary to select one of the possible values that provide for collecting the data in both places, the queue manager and the queue.
- Resist the urge to make all messages persistent, and set standards for which messages or applications can have persistent messages.
- You look at DEFBIND in the *IBM MQ cluster basics* section, and SHARE or NOSHARE in a later slide.

- While you learned that STGCLASS helps assign queue definitions to specific page sets, the STGCLASS attribute is also used to identify queues that are going to IMS.

MQMD: Partial message descriptor attributes from MQPUT

IBM WebSphere MQ (1)

COMMAND ==>

```
Queue Manager      : MQ00
Queue             : ORDER.IN
Forward to Q Mgr : MQ00
Forward to Queue :
```

Message Content :

Message Descriptor

StrucId	:	'MD'
Version	:	000000001
Report	:	000000000
MsgType	:	000000008
Expiry	:	-00000001
Feedback	:	000000000
Encoding	:	000000546
CodedCharSetId	:	000001208
Format	:	'MQSTR'
Priority	:	000000000

(2)

Persistence	:	000000000
MsgId	:	'C3E2D840D4D
.....	:	'
CorrelId	:	'000000000000'.....'
BackoutCount	:	000000001
ReplyToQ	:	' '
ReplyToQMgr	:	'MQ00'
UserIdentifier	:	'TSM0000'
AccountingToken	:	'1A0FD4D8F0F0C3C8C9D5F1F...'
ApplIdentityData	:	' '
PutApplType	:	000000028
PutApplName	:	'WebSphere MQ Client for Java'
PutDate	:	'20140818'
PutTime	:	'18114652'
ApplOriginData	:	' '

© Copyright IBM Corporation 2015

Figure 3-5. MQMD: Partial message descriptor attributes from MQPUT

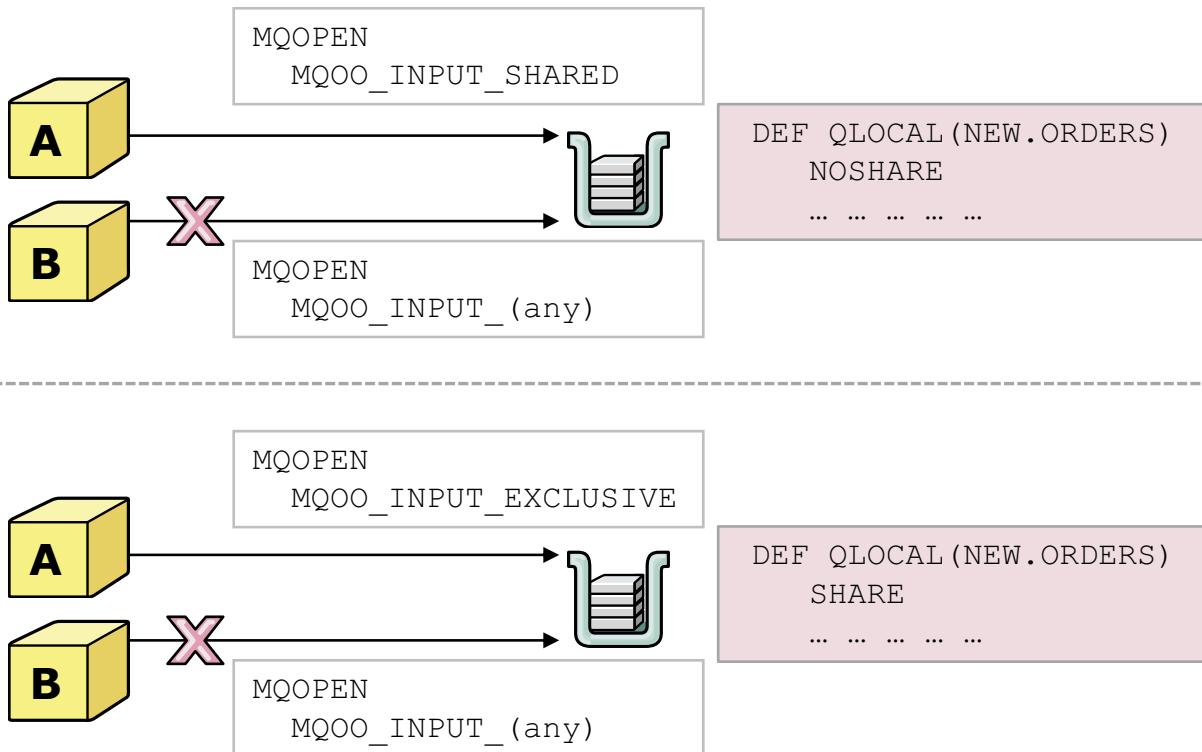
WM3021.1

Notes:

This screen shows the MQMD from a message that the Message Handler sample program prints. This program is accessible by using ISPF option H in the class lab environment.

If you revisit the queue attributes and compare to the fields in the MQMD, you see how many of the attributes are mapped to the MQMD; however, the application can also change the field values.

Application and definition attributes: Exclusive and SHARED



© Copyright IBM Corporation 2015

Figure 3-6. Application and definition attributes: Exclusive and SHARED

WM3021.1

Notes:

When an application opens a queue for input (**MQGET**), it determines through its **OPEN_OPTIONS** whether it wants to access the queue exclusively or share access.

- As noted from the second definition panel, the queue defines whether it allows shared access or not.
- If shared access is allowed, multiple applications might **MQGET** (read) and process messages from that queue at the same time; thus the application design must provide for parallel processing.
- If it is not, all “**OPEN** for input” requests are turned to **EXCLUSIVE** ones, and subsequent **MQOPENS** by other application requests would be rejected.

If an application successfully opens a queue with the **EXCLUSIVE** option, subsequent **MQOPENS** for input are rejected, both for shared and exclusive access.

QREMOTE

- The remote queue points to the name of the transmission queue (XMITQ) that is to hold the messages
- The **RNAME** attribute of the **QREMOTE** denotes the target local queue in the remote queue manager
- The **RQMNAME** attribute is usually the name of the remote queue manager, which might be a different queue manager from what is indicated in the transmission queue
- The **QREMOTE** object is also used to define queue manager aliases

```
CSQM406I +MQ00
QUEUE (REQ.OUT)
TYPE (QREMOTE)
QSGDISP (QMGR)
CLUSTER( )
CLUSNL( )
DESCR( )
PUT (ENABLED)
DEFPRTY (0)
DEFPERSIST (NO)
DEFPRESP (SYNC)
RNAME (SALE.IN)
RQMNAME (MQM3)
XMITQ (MQM3)
DEFBIND (OPEN)
CLWLRANK (0)
CLWLPRTY (0)
ALTDATE (2014-08-19)
ALTTIME (09.17.15)
```

© Copyright IBM Corporation 2015

Figure 3-7. QREMOTE

WM3021.1

Notes:

The QREMOTE was explained in an earlier unit. You revisit the key parts of a QREMOTE definition and its relationships:

- The remote queue does not hold messages. Messages go to the local transmit queue that is associated with this remote queue, in this case, MQM3 in the XMITQ attribute.
- The RQMNAME attribute holds the name of the queue manager to which this remote queue points; in this case, the value is also MQM3. Often, but not always, the transmit queue has the same name as the remote queue manager.
- Finally, the remote queue also points to the intended target queue in the remote queue manager. This queue is the SALE.IN queue, and the information is held in the RNAME attribute.

The QREMOTE object definition is used for purposes other than defining a remote queue. A queue manager alias uses the QREMOTE object for its definition. Queue manager aliases are covered in the *Distributed queuing* unit.



QALIAS

- Provides more decoupling
- Points to a queue or a topic
- **TARGETTYPE** attribute specific to **QALIAS**
- When **TARGETTYPE** is a queue
 - **QREMOTE**
 - **QLOCAL**
- **TARGETTYPE** can also be a publish/subscribe topic

QUEUE (NEW.ORDERS)

TYPE (QALIAS)
 QSGDISP (QMGR)
 CLUSTER ()
 CLUSNL ()

TARGTYPE (QUEUE)

DESCR ()
 PUT (ENABLED)
 DEFPRTY (0)
 DEFPSIST (NO)
 GET (ENABLED)
 DEFREADA (NO)
 DEFPRESP (SYNC)

TARGET (REQ.OUT)

PROPCTL (COMPAT)

DEFBIND (OPEN)

CLWLRank (0)
 CLWLPRTY (0)

© Copyright IBM Corporation 2015

Figure 3-8. QALIAS

WM3021.1

Notes:

ALIAS queues provide a means to map queue names that are used by applications to point to real queues that might be named according to an enterprise-naming convention.

- The queue alias definition might be, for example:

```
DEFINE QALIAS(MY.Q.ALIAS)
DESCR( 'A DEMO ALIAS QUEUE' )
PUT(ENABLED)
DEFPRTY(1)
DEFPSIST(YES)
TARGQ( 'MY.REAL.QUEUE' )
GET(ENABLED)
```

- The target queue must be a
 - Local queue.
 - Remote queue.
 - A topic. You look at topics in Unit 7.

- The target queue cannot be another alias queue or a model queue.

Model and dynamic queues

- A model queue definition specifies a template to create the dynamic queue
 - An application uses the MQOD structure of the MQOPEN call to create the dynamic queue

```
Queue name . . . . . . . . . . . . AMQ.MQEXPLORER.1199353407
Disposition . . . . . . . . . . . : QMGR      MQ00
Description . . . . . . . . . . . : MQ Explorer reply-to queue
... . . . . .
Dynamic queue type . . . . . : T  N=Non-dynamic (Predefined),
T=Temporary, P=Permanent, S=Shared
```

© Copyright IBM Corporation 2015

Figure 3-9. Model and dynamic queues

WM3021.1

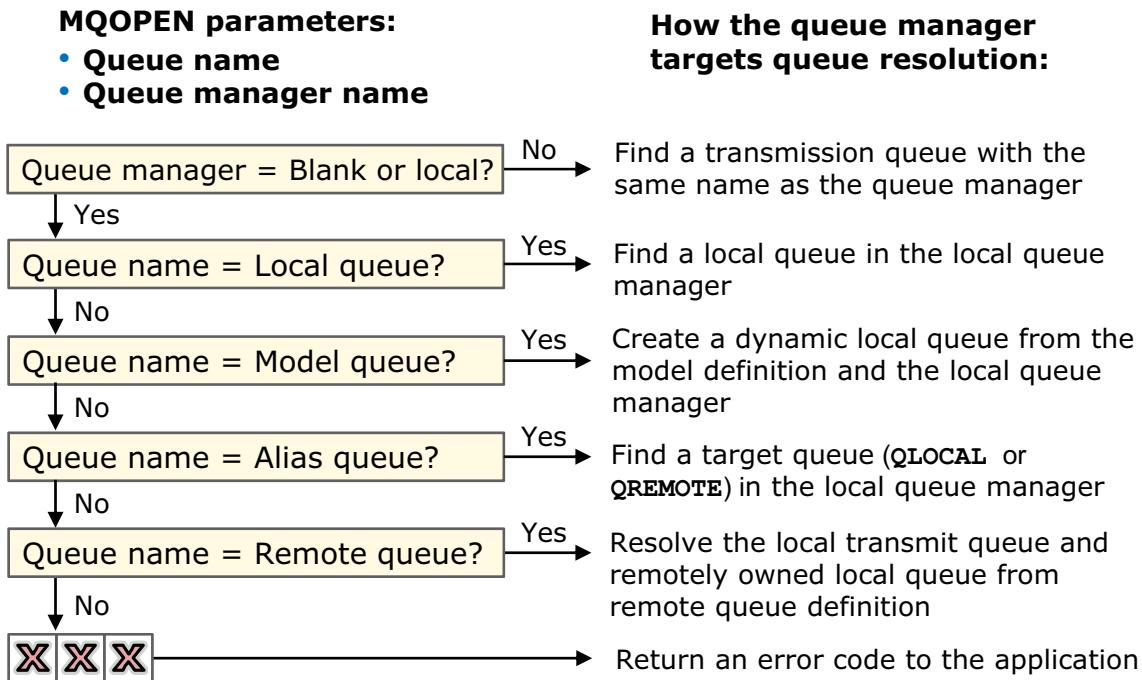
Notes:

Model queues are part of the dynamic queues concept.

- Model queues are used as *definition templates* for queues that are created *dynamically* as part of the processing of an **MQOPEN** call that an application program issues.
 - When an application issues an **MQOPEN** against a **MODEL** queue, IBM WebSphere MQ dynamically creates a local queue, which is based on the **QMODEL** definition.
 - The object handle that is returned to the application points to this new dynamically created queue. Additionally, the name of the new queue is returned.
 - The application can now use this handle for **PUT**ting messages, or it can retrieve the **Objectname** to use it as a **Reply_To_Queue_Name** in a request message.
 - The definition set for a model queue is the same as for a local queue, with one extra parameter for the *definition type*, which can be either *permanent dynamic* or *temporary dynamic*.
 - **DEFTYPE(PERMDYN)** means that the dynamically created queue is handled just as a local queue after creation, but an application can delete it with the **DELETE** or **DELETE_PURGE** option on **MQCLOSE**.

- **DEFTYPE(TERMDYN)** means that the queue is deleted *automatically* when the application that caused its creation (by **MQOPEN**) **MQCLOSE**s it.
- Only predefined or permanent dynamic queues can host persistent messages, so **DEFPSIST(YES)** is not allowed for temporary dynamic queues.
- Dynamic queues are typically used for **reply-to queues**.

Queue name resolution



© Copyright IBM Corporation 2015

Figure 3-10. Queue name resolution

WM3021.1

Notes:

Many variables determine how queue name resolution is done by the queue manager. This diagram summarizes some of the key ideas.

- The queue manager name that is passed in the MQOPEN call, if present, is checked. If the name of this queue manager is not omitted, and does not match the name of the local queue manager, queue resolution looks for a locally defined transmission queue: one with a name that matches the name of the queue manager that is used in the MQOPEN call.
- If the queue manager name was blank, queue resolution checks whether a local queue matches the name of the queue name that is used in the MQOPEN. If such a local queue is found, queue resolution ends, placing the message in the local queue.
- The name of the queue that is used in the MQOPEN should match the name of a model queue, or the name of an alias queue, in that order. If not, then queue name resolution looks for the name of a remote queue that matches the name that is used in the MQOPEN.
- If there is no match, an error is returned to the application, usually a return 2085, unknown object name.

Queue name resolution is revisited in the *Distributed queuing* unit.

Triggering

- Capability of starting a process by:
 - Configuring attributes of the target queue
 - Defining a PROCESS object in the queue manager
- Triggering is engaged if the defined trigger conditions are met
- The trigger monitor for a **message channel** is called the channel initiator
 - Does not require a PROCESS definition
 - Channel name is included in TRIGDATA attribute
- IBM MQ for z/OS provides the following trigger monitors:
 - CKTI for use with CICS
 - CSQQTRMN for use with IMS
 - Channel initiator started task for channels

QUEUE (BILL_IN)

TYPE (QLOCAL)

CRDATE (2014-02-26)

CURDEPTH (1)

DEFBIND (OPEN)

DEFPSIST (NO)

INITQ (SYSTEM.DEFAULT.

INITIATION.QUEUE)

TRIGGER

PROCESS (BILLING.PROCESS)

QDEPTHHI (80)

TRIGDATA()

TRIGDPTH (1)

TRIGTYPE (FIRST)

USAGE (NORMAL)

© Copyright IBM Corporation 2015

Figure 3-11. Triggering

WM3021.1

Notes:

IBM MQ provides the capability to start a process upon the arrival of messages in a queue. This process is called triggering. What is triggered is the local queue.

There are two types of triggering. Triggering done to start an application, and triggering is done to start a channel upon receipt of messages in the channel's associated transmit queue.

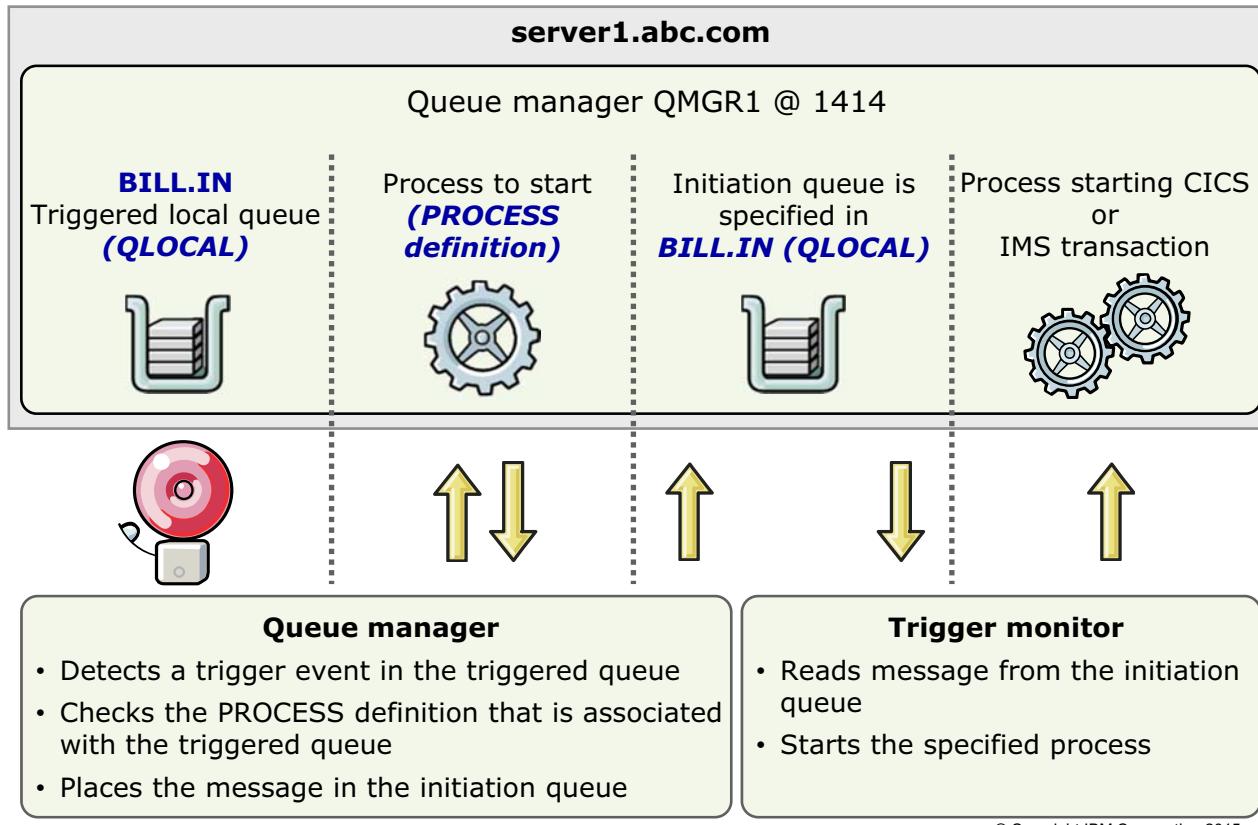
There are two main differences in the processes:

- Triggering an application requires that a process is associated with the triggering definition. Triggering for a channel does not require a process definition.
- The process that reacts to the trigger event and starts the application or process is called a trigger monitor. The process that starts a channel is called a channel initiator.

IBM MQ for z/OS provides a CICS and IMS trigger monitors; however, an application would need to write its own trigger monitor. Distributed IBM MQ has trigger monitors for applications.

You look at how to trigger a process in more detail in the next slide.

Triggering: Process scenario



© Copyright IBM Corporation 2015

Figure 3-12. Triggering: Process scenario

WM3021.1

Notes:

Here are some triggering basics for a process:

- The queue manager detects a “trigger event” in the triggered queue. This event matches the frequency and number of messages that are required to cause the trigger specifications in the queue.
- After detecting the trigger, the queue manager checks the process definition that is associated with the triggered queue. The process definition is a defined IBM MQ object.
- After obtaining the information from the process definition, the queue manager places a specially formatted message with the information required to start the process in the initiation queue that is associated with the triggered queue.
- The trigger monitor that checks the initiation queue reads the message from the initiation queue and starts the process. This process is similar for trigger starting channels, except that the process definition is not required, and the trigger monitor is the channel initiator.

IBM MQ administrative tasks revisited

- Install, apply maintenance, and configure:
 - IBM WebSphere MQ
 - IBM WebSphere MQ Managed File Transfer
- Configure queue managers
- Create queues, clusters, channels, queue-sharing groups, or any objects that users request
- Administer publish/subscribe topologies
- Administer security and SSL
- Back up data sets and object definitions



- Participate in performance analysis
- Monitor buffers and data set spaces
- Perform problem determination
- Participate in capacity planning
- Participate in infrastructure planning sessions
- Participate in naming standards

© Copyright IBM Corporation 2015

Figure 3-13. IBM MQ administrative tasks revisited

WM3021.1

Notes:

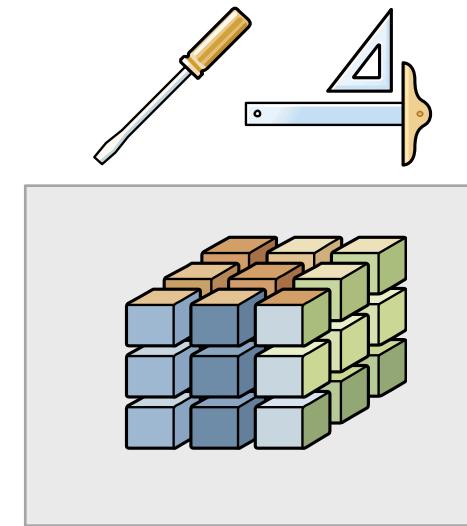
What do you administer, and how do you do it? As an IBM MQ administrator you are involved in different tasks that are related to the health of the queue manager. The tasks that are enclosed in dashed line boxes are included in this unit.

You already learned about the installation process for IBM MQ on z/OS, and configured the queue manager and channel initiator. Now you continue working to tailor the queue manager to the needs of the application. Some of the bullets, like planning and standards, appear best left to management, but an IBM MQ administrator must be involved to provide the correct guidance. What if an application wanted to make all messages persistent, or provide inadequate space for a queue manager? Eventually the resource called to rectify those “problems” introduced by improper planning is the IBM MQ administrator. Therefore, it is best to plan ahead for conditions that affect the IBM MQ infrastructure that you are responsible for.

Capacity planning is not an activity suitable for this course; however, guidance for capacity planning is available in the following publication as a support pack: *Capacity Planning and Tuning for WebSphere MQ for z/OS*, support pack MQ16. At the time of writing of this course, the publication is being updated for IBM MQ V8 for z/OS features. It is a cumulative document, which is refreshed with new releases and features of IBM MQ for z/OS. You now look at the “how.”

IBM MQ for z/OS administrative capabilities

- Equivalent options to perform common actions on IBM MQ objects such as define, alter, display, delete, and show status
- IBM MQ Scripts (MQSC)
 - CSQUTIL COMMAND option
 - SDSF that uses command prefix string
 - ISPF panels
 - Initialization commands
- ISPF operations and control panels
- IBM MQ Explorer
- Programmable command formats
- CSQUTIL
- IBM MQ for z/OS utilities



© Copyright IBM Corporation 2015

Figure 3-14. IBM MQ for z/OS administrative capabilities

WM3021.1

Notes:

IBM MQ for z/OS offers several options for administration. Which option you use depends on what is most appropriate for the task at hand and your personal preference. For instance:

- When creating many objects, maybe associated with a new application, you might want to enter these definitions in a PDS member and define them using the COMMAND option of CSQUTIL.
- If making a single change such as changing MONQ on the queue manager and a single queue, you might find it quicker to use the console with the command prefix string, or CPF.
- If the command string is too long to fit in the log panel, and you use ISPF, you can opt to enter the same command in the ISPF panel command option, which provides ample space.
- If you are checking for the number of messages in a queue or the time a message last arrived in the queue, you might find IBM MQ Explorer quicker to use

Programmable command formats, or PCFs, are more suited for automation, and their use includes application code, so it is less likely to be used unless a specific application is coded. PCFs are included for completeness, as this option is available for IBM MQ for z/OS.

MQSC commands: Most common components

Verbs	IBM MQ objects	Name of the IBM MQ object
ALTER	Queues	<ul style="list-style-type: none"> • Most objects have specific names
CLEAR	Queue managers	<ul style="list-style-type: none"> • Wildcard value can be used for displays
DEFINE	Process definitions	
DELETE	Namelists	
DISPLAY	Authentication information objects	
PING	Channels	
PURGE	Client conn channels	
REFRESH	z/OS storage classes	
RESET	Listeners	
RESOLVE	Topic objects	
RESUME		
SET		
START		
STOP		
SUSPEND		

```

CREATE QLOCAL(APP.IN) DEFPSIST(YES)

DIS QL(APP*) CURDEPTH

START CHANNEL(MQ00.MQ03)

ALTER QMGR(QM00) DEADQ(NEW.DLQ)

DIS QSTATUS(BILL.IN) LPUTDATE
                         LPUTTIME CURDEPTH

```



Use quotation marks to retain lowercase or blank values

© Copyright IBM Corporation 2015

Figure 3-15. MQSC commands: Most common components

WM3021.1

Notes:

IBM MQ script commands are available to define, change, delete, start where applicable, display, or perform most of the day-to-day administration functions. MQSC commands start with a verb, and are followed with an IBM MQ object name. Several attributes or qualifiers can follow the verb-object name combination.

MQSC commands can be used to manage queues, channels, and publish/subscribe objects such as topics or subscriptions; set up clusters; display queue manager details; and numerous other tasks. These commands can be entered in the console, and they are prefixed with the defined CPF string.

The remaining slides in this unit present the administrative options available. In this unit, the examples use queues, but the same capabilities can be used for the queue manager object, channels, and other objects. The one exception is publish/subscribe capabilities, which are not available on the ISPF panels. For all other IBM MQ objects, you find similar capabilities with ISPF, MQSC commands in various ways, and IBM MQ Explorer.



IBM MQ z/OS ISPF panels

IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

```
Action . . . . . . . . . 0      0. List with filter    4. Manage
                                1. List or Display   5. Perform
                                2. Define like       6. Start
                                3. Alter             7. Stop
                                8. Command

Object type . . . . . . . . . QUEUE      +
Name . . . . . . . . . . . *              

Disposition . . . . . . . . . Q Q=Qmgr, C=Copy, P=Private, G=Group,
                                S=Shared, A=All

Connect name . . . . . MQ00 - local queue manager or group
Target queue manager . . . MQ00
                        - connected or remote queue manager for command input
Action queue manager . . . MQ00 - command scope in group
Response wait time . . . 10   5 - 999 seconds
Command ==>
F1=Help F2=Split F3=Exit F4=Prompt F9=SwapNext F10=Messages F12=Cancel
```

© Copyright IBM Corporation 2015

Figure 3-16. IBM MQ z/OS ISPF panels

WM3021.1

Notes:

Most of the options are intuitive. The best way to become familiar with ISPF panels is to use the F1 help. The first option is the action. Option 8 is the command option, which presents you with an ISPF edit panel to enter the command.

If you want to know what names are valid in the object type option, press F1.

Select the queue manager on which to operate on the connect name and other queue manager options. You can select PF1 to discover how the other queue manager options are used.

In this course, you select the same queue manager in all three queue manager name options at the bottom of the ISPF panel screen.

IBM MQ panels: Prompt for object type

Select Object Type - 1

Select one of the following, or press F8 to see other object types.

More: +

1. QUEUE . . . (Q) . . . Any queue
2. QLOCAL. . . (QL) . . . Local queue
3. QREMOTE . . . (QR) . . . Remote queue
4. QALIAS. . . (QA) . . . Alias queue
5. QMODEL. . . (QM) . . . Model queue
6. CLUSQ . . . (QC) . . . Cluster queue
7. QSTATUS . . . (QSTA) . . Queue status

© Copyright IBM Corporation 2015

Figure 3-17. IBM MQ panels: Prompt for object type

WM3021.1

Notes:

This display shows a partial list of the different values that can be entered in the Object Type field. If you notice on the instructions, selecting F8 presents more options of object types.

The screenshot shows a WebSphere Education interface with the title "Queue manager attributes with IBM MQ ISPF panels". The main content area displays the attributes of a queue manager named MQ00. A yellow box highlights the Action, Object type, and Name fields. The output is as follows:

```

Action . . . . 1
Object type . . QMGR
Name . . . . MQ00
Display a Queue Manager - 1

Press F8 to see further fields, or Enter to refresh details.

More: +
Queue manager name . . . . : MQ00

Description . . . . . : MQ00, IBM WebSphere MQ for z/OS
- V8
Default transmission queue : MQ00.DEFXMIT.QUEUE
Dead-letter queue . . . . : MQ00.DEAD.QUEUE
Trigger interval . . . . : 999999999 0 - 999999999 milliseconds
Max open handles . . . . : 256 0 - 999999999
Max uncommitted messages . . : 10000 1 - 999999999
Expiry interval . . . . : OFF 1 - 999999999 seconds or OFF

Command level . . . . . : 700
Coded character set ID . . . : 500
Last alteration time . . . : 2014-08-18 17.54.00

```

© Copyright IBM Corporation 2015

Figure 3-18. Queue manager attributes with IBM MQ ISPF panels

WM3021.1

Notes:

This display uses Action 1 for display, and Object type QMGR for the queue manager object. The first page of the queue manager object is shown. You press PF8 to page through the rest of the queue manager object details.



Queue manager partial attributes that use the MQSC command

DIS QMGR ALL

```
CSQM201I MQ00 CSQMDRTC  DIS QMGR
DETAILS 971
QMNAME (MQ00)
DESCR(IBM ... MQ ... z/OS V8.0.0)
PLATFORM(MVS)
CPILEVEL(100)
CMDLEVEL(800)
VERSION(08000000)
CCSID(500)
CHLAUTH(ENABLED)
MAXPRTY(9)
MAXMSGL(104857600)
SYNCPT(AVAILABLE)
QSGNAME()
COMMANDQ(SYSTEM.COMMAND.INPUT)
DEADQ()
TRIGINT(999999999)
MAXHANDS(256)
MAXUMSGS(10000)
```

```
AUTHOREV(DISABLED)
INHIBTEV(DISABLED)
LOCALEV(DISABLED)
REMOTEEV(DISABLED)
STRSTPEV(ENABLED)
PERFMEV(DISABLED)
CONFIGEV(DISABLED)
CMDEV(DISABLED)
CHLEV(DISABLED)
SSLEV(DISABLED)
BRIDGEEV(DISABLED)
CHADEXIT()
...
SSLFIPS(NO)
CERTLBL(ibmWebSphereMQMQ00)
...
CONNAUTH(SYSTEM.DEFAULT.
          AUTHINFO.IDPWOS)
...
```

© Copyright IBM Corporation 2015

Figure 3-19. Queue manager partial attributes using the MQSC command

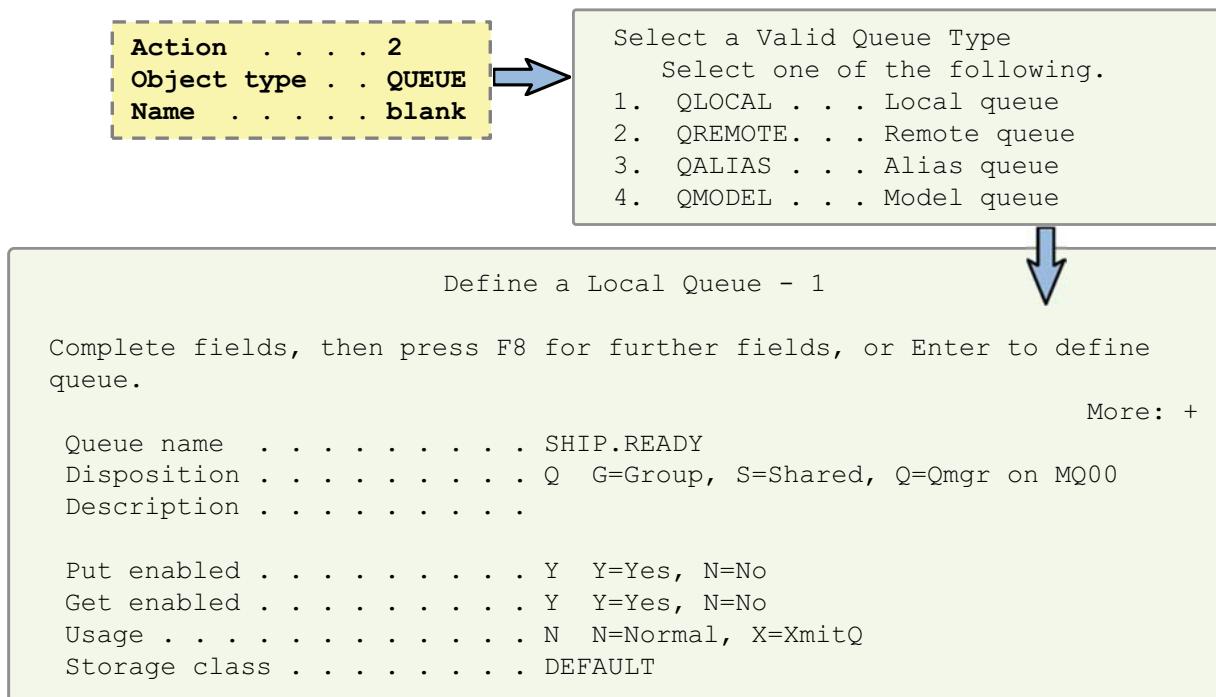
WM3021.1

Notes:

Here are the same queue manager attributes displayed in the SDSF command input screen and checked in the system log.



Define a local queue with ISPF panels



- Leave name blank to omit “LIKE” option
- PF8 for additional QLOCAL attributes

© Copyright IBM Corporation 2015

Figure 3-20. Define a local queue with ISPF panels

WM3021.1

Notes:

Here is a definition of a local queue that uses ISPF panels. The process is similar to a display of a queue manager object. However, you do not enter the name of the queue you need to define in the first ISPF panel. The name to be entered is the name to use as a template or model for the queue that you need to define. If you enter your needed queue name, which should not exist, the reply is that the queue cannot be found.

Leave the queue name blank in the first panel.

If you enter QUEUE as the object type, the second panel to the right pops up and asks what type of queue you want to create.

If you enter QLOCAL as the Object type, you skip the interim queue type selection panel.

When you get to the panel titled “Define a Local Queue”, then you enter the required queue name. Use PF8 to get to the rest of the attributes that are not shown on the first definition panel.

Define a local queue with CPF and MQSC commands

```
HQX7740 ----- SDSF PRIMARY OPTION MENU -----
COMMAND INPUT ===> /MQ00 DEF QLOCAL(PRE.SHIP)           SCROLL ===> CSR
DA      Active users                         INIT   Initiators
I       Input queue                          PR     Printers
..... . . . . .
```

```
HQX7740 ----- SDSF PRIMARY OPTION MENU -- COMMAND ISSUED
COMMAND INPUT ===>                               SCROLL ===> CSR
RESPONSE=MC03          CSQ9022I +MQ00 CSQMAQLC ' DEF QLOCAL' NORMAL COMPLETION
DA      Active users                         INIT   Initiators
..... . . . . .
```

© Copyright IBM Corporation 2015

Figure 3-21. Define a local queue with CPF and MQSC commands

WM3021.1

Notes:

This example uses the SDSF command input to define a queue. You needed only to enter the queue name without changing many attributes. If you need to enter more attributes and lack space in the SDSF command line, some of the options to use MQSC commands would be:

- Enter a slash “/” in the command input line of the SDSF panel.
 - You receive a “System command extension” panel.
 - Proceed to enter your console command in the extension panel.
- Use ISPF panel option 8 for the command input option. You have an ISPF edit screen to enter the command or multiple commands.
- Use CSQUTIL JCL with the COMMAND option to enter the command or multiple commands.



Display a local queue with ISPF panels

Action 1
 Object type . . QUEUE
 Name SHIP* List Queues - MQ00 Row 1 of 1

Type action codes, then press Enter. Press F11 to display queue status.
 1=Display 2=Define like 3=Alter 4=Manage

Name	Type	Disposition
<> SHIP*	QUEUE	PRIVATE MQ00
<u>1</u> SHIP.READY	QLOCAL	QMGR MQ00

***** End of list *****

Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

More: +

Queue name SHIP.READY
 Disposition : QMGR MQ00
 Description :

Put enabled : Y Y=Yes, N=No
 Get enabled : Y Y=Yes, N=No
 Usage : N N=Normal, X=XmitQ
 Storage class : DEFAULT

© Copyright IBM Corporation 2015

Figure 3-22. Display a local queue with ISPF panels

WM3021.1

Notes:

The queue display with ISPF is similar to the queue manager display with ISPF.

Display a local queue with CPF and MQSC commands

```
HQX7740 ----- SDSF PRIMARY OPTION MENU -----
COMMAND INPUT ===> /MQ00 DIS QLOCAL(SHIP*) ALL           SCROLL ===> CSR
..... . . . . .
```

```
HQX7740 ----- SDSF PRIMARY OPTION MENU -- COMMAND ISSUED
COMMAND INPUT ===> LOG                                SCROLL ===>
CSR
RESPONSE=MC03      CSQM293I +MQ00 CSQMDRTC 1 QUEUE FOUND MATCHING REQUEST
RESPONSE=CRITERIA
DA     Active users          INIT   Initiators
..... . . . . .
```

```
SDSF SYSLOG      349.101 MC03 MC03 08/21/2014 0W      8202      COLUMNS 51 130
COMMAND INPUT ===>
CSRCSQM201I +MQ00 CSQMDRTC  DIS QUEUE DETAILS 127
QUEUE(SHIP.READY) ... ...
STGCLASS(DEFAULT) ... ...
PUT(ENABLED) ... ...
DEFPSIST(NO)
OPPROCS(0)
IPPROCS(0)
CURDEPTH(0) ... ... .
```

© Copyright IBM Corporation 2015

Figure 3-23. Display a local queue with CPF and MQSC commands

WM3021.1

Notes:

This screen capture shows how to display a local queue that uses CPF commands from the SDSF panels.

WebSphere Education

IBM

Filters with ISPF (1 of 2)

Action 1
Object type . . QSTATUS
Name SHIP* Filter Queue Status

Type ONE field value, then press Enter to choose a test.

Queue name : *
Disposition : QMGR MQ00
Use counts - Output Input
Current queue depth
Uncommitted messages Y=Yes, N=No
Monitoring : L=Low, M=Medium, H=High, O=Off
Max message age 300 seconds



Filter Test

Choose a test.

5 1. Equal to
 2. Not equal to
 3. Less than
 4. Not less than (Greater than or equal to)
 5. Greater than
 6. Not greater than (Less than or equal to)

Figure 3-24 Filters with JSPE (1 of 2)

WM3021_1

Notes-

When displaying multiple queues, ISPF panels can be set to use a filter, or a set of rules to select specific queues, possibly based on a current queue depth that must be greater than 0.

Filters with ISPF (2 of 2)

```
List Local Queues - Status - MQ00                               Row 1 of 6

Type action codes, then press Enter. Press F11 to display all open queues.
1=Display applications   3=Alter   4=Manage

Queue name                                Disposition    Depth
      Last put time      Last get time      Time on queue    Max age
      Output use   Input use   Uncommitted msgs
><  *                                     QMGR        MQ00
ORDER.IN                                    QMGR        MQ00    2
  2014-08-18 18.03.02 2014-08-19 12.07.09 9999999999 9999999999 257137
SYSTEM.DURABLE.SUBSCRIBER.QUEUE           QMGR        MQ00    3
  2014-08-08 13.30.23 2014-08-08 13.30.22 9999999999 9999999999 1120020
SYSTEM.RETAINED.PUB.QUEUE                 QMGR        MQ00    1
                                         0          0          200129460
```

© Copyright IBM Corporation 2015

Figure 3-25. Filters with ISPF (2 of 2)

WM3021.1

Notes:

Here is the second slide of the ISPF queue display list with filters. The queues were chosen based on the number of messages in the queue.

In busy queue managers, this test would not be practical; it is used here only to illustrate filtering.



Filters with MQSC

Using queues as an example:

QUEUE

DISPLAY QLOCAL WHERE (attribute **operator** value)

QSTATUS

- LT** Less than
- GT** Greater than
- EQ** Equal to
- NE** Not equal to
- LE** Less than or equal to
- GE** Greater than or equal to
- CT** Contains...
- EX** Excludes... } For items
in a list
- LK** Matches...
- NL** Does not match...
-

© Copyright IBM Corporation 2015

Figure 3-26. Filters with MQSC

WM3021.1

Notes:

The same filtering can be done in an MQSC command.

DISPLAY filtering: MQSC examples

- Show the queues that accept messages larger than 100,000 bytes:

```
DISPLAY QLOCAL(*)
  WHERE (MAXMSGL GT 100000)
```

- Show the queues with messages older than 5 minutes:

```
DISPLAY QSTATUS(*)
  WHERE (MSGAGE GT 300)
```

- Show the queues with a currently active *input exclusive* access:

```
DISPLAY QSTATUS(*) TYPE(HANDLE)
  WHERE (INPUT EQ EXCL)
```

© Copyright IBM Corporation 2015

Figure 3-27. DISPLAY filtering: MQSC examples

WM3021.1

Notes:

Here are some examples of filtering with MQSC commands. Filtering can be used on other IBM MQ objects.

Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (1 of 2)

```
Action . . . . 8
Object type . . blank
Name . . . . blank
```

- Enter MQSC commands in resulting ISPF editor screen
 - Commands are written to HLQ.CSQUTIL.COMMANDS
 - Commands execute upon pressing F3
 - Output is saved in HLQ.CSQUTIL.OUTPUT

```
EDIT          TSM0000.CSQUTIL.COMMANDS      Columns 00001 00072
***** **** Top of Data *****
000001 dis qstatus(*) where(LPUTDATE GT 2014-08-17) all

***** **** Bottom of Data *****
Enter/edit MQSC commands. Line Edit (F4) to edit long lines, END (F3) to
process commands, CANCEL (F12) to exit without processing commands, HELP
(F1) for more help.
```

© Copyright IBM Corporation 2015

Figure 3-28. Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (1 of 2)

WM3021.1

Notes:

Here is an option of issuing MQSC filtering commands with the ISPF command option 8.

When using this option, you are presented an ISPF edit screen with a file that is named:
<yourUserID>.CSQUTIL.COMMANDS

Every time that you need to issue a new set of commands with the ISPF 8 option, you use and overwrite the same the edit screen, even if using it for different queue managers.

The CSQUTIL input file is allocated per your ISPF session.

To run the command, you press PF3; to cancel, you press PF12.

Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (2 of 2)

```

EDIT                               TSM0000.CSQUTIL.OUTPUT          Columns 00002 00073
***** Top of Data ****
CSQU001I CSQUTIL Queue Manager Utility - 2014-08-21 11:02:40
COMMAND TGTQMGR(MQ00) RESPTIME(10)
CSQU127I Executing COMMAND using input from CSQUCMD data set
CSQU120I Connecting to MQ00
...
dis qstatus(*) where(LPUTDATE GT 2014-08-17) all
CSQN205I COUNT=      3, RETURN=00000000, REASON=00000000
CSQM441I +MQ00
QSTATUS (ORDER.IN)
TYPE (QUEUE)
OPPROCS (0)
IPPROCS (0)
CURDEPTH (2)
UNCOM (NO)
MONQ (LOW)
QTIME (999999999, 999999999)
MSGAGE (251454)
LPUTDATE (2014-08-18)
LPUTTIME (18.03.02)
LGETDATE (2014-08-19)
LGETTIME (12.07.09) ...

```

© Copyright IBM Corporation 2015

Figure 3-29. Filters: MQSC DISPLAY QSTATUS with ISPF command option 8 (2 of 2)

WM3021.1

Notes:

The results of the ISPF command option commands are displayed in a file that is called <yourUserID>.CSQUTIL.OUTPUT. Like the CSQUTIL input file, this file is reused and overlaid with every iteration of the ISPF command option 8.

For this option, you are issuing the DIS QSTATUS command. You might notice that MONQ is set to low, and you can see some of the statistics, last time, and date that a message was put or taken from this queue. You also see the message age, or MSGAGE attribute, which represents the age, in seconds, of the oldest message in this queue. With the MONQ attribute, this value can also be populated.

The contents of the UNCOM value is platform-specific. For z/OS, NO means no uncommitted messages. If UNCOM is YES on a z/OS queue, it means that at least one message in the queue is uncommitted. On a distributed platform, UNCOM displays the number of uncommitted messages in the queue.

Display QSTATUS handles with ISPF

```
Action . . . . 1
Object type . . QSTATUS
Name . . . . *
```

Type action codes, then press Enter. Press F11 to display all open queues.
 1=Display applications 3=Alter 4=Manage

Queue name	Disposition	Depth	
Last put time	Last get time	Time on queue	Max age
Output use	Input use	Uncommitted msgs	
<> *	QMGR	MQ00	
AMQ.MQEXPLORER.283482628	QMGR	MQ00	0
0	1	N	
CICS01.INITQ	QMGR	MQ00	0
	0	0	0

Display messages

Row 1 of 322

```
CSQM441I MQ00 QSTATUS(AMQ.MQEXPLORER.283482628
) TYPE(HANDLE      ) INPUT(EXCL       ) OUTPUT(NO        ) BROWSE(NO
) INQUIRE(NO        ) SET(NO       ) URID(D4D8F0F0C3C8C9D5
```

© Copyright IBM Corporation 2015

Figure 3-30. Display QSTATUS handles with ISPF

WM3021.1

Notes:

The QSTATUS handle shows what process has an open handle on a queue.

MQSC queue status display default and handle options

DIS QSTATUS (ORDER.IN) ALL

```

000011 QSTATUS (ORDER.IN)
000012 TYPE (QUEUE)
000013 OPPROCS (0)
000014 IPPROCS (0)
000015 CURDEPTH (2)
000016 UNCOM (NO)
000017 MONQ (LOW)
000018 QTIME (0,0)
000019 MSGAGE (17527)
000020 LPUTDATE (2014-08-18)
000021 LPUTTIME (18.03.02)
000022 LGETDATE( )
000023 LGETTIME( )
000024 QSGDISP (QMGR)
000025 CSQ9022I +MQ00 ... ...

```

DIS QSTATUS (ORDER.IN) TYPE (HANDLE)

```

QSTATUS (ORDER.IN)
TYPE (HANDLE)
INPUT (NO)
OUTPUT (NO)
BROWSE (YES)
INQUIRE (NO)
SET (NO)
URID (D4D8F0F0C3C8C9D5 )
QMURID( )
URTYPE (QMGR)
QSGDISP (QMGR)
HSTATE (INACTIVE)
ASTATE (NONE)
USERID (MQUSER)
APPLTAG (MQ00CHIN)
ASID (004A)
APPLTYPE (CHINIT)
CHANNEL (SYSTEM.ADMIN.SVRCONN)
CONNAME (10.4.127.184)

```

© Copyright IBM Corporation 2015

Figure 3-31. MQSC queue status display default and handle options

WM3021.1

Notes:

The DISPLAY QSTATUS command can be issued specifying two types: QUEUE, which is the default, or HANDLE.

In this display, both forms of the QSTATUS command are issued. Observe the differences between the output of the two displays.



ISPF displaying connections

```
Action . . . . 1
Object type . . CONNECT
Name . . . . *
List Connections - MQ00                                     Row 61 of 67

Type action codes, then press Enter.
1=Display handles

Connection          QMgr   Start time           Log time           Handles
Application        ASID   Application information   User ID    UR state
External URID
Origin ID

<> *             MQ00
CDA4F0A4C38C0001 MQ00                                     0
MQ00CHIN CHINIT  0021                                     MQUSER   NONE
CDA4F0A528010001 MQ00                                     3
MQ00CHIN CHINIT  0021 SYSTEM.ADMIN.SVRCONN
                                         10.4.127.184
CDA4F1235AAD0001 MQ00                                     2
TSM0000  BATCH    003E                                     TSM0000   NONE
***** End of list *****
```

© Copyright IBM Corporation 2015

Figure 3-32. ISPF displaying connections

WM3021.1

Notes:

This screen uses the ISPF panels to show the connection display.

ISPF displaying connection handles

```
Action . . . . 1
Object type . . CONNECT
Name . . . . *
```

List Connections - MQ00					Row 61 of 67
Type action codes, then press Enter.					
1=Display handles					
Connection	QMgr	Start time	Log time	Handles	
Application	ASID	Application information		User ID	UR state
External URID		UR type	MQ URID		
Origin ID					
<> * MQ00					
...					
<u>1</u> CDA4F0A528010001 MQ00					3
<u>MQ00CHIN</u> CHINIT	0021	SYSTEM.ADMIN.SVRCONN		MQUSER	NONE
		10.4.127.184			

Object name **Disposition** **Access**
 Type **State**
 *
 QMGR MQ00 - IX --
 QUEUE ACTIVE



© Copyright IBM Corporation 2015

Figure 3-33. ISPF displaying connection handles

WM3021.1

Notes:

This slide displays connection handles with ISPF.




Display page set usage with ISPF command option

```
Action . . . . 1
Object type . . SYSTEM
Name . . . . . blank
```

Select function type, then press Enter to display details.

Function type	2	1. Distributed queuing	CHINIT
		2. Page set usage	
		3. Queue-sharing group	
		4. Data set usage	Logs

```
Action queue manager . . . : MQ00
```

Display messages Row 1 of 2

```
CSQI010I MQ00 Page set usage ...
Page Buffer      Total    Unused   Persistent NonPersist Expansion
set   pool       pages     pages    data pages   data pages   count
-     0          1078      1023      55           0  USER      0
-     1          1078      1061      15           2  USER      0
...
End of page set report
CSQI065I MQ00 Buffer pool attributes ...
Buffer Available Stealable Stealable Page      Location
pool   buffers   buffers  percentage class
-     0        50000    49971      99  4KB      BELOW
...
```

© Copyright IBM Corporation 2015

Figure 3-34. Display page set usage with ISPF command option

WM3021.1

Notes:

In this panel, you see how to display a page with ISPF. If you look at the possible options, you also see the options to display channel initiator information.



Display page set usage with MQSC ISPF command option

```
Action . . . . 8
Object type . . blank
Name . . . . . blank
```

```
EDIT                      TSM0000.CSQUTIL.COMMANDS      Columns 00001
'***** dis usage type(pageset)
```

```
CSQU000I CSQUTIL IBM WebSphere MQ for z/OS V7
CSQU001I CSQUTIL Queue Manager Utility - 2014-08-19 10:00:15
COMMAND TGTQMGR(MQ00) RESPTIME(10)
CSQU127I Executing COMMAND using input from CSQUCMD data set
CSQU120I Connecting to MQ00
CSQU121I Connected to queue manager MQ00
CSQU055I Target queue manager is MQ00
DIS USAGE TYPE(PAGESET)
CSQN205I COUNT=      17, RETURN=00000000, REASON=00000000
CSQI010I +MQ00 Page set usage ...
   Page Buffer      Total     Unused  Persistent NonPersist Expansion
    set   pool      pages      pages   data pages   data pages       count
    -     0        322       275        47          0  USER      0
    -     1        322       308        14          0  USER      0
    -     2        322       312        10          0  USER      0
... ... ... ... ...
```

© Copyright IBM Corporation 2015

Figure 3-35. Display page set usage with MQSC ISPF command option

WM3021.1

Notes:

Here you see the same page display, which uses the ISPF command option and MQSC command. You look at the <yourUserID>.CSQUTIL.COMMANDS input file, and the resulting <yourUserID>COMMAND.OUTPUT files.

ISPF manage a local queue and equivalent MQSC commands

Manage a Local Queue

Select function type, then press Enter.

- | | |
|-------------------------|---|
| Function type | 1. Delete |
| | 2. Create a local copy on MQ00 |
| | 3. Refresh local copy |
| | 4. Clear messages |
| | 5. Move messages to another empty queue |
| | 6. Move messages to another queue |

Queue name : SHIP.READY

Queue type : QLOCAL

Disposition : QMGR MQ00

Description :

To queue name (Move)

MQSC equivalents

Delete: DELETE QLOCAL

Create a local copy: DEFINE QLOCAL LIKE

Refresh: For GROUP objects

Clear messages: CLEAR QLOCAL

Move messages (both forms): MOVE QLOCAL

© Copyright IBM Corporation 2015

Figure 3-36. ISPF manage a local queue and equivalent MQSC commands

WM3021.1

Notes:

This slide shows a comparison of ISPF capabilities and MQSC equivalents.



The screenshot displays the IBM WebSphere MQ Explorer (IBMMQV8) interface. The left pane, titled "MQ Explorer - Navigator", shows a hierarchical tree view of queue managers and their components. Under "MQ00 on 'mvsmc03.ilsvpn.ibm.com(1600)'", the following items are listed: Queues, Topics, Subscriptions, Channels, Listeners, Process Definitions, Namelists, Authentication Information, and Storage Classes. Other entries include "MQ01 on 'mvsmc03.ilsvpn.ibm.com(1601)'" and "MQ03 on 'mvsmc03.ilsvpn.ibm.com(1603)'". The right pane, titled "MQ Explorer - Content", is titled "Queues" and contains a table of queue definitions. The table has columns for "Queue name" and "Queue type". The data is as follows:

Queue name	Queue type
MQ12.REMOTE.QUEUE	Remote
MQ13	Local
MQ13.REMOTE.QUEUE	Remote
MQ14	Local
MQ14.REMOTE.QUEUE	Remote
MQ15	Local
MQ15.REMOTE.QUEUE	Remote
MQ16	Local
MQ16.REMOTE.QUEUE	Remote

- Provides many functions equivalent to ISPF and MQSC commands
- Installed in Windows on Linux

© Copyright IBM Corporation 2015

Figure 3-37. IBM MQ Explorer

WM3021.1

Notes:

You look at more IBM MQ Explorer options throughout other units in this course.

Programmable command formats: PCFs

- Provides a way to automate local and remote administration tasks programmatically
- Same functionality as MQSC commands but not human-readable
- One example of a PCF format is **Inquire Queue**
- To use **Inquire Queue** PCF for remote administration:
 1. A developer writes an MQI program and selects the **Inquire Queue** PCF
 2. The target queue manager must have the command server component active
 3. Channels must exist between the source and target queue manager
 4. The program does an MQPUT with the **Inquire Queue** PCF from the source queue manager to the target queue manager
 5. The target queue manager processes the command and replies to the source queue manager who receives the message with an MQGET

© Copyright IBM Corporation 2015

Figure 3-38. Programmable command formats: PCFs

WM3021.1

Notes:

CSQUTIL parameters and options

- PARM= Queue manager name if the option used requires the queue manager to be active
- PARM= Blank if the option used does not require the queue manager to be active
- Two other parameter options for queue-sharing groups

```
//COMMAND EXEC PGM=CSQUTIL,PARM='MQ##'
```

OPTIONS

- COMMAND option
- FORMAT to format a page set
- PAGEINFO extracts page set information
- COPYPAGE copies page sets to larger pageset (does not reset log)
- RESETPAGE copies a page set and resets the log
- SDEFS produces a copy of IBM MQ define commands for a queue manager or queue-sharing group
- COPY copies messages to a data set while queue manager is running
- SCOPY same as COPY to use when the queue manager is not running
- ANALYSE inspects a data set created by COPY or SCOPY and displays the queue name, number of messages, and length of messages for each queue processed
- EMPTY deletes all messages from a queue, or all queues from a page set
- LOAD and SLOAD restore messages to a queue from a data set created by COPY or SCOPY
- Migrating a channel initiator parameter module (XPARM)
- XPARM
- SWITCH

© Copyright IBM Corporation 2015

Figure 3-39. CSQUTIL parameters and options

WM3021.1

Notes:

CSQUTIL is a primary batch utility for IBM MQ on z/OS. There are numerous options to accomplish different tasks.

The COMMAND option is useful for issuing MQSC commands to create one or more definitions. You look at some of these options now, and other options in a later unit.

CSQUTIL command option: Defining IBM MQ objects

```
//MQ00UTIL JOB ,WM30V1,CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//*
//*- -----
//* RUN THE CSQ UTILITY PGM FOR PROCESSING COMMANDS
//*- -----
//COMMAND EXEC PGM=CSQUTIL,PARM='MQ00'
//STEPLIB    DD DSN=SYS1.WMQ800.SCSQANLE,DISP=SHR
//           DD DSN=SYS1.WMQ800.SCSQAUTH,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *,DCB=BLKSIZE=80
      COMMAND
/*
//CSQUCMD    DD DISP=SHR,DSN=TSM0000.WM30V1.SETUP.CNTL(CSQUTL1)
```

- CSQUCMD DD contains MQSC commands

```
DEFINE QLOCAL(WM302.INPUT)
DEFINE QALIAS(WM302.ALIAS) TARGET(WM302.INPUT)
...
...
```

© Copyright IBM Corporation 2015

Figure 3-40. CSQUTIL command option: Defining IBM MQ objects

WM3021.1

Notes:

This slide shows example JCL that illustrates how to use CSQUTIL to create object definitions. One item to look out for is administrators who use the same JCL and change only the object name; for example, to create a queue. At times, this practice can be error prone if, for instance, you create a queue for an unintended storage class.

Carefully review definitions that are used on a repeated basis to prevent unexpected problems.

CSQUTIL command option: Backing up object definitions in an active queue manager

```
//MQ00UTIL JOB ,WM30V1,CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//* -----
//COMMAND EXEC PGM=CSQUTIL,PARM='MQ##'
//STEPLIB    DD DSN=SYS1.WMQ700.SCSQANLE,DISP=SHR
//          DD DSN=SYS1.WMQ700.SCSQAUTH,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *,DCB=BLKSIZE=80
      COMMAND MADEDEF (DEFOUT)
/*
//CSQUCMD    DD *
      DISPLAY QMGR ALL
      DISPLAY STGCLASS (*) ALL
      DISPLAY QUEUE (*) ALL
      DISPLAY CHANNEL (*) ALL
      DISPLAY NAMELIST (*) ALL
      DISPLAY PROCESS (*) ALL
/*
//DEFOUT DD DSN=TSM00##.WM30V1.SCSQPROC(OBJDEFS),DISP=SHR
```

© Copyright IBM Corporation 2015

Figure 3-41. CSQUTIL command option: Backing up object definitions in an active queue manager

WM3021.1

Notes:

In a queue manager, usually changes are made in various ways, such as using the command line. These changes alter the configuration of the queue manager, but since they are made on the side, no scripts are stored to be able to repeat the change.

Scheduling a periodic backup of all IBM MQ object definitions in the queue manager is necessary for having a side set of definitions to rebuild the existing queue manager environment, if needed.

The CSQUTIL command that is used to back up the IBM MQ objects in the queue manager when the queue manager is running is called MADEDEF. To back up an object, it first needs to be displayed. This screen capture is displaying the queue manager object, along with all storage classes, queues, channels, namelists, and processes. This example is taken from the IBM MQ V7 version, but it would be the same for IBM MQ V8; however, for the newer queue manager, there would also be a display of the CHLAUTH definitions. But CHLAUTH is not discussed until the security unit, so for purposes of this example, the definitions that are shown suffice.

In this batch job, all definitions are placed in member OBJDEFS of the partitioned data set to which the DEFOUT DD statement points.

CSQUTIL SDEFS option: Backing up object definitions in a stopped queue manager

```
//MQ00UTIL JOB ,WM30V1,CLASS=A,MSGCLASS=H,NOTIFY=&SYSUID
//COMMAND EXEC PGM=CSQUTIL,PARM='MQ00'
//STEPLIB   DD DSN=SYS1.WMQ800.SCSQANLE,DISP=SHR
//          DD DSN=SYS1.WMQ800.SCSQAUTH,DISP=SHR
//CSQP0000 DD DISP=OLD,DSN==TSM0000.MQ00.PSID00,DISP=SHR <==== page set 0
//OUTPUT1 DD DISP=OLD,DSN=MY.COMMANDS (CHANNEL)
//OUTPUT2 DD DISP=OLD,DSN=MY.COMMANDS (AUTHINFO)
... ... ...
... ... ...
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
SDEFS OBJECT (CHANNEL) MAKEDEF (OUTPUT1)
SDEFS OBJECT (AUTHINFO) MAKEDEF (OUTPUT2)
SDEFS OBJECT (NAMELIST) MAKEDEF (OUTPUT3)
SDEFS OBJECT (PROCESS) MAKEDEF (OUTPUT4)
SDEFS OBJECT (QALIAS) MAKEDEF (OUTPUT5)
SDEFS OBJECT (QLOCAL) MAKEDEF (OUTPUT6)
SDEFS OBJECT (QMODEL) MAKEDEF (OUTPUT7)
SDEFS OBJECT (QREMOTE) MAKEDEF (OUTPUT8)
SDEFS OBJECT (QUEUE) MAKEDEF (OUTPUT9)
SDEFS OBJECT (STGCLASS) MAKEDEF (OUTPUT0)
SDEFS OBJECT (TOPIC) MAKEDEF (OUTPUTA)
/*

```

© Copyright IBM Corporation 2015

Figure 3-42. CSQUTIL SDEFS option: Backing up object definitions in a stopped queue manager

WM3021.1

Notes:

A different option to back up IBM MQ object definitions for a queue manager is the CSQUTIL SDEFS option. This option is used when the queue manager is stopped because it uses page set zero (where object definitions are kept) to extract all definitions. No display step is involved as the information is extracted directly from page set zero. There is one PDS member per object type.

You are not yet familiar with some objects. AUTHINFO and TOPIC are discussed in later units. A NAMELIST is an option to join, or include more than one name in one object definition; for instance, when two IBM MQ clusters intercept, a namelist is used that includes the two cluster names.

Message handler sample: ISPF option H in class system

Enter information. Press ENTER					
Queue Manager	Name: MQ0A				
Message number	01 of 09				
Msg No	Put Date	Put Time	Format	User Identifier	Put Application Type Name
01	MM/DD/YYYY	HH:MM:SS	MQSTR	NOMAD789	00000011 C:\mqv8\bin64\MQE
02	09/04/2014	21:14:40	MQSTR	NOMAD789	00000011 ls\c\Samples\Bin\
03	09/04/2014	21:19:19	MQSTR	NOMAD789	00000011 ls\c\Samples\Bin\

Queue Manager	MQ00	:
Queue	SHIP.READY	:
Forward to Q Mgr	MQ00	:
Forward to Queue		:
Action :	(D)elete (F)orward	
Message Content :		

PutApplName	'C:\mqv8\bin64\MQExplorer.exe'	:
PutDate	'20140904'	:
PutTime	'21144089'	:
App1OriginData	:	'
Message Buffer	7 byte(s)	
00000000	C1C2 C3C4 C5C6 C7	'ABCDEF'G

© Copyright IBM Corporation 2015

Figure 3-43. Message handler sample: ISPF option H in class system

WM3021.1

Notes:

The previous few slides examined the administration options. However, there are other utilities, some of which are handy when resolving problems. One such utility is the message handler sample program available at ISPF option H in the lab environment. Some of the possible uses for this program are:

- Look up the reason that a message was placed in the dead-letter queue. The formatted dead-letter queue header and return code are shown in the next slide.
- Move messages to another queue.
- Delete a message.
- Check the contents of a message.

You use the message handler sample program in the labs.

Dead-letter queue header with a message handler sample

- Use the message handler sample to view a formatted Dead Letter Header (DLH) and the reason why the message was sent to the dead-letter queue
- DLH is formatted past the MQMD: Use PF8

```

Queue Manager      : MQ0A
Queue             : MQ0A.DEAD.QUEUE
Forward to Q Mgr : MQ00
Forward to Queue :
Action : : (D)elete (F)orward

Message Content :

PutApplName      : 'MQPUTUTP'
PutDate          : '20140901'
PutTime          : '18085379'
ApplOriginData   : ' '

Dead Letter Header
StrucId          : 'DLH '
Version          : 000000001
Reason          : 000002085
DestQName        : 'SALES.RPT'
DestQMgrName     : 'MQ0A 'ABCDEFG

```

Enter information. Press ENTER
 Queue Manager Name: MQ0A
 Queue Name
:MQ0A.DEAD.QUEUE

© Copyright IBM Corporation 2015

Figure 3-44. Dead-letter queue header with a message handler sample

WM3021.1

Notes:

This screen capture shows the message handler display when the message is in the dead-letter queue with the dead-letter header included.

If you look below the “Dead Letter Header” heading, you can see that the reason code was 2085; that is, the IBM MQ object SALES.RPT was not known.

IBM MQ sample put program and parameters

```
BROWSE      TSM0000.WM30V1.STUDENT.SCSQPROC(JCLPUT) - Line 00000003 Col
Command ====>                                         Scroll =
/*
/* PROGRAM CSQ4BCK1 ISSUES MQPUT ON A QUEUE.
/* - FIRST PARM (++QMGR++)  QUEUE MANAGER NAME
/* - SECOND PARM (++QUEUE++) QUEUE NAME
/* - THIRD PARM (++MSGSS++) THE NUMBER OF MESSAGES TO PUT- (9999)
/* - FOURTH PARM (++PAD++)   THE PADDING CHARACTER
/* - FIFTH PARM (++LEN++)    THE LENGTH OF EACH MESSAGE- (9999)
/* - SIXTH PARM (++PERS++)   (P)ERSISTENT/(N)ON PERSISTENT MESSAGES
/* MESSAGES ARE PRINTED TO DD SYSPRINT
/*
//*****EXEC PGM=CSQ4BCK1,REGION=1024K,
// PARM=( 'MQ## EX2.BASEQ 0003 X 0112 P')
/*
//STEPLIB  DD   DSN=SYS2.WMQ.SAMPOLOAD,DISP=SHR
//           DD   DSN=SYS1.SCEERUN,DISP=SHR
```

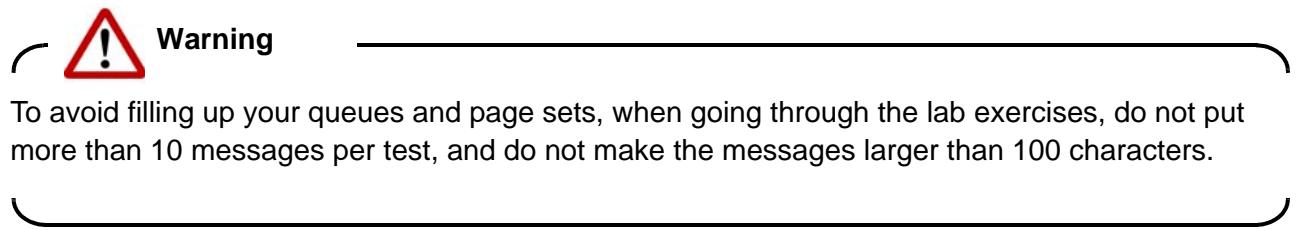
© Copyright IBM Corporation 2015

Figure 3-45. IBM MQ sample put program and parameters

WM3021.1

Notes:

There are sample batch programs to put and get messages. You use these sample programs in the labs. The slide shows one of the put programs. The program takes six parameters that are documented in the JCL.



IBM MQ sample get program and parameters

```

BROWSE      TSM0000.WM30V1.STUDENT.SCSQPROC(JCLPUT) - Line 00000003 Col
Command ===>                                         Scroll =
/* PROGRAM CSQ4BCJ1 ISSUES MQGET ON A QUEUE.
/* (EXEC AND PARM STATEMENTS ARE COMMENTED OUT AS SHIPPED)
/*
/* - FIRST PARM (++QMGR++)   QUEUE MANAGER NAME
/* - SECOND PARM (++QUEUE++)  QUEUE NAME
/* - THIRD PARM (++MSG$++)   THE NUMBER OF MESSAGES TO GET-(9999)
/* - FOURTH PARM (++GET++)    GET TYPE-(B)ROWSE/(D)ESTRUCTIVE
/* - FIFTH PARM (++SYNC++)   (S)YNCPOINT/(N)O SYNCPOINT
/* MESSAGES ARE PRINTED TO DD SYSPRINT
/*
//*****EXEC PGM=CSQ4BCJ1,REGION=1024K,
// PARM=(MQ## EX2.BASEQ 0015 D S')
/*
//STEPLIB  DD  DSN=SYS2.WMQ.SAMPLLOAD,DISP=SHR
//          DD  DSN=SYS1.SCEERUN,DISP=SHR
//          DD  DSN=SYS1.WMQ700.SCSQANLE,DISP=SHR
//          DD  DSN=SYS1.WMQ700.SCSQAUTH,DISP=SHR
//SYSDBOUT DD  SYSOUT=*

```

© Copyright IBM Corporation 2015

Figure 3-46. IBM MQ sample get program and parameters

WM3021.1

Notes:

This screen shows the MQGET JCL for one of the GET message samples. Your JCL member is the same except that it has the IBM MQ V8 libraries.

This program takes five parameters, which are documented in the JCL. There are two other items to notice:

- The fourth parameter presents an option for a destructive get, or a non-destructive (browse) type get. A “destructive” get removes a message from the queue. A browse, or non-destructive get, leaves the message in the queue. In practice, since IBM MQ should not be used like a database, most applications should do destructive gets to remove the messages from the queues.
- The second note is that the messages are printed to the job's SYSPRINT. If the output of this job does not find any messages in the selected queue, it returns a 2033, meaning that there are no messages to get.

Unit summary

- Explain the differences between the various types of queues
- Contrast the effect of selected queue definition attributes and development options on the handling of messages
- Demonstrate use of the Interactive System Productivity Facility (ISPF) panel interface to change queue manager and queue attributes
- Describe how to issue IBM MQ script commands (MQSC) by using the IBM MQ for z/OS CSQUTIL or the command line to create queues, including local, model, and alias queues
- Describe and use the display and monitoring functions that allow inquiry on queue-related application activities
- Demonstrate how to use supplied programs to put and retrieve messages in queues

© Copyright IBM Corporation 2015

Figure 3-47. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. You issue a DIS QSTATUS to check the last date and time a message was put to a queue. No information is shown. What do you do next?
 - a. Issue the DIS QMON command to determine existing settings to check what parameter needs to be set to capture queue statistics, and test again
 - b. Ensure that the QLOCAL MONQ attribute is set to QMGR to capture queue statistics, and test again
 - c. Check the STATIME attribute of the CSQZPARM module, reassemble, and test again
 - d. Ensure that both the queue manager and queue MONQ attribute are set to values that allow capture of queue statistics, and test again

© Copyright IBM Corporation 2015

Figure 3-48. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

1.

Checkpoint questions (2 of 2)

2. An application on queue manager ABC1 puts a message that uses queue manager name XYZ1 and queue name RETURNS.OUT in the MQOPEN. The queue manager did not find a QLOCAL called RETURNS.OUT. Where is the next place the queue manager looks to complete queue name resolution?
 - a. Look for a local transmission queue called ABC1
 - b. Look for a local QREMOTE called RETURNS.OUT
 - c. Look a local transmission queue called XYZ1
 - d. Look for a QMODEL called RETURNS.OUT
3. True or false: When using mixed-case characters in an object definition, you always enclose it in quotation marks to ensure that case is preserved.

© Copyright IBM Corporation 2015

Figure 3-49. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

2.

3.

Checkpoint answers (1 of 2)

1. You issue a DIS QSTATUS to check the last date and time a message was put to a queue. No information is shown. What do you do next?
 - a. Issue the DIS QMON command to determine existing settings to check what parameter needs to be set to capture queue statistics, and test again
 - b. Ensure that the QLOCAL MONQ attribute is set to QMGR to capture queue statistics, and test again
 - c. Check the STATIME attribute of the CSQZPARM module, reassemble, and test again
 - d. Ensure that both the queue manager and queue MONQ attribute are set to values that allow capture of queue statistics, and test again

Answer: d

© Copyright IBM Corporation 2015

Figure 3-50. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

2. An application on queue manager ABC1 puts a message that uses queue manager name XYZ1 and queue name RETURNS.OUT in the MQOPEN. The queue manager did not find a QLOCAL called RETURNS.OUT. Where is the next place the queue manager looks to complete queue name resolution?
 - a. Look for a local transmission queue called ABC1
 - b. Look for a local QREMOTE called RETURNS.OUT
 - c. Look a local transmission queue called XYZ1
 - d. Look for a QMODEL called RETURNS.OUT

Answer: c

3. True or false: When using mixed-case characters in an object definition, you always enclose it in quotation marks to ensure that case is preserved.

Answer: True

© Copyright IBM Corporation 2015

Figure 3-51. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 2



© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

9.1

Figure 3-52. Exercise 2

WM3021.1

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Create, alter, and display queues by using the command prefix string and MQSC commands at the system prompt, and the IBM MQ for z/OS ISPF panels
- Place and retrieve messages from queues with the batch sample programs
- Describe the behavior of various combinations of persistence in the application specification and queue definition attributes

© Copyright IBM Corporation 2015

Figure 3-53. Exercise objectives

WM3021.1

Notes:

Unit 4. Distributed queuing

What this unit is about

This unit describes the IBM MQ components that are required to exchange messages with other queue managers, and follows the path of a message from source to end point.

What you should be able to do

After completing this unit, you should be able to:

- Identify the distributed queuing components
- Describe queue name resolution
- Summarize the use of remote and transmit queues
- Describe message channel agents, listeners, and channel initiators
- Differentiate among the various types of channels
- Describe the object definitions that are necessary to connect queue managers
- Describe connectivity behaviors that client channel definition attributes control
- Demonstrate how to monitor and solve problems with channels

Unit objectives

- Identify the distributed queuing components
- Describe queue name resolution
- Summarize the use of remote and transmit queues
- Describe message channel agents, listeners, and channel initiators
- Differentiate among the various types of channels
- Describe the object definitions that are necessary to connect queue managers
- Describe connectivity behaviors that client channel definition attributes control
- Demonstrate how to monitor and solve problems with channels

© Copyright IBM Corporation 2015

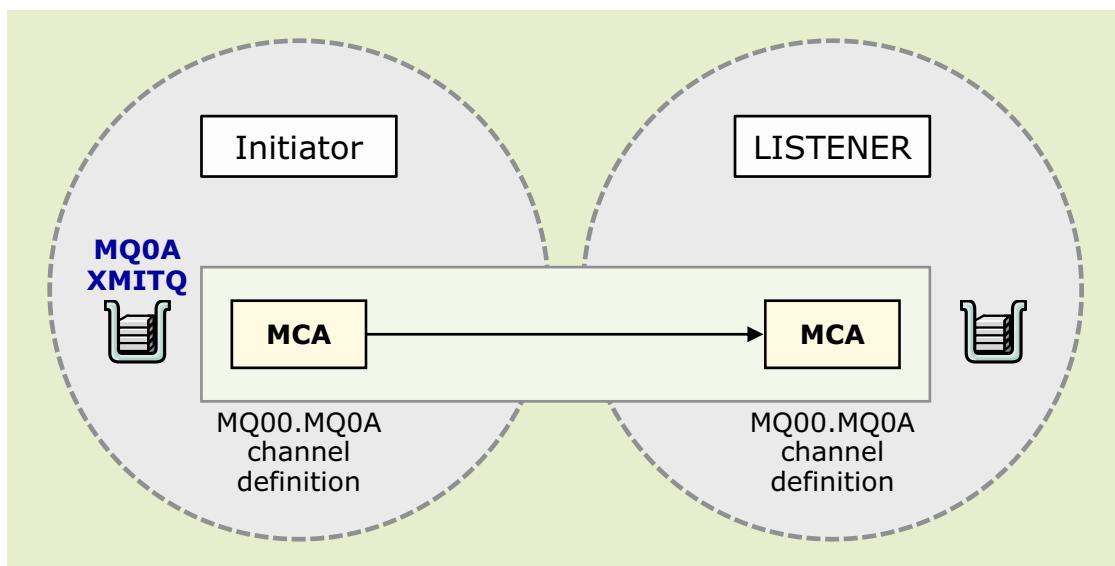
Figure 4-1. Unit objectives

WM3021.1

Notes:

Distributed queuing components

- Transmission queues
- Channel initiators
- Listeners
- Message channel agents
- Channels
- Channel exits



© Copyright IBM Corporation 2015

Figure 4-2. Distributed queuing components

WM3021.1

Notes:

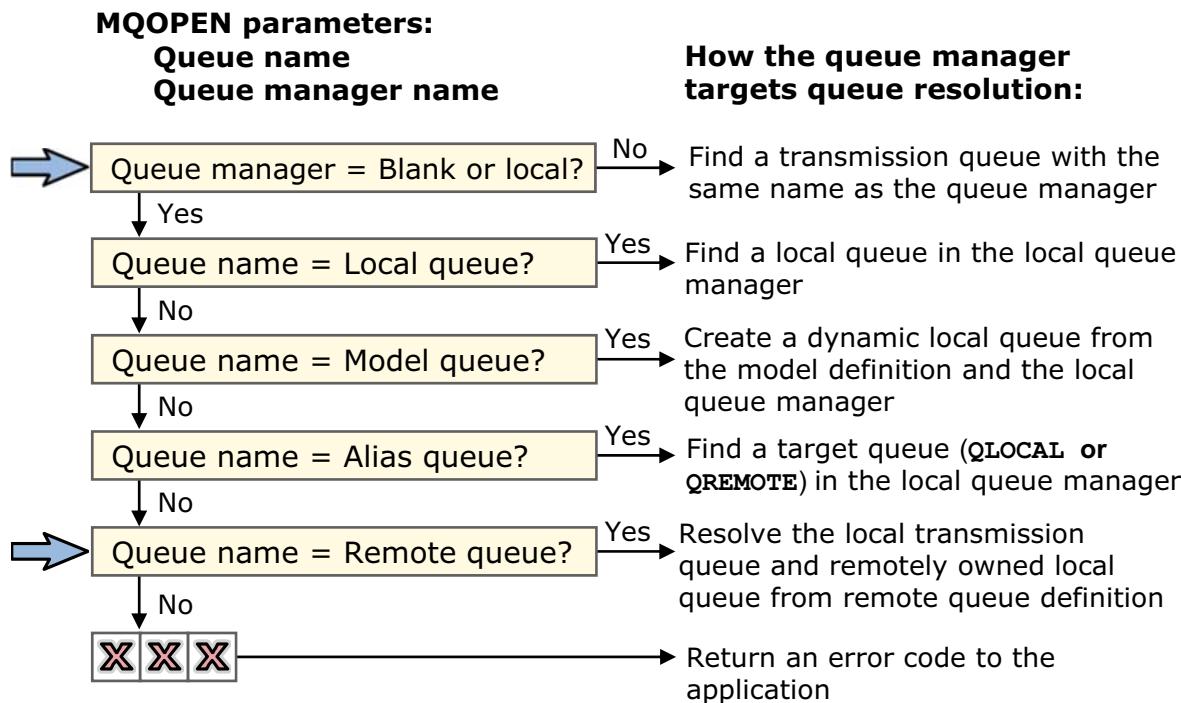
You looked at an IBM MQ summary, and had a closer look at queues in the administration unit. You now take a closer look at how the messages travel to other queue managers that use distributed queuing.

The components of distributed queuing should already be familiar to you:

- The channel initiator and listener, which are already configured in the queue manager
- Transmission queues, which are local queue of usage type XMITQ
- Message channel agent, the process that moves messages, within syncpoint
- Channels, which can be thought of as a predefined instance pair of the message channel agents

You did not yet look at channel exits. There are different types of exits, such as message exits and channel exits. Coding a channel exit becomes another piece of code to support. Many of the reasons that exits were made available have alternatives that are covered in the security unit. Authentication is now made available with the CONNAUTH feature of IBM MQ V8. The CHLAUTH capability handles channel authentication. SSL handles encryption for the channel.

Transmission queues and queue name resolution



© Copyright IBM Corporation 2015

Figure 4-3. Transmission queues and queue name resolution

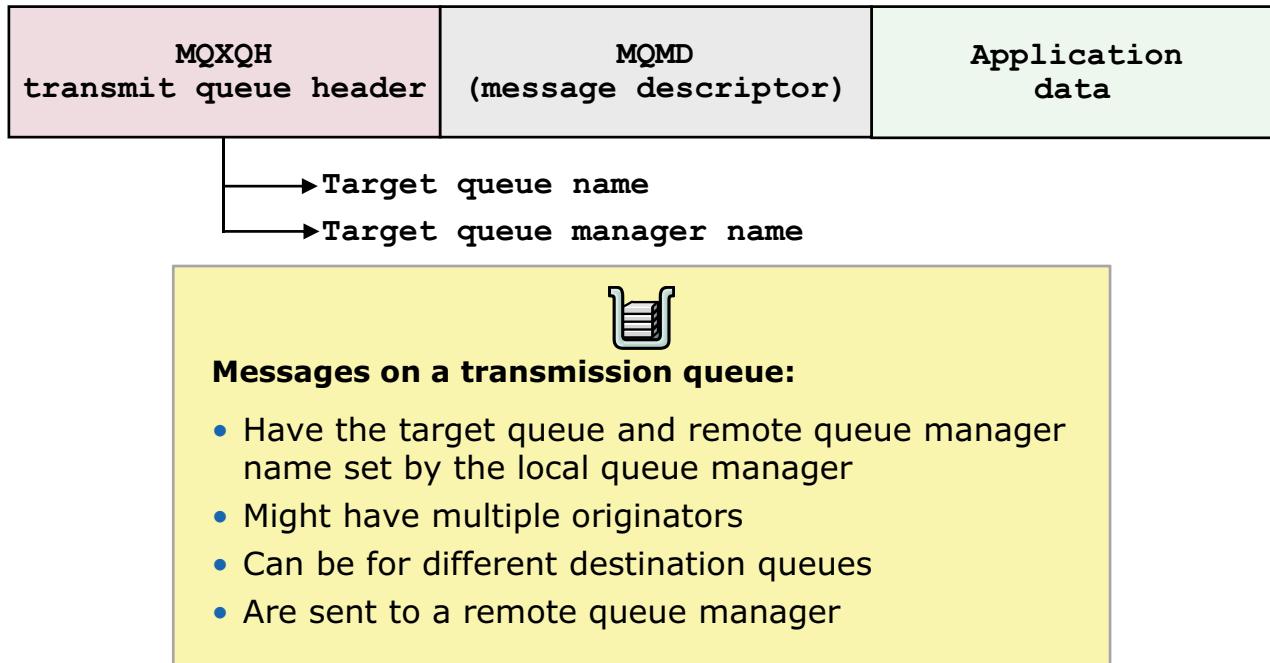
WM3021.1

Notes:

When working with distributed queuing, you always go back to your queue name resolution.

- If an application specifies a queue name in the MQOPEN, this queue name is used regardless of the existence of a QREMOTE. The reason is that queue name resolution first checks for a transmit queue name that matches the name of the queue manager specified.
- Applications that use a QREMOTE definition should not use a queue manager name in the MQOPEN. A QREMOTE definition provides more flexibility.

Transmission header



Only the queue manager should put messages to transmit queues.

© Copyright IBM Corporation 2015

Figure 4-4. Transmission header

WM3021.1

Notes:

As a message is placed on a transmit queue, the queue manager places a transmit queue header in the message. This transmit queue header is used only during the message path to the destination queue manager and is removed from the message when the destination queue is reached.

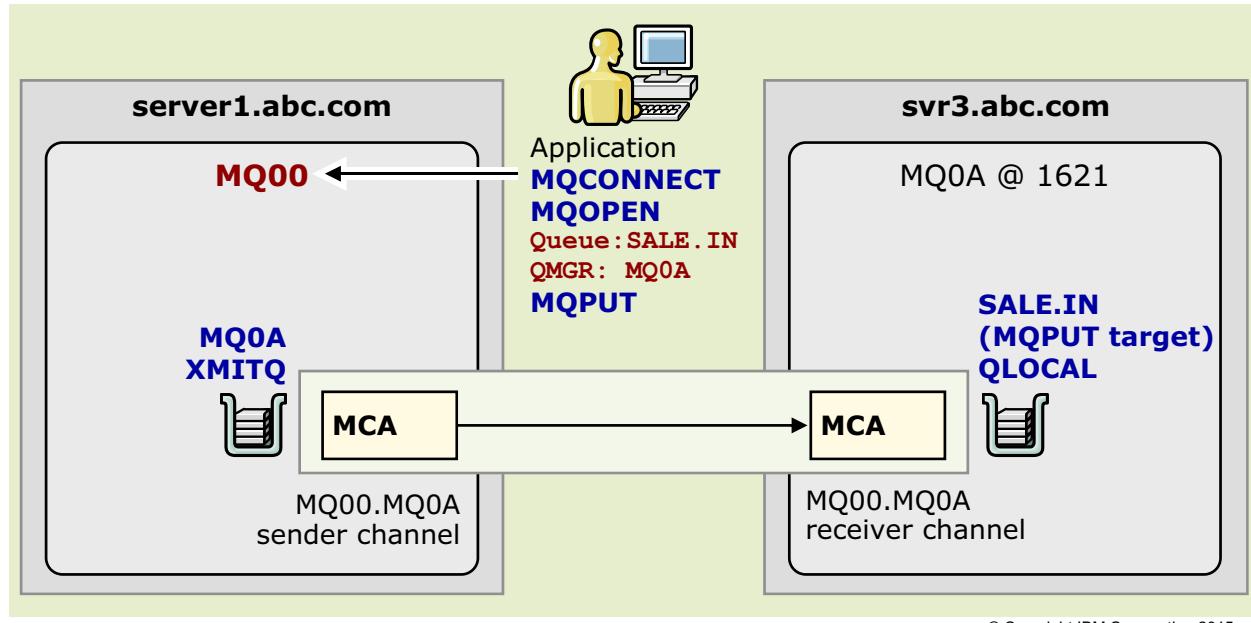
The use of a QREMOTE object to define a queue manager alias influences how the transmit queue header is formatted. There are three roles to a queue manager alias:

- Sending: Remapping the queue manager name
- Sending: Changing or determining the transmit queue
- Receiving: Determining whether the local queue manager is the intended destination for the message

Case 1: Transmission queues and queue name resolution with queue manager name in MQOPEN

What happens when an application:

- Connects to the MQ00 queue manager
- Puts a message to remote queue manager MQ0A, local queue SALE.IN



© Copyright IBM Corporation 2015

Figure 4-5. Case 1: Transmission queues and queue name resolution with queue manager name in MQOPEN

WM3021.1

Notes:

When the MQOPEN uses a queue manager name that matches a transmission queue name, the queue manager places that queue manager name in the transmission queue header.

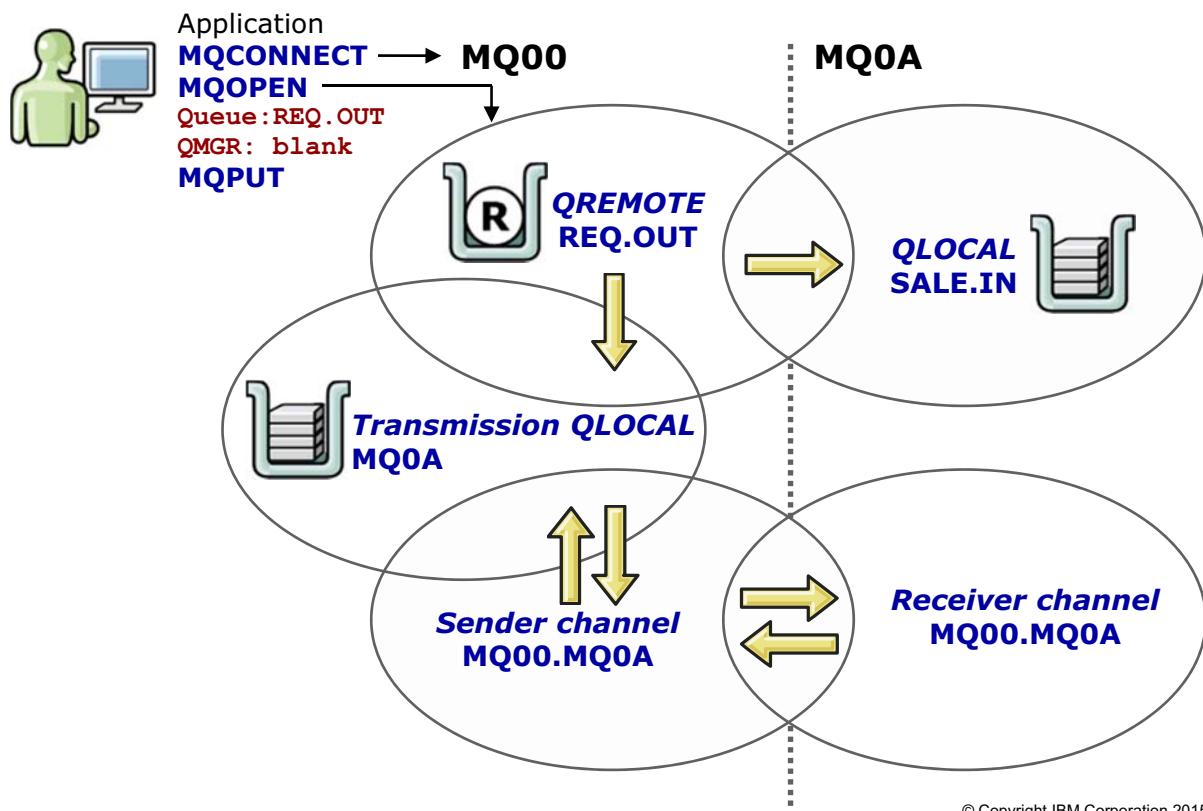
If a match is not found but there is a default transmission queue, the queue manager uses the default transmission queue to place the message.



Important

Applications should never put messages directly in a transmission queue.

Case 2: Message transfer with a QREMOTE definition



© Copyright IBM Corporation 2015

Figure 4-6. Case 2: Message transfer with a QREMOTE definition

WM3021.1

Notes:

A common case is to have the application use a full QREMOTE definition. When the remote queue definition is used, the queue manager uses the name in the RQMNAME field of the QREMOTE definition for the transmit queue header.

Case 3: Multi-hopping

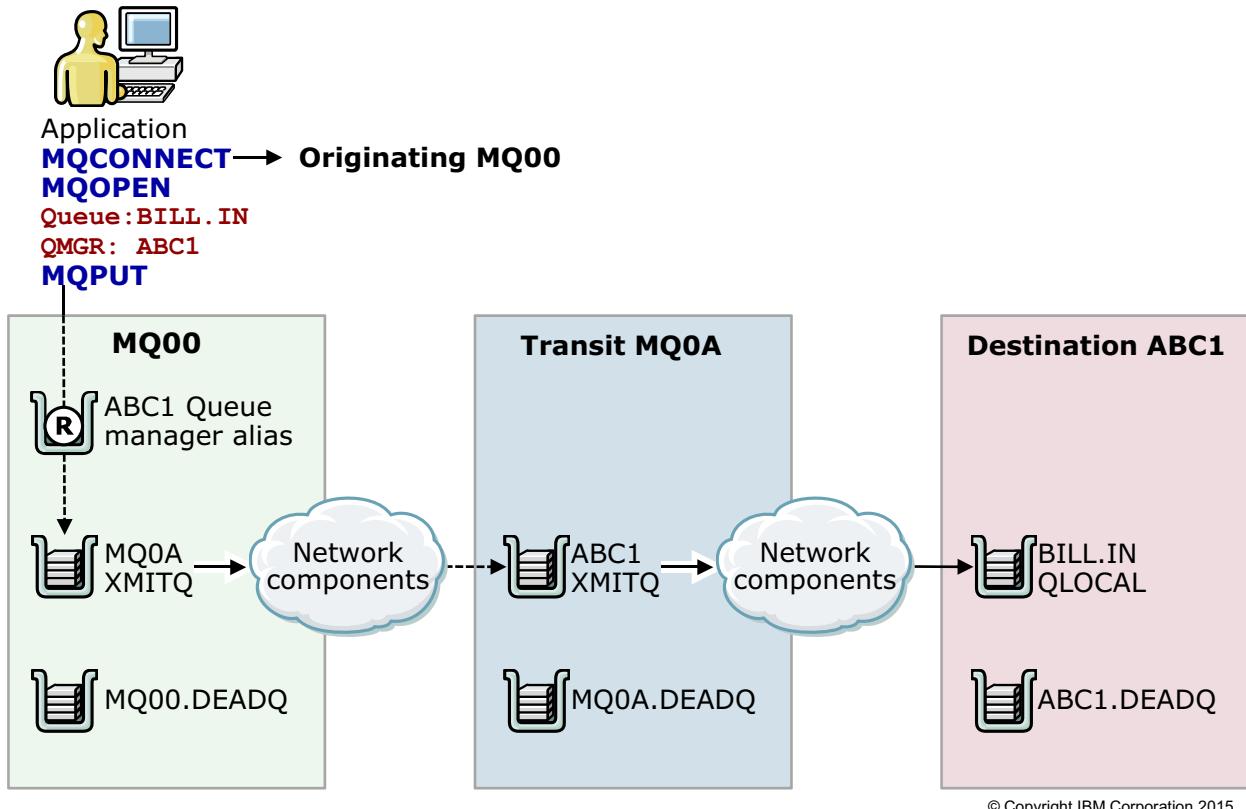


Figure 4-7. Case 3: Multi-hopping

WM3021.1

Notes:

In the case of multi-hopping:

- At MQ00, all messages for ABC1 are intercepted by the ABC1 queue manager alias; value ABC1 is placed in the transmission header. The queue manager alias definition at MQ00 is:


```
DEFINE QREMOTE (ABC1) RNAME(' ') RQMNAME(ABC1) XMITQ(MQ0A)
```
- MQ00 does have a transmission queue defined, which is named MQ0A.
- The MQ0A queue manager receives the messages from MQ00 with ABC1 on the transmission queue header. However, it does not alter the transmission queue header because the name of the destination is already set to queue manager, ABC1, and queue manager MQ0A has a transmission queue called ABC1.

Queue name resolution and queue manager alias scenarios

MQOPEN parameter scenarios that are connected to queue manager MQ00	Given: Transmission queue MQ0A is defined in queue manager MQ00 for all scenarios • QREMOTE definition is:	Results
Case 1: • Queue: SALE.IN • Queue manager: MQ0A	No QREMOTE queue defined	• Message is routed to queue SALE.IN on queue manager MQ0A
Case 2: • Queue: REQ.OUT • Queue manager: Blank	DEF QREMOTE (REQ.OUT) + RQMNAME (MQ0A) + RNAME (SALE.IN) + XMITQ (MQ0A)	• Basic QREMOTE scenario • Message is routed to queue SALE.IN on queue manager MQ0A
Case 3: Multi-hopping scenario • Queue: SALE.IN • Queue manager: ABC1	DEFINE QREMOTE (ABC1) + RNAME (' ') + RQMNAME (ABC1) + XMITQ (MQ0A)	• Routed to MQ0A first • Next, MQ0A has an ABC1 transmission queue definition and routes the message to queue manager ABC1 where it is placed in the SALE.IN queue
Basic QM name remap • Queue: SALE.IN • Queue manager: ABC1	DEF QREMOTE (ABC1) + RQMNAME (MQ0A)	• Message is routed to queue SALE.IN on queue manager MQ0A

© Copyright IBM Corporation 2015

Figure 4-8. Queue name resolution and queue manager alias scenarios

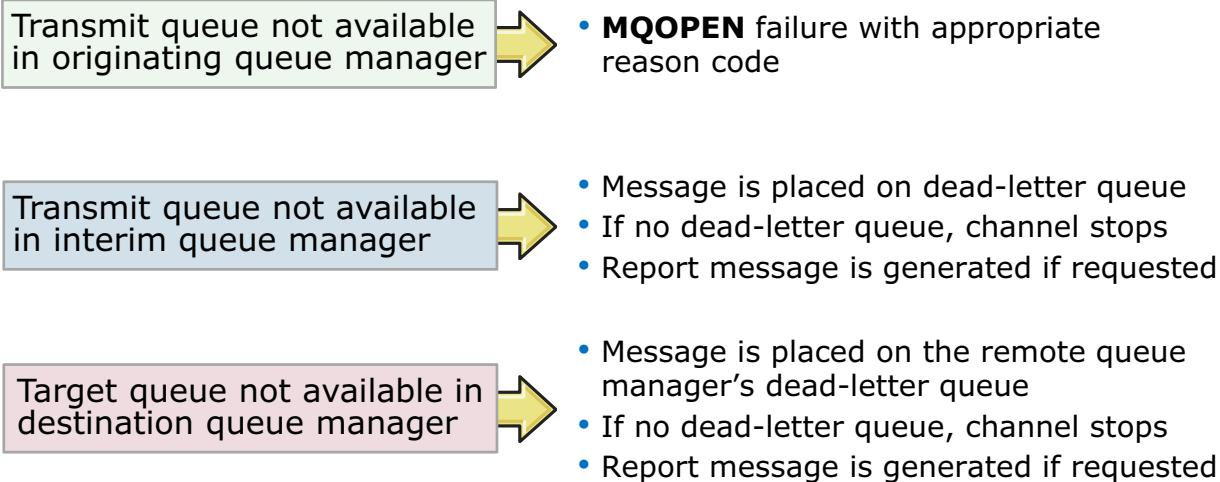
WM3021.1

Notes:

This slide summarizes some of the uses of the QREMOTE object definition.

The assumption in the scenarios in this table is that all channels are correctly defined across the queue managers.

What happens when the destination queue is not found



Reminder

- The queue manager's dead-letter queue is identified:
 - During queue manager configuration in the `CSQ4INYG` member of a queue manager started task by inclusion of `ALTER QMGR DEADQ (QUEUE.NAME)`
 - After configuration issue `ALTER QMGR DEADQ (QUEUE.NAME)`

© Copyright IBM Corporation 2015

Figure 4-9. What happens when the destination queue is not found

WM3021.1

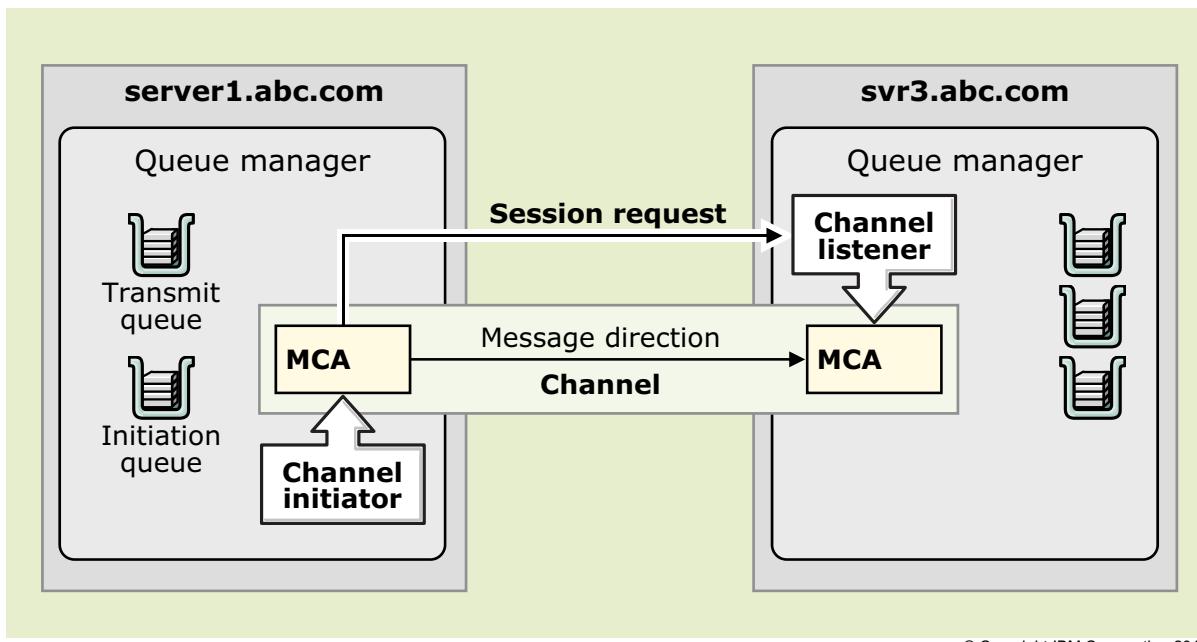
Notes:

This slide corresponds with the three queue managers in the multi-hopping scenario. It walks through what happens if there are failures along the path.

- If the local queue manager cannot find a route to the destination queue manager, assuming that no default transmit queue is specified in the queue manager, the **MQOPEN** fails with a reason code.
- If a transit queue manager cannot route a message, it places the message on the DLQ, and sends a report to the ReplyToQ in the message (if events are enabled and reports requested).
- If the destination queue manager cannot find the queue, the message is placed on the DLQ and a report is returned (if configured).
- If there is no DLQ available, the message remains on the transmit queue and the channel stops.
- If configured, an event is generated and placed on the channel event queue on the system where it took place.

Channel initiators and listeners

- z/OS listeners include TCP/IP and APPC/LU62 implementations



© Copyright IBM Corporation 2015

Figure 4-10. Channel initiators and listeners

WM3021.1

Notes:

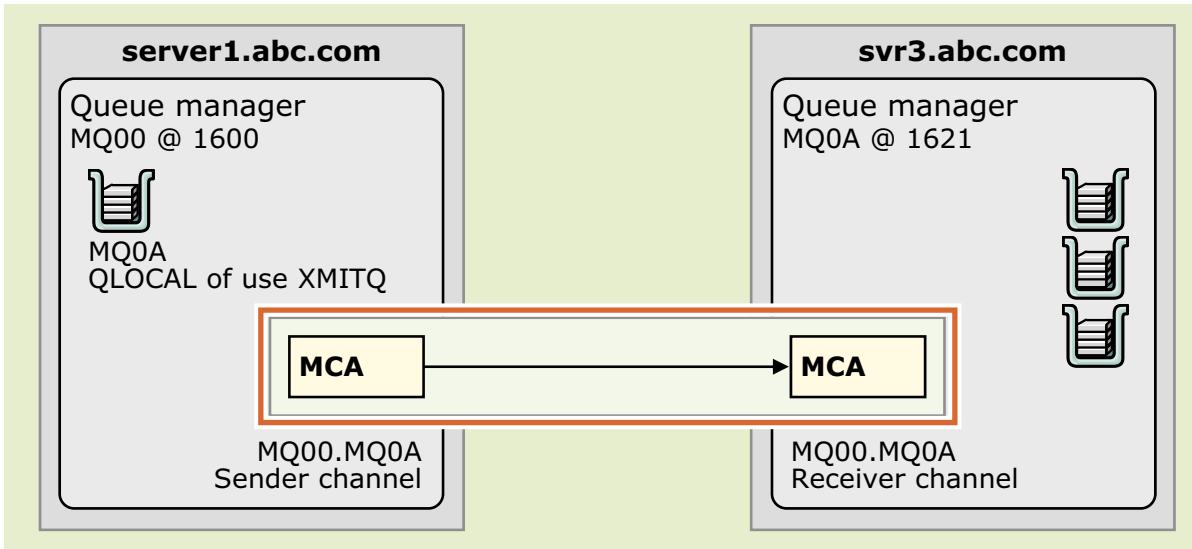
Channel initiators control the resources and message channel agents that move messages across queue managers. Listeners, as the name implies, listen for the arrival of messages for a queue manager.

A *channel initiator* acts as a *trigger monitor* for sender channels because a transmission queue might be defined as a triggered queue. When a message that satisfies the triggering criteria arrives on a transmission queue, a message is sent to the initiation queue, triggering the channel initiator to start the appropriate sender channel. The server channels can also be started in this way if the connection name of the partner is specified in the channel definition. As a result, the channel can be started automatically, based on the messages that arrive on the appropriate transmission queue.

A *channel listener* is required to start the receiving (responder) MCAs. Responder MCAs are started in response to a start request from the caller MCA. The channel listener detects the incoming network requests and starts the associated channels.

Message channel agent

- A message channel agent or MCA is a program that controls message transfer
- A caller MCA is associated with a corresponding responder MCA
- IBM MQ z/OS provides MCA implementations for TCP/IP or APPC/LU62



© Copyright IBM Corporation 2015

Figure 4-11. Message channel agent

WM3021.1

Notes:

A message channel agent sets up communications with its partner channel agent, and picks up the messages from the transmission queue on the sending side of the connection. The receiving channel agent places the messages on the target queues.

Channel categories and types

- Two main categories of channels
 - Message channels: **Distributed** or clustered
 - Client (or MQI) channels
- A **message channel** is a combination of a sending MCA, the network connection, and the receiving MCA
- Message channels are defined in a caller and responder pair
- Distributed message channel types:**
 - Sender
 - Server
 - Receiver
 - Requester

Valid distributed message channel pair combinations

Caller	Flow direction	Responder
Sender		Receiver
Server		Receiver
Server		Requester
Requester		Server
Requester		Sender Callback

© Copyright IBM Corporation 2015

Figure 4-12. Channel categories and types

WM3021.1

Notes:

A sending MCA, the communication link, and the receiving MCA form a one-way message path that is called a channel. There are several types of distributed message channels, and the main difference is the way that they get started. The sender-receiver pair is the most prevalent.

MQI channels are not IBM MQ server channels. They are IBM MQ client channels. They are not one-way but used with two-way communication. MQI channels are discussed in the clients unit.

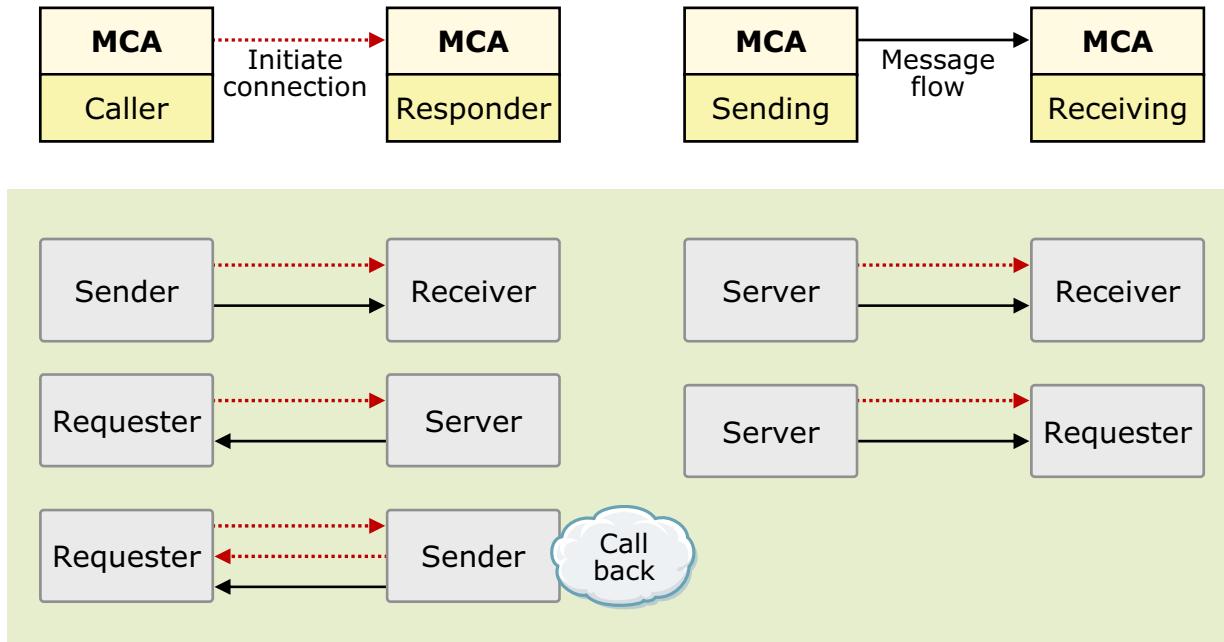
In this course, you work with a sender-receiver pair and learn about a callback setup.



Important

Unless an exceptional condition occurs, always use the sender channel to start and stop sender-receiver pairs. Do not start or stop the receiver side of the channel.

MCA actions per distributed channel pair combinations



© Copyright IBM Corporation 2015

Figure 4-13. MCA actions per distributed channel pair combinations

WM3021.1

Notes:

A message channel agent (MCA) that initiates a communications connection is called a *caller* MCA. Its partner MCA, which is started at the other end of the connection, is called a *responder* MCA.

The definition of a channel at one end must specify a type that is compatible with the type specified in the definition at the other end. The five possible combinations of types are depicted on the visual.

- A sender MCA can initiate a communications connection with a receiver MCA and then send messages to it. This combination is the most common.
- A server MCA can also initiate a communications connection with a receiver MCA and then send messages to it. However, to proceed, the channel definition is available to the server.

MCA must specify sufficient information to enable it to act as a caller MCA. That is, the channel definition must at least specify the CONNAME parameter. Such a server is called a *fully defined*, or *fully qualified*, server.

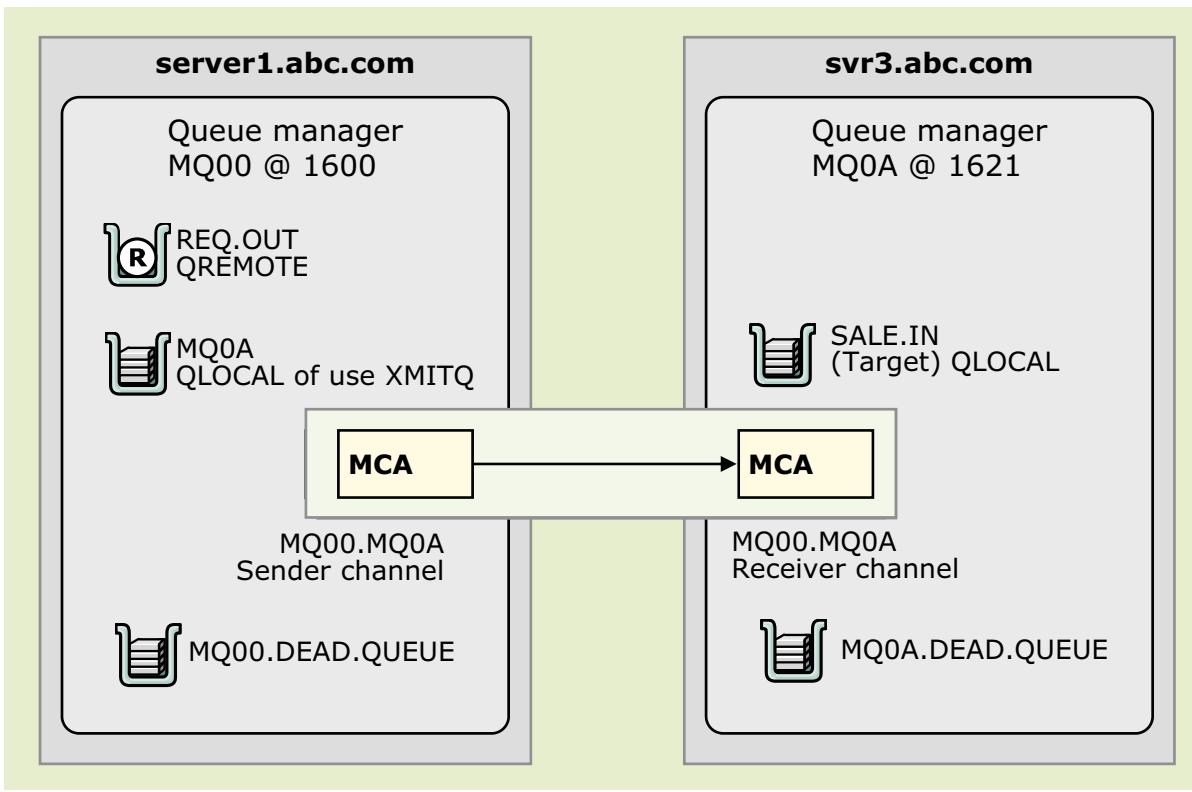
- A requester MCA can initiate a communications connection with a server MCA, which then sends messages to the requester MCA. Therefore, in this combination, the server MCA is a responder and does not need to be fully defined. However, if it is fully defined, a server MCA can initiate a communications connection with a requester MCA and then send messages to it.

- A requester MCA can initiate a communications connection with a sender MCA, which promptly terminates the connection. The sender MCA then initiates a communications connection according to the information in the channel definition at its end and sends messages to the partner MCA that it started. This combination is known as *call-back*.

The relationships among the various roles can be summarized as follows:

- A caller MCA might be a sender, a fully defined server, or a requester.
- A responder MCA might be a sender, a receiver, a server, or a requester.
- A sending MCA might be a sender or a server.
- A receiving MCA might be a receiver or a requester.

Connecting queue managers



© Copyright IBM Corporation 2015

Figure 4-14. Connecting queue managers

WM3021.1

Notes:

Channels are used to transmit messages between queue managers. IBM WebSphere MQ interfaces to networks are called *message channel agents* (MCA).

- The **SENDing** message channel agent **MQGETs** messages from the transmit queue under **SYNCPOINT**.
- It transmits the message to the receiving MCA, which **MQPUTs** it to the destination queue, under **SYNCPOINT**, first **MQOPENing** it if necessary.
- At the end of a batch of messages, the two MCAs agree to the batch status and **MQCMMIT** their queues.
- All of this work might be logged for recovery. It is also possible *not* to use **SYNCPOINT** for nonpersistent messages *only*. The **NPMSSPEED** parameter on the **SENDER** (or **SERVER**) channel definition can be used to allow nonpersistent messages to flow immediately.

Definitions that are required to send messages from MQ00 to MQ0A

At queue manager MQ00

```
DEFINE QLOCAL (MQ0A) +
  USAGE (XMITQ) +
  INITQ (SYSTEM.CHANNEL.INITQ) +
  TRIGGER +
  TRIGDATA (MQ00.MQ0A)

DEFINE QREMOTE (REQ.OUT) +
  RQMNAME (MQ0A) +
  RNAME (SALE.IN) +
  XMITQ (MQ0A)

DEFINE CHANNEL (MQ00.MQ0A) +
  CHLTYPE (SDR) +
  TRPTYPE (TCP) +
  CONNAME ('svr3.abc.com(1621)') +
  XMITQ (MQ0A)
```

At queue manager MQ0A

```
DEFINE QLOCAL (SALE.IN)

DEFINE CHANNEL (MQ00.MQ0A) +
  CHLTYPE (RCVR) +
  TRPTYPE (TCP)
```



© Copyright IBM Corporation 2015

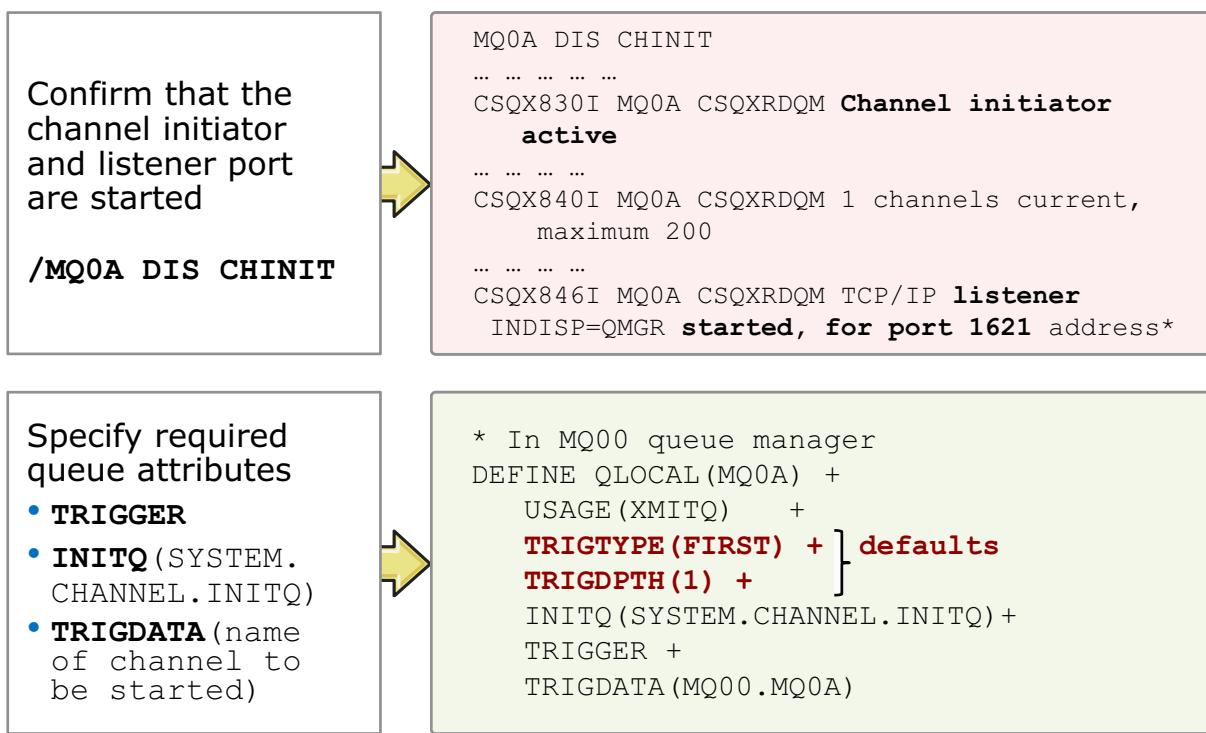
Figure 4-15. Definitions that are required to send messages from MQ00 to MQ0A

WM3021.1

Notes:

In the first unit, you saw a description of the definitions that are needed for a one-way sender-receiver channel pair definition. Definitions that are needed for queue manager MQ00 are on the right, and for queue manager MQ0A are on the left. This definition sends messages from MQ0A to MQ00.

Automatically starting z/OS channels: Triggering



© Copyright IBM Corporation 2015

Figure 4-16. Automatically starting z/OS channels: Triggering

WM3021.1

Notes:

Triggering was introduced in the first unit. You now look at how to trigger a channel.

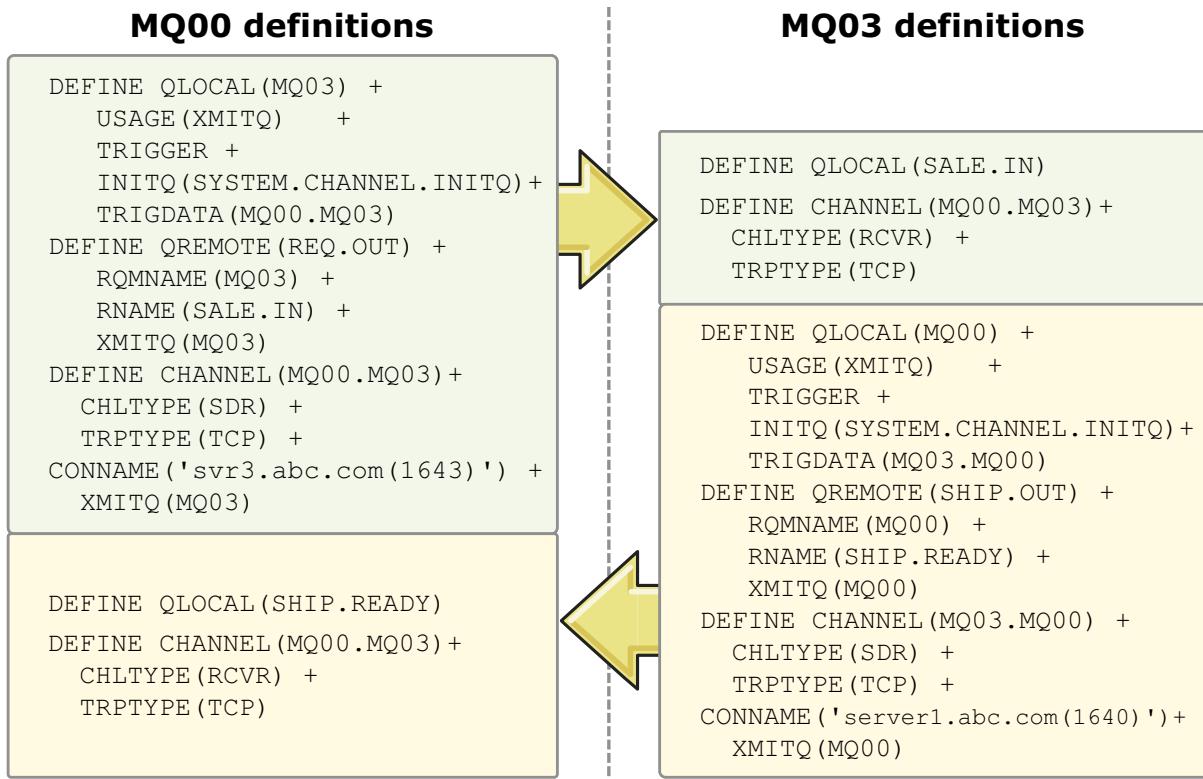
Except for checking the channel initiator and listener, all work is done in the transmit queue:

- Do a **/DIS CHINIT** and ensure that the channel initiator and listener port for the queue manager are running.
- Change the transmit queue QLOCAL to use TRIGGER. TRIGGER is an attribute in itself; toggle either to TRIGGER or to NOTRIGGER. The default attribute is NOTRIGGER, so it must change.
- INITQ is set to SYSTEM.CHANNEL.INITQ
- TRIGDATA holds the name of the channel to be triggered.
- You can leave TRIGTYPE and TRIGDPTH set to the default values.
- Check the channel status. If the status is STOPPED, you need to manually start the channel the first time.

A channel runs, and stops at predetermined intervals if no message traffic is coming across.

Triggering ensures that when messages arrive, the channel restarts automatically.

Two-way definitions MQ00 to MQ03 and MQ03 to MQ00



© Copyright IBM Corporation 2015

Figure 4-17. Two-way definitions MQ00 to MQ03 and MQ03 to MQ00

WM3021.1

Notes:

This slide shows definitions that are needed for two-way message exchange between the two queue managers. Definitions from MQ03 to MQ00 are similar to the definitions from MQ00 to MQ03 with names and some attribute values changed.



Hint

Part of your lab exercise is to define sender-receiver channels to queue manager MQ00. This definition process is similar to the ones that are shown on the MQ03 definition column, bottom box with transmission queue MQ00, QREMOTE, and sender channel. For the lab, the receiver channel is predefined on MQ00. Connectivity information, as shown, is for illustrative purposes only and is not correct for MQ00. The lab exercise has the correct connectivity information.

Callback requester sender definitions MQ0A MQ00

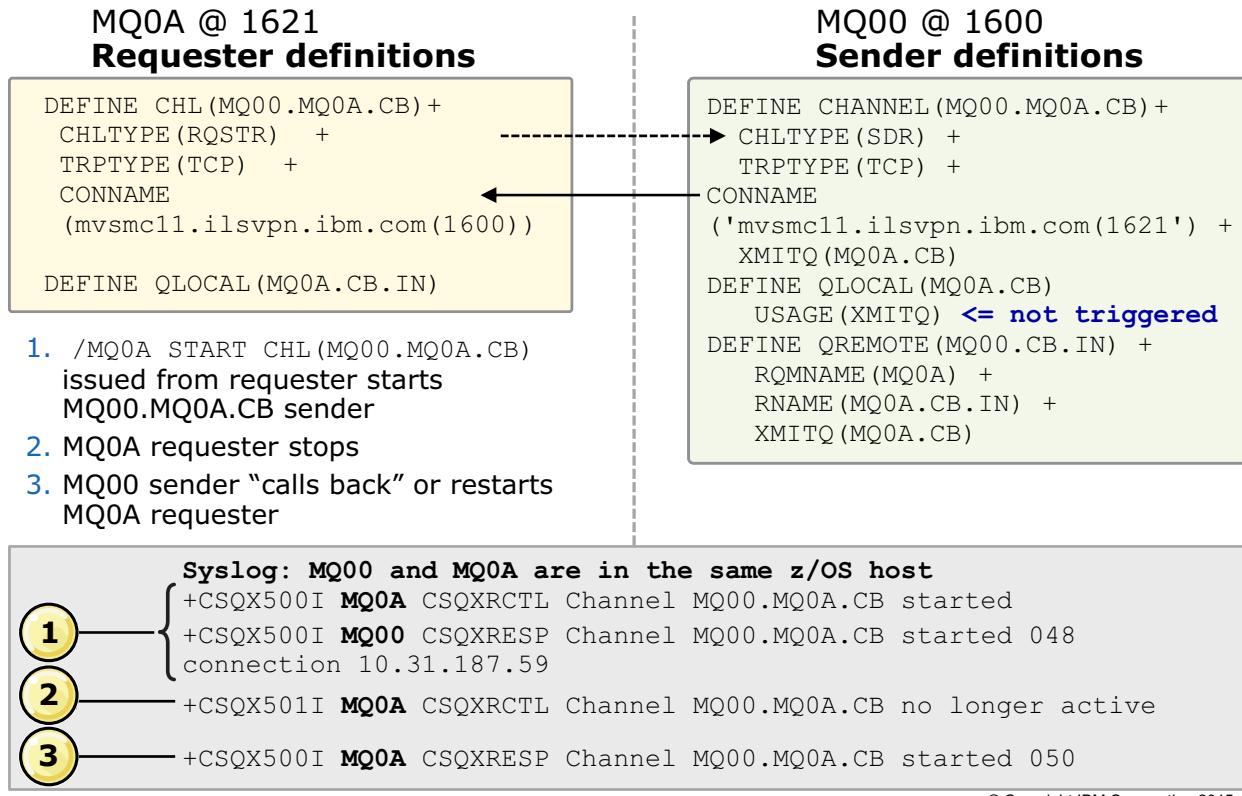


Figure 4-18. Callback requester sender definitions MQ0A MQ00

WM3021.1

Notes:

This slide walks through the definition and start of a “callback,” or requester-sender channel.

While these channels can be confusing, one item that is helpful is to notice how the definition of a requester channel has a CONNNAME attribute, but not a transmission queue (XMITQ). This information indicates that a requester does not send messages.

In a callback, the requester “wakes up” the sender and then stops; then, the sender starts to use the requester as its peer.

MQ00 and MQ0A are in the same z/OS server, so note the queue manager name in the log display.

1. The requester channel is started at the MQ0A queue manager, and in turn it starts the sender channel at the MQ00 queue manager.
2. The requester channel at MQ0A ends.
3. The sender channel restarts the requester channel at MQ0A, so the flow of messages is one-way from MQ00 to MQ0A, and transmit queue MQ0A at the MQ00 queue manager is used.

Channel attributes: Sender channel partial list

```

CHANNEL(MQ00.MQ0A)
CHLTYPE(SDR)
QSGLDISP(QMGR)
XMITQ(MQ0A)
DEFCDISP(PRIVATE)
TRPTYPE(TCP)
CONNNAME(svr3.abc.com(1621))
DESCR( )
MCANAME( )
MODENAME( )
TPNAME( )
DISCINT(6000)
SHORTRTY(10)
SHORTTMR(60)
LONGRTY(999999999)
LONGTMR(1200)
SCYEXIT( )
SCYDATA( )
MSGEXIT( )
MSGDATA( )
SENDEXIT( ) RCVEXIT( )
RCVDATA( )
PROPCtl(COMPAT)
SEQWRAP(999999999)

```

```

... ... ... ...
CONVERT(NO)
BATCHINT(0)
BATCHHB(0)
KAINT(AUTO)
MONCHL(QMGR)
... ...
SSLCIPH( )
SSLPEER( )
CERTLBL( )
BATCHLIM(5000)
USEDLQ(YES)
STATCHL(QMGR)
MCAUSER( )
LOCLADDR( )
BATCHSZ(50)
MAXMSGL(4194304)
COMPHDR(NONE )
COMPMSG(NONE )
HBINT(300)
NPMSPED(FAST)
... ...

```

© Copyright IBM Corporation 2015

Figure 4-19. Channel attributes: Sender channel partial list

WM3021.1

Notes:

This slide contains an example of the channel definition that uses defaults.

When the CONNAME attribute is coded, including a port number, although a definition displays without apostrophes, always use apostrophes to fit the syntax with double parentheses, such as:

```
DEFINE CHANNEL(MQ00.MQ0A) CHLTYPE(SDR) TRPTYPE(TCP) +
CONNNAME('svr3.abc.dom(1621)') XMITQ ... ... ...
```

Here is a description of selected attributes:

- **BATCHSZ**: Batch size, an integer.

The maximum number of messages to be sent before a syncpoint is taken. If the transmission queue is empty before this number of messages is transmitted, a syncpoint is taken as well.

- **DISCINT**: Disconnect interval, a time value.

When there are no messages on the transmission queue for this duration, the sending MCA disconnects the channel. The channel enters the **INACTIVE** status and is restarted automatically when messages arrive for transmission.

- **HBINT**: Heartbeat interval, a time value.

The sending MCA sends heartbeat flows when there are no messages to transmit. The

heartbeat confirms that the receiving MCA is still alive, and offers the receiving MCA a way to disconnect regularly.

- **BATCHINT:** Batch interval, a time value.

During this time, the channel keeps a batch open even if the transmission queue is empty, resulting in a delay of commit processing. The value that is specified does not take effect as a wait interval for every **MQGET** from the transmission queue, but the sum of the **MQGET WAIT** time is limited to that value.

- **NPMSPEED:** Nonpersistent message speed, which is NORMAL or FAST.

Determines whether nonpersistent messages are transmitted within (**NORMAL**) or without (**FAST**) syncpoint control.

- **MAXSMGL:** Maximum message length, an integer number.

The maximum length of a message that can be transmitted on the channel.

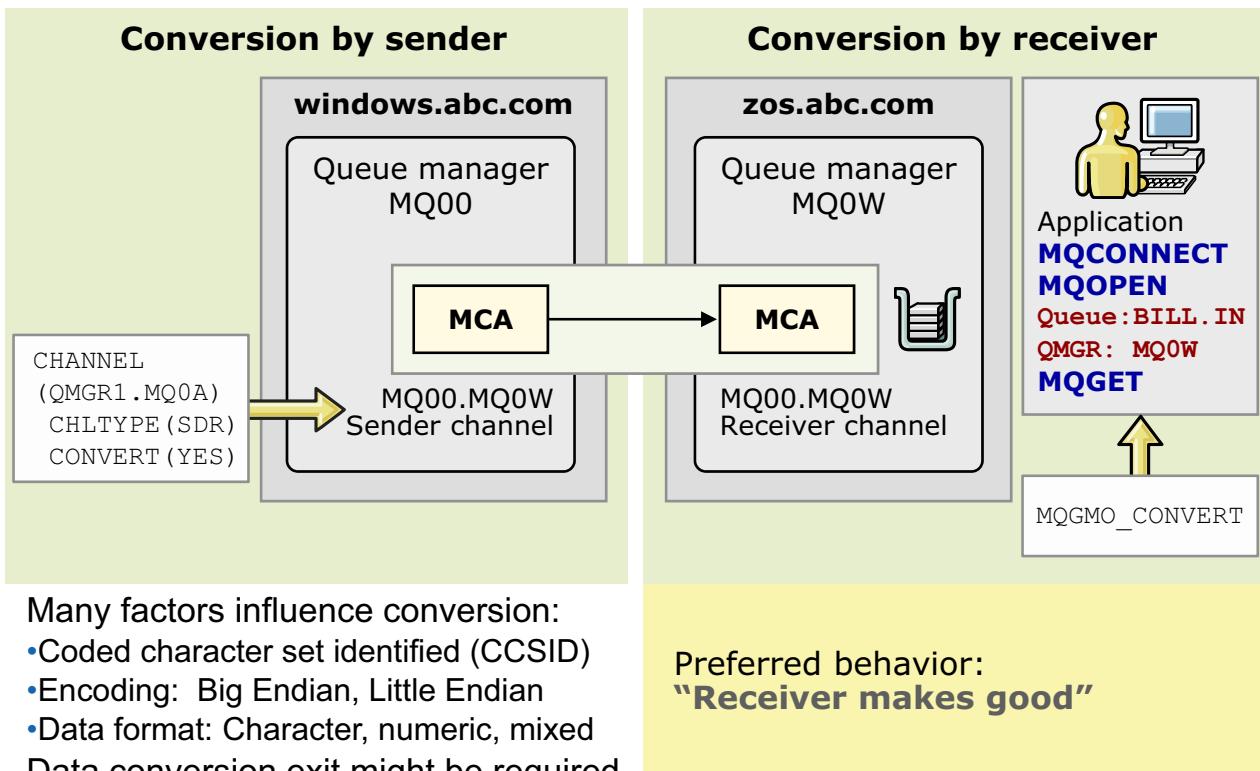
- **MRRTY:** Message retry, an integer number.

The number of times a receiving channel tries again before it decides it cannot deliver the message.

- **MRTMR:** Message retry, a time value (in milliseconds).

The minimum interval of time that must pass before the receiving channel can try the **MQPUT** operation again.

Data conversion: Choose one of two options



© Copyright IBM Corporation 2015

Figure 4-20. Data conversion: Choose one of two options

WM3021.1

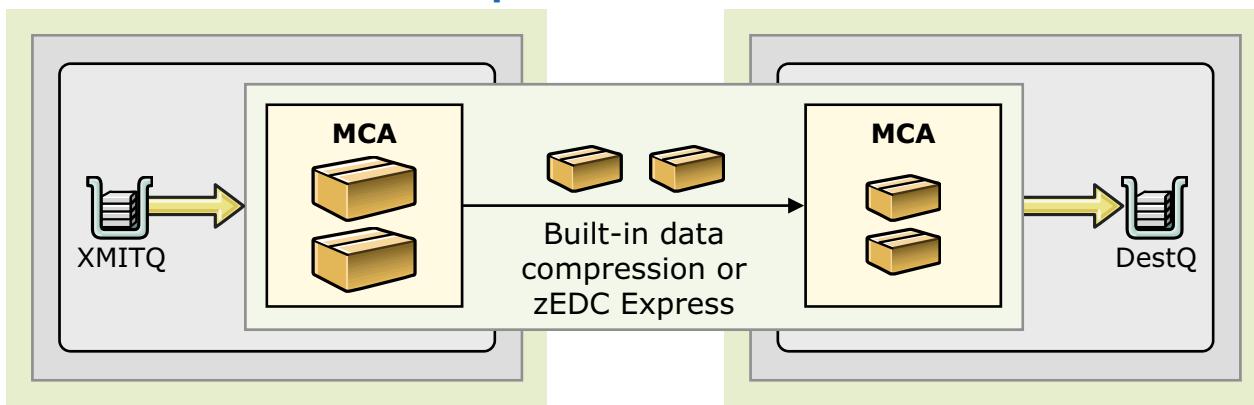
Notes:

Moving application data between different platforms is a major purpose for using IBM WebSphere MQ. Therefore, there is a need for data conversion, as data is represented differently on different platforms.

- IBM WebSphere MQ system structures, such as headers and predefined message formats, are converted automatically, based on fields in the message descriptor.i
- The sending channel (MCA) can do the conversion between **GETting** messages from the transmission queue and forwarding it to the network. Alternatively, the conversion can be done when the application **GETs** its messages from the destination queue. The second method is preferred.
- The reading application must request the data conversion by use of the Get Message Option **MQGMO_CONVERT**.
- Depending on the message structure, it might be necessary to provide exit code to do the data conversion.

The old saying in IBM MQ, since the time when the product was called MQSeries, is that the preferred practice is “receiver makes good.” This choice is the most logical and allows flexibility if, for instance, an application needs to change paths to multi-hop across queue managers.

COMPMSG: Data compression



- Compression options are part of a channel definition
 - For V8 and later, queue managers can offload compression to zEDC Express if the zEDC Express facility is enabled in z/OS
 - Channel initiator requires READ authority to the FPZ.ACCELERATOR.COMPRESSION profile
 - Configure the channel with COMPMSG(ZLIBFAST) at both the sending and receiving ends
- Headers and data can be selected for compression separately
- Optimization can be for speed (performance) or the highest compression rate
- Used methods and results can be displayed as part of the channel monitor status

© Copyright IBM Corporation 2015

Figure 4-21. COMPMSG: Data compression

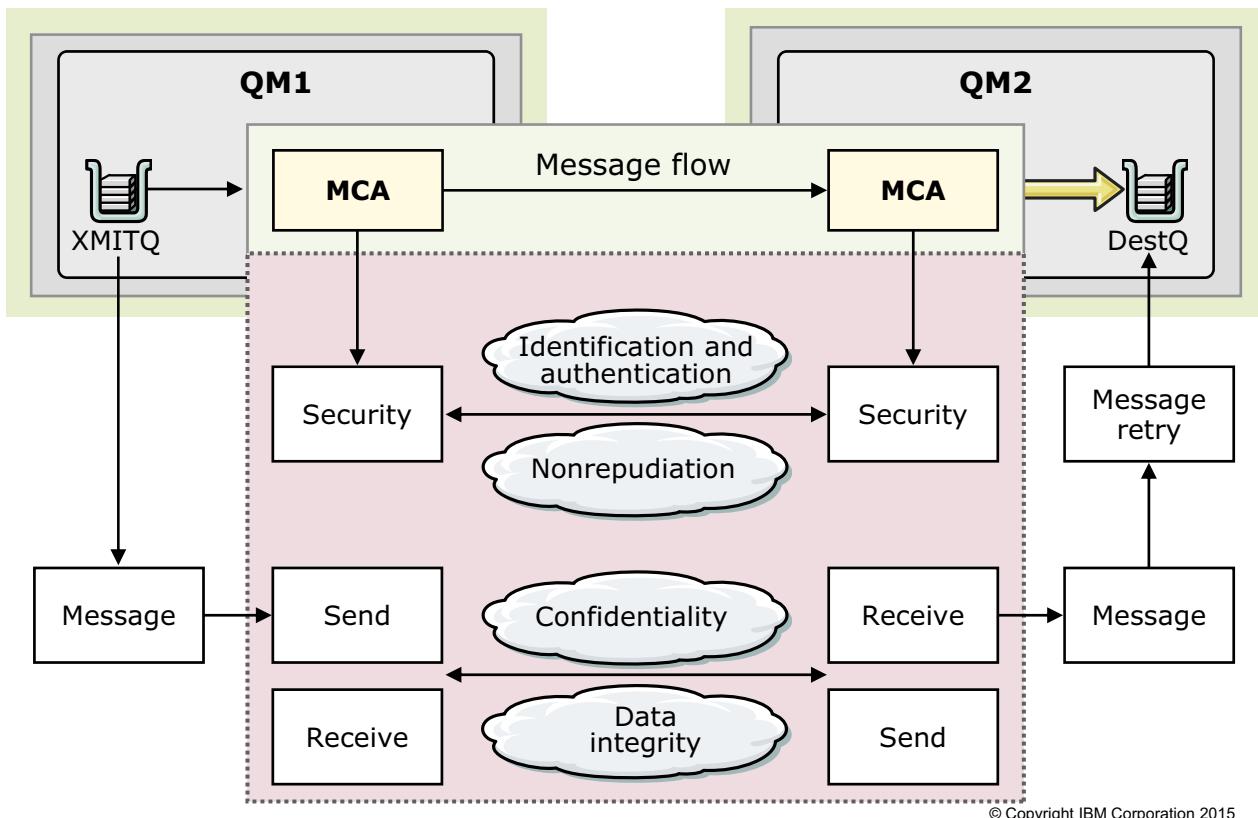
WM3021.1

Notes:

Note the following information about data compression:

- Whether you are going to implement compression, or already have a compression algorithm in use, check whether your channels can benefit from using the z/OS-based zEDC Express compression, if available in your z/OS environment.
- By using channel definition parameters, the channel can determine the data compression, if at all; and, if so, what type of compression is used.
- The MCAs negotiate the actual compression method that is used at run time, on a per-message basis. The message exit on the sending side can be involved in this procedure to take the final decision.
- Channel online monitoring allows for inquiring on the use and effectiveness of the actual compression that is done on each channel.
- Built-in IBM WebSphere MQ compression algorithms are optimized for messages that use XML as application data.

Channel exits



© Copyright IBM Corporation 2015

Figure 4-22. Channel exits

WM3021.1

Notes:

There are several exit points where you can tailor the behavior of the MCAs according to your requirements. All the channel exits normally must be supplied in pairs, which means that if the exit is activated on one side of the channel, then it also must be activated on the other side.



Warning

Before undertaking the effort of creating an exit, which is a customization that must be maintained, check whether the native IBM MQ capabilities can be used instead of the exit, particularly in the areas of:

- Authentication
- Encryption of messages that travel across queue managers (link level)
- Authorization
- Compression and decompression
- Encryption of data at rest (application level)

- **Security exit**

The security exit is called during channel setup. Its purpose is to allow the addition of code, which can check the security credentials of the partner MCA. This process is known as *authentication*. Normally the receiving side invokes security exit data flow, thus forcing the requester side to behave correctly, or else the channel is not allowed to start. Security data flows take place before any messages are transmitted.

- **Message exit**

The message exit is called on the sending side just after a message is read from the transmission queue and on the receiving side just before a message is put on to a destination queue. The message exit receives the entire message, including the MQMD, and can change the message and its descriptor if preferred. Typical purposes are encryption and decryption of data, or *logging* of messages to ensure that a particular message is successfully sent or received.

- **Message retry exit**

The message retry exit is called if a message that is received through a channel cannot be put on the destination queue, for whatever reason. The exit can determine, if at all, how often and at what intervals the delivery MQPUT is retried, by that taking precedence over the appropriate channel attributes (such as MRRTY, MRTMR).

- **Send and receive exits**

The send exit is called just before a buffer of data is transmitted over the network, and the receive exit is called just after a buffer of data is received from the network. So these exits can be called multiple times for a single message. Their typical purpose is data compression and decompression.

Before considering a channel exit



Check whether using CONNAUTH or CHLAUTH removes the need for the exit if:

- Authentication by using ID
- Channel authentications



Check whether using Advanced Message Security removes the need for the exit if requiring:

- Confidentiality
- Data integrity
- Nonrepudiation

© Copyright IBM Corporation 2015

Figure 4-23. Before considering a channel exit

WM3021.1

Notes:

SSL and TLS support for TCP/IP channels

- Industry standard protocol to secure TCP/IP connections
- Based on digital certificates
- Provides
 - Mutual authentication of both the client and the server
 - Message privacy by data encryption
 - Message integrity by message hash functions
- Setup activities
 - Obtain (or create) a digital certificate for the local queue manager: RADCERT if using RACF
 - Identify SSL resources on the queue manager level
 - Specify SSL options per selected channel
- More than one certificate per queue manager with IBM MQ V8 and later queue managers



© Copyright IBM Corporation 2015

Figure 4-24. SSL support for TCP/IP channels

WM3021.1

Notes:

IBM WebSphere MQ for z/OS supports the use of Secure Sockets Layer (SSL).

- To implement SSL, first configure the certificate in the queue manager. IBM MQ V8 and later queue managers can have multiple certificates.
- Using z/OS Security Server (RACF) functions, a key ring database must be created that contains a QMGR certificate and the certificates for the certificate authorities (CA) that possibly signed a received certificate.
- Channel definition attributes can determine the use of SSL:
 - **SSLCIPH** (cipher specifications): Specifies encryption algorithm and strength
 - **SSLPEER** (partner_spec): Specifies filter patterns from allowed partners
 - **SSLCAUTH** (yes/no): Determines whether the client must provide a certificate
 - IBM MQ V8 or higher attribute only: CERTLBL **queue manager** attribute
 - IBM MQ V8 or higher attribute only: CERTQSQL for queue-sharing groups
 - IBM MQ V8 or higher attribute only: CERTLBL channel attribute

- IBM MQ V8 or higher attribute only: SSLCERTI attribute for CHLAUTH records to match the issuer's DN

The security unit describes the IBM MQ V8 attributes.

Administering channels: ISPF or MQSC

List Channels - MQ00 Row 1 of 13

Type action codes, then press Enter. Press F11 to display connection status.

1=Display 2=Define like 3=Alter 4=Manage 5=Perform
6=Start 7=Stop

Name	Type	Disposition	Status
<> *	CHANNEL	QMGR	MQ00
MQ00.MQ0A	SENDER	QMGR	MQ00 INACTIVE
SYSTEM.ADMIN.SVRCONN	SVRCONN	QMGR	MQ00 INACTIVE
SYSTEM.DEF.CLNTCONN	CLNTCONN	QMGR	MQ00
...


```

-DEFINE CHANNEL(MQ0A.MQ00) CHLTYPE(SDR) TRPTYPE(TCP) XMITQ(MQ00) +
  CONNAME('mvsmc11.ilsvpn.ibm.com(1699)')
-ALTER CH(MQ0A.MQ00) CHLTYPE(SDR) CONNAME('mvsmc11.ilsvpn.ibm.com(1600)')
-DIS CHANNEL(MQ0A.MQ00) ALL
-DIS CHS(MQ0A.MQ00)
-START CHANNEL(MQ00.MQ0A)
-STOP CHANNEL(MQ00.MQ0A)
-RESET CHANNEL(MQ00.MQ0A) ...
-RESOLVE CHANNEL ...
-RESOLVE INDOUBT ...

```

 Use of **DIS CHS(*)** can affect performance

© Copyright IBM Corporation 2015

Figure 4-25. Administering channels: ISPF or MQSC

WM3021.1

Notes:

Channels can be managed; that is, operations such as define, start, and resolve can be accomplished in the different equivalent ways you saw in the administration unit. The screen display shows use of the ISPF panels, and also some MQSC commands that can be issued to manage channels.

Sometimes channels do not start on the first try. Most of the channel problems can be categorized in three areas:

- Configuration problems: These types of errors are common right after the channel is defined and goes into retry instead of starting. The problem can be as simple as the partner channel not being defined on the remote side, missing the correct port number, or an error in the IP address or host name.
- Sequence error: Channels keep a count of messages exchanged with their peer channel partner. If an event occurs, such as a delete and redefine of the channel, or perhaps a rebuild of one of the queue managers, this number differs on one side and causes the sequence error.
- Sync point issue, where a batch of messages needs to be committed or backed out of the channel by use of the RESOLVE command.

Channel states

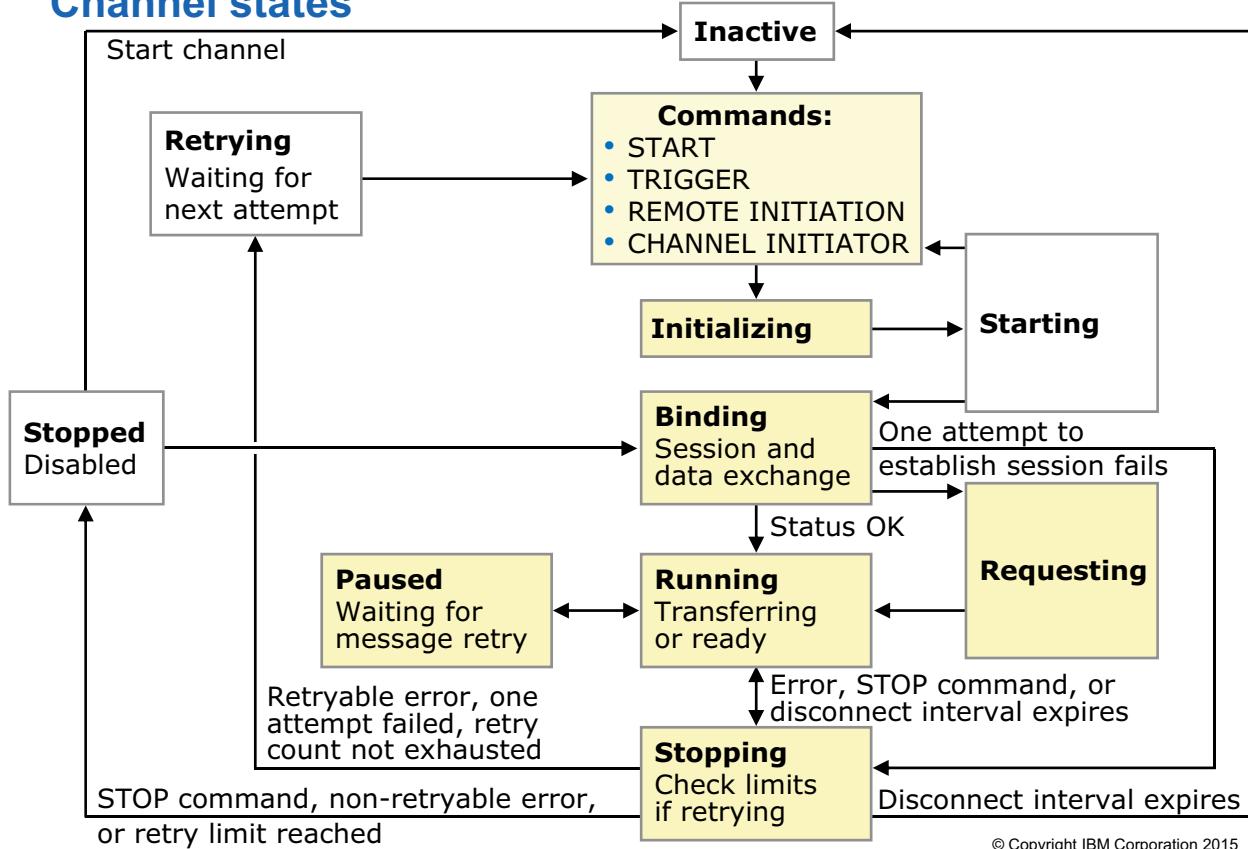


Figure 4-26. Channel states

WM3021.1

Notes:

Channels go across several states before they start. The most common error you find when first starting a channel is “retrying.” Retrying is often the result of a problem with the definition, channel pairs unable to find each other, possibly a wrong port, or a misconfigured host name or IP address. When the problem is rectified, the channel normally starts on its own. If the channel does not automatically start after the correction, then stop the sender (assuming that you have a sender-receiver pair), and then restart it.



Note

If a channel is manually stopped or is in STOPPED state, it must be manually restarted, even if it is triggered.

A brief description of the states in the diagram follows. The diagram is a simplified representation of what happens. The flows between the states, in particular, should not be interpreted too literally.

START is not a state. It indicates a start point for when a command is issued to start a channel, or when the arrival of a message on the transmission queue triggers the start of a channel. It can also indicate a start point when a channel initiator decides it is time for the next attempt to start a channel, or when there is an incoming request to start a channel.

- **INITIALIZING:** Before a channel initiator starts an MCA, it creates an entry for the channel in the channel status table, if an entry does not yet exist, and sets the state of the channel to INITIALIZING. This entry acts as a “placeholder” in case the MCA fails before it has a chance to put an entry in the channel status table itself. If the MCA fails without such an entry, the channel initiator knows nothing about the channel and so there would be no further attempts to start it.
- **STARTING:** A channel waits in this state if no active slot is available. However, if there is no active slot for the requester end of a channel, or the responder end of a channel, the channel fails to start and does not enter the STARTING state.

To start a channel that is in the STARTING state when an active slot does become available, a channel initiator must be running.

- **BINDING:** A channel is establishing a communications connection and is doing the initial data negotiation. However, a requester that is also a caller remains in this state only while it is establishing a communications connection.
- **REQUESTING:** A requester that is also a caller is doing the initial data negotiation.
- **RUNNING:** A channel is transferring messages, or waiting for messages to arrive on the transmission queue.
- **PAUSED:** A channel is waiting for the message-retry interval to elapse before attempting to open a destination queue or to put a message on it. This state does not occur on IBM WebSphere MQ for Windows as message-retry is not supported.
- **STOPPING:** A channel encounters a failure, or a command is issued to stop the channel, or the disconnect interval elapses.

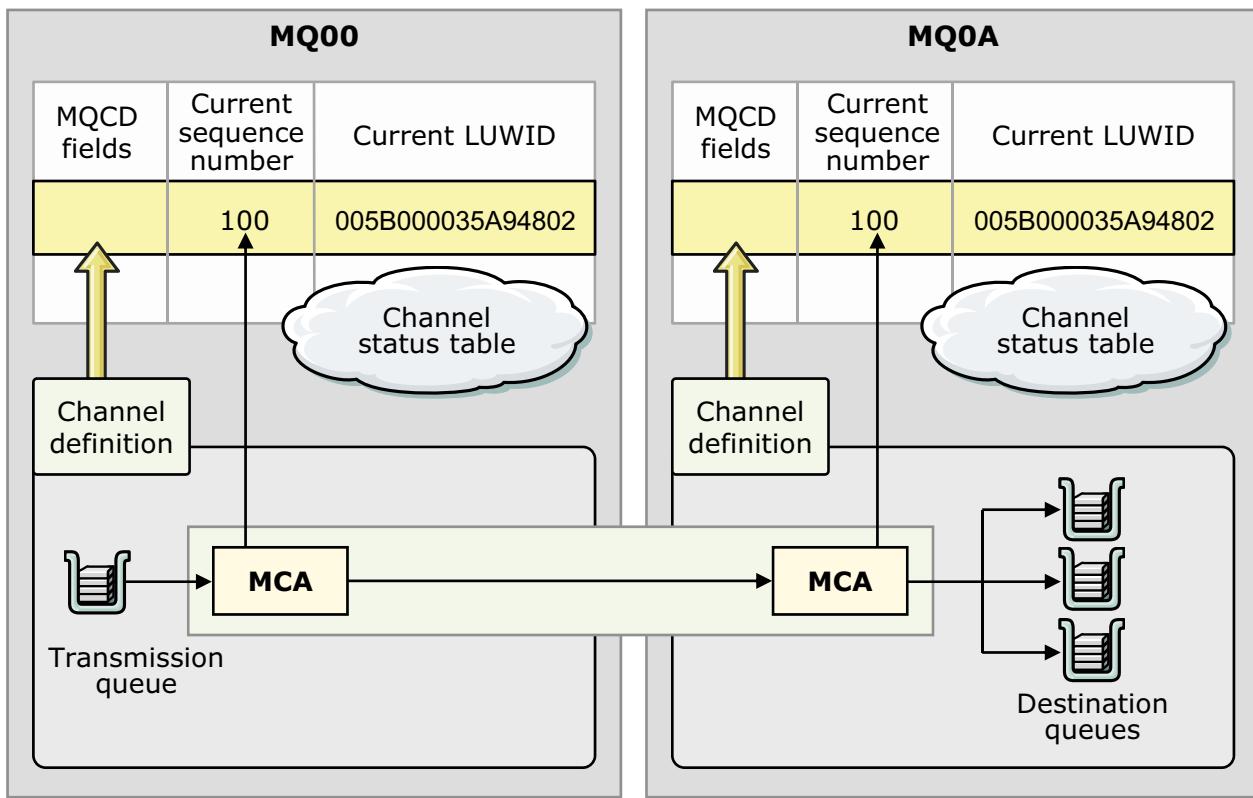
If the channel encounters a failure, it checks whether its retry limits were reached while it is in this state. If the limits were not reached, the channel enters the RETRYING state. If the limits were reached, or the retry counts were set to zero in the channel definition, the channel enters the STOPPED state. However, on a version 5 queue manager, a channel becomes inactive if no channel initiator is running, whether the retry limits were reached or not.

If the disconnect interval elapses, the channel becomes inactive. If a command is issued to stop the channel, the channel enters the STOPPED state.

- **RETRYING:** A channel is waiting until it is time for the channel initiator to make the next attempt to start the channel.
- **STOPPED:** In this state, the channel is disabled and needs manual intervention to start it again.
- **INACTIVE:** An inactive channel is one that was never started, or one for which the disconnect interval elapsed, or one that is considered to be terminated “normally.”

The current state of a channel can be determined by using the `DISPLAY CHSTATUS` command.

Message sequence numbers and logical unit of work identifiers



© Copyright IBM Corporation 2015

Figure 4-27. Message sequence numbers and logical unit of work identifiers

WM3021.1

Notes:

After an MCA is started and connects to the queue manager, it reads the channel definition and uses it to initialize the fields in the channel data structure, MQCD. These fields constitute the initial entry for the channel in the *channel status table*.

Each queue manager has its own channel status table. On IBM WebSphere MQ for z/OS, the channel status table is stored in the channel initiator address space. Therefore, there is a channel status table at each end of a channel, and there is an entry for the channel in each table. The table is stored in main memory and so, if there is a failure, or when the queue manager ends, the information that is held in the table is lost.

The channel status table is used by the various components of distributed queuing to manage channels and for communication among the components. These components include MCAs, channel initiators, IBM WebSphere MQ listeners, and the channel management commands.

Each message that is sent across a channel has an associated *message sequence number (MSN)* which the sending MCA assigns. The current sequence number for a channel is stored in its entry in the channel status table at each end of the channel. At the sending end of the channel, the current sequence number is the sequence number of the last message sent. The sending MCA increments it by one just before sending a message. At the receiving end of the channel, the current sequence

number is the sequence number of the last message received. Thus, a receiving MCA knows which sequence number to expect on the next message it receives. If the sequence number is incorrect, it is assumed that some failure occurred, and the channel terminates.

The usual cause of a sequence number mismatch is when an administrator deletes a channel at one end and then redefines it. Another possible cause is if the administrator deletes a queue manager at one end of a channel and then re-creates it with all of its objects. If one of these situations occurs, the **RESET CHANNEL** command can be used to reset the sequence number at each or either end of a channel. However, the command might be issued at the sender or server end of a channel. If so, the sequence number at the receiver or requester end is automatically reset to the same value the next time the channel starts.

One consequence of continually incrementing a sequence number by one is that, at some time, the sequence number must wrap. The highest value that the sequence number reaches before it restarts at 1 is called the *sequence number wrap value*. The **SEQWRAP** parameter specifies this value on the **DEFINE CHANNEL** command. A channel does not start if the sequence number wrap value at one end of the channel is different from the value at the other end.

Administrators use sequence numbers for monitoring the health of channels and for calculating how many messages were sent across a channel within a certain period.

For efficiency, messages are sent across a channel in batches. At the end of a batch, the receiving MCA confirms to the sending MCA that it safely committed all the messages in the batch on their respective destination queues. After this confirmation, the sending MCA commits the removal of the messages from the transmission queue. This procedure is part of the protocol that ensures that a message is delivered one time and one time only. It is described in more detail on the next visual.

Each batch of messages is assigned a unique identifier by the sending MCA. This identifier is called a *logical unit of work identifier (LUWID)*. At the start of a new batch, the sending MCA communicates the LUWID for the batch to the receiving MCA. The LUWID for the current batch is also stored in the channel status table at each end of the channel.

In addition to using a LUWID to identify a batch, an MCA also uses the sequence number of the last message in the batch. On the next visual, you will see how both identifiers are used. From the point of view of the administrator, LUWIDs are possibly more convenient as batch identifiers because they are more constant than sequence numbers.

Message sequence number error

```
CSQX500I MQ0A CSQXRESP Channel MQ00.MQ0A started 117  
connection 10.31.187.59  
... ... ...  
CSQX526E MQ00 CSQXRCTL Message sequence error for channel MQ00.MQ0A,  
sent=13 expected=1  
... ... ...  
CSQX526E MQ0A CSQXRESP Message sequence error for channel MQ00.MQ0A,  
sent=13 expected=1  
... ... ...  
CSQX599E MQ0A CSQXRESP Channel MQ00.MQ0A ended abnormally 120  
connection 10.31.187.59  
... ... ...  
CSQX558E MQ00 CSQXRCTL Remote channel MQ00.MQ0A not available
```

© Copyright IBM Corporation 2015

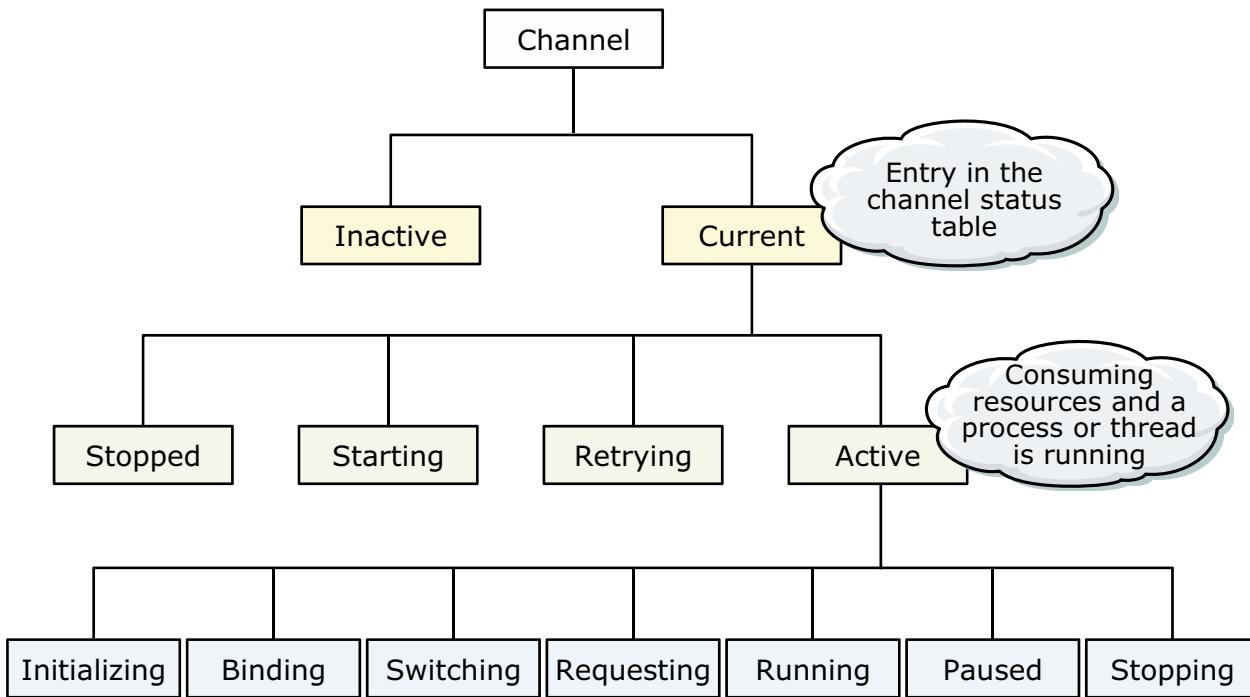
Figure 4-28. Message sequence number error

WM3021.1

Notes:

This screen display is an example of messages that are observed in the system log when there is a message sequence error condition. When you see such an error, the RESET command must be used to set the correct sequence number.

Current and active channels



© Copyright IBM Corporation 2015

Figure 4-29. Current and active channels

WM3021.1

Notes:

Current and saved status data

SAVED status

```
/MQ00 DIS CHS (MQ00.MQ0A) +
SAVED ALL
CHSTATUS (MQ00.MQ0A)
CHLDISP(PRIVATE)
XMITQ (MQ0A)
CONNNAME (MQ0A)
SAVED
CHLTYPE (SDR)
INDOUBT (NO)
LSTSEQNO (12)
LSTLUWID (CDB054432EB...
CURMSGS (0)
CURSEQNO (12)
CURLUWID (CDB152E0B1...
END CHSTATUS DETAILS
```

CURRENT status

```
/MQ00 DIS CHS (MQ00.MQ0A) +
ALL
CHLDISP(PRIVATE)
...
CURRENT
CHLTYPE (SDR)
STATUS (RUNNING)
SUBSTATE (MQGET)
INDOUBT (NO)
LSTSEQNO (20)
LSTLUWID (CDB15283A9...
CURMSGS (0)
CURSEQNO (20)
CURLUWID (CDB152E0B1...
LSTMSGTI (13.08.53)
LSTMSGDA (2014-09-01)
MSGS (8)
BYTSSENT (5144)
BYTSSENT (5144)
BYTSRCVD (592)
BATCHES (2)
```

current continued

```
CHSTATI (13.07.15)
CHSTADA (2014-09-01)
BUFSSENT (10)
BUFSRCVD (4)
LONGRTS (999999999)
SHORTRTS (10)
MONCHL (OFF)
...
KAINT (360)
QMNAME (MQ00)
RQMNAME (MQ0A)
RQMNAME (MQ0A)
SSLCERTI ()
...
RVERSION (08000000)
STATCHL (OFF)
LOCLADDR (10.31.187.59(11
BATCHSZ (50)
MAXMSGL (4194304)
...
...
```

© Copyright IBM Corporation 2015

Figure 4-30. Current and saved status data

WM3021.1

Notes:

When you use the **DISPLAY CHSTATUS** command (or the equivalent ISPF panel option) to display the status of one or more channels, a choice is required. You must specify whether you want the *current status data* or the *saved status data*. By default, current status data is returned if saved status data is not explicitly requested.

The current status data for a channel is data that is derived from the entry for the channel in the channel status table. One consequence of this is that an inactive channel has no current status data.

The saved status data for a channel is data that is derived from the status message for the channel on the synchronization queue, or from the in-doubt status message if the channel is in doubt. Thus, an inactive channel might contain saved status data.

Certain status data is returned only when current status data is requested. These status fields are referred to as *current-only* status fields. The values of current-only status fields are derivable only from data that is stored in the channel status table. They cannot be derived from data that is stored in scratchpad objects or messages on the synchronization queue. The keywords that are used on the **DISPLAY CHSTATUS** command to request these fields are listed on the visual.

The remaining status fields are referred to as *common* status fields because values for these fields can be returned when either current or saved status data is requested. The values of common status fields can be derived either from data that is stored in the channel status table or from data that is stored in scratchpad objects or messages on the synchronization queue. However, the values might be different for current and saved status because data in the channel status table is continually being updated. Data is written to a scratchpad object, or a new status message is put on the synchronization queue, only at certain times during the operation of a channel. The keywords that are used on the `DISPLAY CHSTATUS` command to request these fields are listed on the visual.

All of the keywords that can be specified on the `DISPLAY CHSTATUS` command are described in the *IBM WebSphere MQ Script (MQSC) Command Reference*. However, the following are worthy of note here.

- **CURLUWID:** For the sending or the receiving end of a channel, it is the logical unit of work ID (LUWID) of the current batch. If a channel is in doubt, it is the LUWID of the in-doubt batch. However, if saved status data is requested, its value is meaningful only if a channel is in doubt.
- **CURMSGS:** For the sending end of a channel, it is the number of messages that were sent in the current batch. For the receiving end of a channel, it is the number of messages that were received in the current batch. For an in-doubt channel, it is the number of messages that are in doubt. However, if saved status data is requested, its value is meaningful only if a channel is in doubt.
- **CURSEQNO:** For the sending end of a channel, it is the sequence number of the last message sent. For the receiving end of a channel, it is the sequence number of the last message received. If a channel is in doubt, it is the sequence number of the last message in the in-doubt batch. However, if saved status data is requested, its value is meaningful only if a channel is in doubt.
- **INDOUBT:** For the sending end of a channel, it is YES if the channel is in doubt, and NO otherwise. For the receiving end of a channel, it is always NO.
- **LSTLUWID:** The LUWID of the last committed batch of messages transferred.
- **LSTSEQNO:** The sequence number of the last message in the last committed batch of messages transferred.
- **STATUS:** The state of the channel, STARTING, BINDING and others.

As stated previously, a channel can have multiple instances, either concurrently or over a time period. Therefore, if you enter `DISPLAY CHSTATUS(. . .) CURRENT`, specifying only the name of a channel, the current status data for each instance of the channel is displayed. You can use this display to determine, for example, the number of instances of a receiver that are current at any moment in time. Similarly, if you enter `DISPLAY CHSTATUS(. . .) SAVED`, specifying only the name of a channel, the saved status data for each instance of the channel is displayed.

Message channel agent (MCA) operation and the synchronization queue

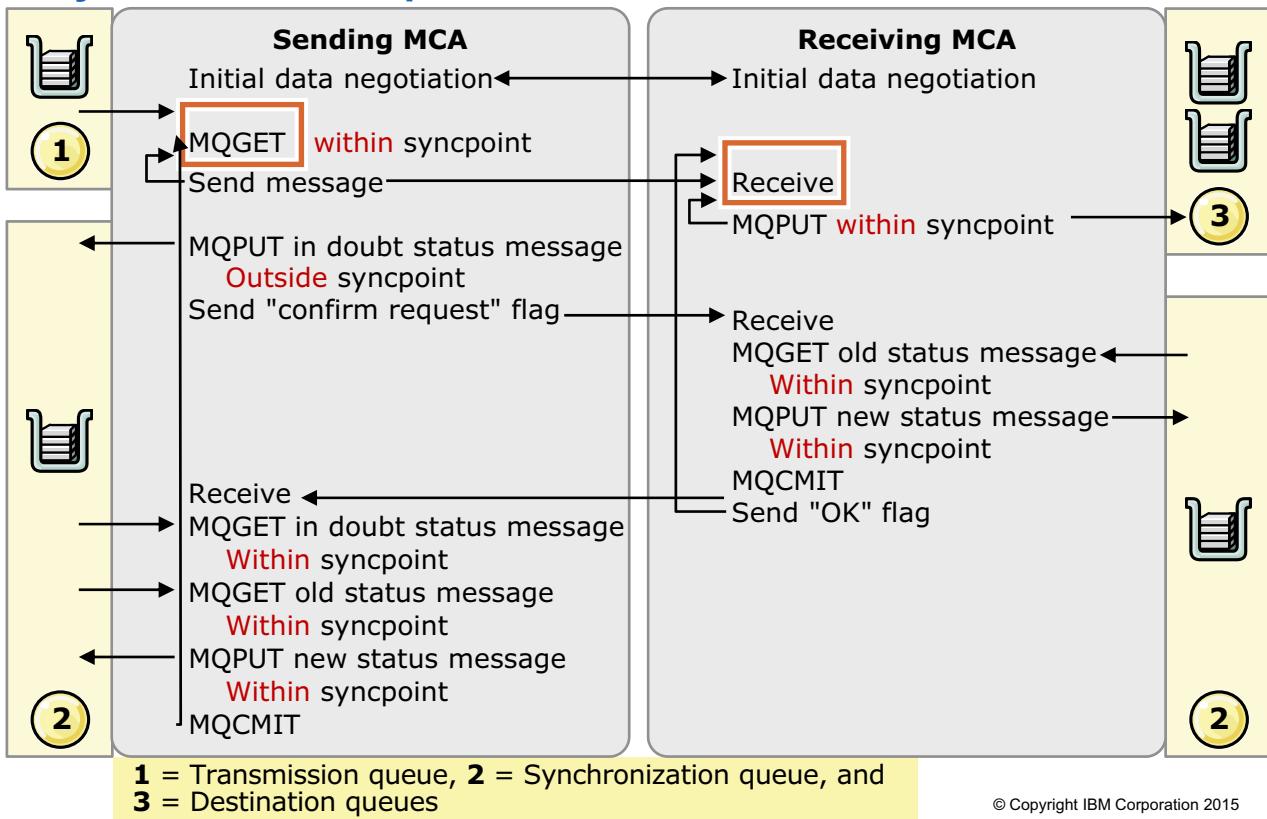


Figure 4-31. Message channel agent (MCA) operation and the synchronization queue

WM3021.1

Notes:

When a channel starts and a communications connection is established, the two MCAs enter a phase that is called the *initial data negotiation* before messages start to flow. During the initial data negotiation, the MCAs exchange certain information about their respective ends of the channel. Certain parameters that control the operation of the channel are negotiated, and the MCAs agree on which one does certain tasks. Here are some examples of what transpires during this phase.

- Checks are made that a channel with the same name is defined at both ends, and that the type of the channel at one end is compatible with the type at the other.
- The two MCAs compare the value of the `SEQWRAP` parameter at each end of the channel. If the two values are not the same, the channel terminates.
- The MCAs agree on which one converts the data in the transmission queue header that accompanies each message that is transferred on the channel. The usual rule is receiver makes good.
- The MCAs agree on the maximum size of a batch of messages. The agreed size is the lower of the values of the `BATCHSZ` parameter at each end of the channel. However, the agreed size is

reduced even further if the maximum number of uncommitted messages that are allowed within a single unit of work at either queue manager is lower.

- The MCAs agree on the maximum length of a message that can be transferred on the channel. The agreed size is the lower of the values of the `MAXMSGL` parameter at each end of the channel.
- Each MCA determines whether its partner supports heartbeat flows. If both MCAs support them, the agreed size of the heartbeat interval is the larger of the values of the `HBINT` parameter at each end of the channel. The use of heartbeat flows is discussed later in this topic.
- The MCAs agree on the class of service for nonpersistent messages on the channel. This information is determined from the value of the `NPMSPED` parameter at each end of the channel. The value can be either `FAST` or `NORMAL`. If the values at the two ends of the channel do not match, or if one end of the channel does not support fast nonpersistent messages, `NORMAL` is used. Fast nonpersistent messages are discussed later in the course.
- The sending MCA determines whether the partner queue manager supports distribution lists. It uses this information to set the `DistLists` attribute of the transmission queue to either `MQDL_SUPPORTED` or `MQDL_NOT_SUPPORTED`.

Transferring a batch of messages

After the initial data negotiation is complete and the two MCAs agree to proceed with the transfer of messages, the sending MCA gets a message from the transmission queue within syncpoint control. It sends the message to the receiving MCA, which puts it on its intended destination queue, also within syncpoint control. The sending MCA then continues to get messages from the transmission queue and send them to the receiving MCA until one of the following conditions is met:

- The agreed maximum number of messages in a batch is reached.
- There are no more messages on the transmission queue and an interval that the `BATCHINT` parameter specified elapses while waiting for a message.

End of batch commitment processing

After the sending MCA closes a batch, it then needs to determine whether the receiving MCA safely committed all the messages in the batch on their respective destination queues. Until the sending MCA receives such confirmation, it does not commit the removal of the messages from the transmission queue, and the channel is said to be *in doubt*. This part of the protocol is important, and the two MCAs use it to ensure that a message is delivered one time and only one time.

The sending MCA puts an in-doubt status message on the synchronization queue, `SYSTEM.CHANNEL.SYNCQ`, at its end of the channel. This action is done outside of syncpoint control. The in-doubt status message contains the following information:

- The LUWID of the in-doubt batch
- The sequence number of the last message in the in-doubt batch
- The number of messages in the in-doubt batch
- The message identifier of each message in the in-doubt batch
- The LUWID of the last committed batch, that is, the previous batch
- The sequence number of the last message in the last committed batch

The sending MCA then sends a confirm request flag on an IBM WebSphere MQ flow to the receiving MCA. If the batch is closed because the agreed maximum batch size is reached, the sending MCA can include the flag on the flow for the last message in the batch. Otherwise, it requires a separate flow.

On each queue manager, the synchronization queue contains a status message for each channel that transfers at least one batch of messages. The status message for a channel contains the following information:

- The LUWID of the last committed batch, that is, the previous batch
- The sequence number of the last message in the last committed batch

When the receiving MCA receives the confirm request flag, it gets the status message for the channel from the synchronization queue within syncpoint control, thus marking it for removal from the queue. It then puts a new status message for the channel on the synchronization queue, also within syncpoint control. This status message contains the LUWID for the current batch and the sequence number of the last message in the current batch. The receiving MCA then calls `MQCMBT` to commit all the messages on their respective destination queues, the new status message for the channel on the synchronization queue, and the removal of the old status message. Finally, the receiving MCA sends the OK flag on an IBM WebSphere MQ flow to the sending MCA, which is waiting on a communications receive in the meantime. The receiving MCA then reverts to waiting on a communications receive.

When the sending MCA receives the OK flag, it gets the in-doubt status message for the channel from the synchronization queue within syncpoint control. In addition, it gets the old status message and puts a new status message, also within syncpoint control. Finally, it calls `MQCMBT` to commit the new status message on the synchronization queue. It also commits the removal of the in-doubt status message and the old status message from the synchronization queue, and the removal of the messages in the batch from the transmission queue.

After processing a batch of messages in this way, the sending MCA starts the processing for the next batch.

Resolve an in-doubt channel manually

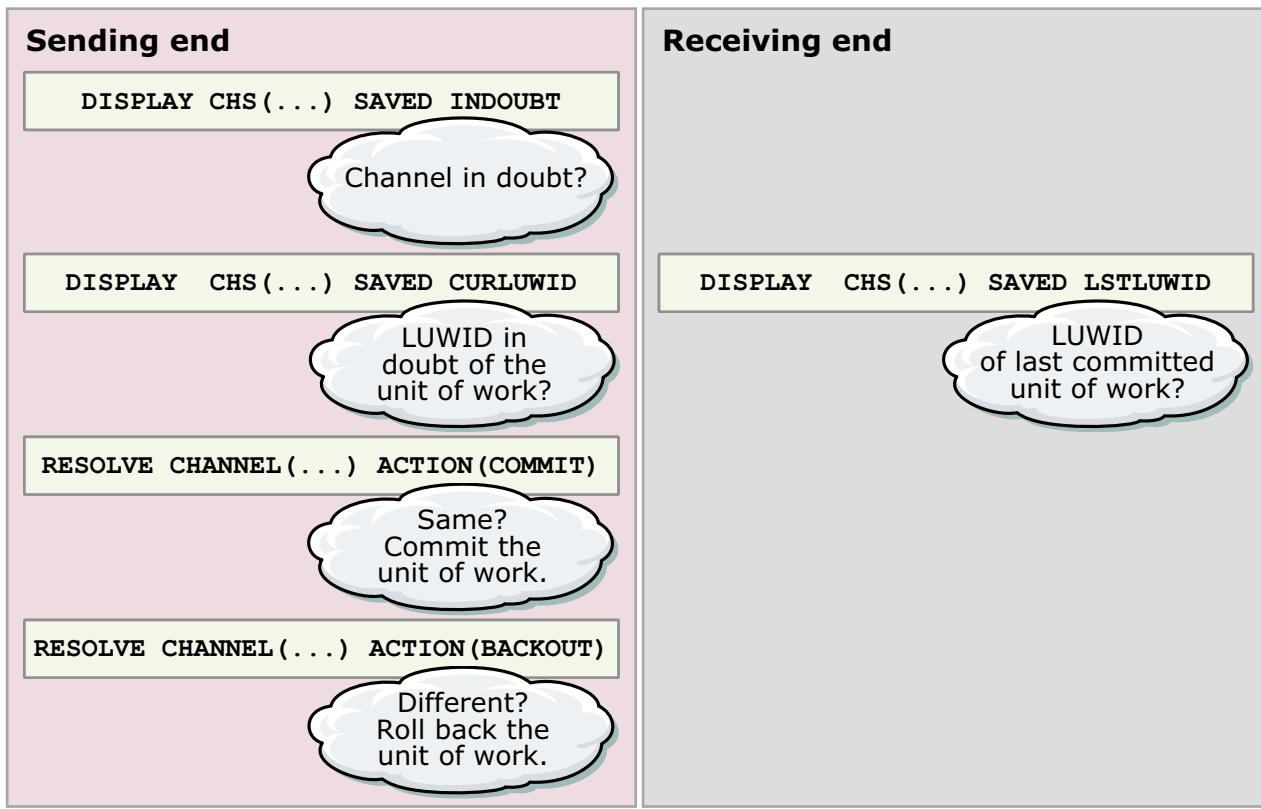


Figure 4-32. Resolve an in-doubt channel manually

WM3021.1

Notes:

The procedure to resynchronize a channel manually is:

- At the sending end of the channel, issue the following command to determine whether the channel is in doubt:

```
DISPLAY CHSTATUS(....) SAVED INDOUBT
```

- If the channel is in doubt, issue the following command at the sending end of the channel to determine the LUWID of the in-doubt batch:

```
DISPLAY CHSTATUS(....) SAVED CURLUWID
```

- At the receiving end of the channel, issue the following command to determine the LUWID of the last committed batch:

```
DISPLAY CHSTATUS(....) SAVED LSTLUWID
```

- If the two LUWIDs are the same, it means that the receiving end already committed the in-doubt messages on their respective destination queues. Therefore, the messages must be removed

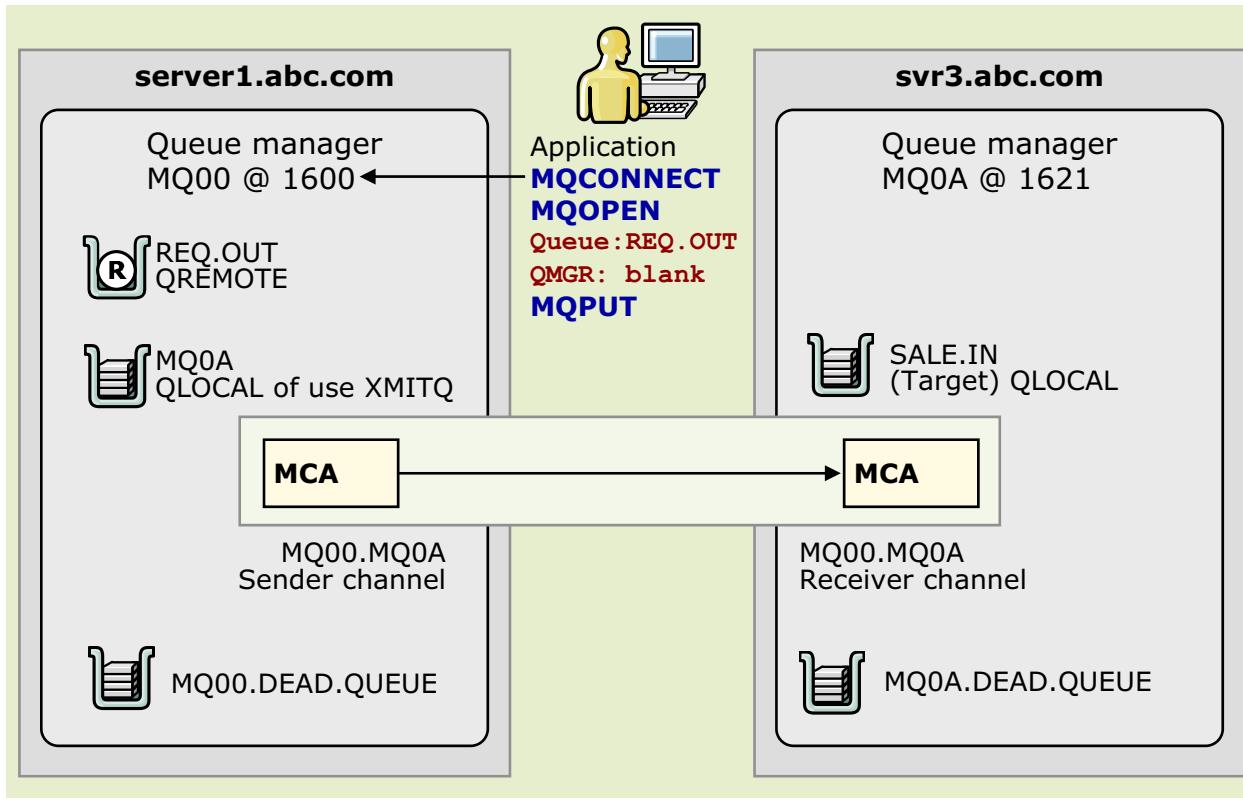
from the transmission queue at the sending end. This result is achieved by issuing the following command:

```
RESOLVE CHANNEL(...) ACTION(COMMIT)
```

5. If the two LUWIDs are not the same, it means that the receiving end did not commit the in-doubt messages and so the messages must be retained on the transmission queue at the sending end. This result is achieved by issuing the following command:

```
RESOLVE CHANNEL(...) ACTION(BACKOUT)
```

Cumulative checkpoint



© Copyright IBM Corporation 2015

Figure 4-33. Cumulative checkpoint

WM3021.1

Notes:

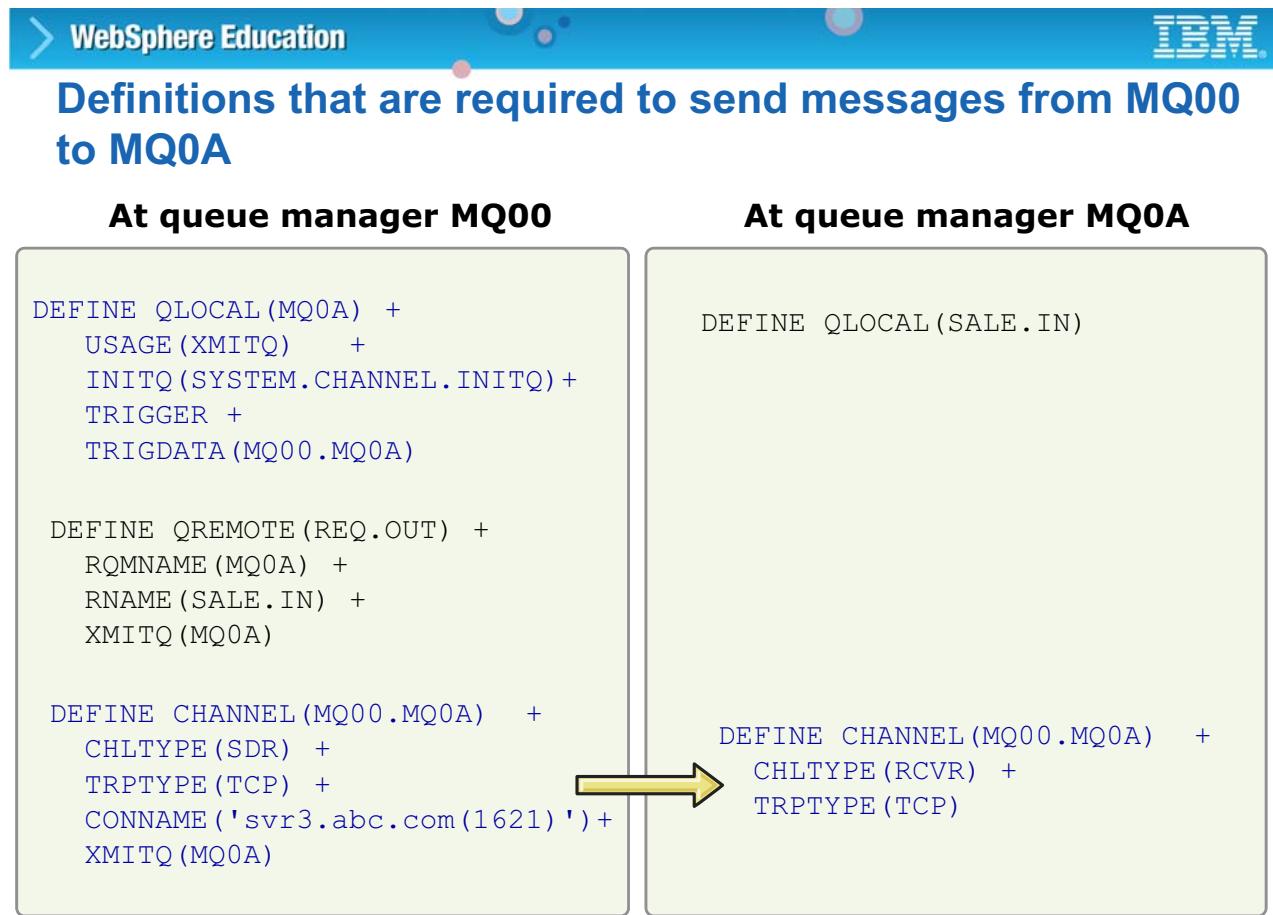
This slide starts a walk-through in a common scenario. An application requests to have a channel and queue defined. Often, the administrator has access to only one of the sides of the connection. As a result, having a basic knowledge of what happens is helpful, as you might be working with a partner administrator who needs some help.

The request from the application is:

- Need a queue to send messages to the sales processing application.
- The administrator in the sales processing organization provided the following information:
 - The queue that is used to receive messages is called: **SALE.IN**
 - Their queue manager is **MQ0A** at **svr3.abc.com**, and the port is **1621**. It is a new connection.
- The application IBM MQ coding standards require that they do **not** use a queue manager name to allow for growth and flexibility.
- You agree on the queue name you create for them: **REQ.OUT**

The application expresses that time is critical, and the IBM MQ administrator for the MQ0A queue manager promises to finish their side of the definitions the same day.

You chat with the MQ0A administrator and proceed to create the requested definitions.



© Copyright IBM Corporation 2015

Figure 4-34. Definitions required to send messages from MQ00 to MQ0A

WM3021.1

Notes:

The MQ0A administrators said that they were in the middle of solving a problem, but the agreed definitions would be completed at end of day. You create all the IBM MQ objects in the MQ00 side, and the MQ0A administrators are to create the definitions on the MQ0A side.

You know that this application is scheduled to start testing in the morning, so later that afternoon you decide to check whether the MQ0A definitions are ready, and you try starting the channel. The channel starts, so all is ready for tomorrow.

As a precaution, since this definition is new, MQ00 was enabled to collect queue status information, and you also enabled transmit queue MQ0A to collect information on activity.

What happens when the destination queue is not found

Symptoms: Channel started. Transmit queue empty. No errors in JCL. Messages did not arrive to expected remote target queue.

MQ00 DIS CHS (MQ*)

...
CHSTATUS (MQ00.MQ0A)
CHLDISP (PRIVATE)
CONNNAME (10.31.187.59)
CURRENT
CHLTYPE (SDR)
STATUS (RUNNING)

MQ00 DIS Q (MQ0A)

CURDEPTH
...
QUEUE (MQ0A)
TYPE (QLOCAL)
QSGDISP (QMGR)
CURDEPTH (0)
END QUEUE DETAILS

MQ0A SYSLOG:

+CSQX548E MQ0A CSQXRESP
Messages sent to local
dead-letter queue, 098
channel MQ00.MQ0A
reason=2085 (MQRC_UNKNOWN_
OBJECT_NAME)

- Browse the dead-letter queue for information about the message
- Message Handler sample program can be used to view the dead-letter header and determine an action to take for the message

Queue Manager : **MQ0A**

Queue : MQ0A.DEAD.QUEUE

Forward to Q Mgr : MQ00

Forward to Queue :

Action : : (D)elete (F)orward

...
Dead Letter Header

StrucId : 'DLH '

Version : 000000001

Reason : 000002085

DestQName : 'SALE.IN'

DestQMgrName : 'MQ0A'

© Copyright IBM Corporation 2015

Figure 4-35. What happens when the destination queue is not found

WM3021.1

Notes:

The application that requested the definitions calls. The sales application got an “error” when they tried to run the code to get the message sent from MQ00. They do not know what the error was, but it was an IBM MQ error.

You check transmit queue MQ0A:

- There are no messages in the queue.
- The history queue shows that the time and date for the MQ0A queue on the MQ00 queue manager matches the time when the application tried to send messages to the sales application on MQ0A. Therefore, you know with certainty that a message was sent to the MQ0A queue manager.
- Additionally, you issued a /DIS CHS('MQ00.MQ0A') SAVED ALL to look at the CURSEQNO, which is a count that the channel keeps of messages that are sent. It did show that a message flowed across the channel.
- Where is the message?

You get on a conference call with both sending and sales applications, and the MQ0A administrators. When the MQ0A administrators look for the SALE.IN queue, it is not defined. The MQ0A administrators also looked at the MQ0A dead-letter queue, where they found the message.

What if the dead-letter queue is not available?

Symptoms: Channel seems to start, but messages are left in the transmit queue. Channel ends abnormally.

```
System log view: MQ00 and MQ0A in same server
+CSQX500I MQ0A CSQXRESP Channel MQ00.MQ0A started 765
connection 10.31.187.59
+CSQX565E MQ0A CSQXRESP No dead-letter queue for MQ0A, channel MQ00.MQ0A
+CSQX527E MQ00 CSQXRCTL Unable to send message for channel MQ00.MQ0A
+CSQX036E MQ0A CSQXRESP Unable to open QUEUE (SALE.IN), 768
MQCC=2 MQRC=2085 (MQRC_UNKNOWN_OBJECT_NAME)
+CSQX599E MQ0A CSQXRESP Channel MQ00.MQ0A ended abnormally 769
connection 10.31.187.59
+CSQX558E MQ00 CSQXRCTL Remote channel MQ00.MQ0A not available
```

```
MQ00 DIS Q(MQ0A) ALL
CSQM293I MQ00 CSQMDRTC 1 QUEUE
... ...
QUEUE (MQ0A)
TYPE (QLOCAL)
...
CURDEPTH (3)
...
GET (DISABLED)
...
```

- Define missing queue, start channel:

```
MQ0A DIS Q(SALE*) CURDEPTH
... ...
QUEUE (SALE.IN)
TYPE (QLOCAL)
CURDEPTH (3)
...
```

- Anything left to complete?

© Copyright IBM Corporation 2015

Figure 4-36. What if the dead-letter queue is not available?

WM3021.1

Notes:

What would happen if the MQ0A administrators overlooked the definition of the SALE.IN queue, and there was no dead-letter queue identified for MQ0A to use?

The call from the application arrives. You check transmit queue MQ0A on queue manager MQ00:

- There are three messages in the MQ0A transmission queue. The application probably tried to keep sending, without success.
- The queue is now GET DISABLED. Why?
- You look at the log for MQ00. There are two messages. The first message states that it is unable to send a message. The next message states that the remote channel is not available.

The clue here is the transmit queue that is being GET disabled. When MQ0A was unable to do anything with the message, it sent it back to MQ00's transmit queue, MQ0A. When a message is returned, the channel stops, and the queue is GET disabled.

On the conference call, the MQ0A administrators tell you that they saw the messages as shown in the screen display, and got the SALE.IN queue defined. Since the queue manager stopped the channel, you had to manually restart it. Messages flow. What should be done after all is working? What else was missing in the MQ0A queue manager? Identifying a dead-letter queue was missing.

Objects are the correct type but the channel fails

```
CSQX500I MQ00 CSQXRCTL Channel MQ00.MQ0B started <== start SENDER
CSQX502E MQ00 CSQXRESP Action not allowed for channel at MQ00
SENDER (MQ00.MQ0B)
CSQX599E MQ00 CSQXRESP Channel MQ00.MQ0B ended abnormally 689
connection 10.31.187.59
CSQX547E MQ00 CSQXRCTL Remote channel MQ00.MQ0B has the wrong type
CSQX599E MQ00 CSQXRCTL Channel MQ00.MQ0B ended abnormally
```

MQ0B DIS CHANNEL (M*)

CSQM293I MQ0B CSQMDRTC
CHANNEL (MQ00.MQ0B)
CHLTYPE (RCVR)

MQ00 @ 1600

MQ00.MQ0B
Sender channel

MQ0B @ 1622

MQ00.MQ0B
Receiver channel

CSQU055I Target queue manager is MQ00

dis channel (MQ00.MQ0B) all
CHANNEL (MQ00.MQ0B)

CHLTYPE (SDR)

XMITQ (MQ0B)

...

TRPTYPE (TCP)

CONNNAME (mvsmc11.ilsvpn.ibm.com(1600)) <= where is sender pointing?

© Copyright IBM Corporation 2015

Figure 4-37. Objects are the correct type but the channel fails

WM3021.1

Notes:

While you were helping with the MQ0A problem, you received another request to connect to the MQ0B queue manager. The MQ0B administrators already completed the channel and queue definitions.

You decide to define the MQ0B channel and queues while waiting on the MQ0A call, and then try to start the MQ00.MQ0B channel. As you start the channel, you see several strange errors. You are already on the phone with the sales application, and are unable to contact the MQ0B administrators.

Later you are able to talk to the MQ0B administrators, but they confirmed that the name of the receiver channel was correct, and sent you a summary of the receiver channel definition. You return to look at the MQ00.MQ0B sender channel, and realize it was pointing to your own MQ00 queue manager.

Soon afterward, you correct the CONNAME on your sender channel, and the problem is corrected.

Considerations when stopping a channel

- Default behavior when a STOP CHANNEL command is issued with only the channel name as a parameter
 - Default if QMNAME or CONNAME are not specified
 - Transmission queue is set to GET DISABLED and NOTRIGGER
 - Channel does not trigger-start until after a manual START CHANNEL is issued
- Default behavior when a STOP CHANNEL command is issued with either queue manager (QMNAME), connection name (CONNAME), or STATUS(INACTIVE)
 - If CONNAME is specified, must match DIS CHSTATUS CONNAME exactly
 - QMNAME refers to a remote queue manager
 - Transmission queue attributes are not altered
 - Triggering continues to work without issuing a START CHANNEL.

© Copyright IBM Corporation 2015

Figure 4-38. Considerations when stopping a channel

WM3021.1

Notes:

Be familiar with your channel states. It is imperative to understand these states to be able to resolve channel problems. For example, you must recognize that a channel in STOPPED state must be manually started to get the channel “out of STOPPED state” regardless of whether the channel is or is not triggered.

Unit summary

- Identify the distributed queuing components
- Describe queue name resolution
- Summarize the use of remote and transmit queues
- Describe message channel agents, listeners, and channel initiators
- Differentiate among the various types of channels
- Describe the object definitions that are necessary to connect queue managers
- Describe connectivity behaviors that client channel definition attributes control
- Demonstrate how to monitor and solve problems with channels

© Copyright IBM Corporation 2015

Figure 4-39. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. What steps need to be completed to trigger-start a channel?
 - a. Ensure that the channel initiator is running
 - b. Include the channel name in the TRIGDATA attribute of the transmission queue
 - c. Ensure that the transmission is defined with TRIGGER and correct initiation queue name
 - d. All of the above
2. True or false: Any in-doubt or sequence error situations in a channel must be rectified before a channel successfully starts.
3. True or false: If a channel is in STOPPED state, it must be started with the START CHANNEL command.

© Copyright IBM Corporation 2015

Figure 4-40. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint questions (2 of 2)

4. Queue manager MQM1 has sender channel MQM1.MQM3, which is defined with the value ('dev21.abc.com') in its connection attribute. Server dev21.abc.com has two queue managers, MQM3 listening on port 1424, and MQM5 listening on port 1414. Both MQM3 and MQM5 queue managers have a receiver channel that is called MQM1.MQM3. What queue manager is MQM1 connecting to through the MQM1.MQM3 sender channel?
 - a. MQM3
 - b. Both MQM1 and MQM3
 - c. MQM5
 - d. None of the above
5. True or false: The QALIAS IBM MQ object is used to define a queue manager alias.

© Copyright IBM Corporation 2015

Figure 4-41. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

4.

5.

Checkpoint answers (1 of 2)

1. What steps need to be completed to trigger-start a channel?
 - a. Ensure that the channel initiator is running
 - b. Include the channel name in the TRIGDATA attribute of the transmission queue
 - c. Ensure that the transmission is defined with TRIGGER and correct initiation queue name
 - d. All of the above

Answer: d

2. True or false: Any in-doubt or sequence error situations in a channel must be rectified before a channel successfully starts.

Answer: True

3. True or false: If a channel is in STOPPED state, it must be started with the START CHANNEL command.

Answer: True

© Copyright IBM Corporation 2015

Figure 4-42. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

4. Queue manager MQM1 has sender channel MQM1.MQM3, which is defined with the value ('dev21.abc.com') in its connection attribute. Server dev21.abc.com has two queue managers, MQM3 listening on port 1424, and MQM5 listening on port 1414. Both MQM3 and MQM5 queue managers have a receiver channel that is called MQM1.MQM3. What queue manager is MQM1 connecting to through the MQM1.MQM3 sender channel?
- a. MQM3
 - b. Both MQM1 and MQM3
 - c. MQM5
 - d. None of the above

Answer: c

5. True or false: The QALIAS IBM MQ object is used to define a queue manager alias.

Answer: False. A QREMOTE is used to define a queue manager alias.

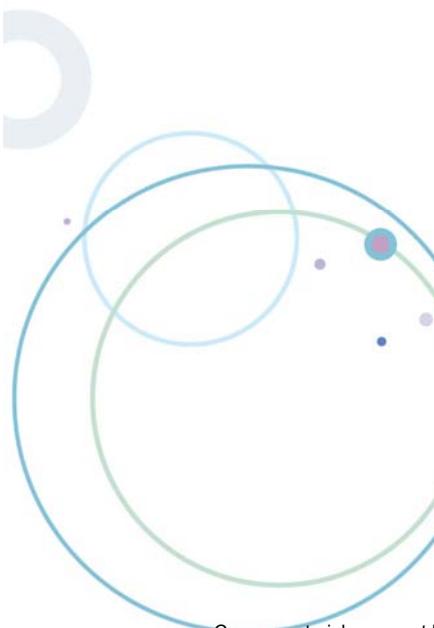
© Copyright IBM Corporation 2015

Figure 4-43. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 3



Working with channels

© Copyright IBM Corporation 2015
Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 4-44. Exercise 3

WM3021.1

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Define transmission queues
- Define sender-receiver channels
- Define remote queues
- Start channels
- Trigger-start channels
- Determine the status of a channel
- Demonstrate how to track the path of a message
- Use the message handler sample to process messages in the dead-letter queue
- Define a multi-hopping configuration

© Copyright IBM Corporation 2015

Figure 4-45. Exercise objectives

WM3021.1

Notes:

Unit 5. IBM MQ clients

What this unit is about

This unit covers differences between an IBM MQ client and an IBM MQ server, and explains connectivity options.

What you should be able to do

After completing this unit, you should be able to:

- Distinguish between an IBM MQ server and an IBM MQ client
- Describe the various ways of connecting WebSphere MQ clients to IBM MQ servers
- Describe security considerations of an IBM MQ client
- Summarize transactional capabilities of an IBM MQ client



Unit objectives

- Distinguish between an IBM MQ server and an IBM MQ client
- Describe the various ways of connecting WebSphere MQ clients to IBM MQ servers
- Describe security considerations of an IBM MQ client
- Summarize transactional capabilities of an IBM MQ client

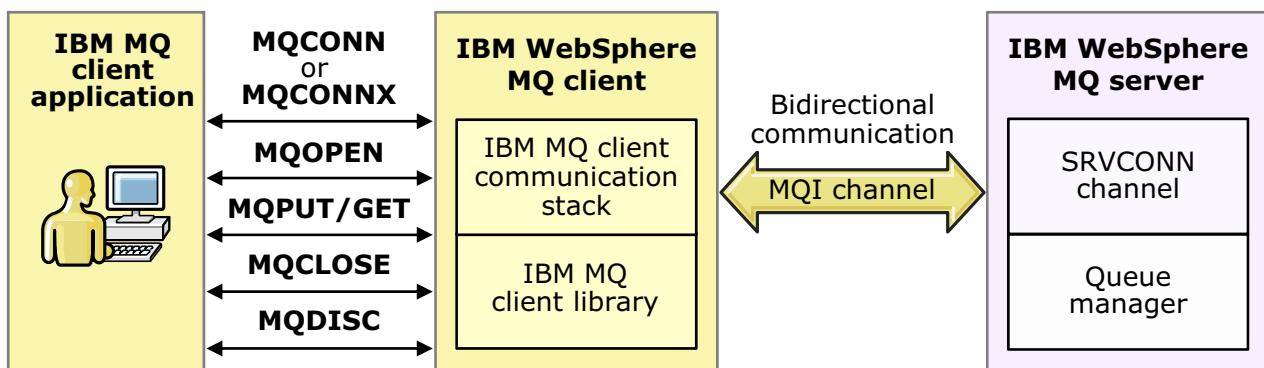
© Copyright IBM Corporation 2015

Figure 5-1. Unit objectives

WM3021.1

Notes:

IBM MQ clients



- An IBM MQ client has IBM MQ client libraries installed
- No queue managers or queues are defined
- Uses the same MQI calls as the IBM MQ server
- Flow of messages in an MQI channel is bidirectional
- Transactional capabilities

© Copyright IBM Corporation 2015

Figure 5-2. IBM MQ clients

WM3021.1

Notes:

An IBM MQ client is an option that allows installation of a different set of IBM MQ libraries that run in client mode. The client does not have queues or queue managers. An IBM MQ client application and a server queue manager communicate with each other by using an MQI channel. An MQI channel does not use a channel pair, but rather an MQI connection.

An MQI channel starts when the client application issues an MQCONN or MQCONNX call to connect to the queue manager, and ends when the client application issues an MQDISC call to disconnect from the queue manager.



Note

Do not confuse the term “client” as used in the general architectural mode, with an “IBM MQ client.” An application that is compiled with IBM MQ server libraries might serve as a client application, but uses distributed channel pairs. An IBM MQ client is compiled with IBM MQ client libraries, uses MQI channels, and must connect to a queue manager SVRCONN channel.

For IBM MQ for z/OS, before IBM MQ V8:

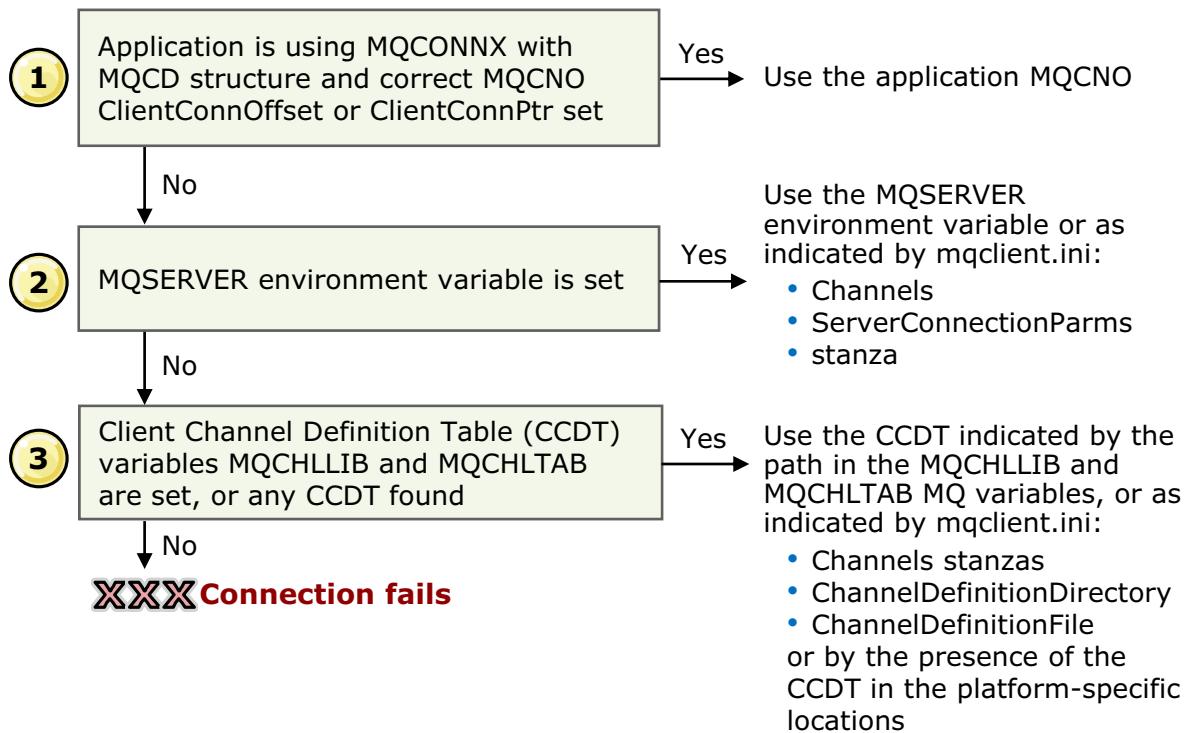
- Separate software had to be installed to enable IBM MQ client connections for IBM MQ clients. This installation of separate software is no longer the case with IBM MQ V8 for z/OS, as clients are included in the IBM MQ z/OS server.
- There was no option to create client channel definition tables (CCDT, covered later in this unit) in the IBM MQ client side. Client definitions in the CCDT had to be downloaded from an IBM MQ server to the IBM MQ client.
- IBM MQ for z/OS did not provide IBM MQ clients, but an IBM MQ client in other platforms was able to connect to an IBM MQ for the z/OS queue manager.



Important

This course focuses on IBM MQ client behavior for IBM MQ V8 only; however, the topic on creating a CCDT on the z/OS side and downloading to a client is applicable to pre-V8 IBM MQ.

IBM MQ client connectivity options and search order



© Copyright IBM Corporation 2015

Figure 5-3. IBM MQ client connectivity options and search order

WM3021.1

Notes:

IBM MQ client applications have several options to connect to a queue manager.

1. If MQCONN is used in the application, the connectivity fields of the MQCNO structure are examined. If present, these fields identify structure to be used as the definition of the CCDT. If MQCONN is used, it follows the next options.
2. If the MQSERVER environment variable is set, the channel information is used to connect.
3. If the MQCHLLIB and MQCHLTAB environment variables are set, the CCDT referred to by these variables is used to connect.

If no information is found, the client looks for an mqs.ini file for the location of the CCDT.

Then, if no information is found, the client looks for the CCDT in the default location according to the platform-specific convention for IBM MQ CCDT locations on the client side.

MQI channel objects

- SVRCONN
 - Channel type that all clients use
 - MCAUSER security implications, SVRCONN attributes, and performance considerations
 - Use standard facilities to define and display status on IBM MQ server
- CLNTCONN with runmqsc -n option
 - Correct method to use to create CCDT for IBM MQ for z/OS V8 and later
 - CCDT created at the client side, no CCDT download necessary
- CLNTCONN with CSQUTIL MAKECLNT option
 - MAKECLNT last update before IBM MQ V8
 - CCDT had to be downloaded to client
 - Do not use MAKECLNT for IBM MQ V8 for z/OS and later

© Copyright IBM Corporation 2015

Figure 5-4. MQI channel objects

WM3021.1

Notes:

On the z/OS queue manager side, the SVRCONN channel is defined for use by clients. It is not necessary to have a CLNTCONN to connect to an SVRCONN channel. However, the use of a CLNTCONN and a CCDT provides more capabilities, such as using SSL, or being able to define an alternative queue manager to connect to, perhaps for a failover situation.

The setting of the SVRCONN SHARECNV attribute can be tuned for performance. Having a single socket host multiple connections is called multiplexing, and might enhance the performance of an application. The following sections of the IBM MQ Knowledge Center explains the settings and application considerations for the SHARECONV settings, including enhancements for distributed platforms in IBM MQ V8:

- Tuning client and server connection channels
- MQI client: Default behavior of client-connection and server-connection channels

When displaying channel status, the SVRCONN-specific attribute CURSHCNV can be used to determine whether applications are using multiplexed connections.

When defining CLNTCONN channels on a server side, all channels must be defined in the same queue manager. That is, when using queue manager CSQ1, even if there are two more CLTCONN

channels for queue managers CSQ2 and CSQ3, these CLNTCONN definitions must also be defined in the CSQ1 queue manager. The reason is that CLNTCONN creates the file that needs to be downloaded to the client side. This file cannot be concatenated or patched up from several queue managers; it must be one file from the same queue manager.

A new option with IBM MQ V8 is the ability to generate CLNTCONN channels in the client side. This option is described in the notes, since it is outside the IBM MQ z/OS environment. The following example uses a Windows platform.



Note

Defining a CLNTCONN on the client side. Use the “-n” option of runmqsc to create a CCDT AMQCLCHL.TAB file.

The SVRCONN on the IBM MQ server queue manager must match the name of the CLNTCONN, in this case MQ00.CL.

```
C:\>runmqsc -n
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting local MQSC for 'AMQCLCHL.TAB'.
DEF CHL(MQ00.CL) CHLTYPE(CLNTCONN) TRPTYPE(TCP)
CONNNAME( 'mvsmpc11.ilsvpn.ibm.com(1600)' )
    1 : DEF CHL(MQ00.CL) CHLTYPE(CLNTCONN) TRPTYPE(TCP) CONNNAME( 'mvsmpc11.ilsvpn
.ibm.com(1600)' )
AMQ8014: WebSphere MQ channel created.
end
    2 : end
```

No commands have a syntax error.

For a Windows platform, the file was created in the MQ directory; for the installation that was used, it was:

```
C:\IBM\WebSphere MQ.
```

If `mqs.ini` was previously set to another location, update `mqs.ini` with the location of the new CCDT.

It is possible that the channel might work, but the connection might be rejected. Always check the return code by using a sample test application on the client side. If the z/OS log shows a connection, the CLNTCONN definition was good, as shown in the next example. Now if a 2035 return code occurred on the client side, you would need to work with z/OS security, which you do in Unit 11.

For results of testing the channel on the z/OS side, notice that although the connection was blocked (resulting in a 2035 code), the channel did initially start, so the definition was good.

```
+CSQX511I MQ00 CSQXRESP Channel MQ00.CL started 438  
connection 10.4.127.184  
+CSQX777E MQ00 CSQXRESP Channel MQ00.CL from 10.4.127.184 439  
(10.4.127.184) has been blocked due to USERSRC(NOACCESS), Detail:  
CLNTUSER(nomad789)
```



MQI environment variables in IBM MQ client side

MQSERVER

- Set MQSERVER environment variable where ChannelName matches the SVRCONN name
set MQSERVER=ChannelName/TransportType/ConnectionName
- Example:**
MQSERVER=MQ00.CLIENT/TCP/mvsmc11.ilsvpn.ibm.com(1600)
- The SVRCONN on the server side must be called MQ00.CLIENT

MQCHLLIB and MQCHLTAB (CCDT)

- Set MQCHLLIB and MQCHLTAB environment variables
- Examples:
MQCHLLIB=C:\IBM\WEBSPH~1\Qmgrs\QMGR1\@ipcc
MQCHLTAB=AMQCLCHL.TAB

MQCLNTCF

- mqclient.ini provides an alternative to using MQSERVER, MQCHLLIB, or MQCLTAB
- It is not necessary to use the **MQCLNTCF** variable to locate mqclient.ini
- MQCLNTCF** can be used to specify an alternative location for mqclient.ini

© Copyright IBM Corporation 2015

Figure 5-5. MQI environment variables in IBM MQ client side

WM3021.1

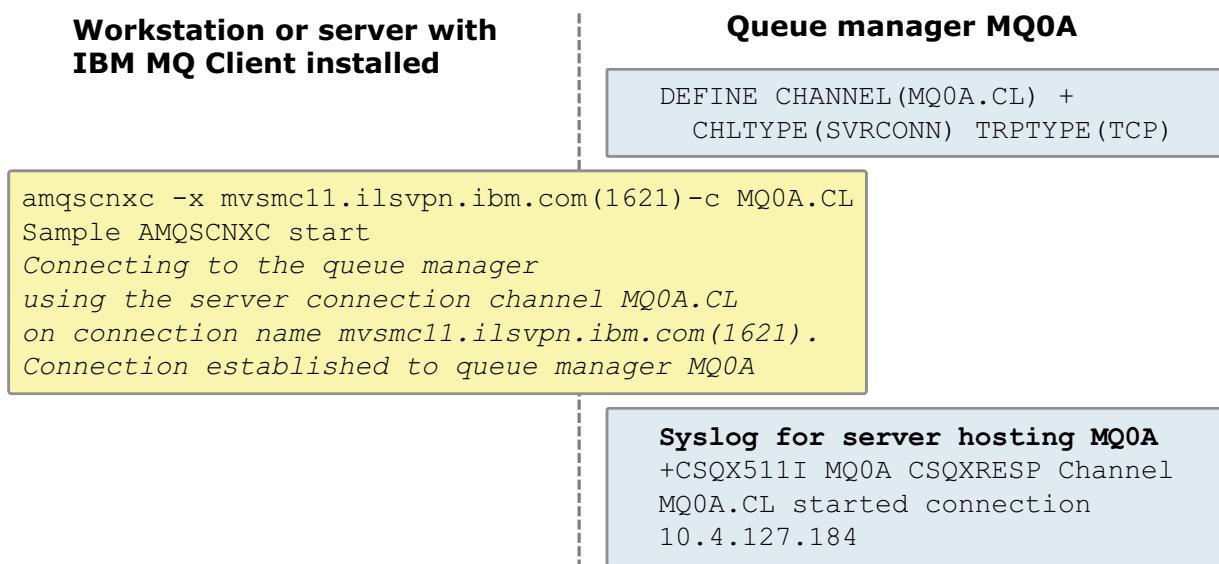
Notes:

Remember to update your variables if any changes are made to the SVRCONN channel location or to the location of the CCDT file that your applications use.



Testing client to server connectivity with MQCONN

- Components required:
 - SVRCONN channel on the IBM MQ server side
 - IBM MQ client is installed on the client side
 - IBM MQ sample client program to test connectivity: amqscnxc



© Copyright IBM Corporation 2015

Figure 5-6. Testing client to server connectivity with MQCONN

WM3021.1

Notes:

This slide shows a connection that uses an MQCONN call that the IBM MQ sample application amqscnxc issues in a Windows platform, to the SVRCONN channel MQ0A.CL defined in the IBM MQ for z/OS queue manager MQ0A.

On the client side, there was no MQSERVER variable or client connection definition table (CCDT) used. The MQCONN call in the application contains all the information necessary to connect to the MQ0A.CL SVRCONN channel in the IBM MQ server queue manager, without any variables set or CCDT available.

Connect with CCDT and put a message (1 of 2)

- Required:
 - SVRCONN and local queue object definitions on IBM MQ queue managers
 - CCDT with all CLNTCONN definitions created with runmqsc -n on client side
 - For this example, CCDT information is specified on mqclient.ini without any variables set
 - Sample IBM MQ client put program on client side: amqspputc

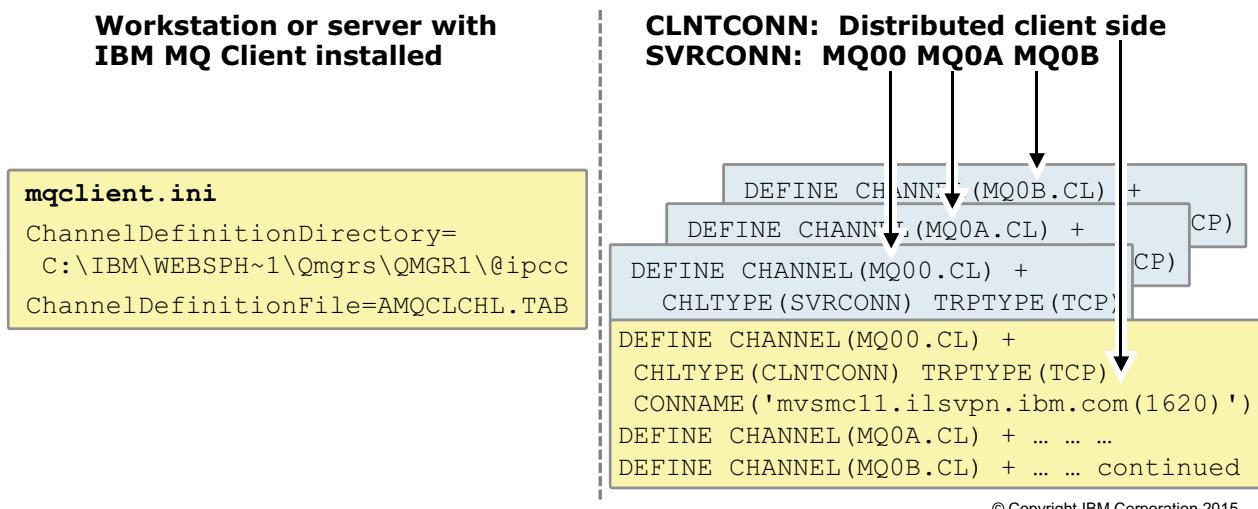


Figure 5-7. Connect with CCDT and put a message (1 of 2)

WM3021.1

Notes:

These two slides walk through the setup of a CCDT on the z/OS side.

First, SVRCONN channels are defined in each queue manager where a client channel is defined.

- SVRCONN MQ00.CL is defined in the MQ00 queue manager.
- SVRCONN MQ0A.CL is defined in the MQ0A queue manager.
- SVRCONN MQ0B.CL is defined in the MQ0B queue manager.

Now choose a queue manager for the CLNTCONN definitions and use MQ00 to define all the CLNTCONN channels in the same queue manager.

You continue on slide 2.

Connect with CCDT and put a message (2 of 2)

Distributed client side

```
runmqsc -n
DEFINE CHANNEL (MQ00.CL) CHLTYPE (CLNTCONN) +
TRPTYPE (TCP) QMNAME (MQ00)
CONNAME ('mvsmc11.ilsvpn.ibm.com(1620)')
DEFINE CHANNEL (MQ0A.CL) CHLTYPE (CLNTCONN) +
TRPTYPE (TCP) QMNAME (MQ0A)
CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)')
DEFINE CHANNEL (MQ0B.CL) CHLTYPE (CLNTCONN) +
TRPTYPE (TCP) QMNAME (MQ0A)
CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)')
```



C:\IBM\WEBSPH~1\Qmgrs\QMGR1\@ipcc

```
amqsputc ORDERS.AT.MQ00 MQ00
Sample AMQSPUTO start
target queue is ORDERS.AT.MQ00
Hello test CCDT setup
Sample AMQSPUTO end
```

+CSQX511I MQ00 CSQXRESP Channel MQ00.CL
started ...

/MQ00 DIS Q(ORDERS.AT.MQ00) CURDEPTH
QUEUE(ORDERS.AT.MQ00)... CURDEPTH(1)

1. Configure all CLNTCONN channels in the distributed client
2. If necessary, move the CCDT to the directory indicated by mqclient.ini
3. Use amqsputc to test
4. Check IBM MQ server side log and queue

IBM MQ for z/OS server side

© Copyright IBM Corporation 2015

Figure 5-8. Connect with CCDT and put a message (2 of 2)

WM3021.1

Notes:

You again look at the three CLNTCONN definitions that are made for MQ00, MQ0A, and MQ0B on queue manager MQ00. However, the MQ0B and MQ0A channel all have QMNAME as MQ0A because MQ0B is used as an alternative queue manager for MQ0A, should MQ0A not be available. This type of definition with two queue managers with the same QMNAME is called a queue manager group.

You now run the CSQUTIL with the MAKECLNT command to generate the CCDT. After the CCDT is created, download the file to the client side and rename the file, preferably to: AMQCHCHL.TAB

The renamed file should be placed in the directory indicated by the mqclient.ini file. If not using the mqclient.ini, check IBM Knowledge Center to confirm the directory where the IBM MQ client searches for this file as a default location, and place it there. For this slide, you used the mqclient.ini and specified the default location for the CCDT on a Windows platform.

The client uses the MQ00.CL channel to connect to the MQ00 queue manager (see z/OS log message CSQX511I for the channel name), and the message is put to the ORDERS.AT.MQ00 queue.

If the MQSERVER environment is set in the client side, the messages go to the queue manager indicated by the channel in the MQSERVER variable; if different, then they go to MQ00.CL.

CCDT and queue manager groups

- The subset of definitions in the CCDT that share the same QMNAME attribute is called a queue manager group
 - Use for connecting to alternative servers if a failure occurs
 - Ability to assign different weights to influence load balancing
 - Ability to change queue managers without having to change application code

```

DEFINE CHANNEL (MQ00.CL) CHLTYPE (CLNTCONN) +
  TRPTYPE (TCP) QMNAME (MQ00)
  CONNAME ('mvsmc11.ilsvpn.ibm.com(1620)')
DEFINE CHANNEL (MQ0A.CL) CHLTYPE (CLNTCONN) +
  TRPTYPE (TCP) QMNAME (MQ0A) CLNTWGHT (2)
  CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)')
DEFINE CHANNEL (MQ00.CL) CHLTYPE (CLNTCONN) +
  TRPTYPE (TCP) QMNAME (MQ0A) CLNTWGHT (4)
  CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)')

```

© Copyright IBM Corporation 2015

Figure 5-9. CCDT and queue manager groups

WM3021.1

Notes:

Use MQSERVER environment variable

- Components required:
 - SVRCONN channel and local queue on the IBM MQ server (svr) side
 - IBM MQ client is installed on the client side
 - IBM MQ sample client program: amqsputc or amqsgetc

Client

```
C:\>set MQSERVER=MQ00.CL/TCP/mvsmc11.ilsvpn.ibm.com(1600)
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
target queue is BACK.ORDER
abc
Sample AMQSPUTO end
```

Server

+CSQX511I MQ00 CSQXRESP Channel MQ00.CL started 637 connection 10.4.127.184	/MQ00 DIS Q(ORDERS.AT.MQ00) CURDEPTH QUEUE(ORDERS.AT.MQ00)... CURDEPTH(2)
---	--

Client

```
Example of running amqsgetc
C:\IBM\WebSphere MQ\tools\c\Samples\Bin>amqsgetc ORDERS.AT.MQ00
Sample AMQGET0 start
message <Hello test CCDT setup>
message <abc>
no more messages
Sample AMQGET0 end
```

© Copyright IBM Corporation 2015

Figure 5-10. Use MQSERVER environment variable

WM3021.1

Notes:

Components that are required for this slide include the sample programs, assuming that there are no application programs available. Should there be application programs, they might be used instead of the samples. However, if an application program does not work, then use the sample to confirm that there are no problems with the application.

A Windows platform server represents the client side. You set the MQSERVER client to the host name and port of the IBM MQ server SVRCONN channel that you need to connect to.

As soon as the MQSERVER is set, use a sample application to test the program. It is only required to pass the queue name to the amqsputc sample application. If the MQSERVER definition is correct, you see the result of accessing the application in the IBM MQ for z/OS log. If the channel starts, you know that the MQSERVER was set correctly. If there are any security or other problems, the channel comes down, and you need to check the IBM for z/OS log information for the queue manager where the connection was directed for any additional information. The connection in this display worked.

You also display the target queue and see there are two messages. You can use the sample amqsgetc to get these messages from the target queue manager. The demonstrated process would make little sense as an application. It is used here only to show how to use the IBM MQ client side.

IBM MQ client connection capabilities summary

Connection capability	MQCONNX - MQCNO	MQSERVER	CCDT
Connect to alternate queue managers	NO	YES* <i>if ConnectionName is a list</i>	YES
Security exits	YES	NO	YES
Use a subset of channel attributes	YES	Few attributes available	YES
SSL <i>Distributed platforms only</i>	YES	NO	YES
Server failover detection, load balance, and weighting	NO	NO* <i>ConnectionName list helps failover</i>	YES

© Copyright IBM Corporation 2015

Figure 5-11. IBM MQ client connection capabilities summary

WM3021.1

Notes:

This table summarizes the IBM MQ client capabilities per selected connectivity option.

The CCDT version can be implemented in either of the following ways:

- Create the CCDT in the queue manager and download it to the client.
- If you have IBM MQ V8 or later, use `runmqsc -n` to generate the CCDT in the client.

The names of the CLNTCONN and SVRCONN channels must match.

IBM MQ client security considerations

Authentication:

- MCAUSER attribute of SVRCONN
 - If left blank, a platform-dependent ID is used 
 - On z/OS, it is the same ID as CHINIT and might grant similar rights
 - Channel authentication records can be used to map the MCAUSER of the same SVRCONN channel to different user IDs with different authorizations
- mqccred exit
 - Assumption: IBM MQ server has CONNAUTH enabled
 - Use with client applications that do not currently send a user ID and password
 - mqccred.ini file and MQCCRED variable
 - Password obfuscation for mqccred.ini file
 - Update CLNTCONN definition with SCYEXIT("mqccred(ChIExit)") attribute
- Message exits

Authorization:

- Business as usual on the server side
- Consider use of CHLAUTH

© Copyright IBM Corporation 2015

Figure 5-12. IBM MQ client security considerations

WM3021.1

Notes:

Some IBM MQ details are “well known” in the user community, which includes anyone who is trying to hack into an environment. IBM MQ default port 1414, and SYSTEM.ADMIN.SVRCONN channels, are among these known details.

When connecting with client channels, applications should take care to use a different port for their queue managers and a different name for their SVRCONN channels. Changing port number and channel name alone does not remedy the situation. The MCAUSER value for the SVRCONN should be set to a user with few privileges. A blank MCAUSER on the z/OS platform runs under the same ID as the channel initiator and would have similar rights.

The mqccred exit and CHLAUTH are explained in Unit 11.

WebSphere MQ Base and Extended transactional client

- A WebSphere MQ transactional client:
 - Is a WebSphere MQ Base client with extra transactional capabilities
 - Is supplied with WebSphere MQ V8
 - Is a separate installation for WebSphere MQ V7.5 and earlier
 - The WebSphere MQ transactional client is available for Linux, UNIX, and Windows platforms
- The differences in transactional capabilities between WebSphere MQ Base and WebSphere MQ Extended transactional client are:

WebSphere MQ Base client	WebSphere MQ Extended transactional client
<ul style="list-style-type: none"> • Can participate in a unit of work that is managed by the queue manager to which it is connected • Able to use MQCMIT or MQBACK to complete the unit of work • Unable to update resources that owned by another resource manager, such as a database 	<ul style="list-style-type: none"> • Is able to put to and get messages from the queue manager to which it is connected • Is able to update resources that are owned by another resource manager in the same unit of work • Unit of work must be managed by an external transaction manager that is collocated with the client application • Uses external transaction manager API, no MQCMIT, or MQBACK

© Copyright IBM Corporation 2015

Figure 5-13. WebSphere MQ Base and Extended Transactional Client

WM3021.1

Notes:

All IBM MQ clients were not created equal, but starting with IBM MQ V8, it is no longer necessary to install a separate client to get the capabilities provided by the transactional client.

The two tables in this slide summarize the differences between the basic IBM MQ client and the IBM MQ extended transactional client.

A client application that uses a transactional client can connect to a queue manager on IBM MQ for z/OS.

Unit summary

- Distinguish between an IBM MQ server and an IBM MQ client
- Describe the various ways of connecting WebSphere MQ clients to IBM MQ servers
- Describe security considerations of an IBM MQ client
- Summarize transactional capabilities of an IBM MQ client

© Copyright IBM Corporation 2015

Figure 5-14. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. True or False: An IBM MQ client installation does not have any configured queue manager or queues.

2. Which statements are true about IBM MQ V8 client support for connection authentication?
 - a. The mqccred message exit can be used to provide credentials to enable CONNAUTH for existing client applications without modifying the client code.
 - b. There are no provisions to obfuscate the password sent to the IBM MQ server when using CONNAUTH; passwords are always sent as clear text.
 - c. Sample IBM MQ client programs have different capabilities to test connection authentication that is configured on the IBM MQ server.

© Copyright IBM Corporation 2015

Figure 5-15. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

1.

2.

Checkpoint questions (2 of 2)

3. True or False: IBM MQ for z/OS must continue to use the CSQUTIL MAKECLNT option to create a client channel definition table to use with IBM MQ V8 clients.

4. The entries that are listed are defined in an IBM MQ client. Which connection is used when an application puts a message to the IBM MQ server queue *without* connectivity details in an MQCD structure?
 - a. Client channel ABC.CONN in the CCDT file found at the default location
 - b. The MQSERVER environment variable that points to the XYZ.CONN channel
 - c. The default client channel connection file
 - d. Client channel DEF.CONN, the CCDT file at the location pointed to by the mqclient.ini file

© Copyright IBM Corporation 2015

Figure 5-16. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

Checkpoint answers (1 of 2)

1. True or False: An IBM MQ client installation does not have any configured queue manager or queues.

Answer: True

2. Which statements are true about IBM MQ V8 client support for connection authentication?

- a. The mqccred message exit can be used to provide credentials to enable CONNAUTH for existing client applications without modifying the client code.
- b. There are no provisions to obfuscate the password sent to the IBM MQ server when using CONNAUTH; passwords are always sent as clear text.
- c. Sample IBM MQ client programs have different capabilities to test connection authentication that is configured on the IBM MQ server.

Answer: a, c. Only V8 and later clients can obfuscate the password.

© Copyright IBM Corporation 2015

Figure 5-17. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. True or False: IBM MQ for z/OS must continue to use the CSQUTIL MAKECLNT option to create a client channel definition table to use with IBM MQ V8 clients.

Answer: False. IBM MQ V8 on z/OS must discontinue use of MAKECLNT.

4. The entries that are listed are defined in an IBM MQ client. Which connection is used when an application puts a message to the IBM MQ server queue *without* connectivity details in an MQCD structure?
- a. Client channel ABC.CONN in the CCDT file found at the default location
 - b. The MQSERVER environment variable that points to the XYZ.CONN channel
 - c. The default client channel connection file
 - d. Client channel DEF.CONN, the CCDT file at the location pointed to by the mqclient.ini file

Answer: b

© Copyright IBM Corporation 2015

Figure 5-18. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 4



Working with IBM MQ clients

© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 5-19. Exercise 4

WM3021.1

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Use the IBM MQ V8 client-side **runmqsc** command to configure a client channel definition table (CCDT) on the client side
- Define a client connection to a queue manager with the **MQSERVER** variable
- Use sample IBM MQ client programs to place messages in an IBM MQ server queue
- Configure IBM MQ Explorer to administer your queue manager, and browse the MQ00, MQ0A, and MQ0B queue managers

© Copyright IBM Corporation 2015

Figure 5-20. Exercise objectives

WM3021.1

Notes:



Exercise instructions

- In this exercise you use:
 - The IBM MQ for z/OS administrative capabilities
 - Your VMware image Windows command prompt
 - IBM MQ Explorer

© Copyright IBM Corporation 2015

Figure 5-21. Exercise instructions

WM3021.1

Notes:

Unit 6. IBM MQ cluster basics

What this unit is about

This unit reviews what an IBM MQ cluster is, describes cluster configuration, identifies considerations to observe, and provides cluster troubleshooting tips.

What you should be able to do

After completing this unit, you should be able to:

- Describe what an IBM MQ cluster is
- List the information to review before implementing IBM MQ clusters
- Describe the components of an IBM MQ cluster
- Define a basic cluster
- Interpret the contents of a cluster display and other cluster commands
- Recognize and resolve cluster problems
- Define and put messages to cluster queues
- Describe the commands and options that are used to administer a cluster
- Summarize how to influence cluster workload balancing
- Describe how to define a connection from outside the cluster to a cluster gateway
- List cluster best practices

Unit objectives

- Describe what an IBM MQ cluster is
- List the information to review before implementing IBM MQ clusters
- Describe the components of an IBM MQ cluster
- Define a basic cluster
- Interpret the contents of a cluster display and other cluster commands
- Recognize and resolve cluster problems
- Define and put messages to cluster queues
- Describe the commands and options that are used to administer a cluster
- Summarize how to influence cluster workload balancing
- Describe how to define a connection from outside the cluster to a cluster gateway
- List cluster best practices

© Copyright IBM Corporation 2015

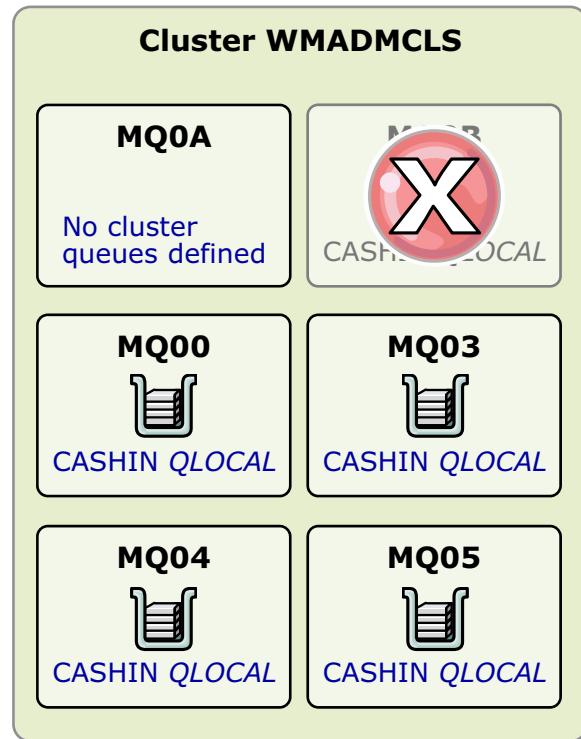
Figure 6-1. Unit objectives

WM3021.1

Notes:

IBM MQ clusters

- Simplified administration
 - Reduce number of remote queues, transmit queues, and channel definitions
- Workload balancing
 - Same queue can be hosted in several queue managers
 - Route around failures
- Scalable
- Contribute to high availability
- MQPUT to local queue manager if queue exists locally
 - If the queue is not found locally, an algorithm is used for target queue selection
- MQGET done to local queue manager
- Publish/subscribe can use clusters



© Copyright IBM Corporation 2015

Figure 6-2. IBM MQ clusters

WM3021.1

Notes:

An IBM MQ cluster is a grouping of queue managers such that the queue managers share knowledge of each other's queues. They can exchange messages across queue managers without the need to define new channels as soon as the cluster is set up. After a queue manager “joins” a cluster, it learns about the queues in other clusters.

If an application needs to put a message to the CASHIN queue from MQ0A, this queue would be available in queue managers MQ00, MQ03, MQ04, and MQ05. IBM MQ recognizes the member queue manager that is unavailable and excludes that queue manager from the potential target queue managers.

As you look at how the clusters are defined, the reduction in the number of required IBM MQ object definitions becomes apparent.

This slide qualifies the statement “contribute to high availability” because a cluster is not a high availability solution. Messages in cluster queues in an unavailable cluster queue manager are inaccessible unless the queue manager is part of a queue-sharing group or configured for high availability. However, clustering contributes to high availability, scalability, and workload balancing for new incoming requests.

Queue-sharing groups are discussed in a later unit. A queue part of a queue-sharing group (called a shared queue) can be clustered.

When working with clusters, the significant parts are the MQOPEN and the MQPUT, which are where cluster workload balancing occurs, particularly the MQOPEN.

MQGET calls are always sent to the local queue manager.

Before you start

- What is the strategic plan for the cluster?
- Does the decision include comparison with queue-sharing groups?
- Is there adequate communication with the IBM MQ administrator staff about standards for use of the cluster?
- Are there any known queue manager affinities in the applications?
- Is there adequate communication with development?
- Is a team identified to baseline workload balancing?
- What are the expected message sizes?
- Do system support objects need to change sizes if clustered?
 - Dead letter queue
 - SYSTEM.CLUSTER.TRANSMIT.QUEUE
- Is the staff trained to support the cluster?
- Are adequate servers to hold full cluster repository queue managers and gateways identified?

© Copyright IBM Corporation 2015

Figure 6-3. Before you start

WM3021.1

Notes:

When a decision is made to implement a cluster, many times the impact to the entire organization is underestimated.

If the only purpose of the cluster is to simplify definitions, perhaps the effort is smaller; however, as soon as a cluster is in place, users expect workload balancing to follow.

If workload balancing is an objective, applications need to look at their code, first for any affinities, and also for any attributes in use that might interfere with the default or configured workload balancing. In this case, IBM Integration Bus, WebSphere Message Broker, or the process that creates messages to go on a clustered queue is also considered an application.

There are also considerations as to which IBM MQ server should host the cluster repositories, and what the IBM MQ upgrade schedules are. These considerations arise because the queue managers that are holding the full cluster repositories should be at the highest level of IBM MQ within the cluster.

Another consideration is the size of the messages. If one application exceeds the default maximum message length, then all cluster-related `SYSTEM.*` queues should reflect the largest possible

message size to be exchanged within the cluster. The `SYSTEM.*` queues in this case also include the dead-letter queue.

Who has control of the cluster definitions? Is it a subset of IBM MQ administrators, or can anyone alter definitions that might affect the cluster?

What about interfacing applications: does your organization plan to allow them to join the cluster, or does it allow an outside cluster to overlap or join your cluster?

Clustering itself is straightforward, but the organizational issues that affect the cluster need to be resolved before an implementation plan is completed.

IBM MQ cluster components

- Repositories
 - Collection of information about the cluster
 - Cluster queue managers can hold a full or a partial repository
- Queue manager system support objects
 - SYSTEM.CLUSTER.COMMAND.QUEUE
 - SYSTEM.CLUSTER.REPOSITORY.QUEUE
- CLUSRCVR channels
- CLUSSDR channels
 - Point to one and only one full repository
 - Communicate any cluster-related changes to the full repository
- Transmission queues
 - SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Alternative cluster transmit queue
- Clustered queues

© Copyright IBM Corporation 2015

Figure 6-4. IBM MQ cluster components

WM3021.1

Notes:

IBM MQ clusters rely on several components. This slide omits a most important one, which is the cluster or repository manager. In distributed systems, the repository is a process that is called amqrrmfa. As you learned in Unit 2, in z/OS the repository manager is part of the channel initiator. The following aspects are important:

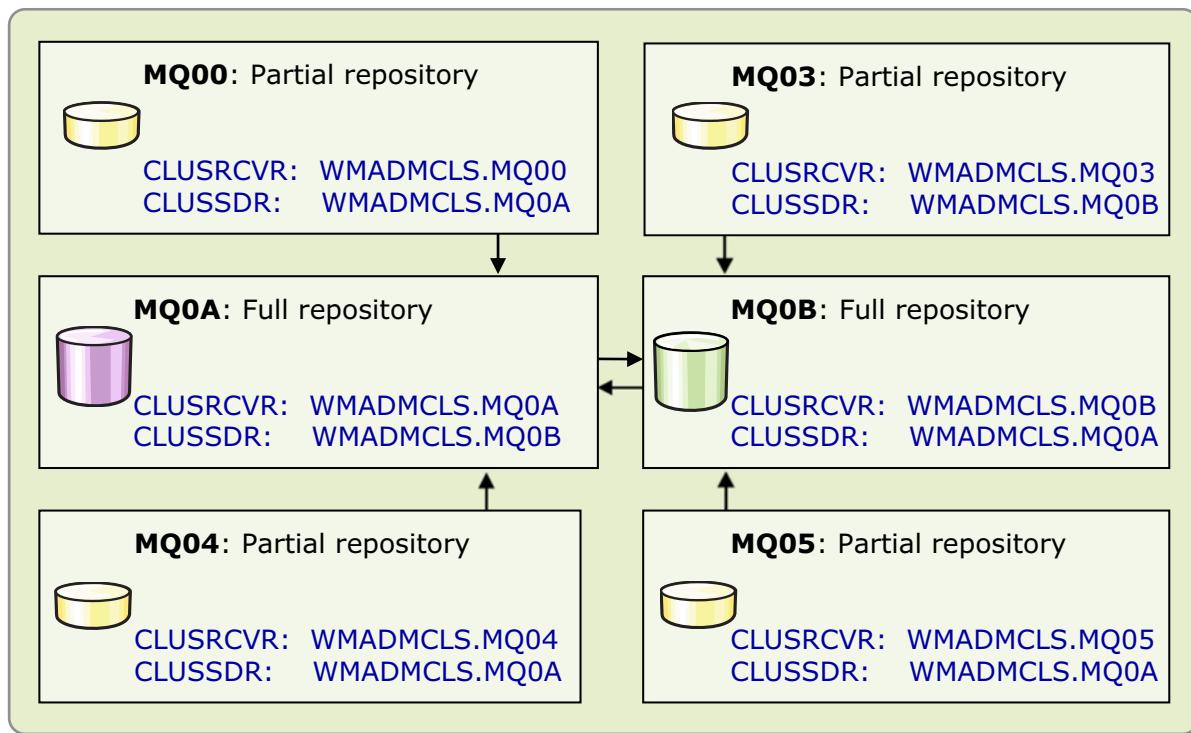
- Repositories:
 - The repository is where all information about the cluster is stored. There are two types of repositories:
 - A full repository, which is held by a queue manager with the REPOS queue manager attribute set to the cluster name.
 - A partial repository, which is held by a member queue manager. The full repositories are critical. There should always be two queue managers that are designated as full cluster repositories, and the CLUSSDR channels for these two queue managers should point to each other.
 - Full repositories have information about the entire cluster, while partial repositories keep information about those objects that are of interest to what they need to process.

- If one full repository is available, it is possible for a cluster to continue functioning. That is, although the optimal case is to have the two full repository queue managers available, it is not essential. If one queue manager is brought down for maintenance, the cluster is able to operate with the remaining full cluster repository queue manager active.
 - When a member queue manager defines a queue, it tells the repository that it connects to, about this new queue.
- Support objects:
 - The queue managers have two SYSTEM.* queues that support the cluster.
 - The SYSTEM.CLUSTER.REPOSITORY queue, as its name implies, keeps the cluster repository. This queue should never be empty for an active cluster.
 - The SYSTEM.CLUSTER.COMMAND.QUEUE receives commands and information that the repository manager processes, and it should normally be empty. When messages accumulate in this queue, it is a good indication that the repository manager is not picking them up. Therefore, if there are any cluster problems, you know to check the channel initiator started task for any errors. If you are working with a distributed administrator, you can also check whether the amqrrmfa process is running or is not running.
 - Cluster channels:
 - Understanding the channels is key to learning about clusters. There are two cluster-related channel definitions:
 - CLUSRCVR channels. Unlike distributed RCVR channels, the CLUSRCVR has the connection information. This CLUSRCVR connection information is what cluster member queue managers use to create a dynamic CLUSSDRx channel to the queue manager that owns the CLUSRCVR. The dynamic CLUSSDR channel is of type CLUSSDRA or CLUSSDRB. That is, when using clusters, a member queue manager that just joined a cluster checks the repository for the information on a queue manager that it needs to send messages to. It then uses the connection name in the CLUSRCVR to create a dynamic channel to the required queue manager.
 - CLUSSDR channels. A queue manager that joins the cluster should have one CLUSSDR channel, and only one CLUSSDR channel, which is pointing to **one** of the two full cluster repository queue managers.
 - Thus, each queue manager that joins a cluster has one CLUSRCVR and one CLUSSDR channel, and can create dynamic `CLUSSDRx` channels to other cluster member queue managers.

What is the “x” in `CLUSSDRx`?

- A local queue manager might create a dynamic channel to a remote queue manager, and the CLUSSDR in that local queue manager is **not** pointing to the target remote queue manager. In this case, the CLUSSDR channel displays as a CLUSSDRA.
- When this local queue manager has a CLUSSDR channel that is pointing to the target remote queue manager, the dynamic channel that is created to that remote queue manager where CLUSSDR is pointing displays as CLUSSDRB.

Queue manager cluster definition (1 of 2)



Note: Arrows represent the full repository that CLUSSDR channels point to

© Copyright IBM Corporation 2015

Figure 6-5. Queue manager cluster definition (1 of 2)

WM3021.1

Notes:

Before defining a cluster, it is important to draw a plan and determine what queue managers host the full cluster repositories. For the WMADMCLS cluster, MQ0A and MQ0B are designated as full cluster repositories. MQ0A and MQ0B have:

- The queue manager REPOS attribute set to the cluster name WMADMCLS
- The CLUSRCVR channel that points to their own connection information
- **One** CLUSSDR channel for each, which points to the other

The rest of the queue managers that join the cluster have:

- The CLUSRCVR channel that points to their own connection information
- One CLUSSDR channel that points to **one** of the full repository queue managers
- No changes to the queue manager object

The naming convention that is used for the channels includes the cluster name as the first node, which represents one of the cluster “best practices.”

Queue manager cluster definition (2 of 2)

At MQ0A: Full repository

```
ALTER QMGR REPOS (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') cluster (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') cluster (WMADMCLS)
```

At MQ0B: Full repository

```
ALTER QMGR REPOS (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') cluster (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') cluster (WMADMCLS)
```

At MQ00: Partial repository

```
DEFINE CHL (WMADMCLS.MQ00) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1600)') CLUSTER (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0A) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1621)') CLUSTER (WMADMCLS)
```

At MQ03: Partial repository

```
DEFINE CHL (WMADMCLS.MQ03) CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1603)') CLUSTER (WMADMCLS)
DEFINE CHL (WMADMCLS.MQ0B) CHLTYPE (CLUSSDR) TRPTYPE (TCP) +
    CONNAME ('mvsmc11.ilsvpn.ibm.com(1622)') CLUSTER (WMADMCLS)
```

© Copyright IBM Corporation 2015

Figure 6-6. Queue manager cluster definition (2 of 2)

WM3021.1

Notes:

If you created the cluster in the previous slide, these definitions would be the ones for queue managers MQ0A, MQ0B, MQ00, and MQ03.

Should there be an issue with large messages that come across the cluster, there are three other definitions you might want to add to each queue manager:

- Assuming that `SYSTEM.DEAD.LETTER.QUEUE` is your designated DLQ, and you are not using separate cluster transmit queues:

```
ALTER QMGR MAXMSGL(104857600)
ALTER QL(SYSTEM.DEAD.LETTER.QUEUE) MAXMSGL(104857600)
ALTER QL(SYSTEM.CLUSTER.TRANSMIT.QUEUE) MAXMSGL(104857600)
```

- These additional definitions should not be added if not needed because the larger sizes consume more system resources for the channels.

When the cluster definitions are completed, you always need to thoroughly check the results by using the `DIS CLUSQMGR` command. You now walk through the verification or checking process.

MQ0A definition check 1 of 2: What happens in the cluster

/MQ0A DIS CLUSQMGR(*) ALL (partial display)

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0A) ————— This definition is for the MQ0A queue manager itself
 CLUSTER (WMADMCLS)
 CHANNEL (WMADMCLS.MQ0A)
 QMID (MQ0A.CDA33A44AE870B2B)
 DEFTYPE (CLUSRCVR) ————— CLUSRCVR channel type definition for MQ0A
 QMTYPE (REPOS) ————— QMTYPE REPOS: This queue manager (MQ0A) holds a full repository
 ...

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ0B) ————— The partner queue manager for this MQ0A connection is MQ0B
 CLUSTER (WMADMCLS)
 CHANNEL (WMADMCLS.MQ0B)
 QMID (MQ0B.CDA2735757A9FEA9)
 DEFTYPE (CLUSSDRB) ————— A CLUSSDRB is the combination of both MQ0B's CLUSRCVR and MQ0A's CLUSSDR; MQ0A defined CLUSSDR to MQ0B
 ...
 QMTYPE (REPOS) ————— The MQ0B queue manager holds the second full repository
 STATUS (RUNNING)
 ...

© Copyright IBM Corporation 2015

Figure 6-7. MQ0A definition check 1 of 2: What happens in the cluster

WM3021.1

Notes:

From one of the full cluster repository queue managers, you issue the `DIS CLUSQMGR(*) ALL` command. You use MQ0A. The output from the command spans two slides. What do you check for?

- Immediately after the cluster is created, you expect to see the cluster channels with a `RUNNING` status. For the MQ0A display, it looks good.
- If any errors have a “`SYSTEM TEMP .xxxx`” name as the object, there is a problem. These error entries are usually:
 - `CLUSQMGR (SYSTEM TEMPQMGR.xxxx.xx.xxxx)`
 - `QMID (SYSTEM TEMPUUID.xxxx.xx.xxxx)`
 - No `CLUSSDR` type channels should display. They should all be `CLUSSDRA` or `CLUSSDRB`. A `CLUSSDR` in the `DIS CLUSQMGR` indicates a problem.
- So far all looks well. Also, remember that when an error is made in the connectivity information of the cluster channel, the correction is similar to distributed channels. Therefore, if you see one of these `SYSTEM TEMP .xxxx.xx.xxxx` messages, the first place to check is the `CONNNAME` attribute for the `CLUSRCVR` and the `CLUSSDR`.

Review the notes that are embedded in the slide for the selected DIS CLUSQMGR attributes. By the end of the display in this slide, you know that the cluster connection from MQ0A to MQ0B is good and that MQ0A and MQ0B are both full repositories (QMTYPE - REPOS). Also, MQ0B created a dynamic CLUSSDRB channel to MQ0A, which is in RUNNING status. Why is this CLUSSDR channel type from MQ0A to MQ0B a CLUSSDRB?

MQ0A definition check 2 of 2: What happens in the cluster

CSQM201I MQ0A CSQMDRTC DIS ...

CLUSQMGR (MQ00) _____ The partner queue manager for this MQ0A connection is MQ00

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ00)

QMID (MQ00.CDA33FC1E2935036)

DEFTYPE (CLUSSDRA) _____ CLUSSDRAs show a dynamic connection from MQ0A to MQ00's CLUSRCVR; MQ0A has no defined CLUSSDR to MQ0A
...

QMTYPE (NORMAL) _____ The MQ00 queue manager holds a partial repository

STATUS (RUNNING)

VERSION (08000000)

XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE)

TRPTYPE (TCP)

CONNNAME (mvsmc11.ilsvpn.ibm.com(1600))

...

/MQ0B DIS CLUSQMGR(*) ALL output is similar to MQ0A and bypassed for brevity

© Copyright IBM Corporation 2015

Figure 6-8. MQ0A definition check 2 of 2: What happens in the cluster

WM3021.1

Notes:

You now look at the rest of the DIS CLUQMGR display. In this part of the output you see that:

- The information is what cluster queue manager MQ0A learns about the MQ00 cluster queue manager.
- Channel WMADMCLS.MQ00 is a dynamically defined CLUSSDRA type channel from MQ0A to MQ00. This information indicates that queue manager MQ0A learned how to find MQ00 from the information in the repository because the CLUSSDR for MQ0A points to MQ0B.
- Since MQ0A has no CLUSSDR defined to MQ00, the DEFTYPE for the dynamic channel that MQ0A creates to MQ00 is a CLUSSDRA type channel.
- The information that MQ0A has about MQ00 also indicates that MQ00 does not hold a full repository, and the QMTYPE shown for MQ00 is NORMAL, not REPOS.

You are finished looking at the output of DIS CLUSQMGR issued from MQ0A. The display from MQ0B is skipped for brevity, as it is similar to MQ0A. It is the view from a full repository queue manager that is pointing to another full repository.

But where is the information for the MQ03 queue manager?

MQ00 definition check 1 of 2: What happens in the cluster

```
/MQ00 DIS CLUSQMGR(*) ALL (partial display)
```

CSQM201I MQ00 CSQMDRTC DIS

CLUSQMGR (MQ0A)

The partner queue manager for this MQ00 connection is MQ0A

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0A)

QMID (MQ0A.CDA33A44AE870B2B)

A CLUSSDRB is the combination of both MQ0A's CLUSRCVR and MQ00's CLUSSDR; MQ00 has a defined CLUSSDR to MQ0A

DEFTYPE (CLUSSDRB)

...

QMTYPE (REPOS)

The MQ0A queue manager holds a full repository

CLUSDATE (2014-09-08)

STATUS (RUNNING)

VERSION (08000000)

XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE)

TRPTYPE (TCP)

CONNNAME (mvsmc11.ilsvpn.ibm.com(1621))

CSQM201I MQ00 CSQMDRTC DIS

CLUSQMGR (MQ0B)

The partner queue manager for this MQ00 connection is MQ0B

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ0B)

QMID (MQ0B.CDA2735757A9FEA9)

CLUSSDRAs show a dynamic connection from MQ00 to MQ0B's CLUSRCVR; MQ00 has no defined CLUSSDR to MQ0B

DEFTYPE (CLUSSDRA)

...

QMTYPE (REPOS)

The MQ0B queue manager holds the second full repository

STATUS (RUNNING) ...

© Copyright IBM Corporation 2015

Figure 6-9. MQ00 definition check 1 of 2: What happens in the cluster

WM3021.1

Notes:

You now issue `DIS CLUSQMGR(*) ALL` in the MQ00 queue manager. This queue manager has a partial repository, and its CLUSSDR channel points to MQ0A. Currently, you know that:

- The WMADMCLS.MQ0A channel that you see here should be DEFTYPE CLUSSDRB. The CLUSSDR channel for MQ00 selected MQ0A as its full cluster repository, so the dynamic channel from MQ00 to MQ0A should be a CLUSSDRB. You can see that this statement checks out.
- Dynamic cluster sender channels to MQ0B and MQ03 should be type CLUSSDRA because MQ00 does not have any explicitly defined CLUSSDR channels to MQ0B or MQ03. This statement checks out for MQ0B, but where is MQ03?
- The CLUSRCVR channel for MQ00 should point to itself. Check the next slide.

MQ00 definition check 2 of 2: What happens in the cluster

CSQM201I MQ00 CSQMDRTC DIS ...

CLUSQMGR (MQ00) ————— This definition is for the MQ00 queue manager itself

CLUSTER (WMADMCLS)

CHANNEL (WMADMCLS.MQ00)

QMID (MQ00.CDA33FC1E2935036)

DEFTYPE (CLUSRCVR) ————— CLUSRCVR channel type definition for MQ00

...

QMTYPE (NORMAL) ————— The MQ00 queue manager holds a partial repository

CLUSDATE (2014-09-08)

CLUSTIME (09.06.23)

SUSPEND (NO)

VERSION (08000000)

TRPTYPE (TCP)

CONNNAME (mvsmc11.ilsvpn.ibm.com(1600))

...

© Copyright IBM Corporation 2015

Figure 6-10. MQ00 definition check 2 of 2: What happens in the cluster

WM3021.1

Notes:

As expected, the CLUSRCVR for MQ00 shows it pointing to itself. But there are no entries for the MQ03 queue manager.

You now go to MQ03 and issue the DIS CLUSQMGR(*) ALL command.

MQ03 definition check: What happens in the cluster

```
/MQ00 DIS CLUSQMGR(*) ALL (partial display)
CSQM201I MQ03 CSQMDRTC  DIS CLUSQMGR DETAILS 636
CLUSQMGR (MQ03) _____ This definition is for the MQ03 queue manager
CLUSTER (WMBADMCLS)
CHANNEL (WMADMCLS.MQ03)
QMID (MQ03.CDB68FBADA5634A2)
DEFTYPE (CLUSRCVR) _____ CLUSRCVR channel type definition for MQ03
QMTYPE (NORMAL)
...
CONNNAME (mvsmc11.ilsvpn.ibm.com(1603))
...
CSQM201I MQ03 CSQMDRTC  DIS CLUSQMGR DETAILS 637
CLUSQMGR (SYSTEM.TEMPQMGR.mvsmc11.ilsvpn.ibm.com(1622))
CLUSTER (WMADMCLS)
CHANNEL (WMADMCLS.MQ0B)
QMID (SYSTEM.TEMPUUID.mvsmc11.ilsvpn.ibm.com(1622))
DEFTYPE (CLUSSDR) _____ Channel DEFTYPE should be a CLUSSDRB to MQ0B
QMTYPE (REPOS)
...
STATUS (INACTIVE)
XMITQ (SYSTEM.CLUSTER.TRANSMIT.QUEUE)
TRPTYPE (TCP)
CONNNAME (mvsmc11.ilsvpn.ibm.com(1622))
...
```

 Can you spot the problem?

Always check
new cluster queue
managers before
proceeding

© Copyright IBM Corporation 2015

Figure 6-11. MQ03 definition check: What happens in the cluster

WM3021.1

Notes:

You issue the DIS CLUSQMGR for MQ03. Note what happens:

- MQ03 is directing its CLUSSDR channel to the MQ0B full repository, but the display for the CLUSSDR* QMTYPE has three obvious problems:
 - CLUSQMGR indicates a SYSTEM.TEMPQMGR.xxx name, a red flag.
 - QMID also indicates SYSTEM.TEMPUUID.xxx name, another red flag.
 - The channel DEFTYPE shows CLUSSDR, another red flag. It should be a CLUSSDRB because there is an explicitly defined CLUSSDR to the full repository, MQ0B.

You check all definitions, but the connection information for the CLUSSDR to MQ0B looks correct, and so does the connection information in the CLUSRCVR channel.

After checking the channel connection attributes, the error was not obvious, so it was decided to refresh the cluster. You proceed to issue this command.

MQ03 definition check: Log messages

```
From syslog: /MQ03 refresh cluster (WM*)
CSQ9022I MQ03 CSQMRCLU 'REFRESH CLUSTER' NORMAL COMPLETION
+CSQX875I MQ03 CSQXREPO REFRESH CLUSTER processing started for cluster *
+CSQX420I MQ03 CSQXREPO No repositories for cluster WMBADMCLS
+CSQX419I MQ03 CSQXREPO No cluster-receivers for cluster WMADMCLS
+CSQX442I MQ03 CSQXREPO Phase one of REFRESH CLUSTER has completed, 371
```

CSQM201I MQ03 CSQMDRTC DIS CL ...
 CLUSQMGR (MQ03)
CLUSTER (WMBADMCLS)
 CHANNEL (WMADMCLS.MQ03)
 DEFTYPE (CLUSRCVR)

CSQM201I MQ03 CSQMDRTC DIS CL ...
 CLUSQMGR (MQ03)
CLUSTER (WMADMCLS)
 CHANNEL (WMADMCLS.MQ03)
 DEFTYPE (CLUSRCVR)

/MQ00 DIS CLUSQMGR(*) ALL partial
 CSQM201I MQ03 CSQMDRTC DIS CL ...
 CLUSQMGR (MQ0A)
 CLUSTER (WMADMCLS)
 CHANNEL (WMADMCLS.MQ0A)
 DEFTYPE (CLUSSDRA)
 CSQM201I MQ03 CSQMDRTC DIS CL ...
 CLUSQMGR (MQ0B)
 CLUSTER (WMADMCLS) ...continued ->

CHANNEL (WMADMCLS.MQ0B)
 DEFTYPE (CLUSSDRB)
 CSQM201I MQ03 CSQMDRTC DIS CL ...
 CLUSQMGR (MQ03)
 CLUSTER (WMADMCLS)
 CHANNEL (WMADMCLS.MQ03)
 DEFTYPE (CLUSRCVR)

© Copyright IBM Corporation 2015

Figure 6-12. MQ03 definition check: Log messages

WM3021.1

Notes:

After completing due diligence and carefully checking the channel definitions, only now you try refresh the partial repository for the MQ03 queue manager by issuing the command `/MQ03 REFRESH CLUSTER(WM*)` without any additional parameters.

Take a moment to look at the output of the command. There are two clues:

- First, no repositories were found for cluster WMBADMCLS, but this name is not the intended name of the cluster used.
- Second, IBM MQ is reporting that there are no cluster receivers for the WMADMCLS cluster, which is the correct cluster name.

These messages bring you to look at the MQ03 CLUSRCVR information, and you now find a typographical error in the CLUSTER name attribute for the MQ03 CLUSRCVR definition. You now alter the CLUSRCVR channel to correct the typographical error on the CLUSTER attribute, and reissue the DIS CLUSQMGR. What are the expected results?

- Dynamic CLUSSDRB channel WMADMCLS.MQ0B was created to MQ0B.
- Dynamic CLUSSDRA channels WMADMCLS.MQ0A and WMADMCLS.MQ00 channels are created to MQ0A and MQ00.

- **Note:** Not all channels were able to be shown on the reduced display.

Cluster queues

At MQ0A:

```
DEFINE QLOCAL(CASHIN) CLUSTER(WMADMCLS) DEFBIND(NOTFIXED)
```

At MQ0B:

```
DEFINE QLOCAL(CASHIN) CLUSTER(WMADMCLS) DEFBIND(NOTFIXED)
```



MQCONNECT
to MQ00
MQOPEN
to CASHIN
MQPUT

```
/MQ0A DIS Q(CASHIN) CURDEPTH
CSQM201I MQ0A CSQMD
QUEUE(CASHIN)
CURDEPTH(15)
```

```
/MQ0B DIS Q(CASHIN) CURDEPTH
CSQM201I MQ0B CSQMD
QUEUE(CASHIN)
CURDEPTH(9)
```

```
/MQ00 DIS QCLUSTER(CA*) ALL (partial)
QUEUE(CASHIN)
CLUSTER(WMADMCLS)
DEFBIND(NOTFIXED)
CLWLRANK(0)
CLWLPRTY(0)
CLUSQMGR(MQ0A)
QMID(MQ0A.CDA33A44AE870B2B)
```

```
-----
```

```
QUEUE(CASHIN)
CLUSTER(WMADMCLS)
DEFBIND(NOTFIXED)
CLWLRANK(0)
CLWLPRTY(0)
CLUSQMGR(MQ0B)
QMID(MQ0B.CDA2735757A9FEA9)
```

© Copyright IBM Corporation 2015

Figure 6-13. Cluster queues

WM3021.1

Notes:

As part of the cluster setup, you defined some cluster queues. If the applications that consume the messages are available in each queue manager, the same cluster queues can be defined in several cluster member queue managers.

The default QLOCAL DEFBIND attribute is OPEN, which means that when an application starts putting messages to a queue, they all go to the queue where the first MQOPEN bound these messages.

If there are no affinities, the DEFBIND can be, and should be, changed to NOTFIXED to allow workload balancing.

The command `DIS QCLUSTER(CAS*)` shows all instances of queues with names that start with CAS in the cluster. However, the individual cluster queue managers might not learn about the newly defined queue until an MQPUT to the queue is attempted.

Notice that the `DIS QCLUSTER` command also indicates the queue manager that each instance of the clustered queue belongs to in the CLUSQMGR attribute.

Cluster definition recap

1. On MQ0A
 - Define CLUSRCVR pointing to itself
 - Define CLUSSDR pointing to other full repository at MQ0B
 - Alter queue manager to make it a full repository: REPOS(WMADMCLS)
2. On MQ0B
 - Define CLUSRCVR pointing to itself
 - Define CLUSSDR pointing to other full repository at MQ0A
 - Alter queue manager to make it a full repository
3. Issue DIS CLUSQMGR on both full repository queue managers
4. For each partial repository queue manager added
 - Define CLUSRCVR pointing to queue manager itself
 - Define CLUSSDR pointing to one of the two full cluster repositories
 - Issue DIS CLUSQMGR to confirm successful add
5. Create cluster queues
6. Test applications

© Copyright IBM Corporation 2015

Figure 6-14. Cluster definition review

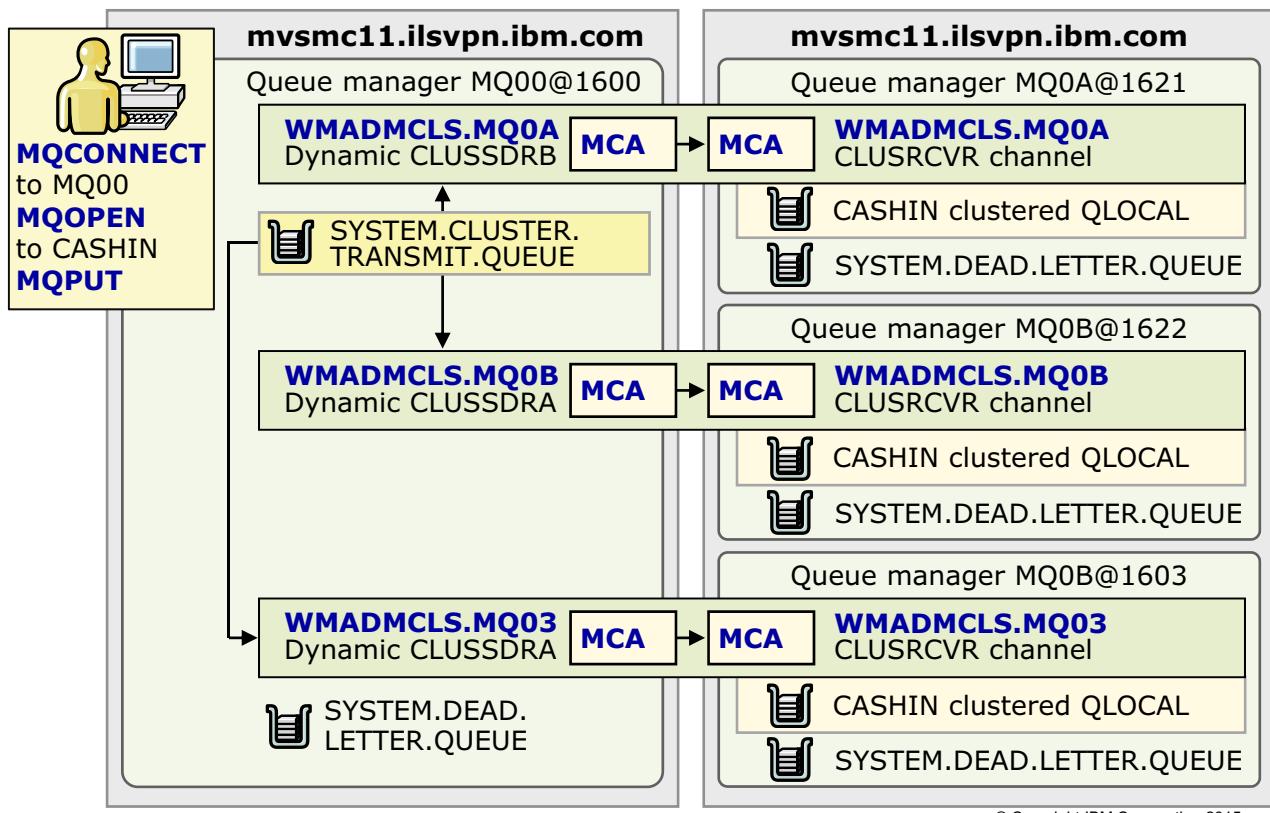
WM3021.1

Notes:

This slide reviews the steps that are taken to create the cluster.

Steps 1 – 3 are complete. In the lab exercise, you add your queue manager to the cluster as noted in step 4.

Cumulative checkpoint



© Copyright IBM Corporation 2015

Figure 6-15. Cumulative checkpoint

WM3021.1

Notes:

How does clustering compare to the distributed channel checkpoint? What path do cluster messages take?

Other than workload balancing, following the path of a message in a cluster queue is similar to working with distributed queuing except that there are some additional cluster queues where the message might go. In this walk-through, you must make the following assumptions:

- That the queue manager is not set to allow the messages to be placed outside the local version of the CASHIN queue if one is available. This setting is explained in a later slide.
- The MQ00 queue manager does not host the CASHIN queue.
- The queue managers all use the default transmit queue.

An application issues MQOPEN of clustered queue CASHIN for MQ00. The MQ00 queue manager does not host the CASHIN queue. Cluster workload balancing determines the best candidate queue manager that is hosting the CASHIN cluster queue to be the target for the message.

- Assume that the message goes to CASHIN at MQ0A. The message is first placed on SYSTEM.CLUSTER.TRANSMIT.QUEUE at MQ00. MQ00 has its CLUSSDR channel pointed to MQ0A, so a dynamic channel of type CLUSSDRB called WMADMCLS.MQ0A picks up the

message and if no problems are encountered, places this message in the CASHIN queue at MQ0A.

- If there is a problem with the channel, the message would wait in the SYSTEM.CLUSTER.TRANSMIT.QUEUE at MQ00.
- If there is a different problem, the message might be forwarded to the dead-letter queue, and identified for either MQ00 or MQ0A, depending on the problem.

This trajectory is similar to tracing the route of a message in a distributed channel. If the workload algorithm selects MQ0B or MQ03 as the target for the message, then a CLUSSDRA instead of a CLUSSDRB dynamic channel carries the message to its target.

If you are using a dedicated transmit queue other than SYSTEM.CLUSTER.TRANSMIT.QUEUE, then instead of looking in SYSTEM.CLUSTER.TRANSMIT.QUEUE you would check in the dedicated named transmit queue. Using different transmit queues is discussed in a later slide.

If it is critical to find which of the clustered queues the message was sent to, the status for the potential candidate cluster queues can be displayed, for instance. In the display, it is assumed that queue monitoring is active for the queue manager and also for the queue.

```
MQ0A DIS QSTATUS(CASHIN) ALL
QSTATUS(CASHIN)
TYPE(QUEUE)
OPPROCS(0)
IPPROCS(0)
CURDEPTH(17)
UNCOM(NO)
MONQ(LOW) <==== queue monitoring enabled, but need to check queue manager
QTIME(0,0)
MSGAGE(3138292)
LPUTDATE(2014-10-15) <==== date message was last put
LPUTTIME(19.52.35) <==== date message was last put
LGETDATE()
```

Since you did get the last put date and time populated in the DIS QSTATUS display, it is obvious that the queue manager was enabled for monitoring information. However, the command to check whether the queue manager is enabled is shown here in case it is needed.

```
MQ0A DIS QMGR MONQ
QMNAME(MQ0A)
MONQ(LOW) <==== queue manager is enabled
```

IBM MQ cluster commands

Commands	
DIS CLUSQMGR	Shows cluster information and detail on connections
DIS QCLUSTER	Shows cluster queues and pertinent details
REFRESH CLUSTER	 <ul style="list-style-type: none"> Updates the cluster repository Impacts performance Refrain from unnecessary use Read all usage notes
SUSPEND QMGR	Notifies cluster member queue managers to stop sending messages to the queue manager that is specified in the command
RESUME QMGR	Informs cluster member queue managers that the queue manager specified in the command is back and available to receive messages
RESET CLUSTER	Forces the removal of a queue manager from a cluster

© Copyright IBM Corporation 2015

Figure 6-16. IBM MQ cluster commands

WM3021.1

Notes:

Several commands are used to manage the cluster. Under most circumstances, **DIS CLUSQMGR** and **DIS QCLUSTER** are the most commonly used commands.

REFRESH CLUSTER can lead to performance problems, and should not be used unless the administrator is experienced with clusters and reads all the warnings in the documentation. Some forms of this command should never be used from a full repository. Allowing the cluster a few moments to update itself usually is a better option than trying to use **REFRESH CLUSTER**.

The rest of the commands do as noted in the slide. It is imperative to become familiar with the warnings and details that are associated with each command as noted in the IBM MQ Knowledge Center. For instance, some forms of **REFRESH CLUSTER** should never be issued from a full cluster repository.

On the **RESET CLUSTER**, in the unlikely event that two queue managers with the same name were defined in the same cluster, it is important to obtain the queue manager ID, or QMID. This ID qualifies the **RESET CLUSTER** command, ensuring that the correct queue manager is removed.

The case against use of REFRESH CLUSTER: System console after cluster correction

```
+CSQX420I MQ03 CSQXREPO No repositories for cluster WMBADMCLS
... (Problem with MQ03 CLUSRCVR corrected)
+CSQX500I MQ03 CSQXRCTL Channel WMADMCLS.MQ0B started
+CSQX500I MQ0B CSQXRESP Channel WMADMCLS.MQ0B started 893
connection 10.31.187.59
+CSQX500I MQ0B CSQXRCTL Channel WMADMCLS.MQ03 started
+CSQX500I MQ03 CSQXRESP Channel WMADMCLS.MQ03 started 895
connection 10.31.187.59
+CSQX500I MQ03 CSQXRCTL Channel WMADMCLS.MQ0A started
+CSQX500I MQ0A CSQXRESP Channel WMADMCLS.MQ0A started 897
connection 10.31.187.59
+CSQX407I MQ0B CSQXREPO Cluster queue CASHIN definitions inconsistent
+CSQX500I MQ0A CSQXRCTL Channel WMADMCLS.MQ03 started
+CSQX407I MQ03 CSQXREPO Cluster queue CASHIN definitions inconsistent
+CSQX500I MQ03 CSQXRESP Channel WMADMCLS.MQ03 started 901
connection 10.31.187.59
+CSQX500I MQ0B CSQXRCTL Channel WMADMCLS.MQ00 started
+CSQX407I MQ0A CSQXREPO Cluster queue CASHIN definitions inconsistent
+CSQX500I MQ00 CSQXRESP Channel WMADMCLS.MQ00 started 904
```

All channels start correcting the cluster name in the CLUSRCVR without issuing a REFRESH. The CASHIN cluster queues were defined on MQ0A, MQ0B, and MQ03 before corrections were made to the MQ03 CLUSRCVR cluster name, so there were no errors on MQ03 queues.

© Copyright IBM Corporation 2015

Figure 6-17. The case against use of REFRESH CLUSTER: System console after cluster correction

WM3021.1

Notes:

The IBM MQ Knowledge Center has a significant amount of documentation that details why it is best to refrain from using a REFRESH CLUSTER without absolute justification.

Given some time, depending on the size of the cluster, when a definition or correction is made, the cluster mechanism shares this information without any intervention.

The display picks up after the last CSQX420I message from the repository (CSQXREPO) stating “no repository for cluster WMBADMCLS.”

Notice what happened right after the CLUSRCVR was corrected to contain the correct cluster name in the CLUSTER attribute:

- Channels to all the cluster queue manager members – MQ0B, MQ0A, and MQ03 – started.
- But you now have a new error. Why are the definitions for cluster queue CASHIN inconsistent?

DIS QCLUSTER(CASH*) DEFBIND CLUSQMGR

CSQX407I

csect-name Cluster queue q-name definitions inconsistent

- **Explanation:** The definition of a cluster queue has different values for the DEFPRTY, DEFPSIST, DEFPRESP, and DEFBIND attributes on the various queue managers in the cluster

```
CSQM201I MQ00 CSQMDRTC DIS QCLUSTER DETAILS 956
QUEUE (CASHIN)
TYPE (QCLUSTER)
DEFBIND (NOTFIXED)
CLUSQMGR (MQ0A)
END QCLUSTER DETAILS
CSQM201I MQ00 CSQMDRTC DIS QCLUSTER DETAILS 957
QUEUE (CASHIN)
TYPE (QCLUSTER)
DEFBIND (NOTFIXED)
CLUSQMGR (MQ0B)
END QCLUSTER DETAILS
CSQM201I MQ00 CSQMDRTC DIS QCLUSTER DETAILS 958
QUEUE (CASHIN)
TYPE (QCLUSTER)
DEFBIND (OPEN) ←
CLUSQMGR (MQ03)
END QCLUSTER DETAILS
```

© Copyright IBM Corporation 2015

Figure 6-18. DIS QCLUSTER(CASH*) DEFBIND CLUSQMGR

WM3021.1

Notes:

After a review of the error in the IBM MQ Knowledge Center, you find the reason for the error. When defining a clustered queue, certain attributes must match.

Notice CSQX407I is only an informational error, as noted by the “I” at the end of the message number, CSQX407I. However, you know that this consuming application does not have any known message affinities. Since the requirement is to have workload balancing, you change DEFBIND to NOTFIXED for the CASHIN clustered queue at the MQ00 queue manager.



Cluster administrative options

- ISPF panels and MQSC
- IBM MQ Explorer presents organized display
 - Cluster full and partial repositories
 - QCLUSTER tab
 - CLUSQMGR sender and receiver tabs
 - Publish/subscribe topics

The screenshot shows the IBM WebSphere MQ Explorer interface. The top menu bar includes File, Edit, Window, Help, and a toolbar with icons for New, Open, Save, and Print. The left sidebar is titled 'MQ Explorer - Navigator' and contains a tree view of MQ objects:

- IBM WebSphere MQ
 - Queue Managers
 - Queue Manager Clusters
 - WM102
 - WMADMCLS
 - Full Repositories
 - MQ0A
 - MQ0B
 - Partial Repositories
 - MQ00
 - MQ03

The main content area is titled 'Repository data for queue manager MQ00'. It has tabs for Cluster Queues, Cluster Topics, Cluster-sender Channels, and Cluster-receiver Channels. The 'Cluster-sender channels:' tab is selected, showing a diagram where MQ00 connects to MQ03 via a channel labeled '1'. Below the diagram is a table of cluster-sender channels:

Channel name	Cluster queue manager	Queue manager type	Definition type	Xmit protocol	Channel s
WMADMCLS.MQ03	MQ03	Normal	Auto cluster-sender	TCP	Running
WMADMCLS.MQ0A	MQ0A	Repository	Auto explicit cluster-sender	TCP	Running
WMADMCLS.MQ0B	MQ0B	Repository	Auto cluster-sender	TCP	Running

© Copyright IBM Corporation 2015

Figure 6-19. Cluster administrative options

WM3021.1

Notes:

IBM MQ clusters can be administered with MQSC commands, ISPF, and IBM MQ Explorer. For clusters, IBM MQ Explorer lays out the cluster objects in an organized way.

For this display, you are looking at queue manager MQ00. The navigator menu shows MQ00 as a partial repository. Selecting the MQ00 queue manager brings several tabs:

- The Cluster Queues tab is equivalent to DIS QCLUSTER.
- Cluster topics are discussed in the publish/subscribe unit.
- Cluster-sender channel contains CLUSSDR information equivalent to: DIS CLUSQMGR (*) ALL
- Cluster-receiver channel shows CLUSRCVR information equivalent to: DIS CLUSQMGR(*) ALL

Right-clicking the queue manager brings up other commands such as REFRESH and RESET (named REMOVE), and SUSPEND.

Workload balancing

- Use of workload algorithm depends on MQOPEN or MQPUT
 - Queue manager name included in MQOD: Messages go to specified queue manager
 - Queue manager name not included in MQOD: Queue manager decides
- When the queue manager decides:
 - Always put to local queue manager if it hosts the queue (unless CLWLUSEQ is set to ANY)
 - If the clustered queue is not local, a round-robin workload management algorithm is used:

MQOPEN	When cluster workload management algorithm used
MQOO_BIND_ON_OPEN	Workload algorithm is invoked when MQOPEN issued
MQOO_BIND_NOT_FIXED	Workload algorithm is invoked for every MQPUT
MQOO_BIND_ON_GROUP	Workload algorithm is invoked at start of a message group
MQOO_BIND_AS_Q_DEF	Default is to use queue DEFBIND attribute

- The algorithm is further influenced from specific object attributes and by the use of custom exits

Queues	Channels	Queue managers
Priority: CLWLPRTY	Priority: CLWLPRTY, NETPRTY	CLWSUSEQ
Rank: CLWLRANK	Rank: CLWLRANK	CLWLMRUC
Use remote: CLWLUSEQ	Channel weight: CLWLWGHT	Availability status
PUT enabled or disabled	Channel status	

© Copyright IBM Corporation 2015

Figure 6-20. Workload balancing

WM3021.1

Notes:

The topic in this slide is intended to raise awareness of the factors that influence workload balancing. Due to the scope of the course, it is not intended to offer comprehensive coverage of the subject.

Unless compelling reasons exist, a simple approach without altering the default queue manager settings is best when starting to implement a cluster workload balancing solution. Altering the workload balancing attributes without adequate analysis or governance can lead to situations that are complex to adjust or rectify.

There might be a valid reason to alter the defaults, such as having a server with more or less capacity than another server. Even in such a case, a baseline of the message distribution of the existing applications should always be scheduled before changing any settings. It is important to confirm, for example:

- Whether any applications have affinities
- Whether any applications are inadvertently using the MQOO_BIND_ON_OPEN option
- Whether queues are inadvertently using DEFBIND=OPEN and applications are using the MQOO_BIND_AS_Q_DEF option, which would bind messages to the same queue

- And many other factors that might skew the distribution

If an attribute is defined on a QALIAS definition, it overrides attributes that are defined in a clustered QLOCAL queue.

Default behaviors for queue attributes and application development are:

- Queue definition: DEFBIND=OPEN
- Application: MQOO_BIND_AS_Q_DEF



Warning

The cluster workload algorithm evaluates many factors. Before use of any of the attributes that are described, it is critical to understand how these attributes are handled in relation to other factors in the algorithm. Some of these factors might be as subtle as determining whether a clustered queue is PUT disabled, or as complex as the combined application and queue definition options. Become familiar with the cluster workload balancing algorithm documentation in the IBM MQ Knowledge Center before introducing these attributes to your infrastructure. It is also critical to baseline your applications.

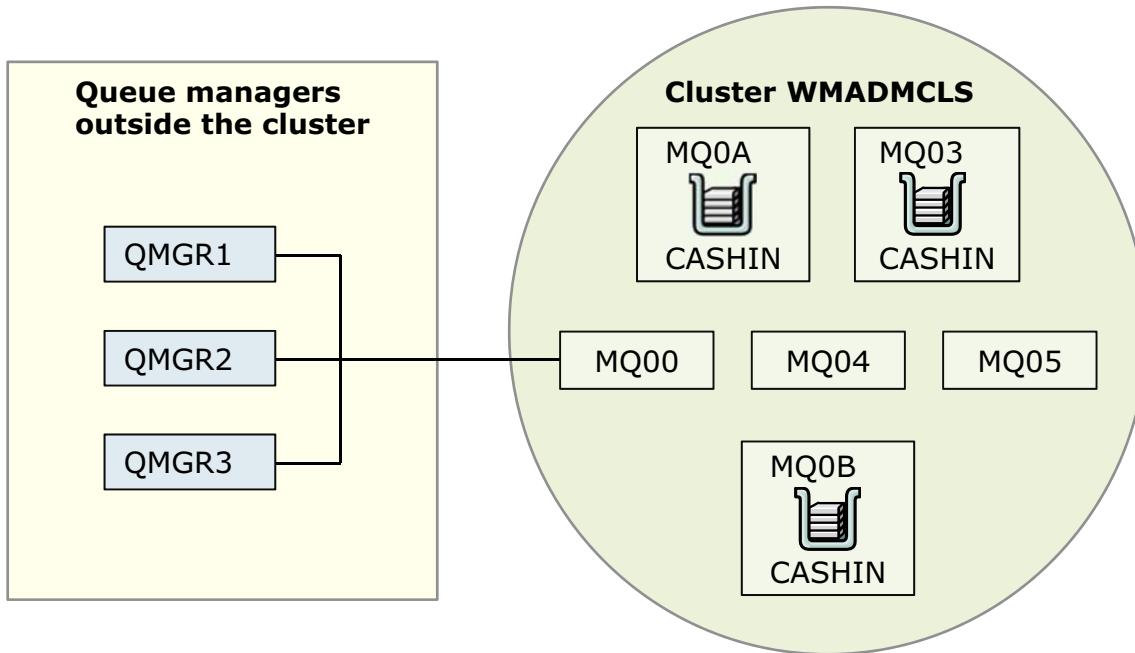
When conducting a baseline, it is important to note that even if a queue definition has the DEFBIND attribute set to NOTFIXED, if the application uses the MQOO_BIND_ON_OPEN option, it overrides the queue definition. That is, the messages will be bound to the same queue manager until the next MQOPEN.

The cluster workload balancing object attributes are as follows:

- Queue attributes:
 - CLWLPRTY: Sets a priority on local, remote, or alias queues. Possible values are 0 – 9, where 9 is the highest priority.
 - CLWLRANK: Sets a rank differentiator on a local, remote, or alias queue for cluster workload distribution. Range is 0 – 9, with 9 being the highest rank.
 - CLWLUSEQ: Determines whether a local instance of a queue is favored as a target against other instances of the same queue in the cluster. Values are: LOCAL goes to local queue. QMGR takes the value of the CLWLUSEQ queue manager attribute. ANY allows the algorithm to treat the local queue as any of the same queue instances available in the cluster.
- Queue manager attributes:
 - CLWLUSEQ: Determines whether a local instance of a clustered queue is given preference over other instances of the queue in a cluster. Possible values are LOCAL or ANY, and work as described in the queue attribute of the same name.
 - CLWLMRUC: Is used to restrict the number of active outbound cluster channels. Default is 999,999,999.
- Channel attributes
 - CLWLPRTY: Sets priority for the channel. Range is 0 – 9, where 9 is the highest.

- CLWLRANK: Sets a range for a rank. Range is 0 – 9, where 9 is the highest.
- CLWLWGHT: Sets a percentage “weight” to influence the CLUSSDR and CLUSRCVR channels for workload distribution. Range is 1 – 99, where 99 is the highest.
- NETPRTY: Sets priority for a CLUSRCVR channel. Range is 0 – 9, where 9 is the highest.

Topology with a cluster gateway



© Copyright IBM Corporation 2015

Figure 6-21. Topology with a cluster gateway

WM3021.1

Notes:

There are numerous ways to design the infrastructure with IBM MQ clusters.

There are such configurations as overlapping clusters, that is, where two clusters are joined.

Overlapping clusters are mentioned for completeness, but they are excluded from the scope of this course.

However, you look at a common configuration where a cluster uses a gateway queue manager to interface with external applications.

WMADMCLS might be the cluster for the ORDERS application. Within the ORDERS application, there might be:

- Redundant servers that host the CASHIN queue for the billing department
- Other servers that deal with fulfillment or shipping, but all interfacing internally

The ORDERS application does not allow users external to the application to join their cluster, so they created MQ00 as a gateway queue manager into the cluster. This configuration is common.

As shown, MQ00 would be a single point of failure. Therefore, it is assumed that MQ00 is set up with a high-availability solution, or it might be part of a queue-sharing group, discussed in a later unit. Next, you look at how to configure this architecture.



Definitions to MQ00 gateway from an external queue manager

At queue manager QMGR1

```
DEFINE QLOCAL(MQ00) +
  USAGE(XMITQ) +
  INITQ(SYSTEM.CHANNEL.INITQ) +
  TRIGGER +
  TRIGDATA(QMGR1.MQ00)

DEFINE QREMOTE(PAYMENTS) +
  RQMNAME(GATEWAY0) +
  RNAME(CASHIN) +
  XMITQ(MQ00)

DEFINE CHANNEL(QMGR1.MQ00) +
  CHLTYPE(SDR) +
  TRPTYPE(TCP) +
  CONNAME(mvsmc...ibm.com(1600)) +
  XMITQ(MQ00)
```

At queue manager MQ00

```
DEFINE QREMOTE(GATEWAY0) +
  RQMNAME(' ') RNAME(' ') +
  XMITQ(' ')

DEFINE CHANNEL(QMGR1.MQ00) +
  CHLTYPE(RCVR) +
  TRPTYPE(TCP)
```

Local queue CASHIN is defined in other queue manager members of the WMADMCLS cluster

© Copyright IBM Corporation 2015

Figure 6-22. Definitions to MQ00 gateway from an external queue manager

WM3021.1

Notes:

In the “Distributed queuing” unit, the use of the QREMOTE object as a queue manager alias or to remap to another queue manager was introduced. For this type of definition, a similar use is made of the QREMOTE object.

The gateway queue manager, in this case MQ00, does not host any clustered queues. However, it is given a queue manager alias definition, GATEWAY0. This definition helps to resolve queue resolution for inbound messages that are not part of the cluster. It helps them to select an instance of the CASHIN clustered queue from one of the clustered queue managers that host the CASHIN queue.

Remote queue manager QMGR1 is not part of the WMADMCLS cluster.

Without the GATEWAY0 object at the MQ00 queue manager, messages that are arriving to MQ00 go to the MQ00 designated dead-letter queue.

On the QMGR1 side, the QREMOTE definition must have the name of the MQ00 queue manager alias, GATEWAY0, in its RQMNAME attribute. If omitted, messages also end up in the dead-letter queue when they arrive at the MQ00 queue manager.

Cluster transmission queue options

1. Use default SYSTEM.CLUSTER.TRANSMIT.QUEUE
2. Automatically associate all cluster sender channels for a queue manager to be associated to a separate transmission queue
 - Set queue manager attribute DEFCLXQ to CHANNEL
 - Queue names are going to be SYSTEM.CLUSTER.TRANSMIT.ChannelName where ChannelName matches the CLUSSDR channel of the queue manager
3. Manually set selected cluster-sender channels to be served by a single transmission queue
 - Create a transmission queue
 - Set the CLCHNAME attribute to the name of the cluster-sender channel
4. Select a group of cluster-sender channels to be served by a single cluster transmission queue
 - Create the transmission queue
 - Set CLCHNAME to a generic name with a wildcard, such as ClusterName.*

© Copyright IBM Corporation 2015

Figure 6-23. Cluster transmission queue options

WM3021.1

Notes:

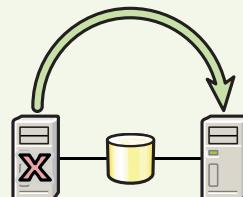
The ability to provide alternative transmit queues other than SYSTEM.CLUSTER.TRANSMIT.QUEUE was a requirement that many customers requested. When this capability was initially introduced, the biggest justification was performance. Now that the ability to change the cluster transmit queue is available, people are finding that providing a separate transmit queue does not have a significant influence on performance.

Having a separate transmit queue is now considered to be more of a way to have application isolation than a performance solution.

The options for cluster transmit queues are listed on this slide. Manually defined queues (option 3) override automatically defined alternative queues (option 1).

Highly available WebSphere MQ topologies

- WebSphere MQ clusters offer scalability by having multiple queue managers to service a request; however:
 - Clustering provides availability for new incoming requests
 - When a queue manager fails, messages in transit for that queue manager are marooned
- If the queue manager is a critical resource such as a gateway, a failover capability compatible with the platform where IBM MQ is running must be implemented
- Explore use of queue-sharing groups for the z/OS platform



© Copyright IBM Corporation 2015

Figure 6-24. Highly available WebSphere MQ topologies

WM3021.1

Notes:

Best practices

- Limit use of the REFRESH CLUSTER command
- Isolate full cluster repositories to dedicated queue managers
- The preferred number of full cluster repositories is two
- Queue managers that hold full cluster repositories should be kept at the highest maintenance level
 - Keep up maintenance
 - Both full cluster repositories should be at the same maintenance level
- Adopt a naming convention for cluster channels that includes the cluster name and the queue manager they point to
- Define only one CLUSSDR channel per queue manager
- Queue manager affinities
 - Set up IBM MQ administration standards about the DEFBIND attribute
 - Communicate with development regarding DEFBIND settings in code
- Baseline existing infrastructure before changing cluster weights and attributes
- Unless there are compelling reasons against it, keep the default cluster transmit queue SYSTEM.CLUSTER.TRANSMIT.QUEUE to service cluster channel traffic
- Unless compelling reasons exist, keep workload balancing simple

© Copyright IBM Corporation 2015

Figure 6-25. Best practices

WM3021.1

Notes:

Clustering is not new, and it provides significant benefits to the organization. However, adequate interface with applications, established administrative procedures, and clear architecture documentation are critical.

After considering the different aspects of IBM MQ clusters, the importance of following these suggested practices should be apparent.

Remember that when a cluster problem surfaces, the administrator is usually looked at to provide resolution. Many times, the problem is a symptom; the real culprit might be with the application code, or with unauthorized changes to the cluster.

Unit summary

- Describe what an IBM MQ cluster is
- List the information to review before implementing IBM MQ clusters
- Describe the components of an IBM MQ cluster
- Define a basic cluster
- Interpret the contents of a cluster display and other cluster commands
- Recognize and resolve cluster problems
- Define and put messages to cluster queues
- Describe the commands and options that are used to administer a cluster
- Summarize how to influence cluster workload balancing
- Describe how to define a connection from outside the cluster to a cluster gateway
- List cluster best practices

© Copyright IBM Corporation 2015

Figure 6-26. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. A cluster is created and a DIS CLUSQMGR(*) issued. If there are no errors in the definitions, which of the following channel DEFTYPEs should appear in the display? Select all that apply for an error-free implementation:
 - a. CLUSSDRA
 - b. CLUSRCVR
 - c. CLUSSDR
 - d. CLUSSDRB

2. True or false: The CLUSRCVR channel does not require a CONNAME attribute.

© Copyright IBM Corporation 2015

Figure 6-27. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.

- 2.

Checkpoint questions (2 of 2)

3. A new cluster that is composed of queue managers MQA1, MQA2, MQA3, and MQA4 is defined. DIS CLUSQMGR output from both MQA1 and MQA2, the two full repositories, shows MQA4, but no signs of MQA3. What should the administrator do next?
 - a. Check the cluster channel definitions in the full repository queue managers
 - b. Issue REFRESH CLUSTER commands from one of the full repositories until MQA3 shows up
 - c. Issue REFRESH CLUSTER from MQA3 until the problem is resolved
 - d. Check the MQA3 cluster channel definitions
4. True or false: Information about newly defined cluster queues is not automatically sent to partial repository queue managers until these queue managers have a “need to know” about the queue.

© Copyright IBM Corporation 2015

Figure 6-28. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

Checkpoint answers (1 of 2)

1. A cluster is created and a DIS CLUSQMGR(*) issued. If there are no errors in the definitions, which of the following channel DEFTYPEs should appear in the display? Select all that apply for an error-free implementation:

- a. CLUSSDRA
- b. CLUSRCVR
- c. CLUSSDR
- d. CLUSSDRB

Answers: a, b, d

2. True or false: The CLUSRCVR channel does not require a CONNAME attribute.

Answer: False. A CONNAME attribute is required.

© Copyright IBM Corporation 2015

Figure 6-29. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. A new cluster that is composed of queue managers MQA1, MQA2, MQA3, and MQA4 is defined. DIS CLUSQMGR output from both MQA1 and MQA2, the two full repositories, shows MQA4, but no signs of MQA3. What should the administrator do next?
 - a. Check the cluster channel definitions in the full repository queue managers
 - b. Issue REFRESH CLUSTER commands from one of the full repositories until MQA3 shows up
 - c. Issue REFRESH CLUSTER from MQA3 until the problem is resolved
 - d. Check the MQA3 cluster channel definitions

Answer: d

4. True or false: Information about newly defined cluster queues is not automatically sent to partial repository queue managers until these queue managers have a “need to know” about the queue.

Answer: True

© Copyright IBM Corporation 2015

Figure 6-30. Checkpoint answers (2 of 2)

WM3021.1

Notes:



Working with IE

© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without

Figure 6-31. Exercise 5

WM3021.1

Notes:

- Add a queue manager to an existing cluster
- Interpret the information in a cluster display
- Put messages to a clustered queue
- Review the IBM MQ Explorer cluster admin

Figure 6-32. Exercise objectives

WM3021.1

Notes:

Unit 7. Publish/subscribe basics

What this unit is about

This unit describes the publish/subscribe messaging style, analyzes the evolution of publish/subscribe, and details ways to configure and use the technology.

What you should be able to do

After completing this unit, you should be able to:

- Differentiate between publish/subscribe and point-to-point messaging
- Summarize the history of publish/subscribe and how it influences current functionality
- Identify the basic components of publish/subscribe
- Describe key properties of topics, subscriptions, and publications
- Summarize managed and unmanaged subscriptions
- Describe ways to administer publish/subscribe
- Summarize publish/subscribe life cycles
- Describe distributed publish/subscribe topologies and when to use them
- Identify factors to consider when implementing or maintaining publish/subscribe

Unit objectives

- Differentiate between publish/subscribe and point-to-point messaging
- Summarize the history of publish/subscribe and how it influences current functionality
- Identify the basic components of publish/subscribe
- Describe key properties of topics, subscriptions, and publications
- Summarize managed and unmanaged subscriptions
- Describe ways to administer publish/subscribe
- Summarize publish/subscribe life cycles
- Describe distributed publish/subscribe topologies and when to use them
- Identify factors to consider when implementing or maintaining publish/subscribe

© Copyright IBM Corporation 2015

Figure 7-1. Unit objectives

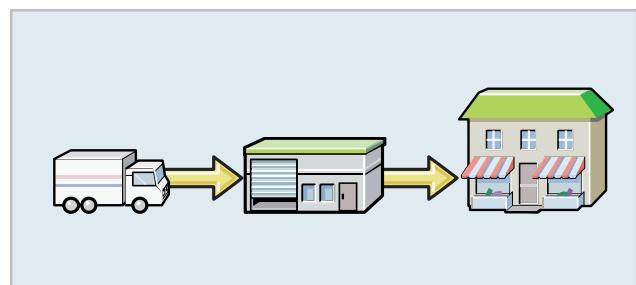
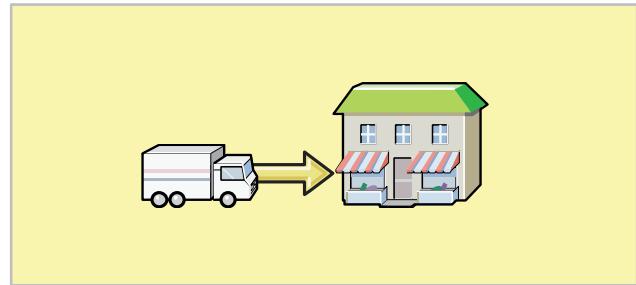
WM3021.1

Notes:

Point-to-point and publish/subscribe

- Point-to-point application
 - Sends messages to a pre-defined destination
 - Application does not need to know where the destination is
 - WebSphere MQ locates the target through object definitions

- Publish/subscribe application
 - Publishes messages to an interim destination according to a topic
 - Interested recipients subscribe to the topic
 - No explicit connection between publishing and subscribing applications



© Copyright IBM Corporation 2015

Figure 7-2. Point-to-point and publish/subscribe

WM3021.1

Notes:

In the distributed queuing exercise, you saw how to connect applications with point-to-point distributed and cluster channels. You also put a message and knew the intended destination of this message. Although that message might be placed in an alias or remote queue with a degree of decoupling, you still knew the trajectory of the message and where it should end up.

With publish/subscribe, the topic tree is used as an intermediary for the sending or publishing of messages and the subscription or getting of messages. It is not apparent where the message is going, or how it shows up; the publish/subscribe process handles those details. Publish/subscribe uses distributed and cluster channels, but the intermediary publish/subscribe process determines the source and destination of the messages. Publish/subscribe adds another level of application decoupling.

Distributed publish/subscribe patterns

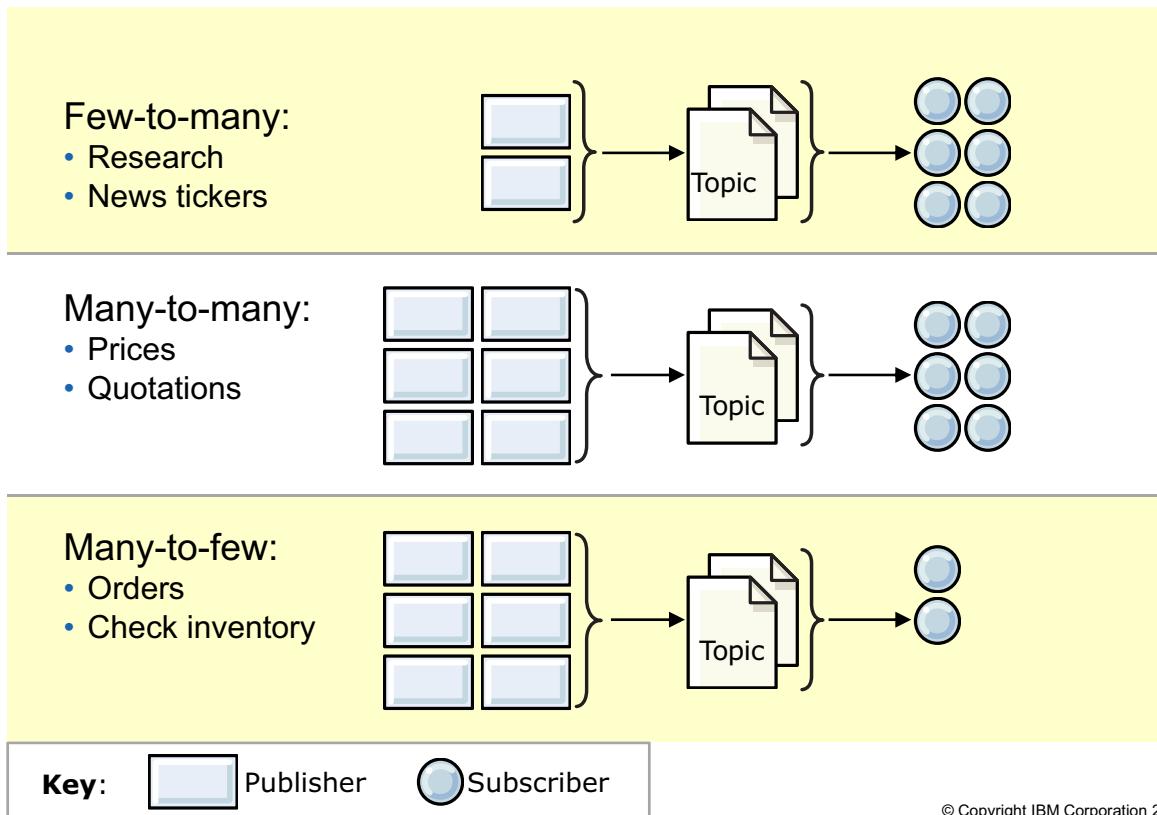


Figure 7-3. Distributed publish/subscribe patterns

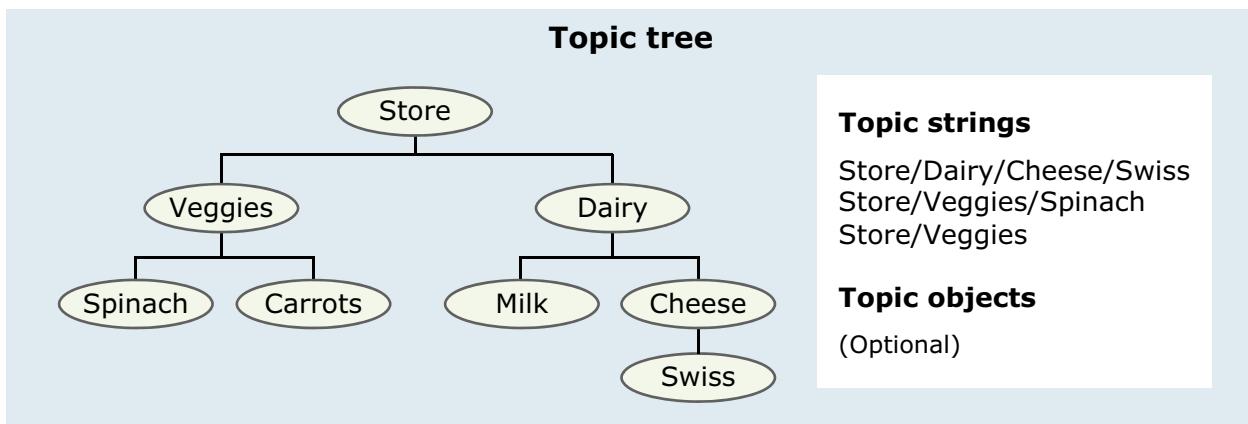
WM3021.1

Notes:

One of the reasons publish/subscribe gained popularity was the flexibility of configurations that are possible to achieve. The publish/subscribe patterns apply to stocks, sports, sales, and other popular applications.

Forms of publish/subscribe are even embedded in software products such as IBM Integration Bus, formerly WebSphere Message Broker, IBM MQ Managed File Transfer, and telemetry mobile applications.

Publish/subscribe basic components



Publishing application

Use:

MQOPEN
topic Store/Veggies/Spinach

MQPUT

- Point-to-point opens a queue
- Publish/subscribe opens a topic

Subscribing application

Use:

MQSUB verb
to Store/Veggies

Figure 7-4. Publish/subscribe basic components

© Copyright IBM Corporation 2015

WM3021.1

Notes:

At the center of publish/subscribe is the topic tree. A topic tree is a hierarchical structure that contains and organizes topics. The way that the topics are created influences a topic tree's organization. The topic tree can be compared to a file system with directories and subdirectories.

Topics can be created by using the create topic commands or by specifying the topic for the first time in a publication or subscription. Topics are structured into topic strings. Topic strings have separate categories that are separated by a slash (/) character.

When messages are sent in a publish/subscribe application, they are published, or put to a topic instead of a queue. A topic can be an explicitly defined object, or a topic that is dynamically created by a publish or a subscribe to the topic string. Published messages are called publications.

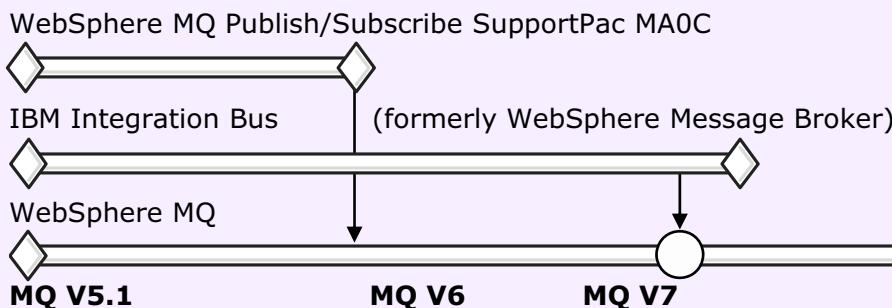
To consume these publications, applications subscribe to the topics or topic strings. The subscriptions can be an explicitly defined object, or a dynamic subscription that an application creates.

So you have the topic tree, topics, publications and publishers, and subscribers and subscriptions.

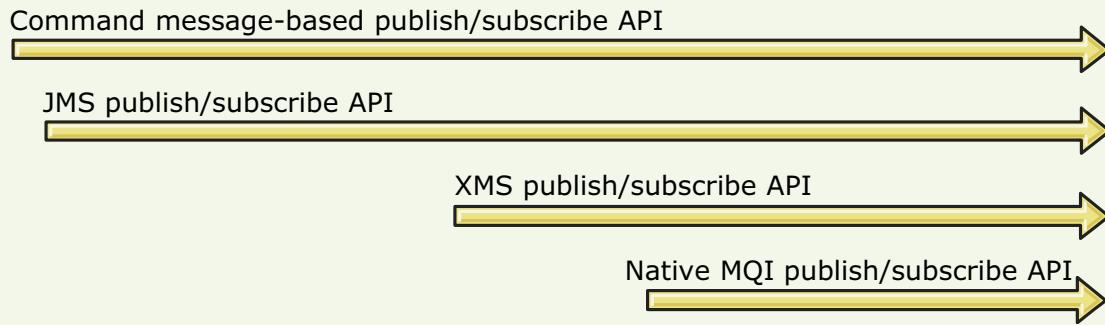
To better understand today's publish/subscribe implementation, you look at publish/subscribe over the years.

Current publish/subscribe functionality: Some history

Publish/subscribe brokers



IBM MQ publish/subscribe APIs



© Copyright IBM Corporation 2015

Figure 7-5. Current publish/subscribe functionality: Some history

WM3021.1

Notes:

The implementation and terminology of publish/subscribe can be confusing. Queue managers have a “mode” or PSMODE parameter. There is mention of “managed queues” and “queued publish/subscribe”. Why such terminology?

Publish/subscribe started out as IBM MQ Support pack MA0C. Publishing and subscribing with the original support pack involved the use of the RFH command message-based API to send subscription and publication requests. WebSphere Message Broker also had a separate publish/subscribe mechanism and used WebSphere Message Broker capabilities to do the publish/subscribe actions. Both the support pack and WebSphere Message Broker referred to the publish/subscribe engine as a “broker”.

As publish/subscribe evolved, it was included in the IBM MQ product. The publish/subscribe API was also changed and augmented. This course does not address migration of the “old” publish/subscribe applications. Nevertheless, it is good to keep in mind that the need to address or handle a migration might surface if you work with an existing application that uses the “old” API and broker function.

A good example of the publish/subscribe heritage is the subscription object PSPROP attribute, which deals with the ways that related message properties are added to messages. You look at PSPROP later in this unit.

Publish/subscribe terminology baseline (1 of 2)

- Terms that are used to refer to the original publish/subscribe support pack engine or to the WebSphere Message Broker engine:
 - Broker
 - Queued publish/subscribe, command-message based publish/subscribe
- The “new” V7+ publish/subscribe is referred to as:
 - Integrated publish/subscribe
 - Publish/subscribe interface
 - Publish/subscribe MQI
- Managed publish/subscribe
 - Publish/subscribe that is used dynamically without creating defined queues or topics
 - “Managed” can also apply to a dynamically created queue or a dynamically created topic

© Copyright IBM Corporation 2015

Figure 7-6. Publish/subscribe terminology baseline (1 of 2)

WM3021.1

Notes:

This slide covers some of the terminology that you might see in the IBM MQ Knowledge Center and also notice when the channel initiator comes up.

Publish/subscribe terminology baseline (2 of 2)

- Administered
 - Creating an explicitly defined object for publish/subscribe such as a topic or a subscription
 - An administered object such as a subscription can have a managed queue
- Streams
 - Concept that is used in queued publish/subscribe to separate the flows for different topics
 - Default stream available to V7+ queue managers if queued publish/subscribe is enabled
 - Named streams require a queue definition that is added to
SYSTEM.QPUBSUB.QUEUE.NAMELIST

© Copyright IBM Corporation 2015

Figure 7-7. Publish/subscribe terminology baseline (2 of 2)

WM3021.1

Notes:

Publish/subscribe functionality

- **PSMODE** queue manager attribute controls publish/subscribe versions
 - **ENABLED**: The default, which indicates that both the **publish/subscribe engine** and the **queued publish/subscribe** are running
 - **COMPAT**: Publish/subscribe engine is active, and queued publish/subscribe is stopped
 - **DISABLED**: Both publish/subscribe engine and queued publish/subscribe are stopped

```
CSQX000I MQ11 CSQXJST IBM WebSphere MQ for z/OS V8.0.0
CSQX001I MQ11 CSQXJST Channel initiator starting
...
+CSQT975I MQ11 CSQXDPSC Distributed Pub/Sub Controller has started
...
+CSQT975I MQ11 CSQXDPSC Distributed Pub/Sub Fan Out Task has started
+CSQT975I MQ11 CSQXDPSC Distributed Pub/Sub Command Task has started
+CSQT975I MQ11 CSQXDPSC Distributed Pub/Sub Publish Task has started
...
+CSQT806I MQ11 CSQXFCTL Queued Pub/Sub Daemon started
+CSQX023I MQ11 CSQXLSTT Listener started, 664
```

© Copyright IBM Corporation 2015

Figure 7-8. Publish/subscribe functionality

WM3021.1

Notes:

PSMODE is a queue manager attribute across the z/OS and distributed platforms that controls the type of publish/subscribe functionality that is enabled in a queue manager.

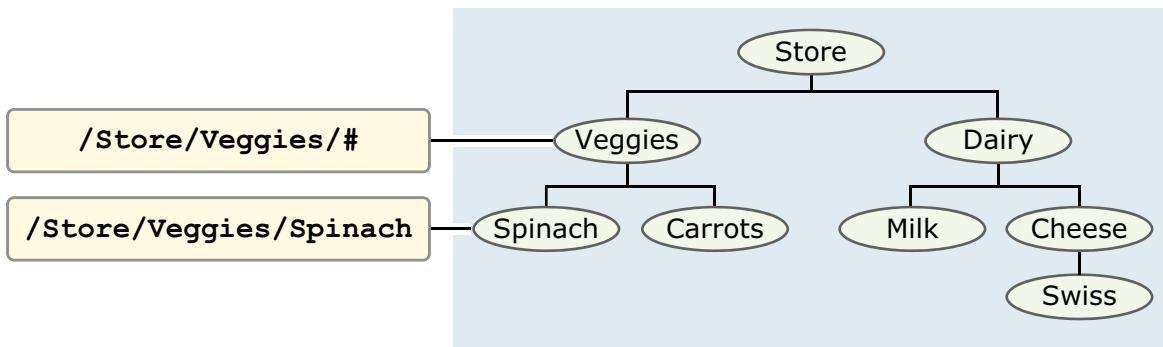
In the snapshot of the channel initiator start console messages, you can see how the publish/subscribe engine starts, then later the “old” publish/subscribe, that is, the “queued” publish/subscribe process, also starts.

The predominant modes are:

- **ENABLED**: Use this mode if you are not sure what the mix of applications is. It supports the old and new publish/subscribe, within any limitations that are documented in IBM MQ Knowledge Center.
- **COMPAT**: The “old” publish/subscribe is not active. Only the new engine is started. Use this mode if you are sure that no old publish/subscribe applications are running.
- **DISABLED**: If publish/subscribe is not being used, startup of the publish/subscribe process can be stopped. **However**, other software components outside the applications, such as WebSphere Message Broker or IBM MQ Managed File Transfer, might require queued

publish/subscribe to be active. Always confirm what runs in the environment before setting PSMODE to DISABLED.

Topic tree, topic strings, topic nodes, and topic objects



- Topic strings are the common sets of slash structured text that applications publish and subscribe to
- Topic nodes are a segment of the topics
- Topic objects are explicitly defined topics, and are also called administered topics
- The topic tree is a hierarchical structure that holds topic strings and nodes
 - When a topic is published to, subscribed to, or defined as an administered topic object the topic tree expands to include the new topic



It is **critical** to implement governance and exercise control over the topic tree.

© Copyright IBM Corporation 2015

Figure 7-9. Topic tree, topic strings, topic nodes, and topic objects

WM3021.1

Notes:

This slide elaborates on topic-related terminology.

Topic trees can take many “shapes.” While some topic trees are hierarchical, others can be “flat”; for instance, they can have more horizontal nodes.

Since topics nodes can be added in so many ways, it is easy for a topic tree to grow larger than expected. When an application publishes or subscribes, it must search the topic tree. Searching a large topic tree, or a topic tree with unused nodes, eventually results in performance impacts to an application.

There is a distinction between a topic node and a topic object, as the topic object is defined as an IBM MQ object with the MQSC command “DEFINE TOPIC.”

The next slide shows the advantages of a defined topic object.

Topic objects

- Provide an administrative control point for the topic tree
 - Configuration attributes
 - Security profiles
 - Topic tree isolation
- Provide flexibility if the topic tree changes later
- Defined topic objects for publish/subscribe are optional

Use:
MQSUB
to Store/Veggies

Use:
MQSUB
to FOODS.ANCHOR + Veggies

```
TOPIC (FOODS.ANCHOR)
TOPICSTR ('Store')
CLUSTER( )
DURSUB (ASPARENT)
DEFPSIST (ASPARENT)
```

...

```
TOPIC (FOODS.ANCHOR)
TOPICSTR ('State/County/City/Store')
CLUSTER( )
DURSUB (ASPARENT)
DEFPSIST (ASPARENT)
```

...

© Copyright IBM Corporation 2015

Figure 7-10. Topic objects

WM3021.1

Notes:

The ability to dynamically add a topic provides significant flexibility to an organization, but topic objects provide extra flexibility.

Imagine that an organization started locally with one store. This store was successful and is now a national entity. What happens to all applications that published or subscribed to the original store?

By having an explicitly defined topic object as an anchor topic, an application can use the anchor topic with other nodes in the topic string, and automatically be part of the larger scope. If the applications used the FOODS.ANCHOR topic to start, there are no concerns if extra nodes must be added ahead of the original hierarchy.

The defined topic also helps as a point to establish “parental” attributes and to apply security profiles.

Topic object attributes: DISPLAY TOPIC view

- **ASPARENT** values
 - Taken from the first parent administrative node that is found in the topic tree
 - If no administrative nodes are found, values are taken from SYSTEM.BASE.TOPIC
- **DEFPSIST** is persistence that is used when the application uses the **MQPER_PERSISTENCE_AS_TOPIC_DEF** option

<pre>CSQM201I MQ11 CSQMDRTC DIS TOPIC(FOODS*) ALL TOPIC(FOODS.ANCHOR) TYPE(LOCAL) QSGDISP(QMGR) CLUSTER() TOPICSTR(Store) DEFPRTY(ASPARENT) DEFPSIST (ASPARENT) DURSUB (ASPARENT) NPMSGDLV(ASPARENT) PMMSGDLV(ASPARENT) MDURMDL()</pre>	<pre>PUB(ASPARENT) SUB(ASPARENT) PUBSCOPE (ALL) SUBSCOPE (ALL) PROXYSUB (FIRSTUSE) WILDCARD(PASSTHRU) CLROUTE (DIRECT) DESCR() DEFPRESP(ASPARENT) USEDLQ (ASPARENT) CUSTOM() ALTDATE(2014-09-15)</pre>
---	---

© Copyright IBM Corporation 2015

Figure 7-11. Topic object attributes: DISPLAY TOPIC view

WM3021.1

Notes:

This slide shows a topic object display and introduces the concept of inheritance.

At the base, or starting topic, of the topic tree is a topic object called SYSTEM.BASE.TOPIC. If topic objects are defined to use default attributes, and there are no other topic objects between the topic string and the SYSTEM.BASE.TOPIC that change the attributes, the object attributes are inherited from the SYSTEM.BASE.TOPIC object.

When a topic is displayed with the DIS TOPIC command, you see the value ASPARENT in many of the attributes. This value indicates that the topic object definition takes its attributes from the next higher topic object definition.

The default values for topic attributes are listed in the IBM MQ Knowledge Center.

Topic object attributes: DISPLAY TPSTATUS view

- **TPSTATUS** shows some actual values in place of **ASPARENT**

```

TPSTATUS( Store/Veggies/Spinach )
TYPE(TOPIC)
DEFPRESP(SYNC)
DEFPSIST(NO)
DEFPRTY(0)
DURSUB(YES)
PUB(ENABLED)
SUB(ENABLED)
ADMIN( )
MDURMDL(SYSTEM.DURABLE.MODEL.QUEUE)
MNDURMDL(SYSTEM.NDURABLE.MODEL.QUEUE)
... ... ...

```

```

... ...
NPMSGDLV(ALLAVAIL)
PMMSGDLV(ALLDUR)
RETAINED(YES)
PUBCOUNT(0)
SUBCOUNT(2)
PUBSCOPE(ALL)
SUBSCOPE(ALL)
CLUSTER( )
USEDLQ(YES)
CLROUTE(NONE)
CSQM443I MQ11

```

© Copyright IBM Corporation 2015

Figure 7-12. Topic object attributes: DISPLAY TPSTATUS view

WM3021.1

Notes:

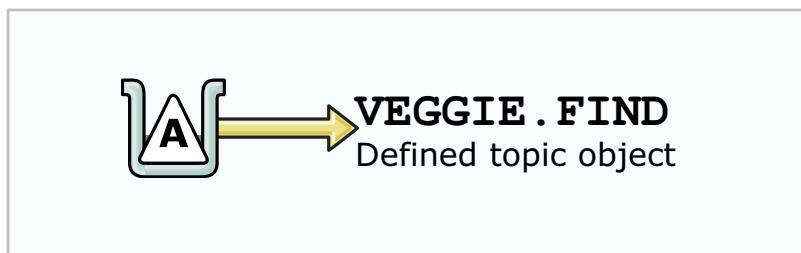
To check the status of a topic, use the **DISPLAY TPSTATUS** command. When displaying the status, the actual inherited attributes, rather than “**ASPARENT**”, are shown on the display.

If you need to match all topic status, you use the number sign (#) instead of the asterisk, such as:

DIS TPSTATUS('#') ALL

Remember that if the topic tree is exceptionally large, the command might cause a great deal of searching.

Topic alias



- Alias queue that points to an explicitly defined topic object
- Topic object must be defined in same queue manager as the alias queue
- Set TARGTYPE attribute to TOPIC in the QALIAS definition

© Copyright IBM Corporation 2015

Figure 7-13. Topic alias

WM3021.1

Notes:

If an application currently puts messages onto a queue, it can be made to publish to a topic by making the queue name an alias for the topic.

Topic administration options

- Administer topics with MQSC commands and IBM MQ Explorer
- **DEFINE, ALTER, DISPLAY, DELETE, and DISPLAY TPSTATUS**
- Wildcard character for topic status is "#": **DIS TPSTATUS (#)**

MQ11 DIS PUBSUB ALL

```
CSQM201I MQ11 CSQMDRTC DIS
TYPE (LOCAL)
QMNAME (MQ11)
STATUS (ACTIVE)
SUBCOUNT (8)
TPCOUNT (11)
```

```
DIS TPSTATUS (#) SUBCOUNT WHERE (SUBCOUNT GT 0)
CSQM443I MQ11
TPSTATUS (SYSTEM.BROKER.ADMIN.STREAM/MQ/MQ11 /StreamSupport )
TYPE (TOPIC) SUBCOUNT (1)
CSQM443I MQ11
TPSTATUS (Store/Veggies)
TYPE (TOPIC) SUBCOUNT (2)
CSQM443I MQ11
TPSTATUS (Store/Veggies/Spinach)
TYPE (TOPIC) SUBCOUNT (2)
```

© Copyright IBM Corporation 2015

Figure 7-14. Topic administration options

WM3021.1

Notes:

To administer topics along with subscriptions, MQSC commands can be used as console commands, or CSQUTIL JCL. The WebSphere MQ Explorer can also be used to administer publish/subscribe objects.

For IBM MQ for z/OS, the ISPF panels do not support publish/subscribe.

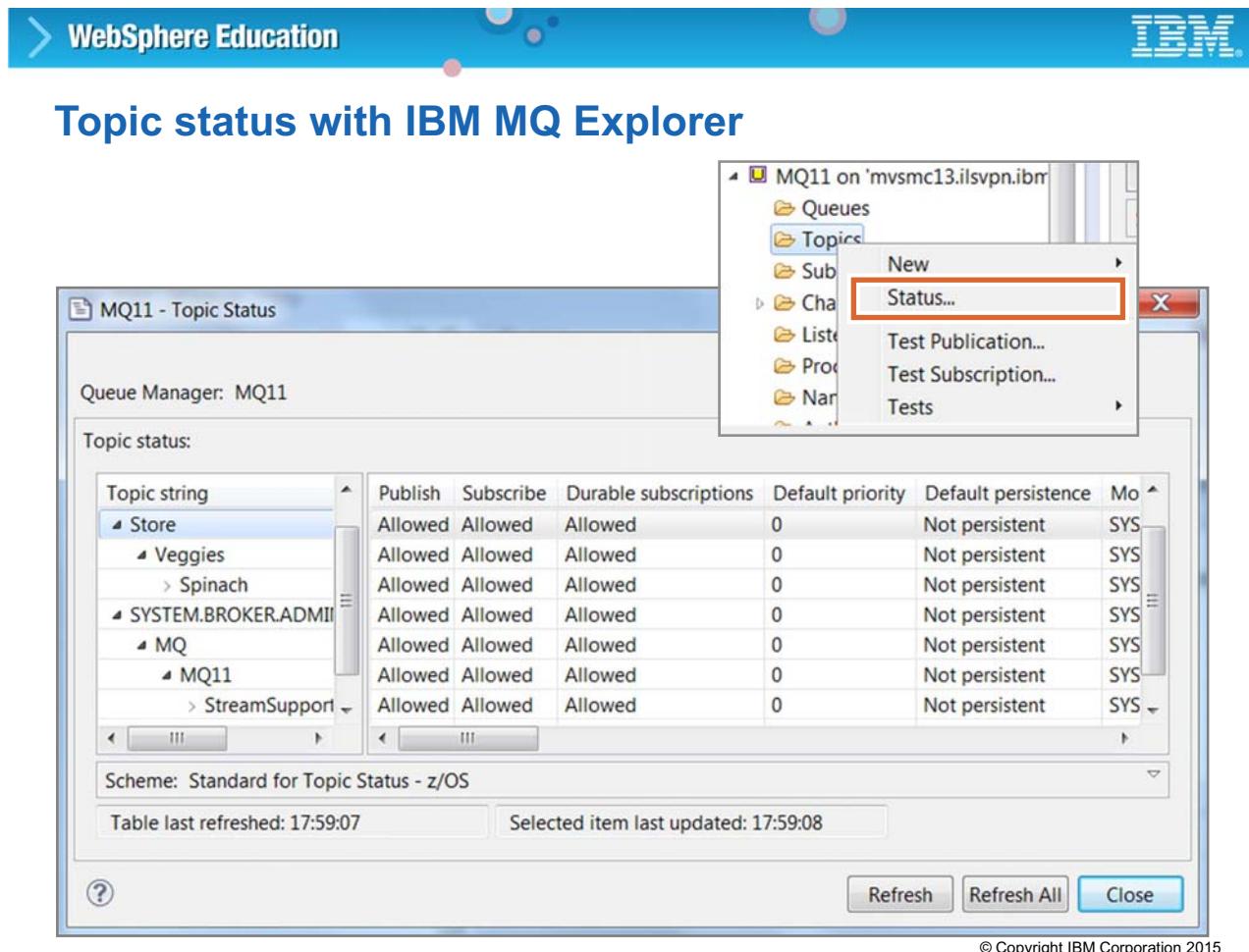


Figure 7-15. Topic status with IBM MQ Explorer

WM3021.1

Notes:

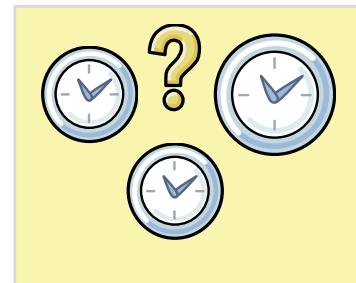
IBM MQ Explorer provides capabilities to create topics, display the status, and also offers the capability to test a publication and subscription.

The topic string hierarchy can be seen in the IBM MQ Explorer display.

When using the test panels, the Test Subscription should be started before the Test Publication.

Topic administration: Topic scavenger

- What topic scavenger is
 - Process to clean up unused topics
 - Runs automatically at intervals set by queue manager parameter **TREELIFE**
- Why topic scavenger needs to run
 - Mitigate time that is used to subscribe to applications
 - Remove unneeded topic strings
- How to regulate topic scavenger execution frequency
 - Defaults to run every 30 minutes
 - Running it at improper intervals can lead to subscription performance delays
 - Running it too frequently can lead to performance issues due to CPU consumption
- Consult support pack MQ1H: WebSphere MQ for z/OS V7.1.0 Performance report



© Copyright IBM Corporation 2015

Figure 7-16. Topic administration: Topic scavenger

WM3021.1

Notes:

The importance of keeping the topic tree without needless nodes was discussed at the start of this unit.

IBM MQ has a process that is called the “topic scavenger” to help rid the topic tree of unused subscriptions, thus keeping the searches more efficient. The default cycle for topic scavenger runs is every 30 minutes, and the TREELIFE queue manager attribute can regulate it. However, depending on your application behavior, it might need to run more or less often, or even not run at all, if publish/subscribe is not used.

Support Pack MQ1H offers guidance on the topic scavenger frequency.

Subscriptions: The three aspects (1 of 2)

- Subscriptions link topics to queues
- Three aspects to subscriptions:
 1. Subscription creation and deletion
 2. Subscription queue management
 3. Subscription lifetime
- Subscription creation and deletion
 - Application-created subscription uses **MQSUB** to create and **MQCLOSE** to delete
 - Administrative or explicitly defined subscription object
- Subscription queue management
 - Managed where the subscription creates and deletes the queue: **DESTCLASS MANAGED**
 - Unmanaged where a queue is explicitly defined or provided: **DESTCLASS PROVIDED**

SUB (VEGETARIAN.SEARCH)

DURABLE (YES)

SUBTYPE (ADMIN)

TOPICSTR (Store/Veggies)

**DEST (SYSTEM.MANAGED.DURABLE.CDC
456AF9B060E32)**

TOPICOBJ (FOODS.ANCHOR)

DESTCLAS (MANAGED)

EXPIRY (UNLIMITED)

SUB (VEGGIE.FIND)

DURABLE (YES)

SUBTYPE (ADMIN)

TOPICSTR (Store/Veggies)

DEST (VEGGIES.QLOCAL)

TOPICOBJ (FOODS.ANCHOR)

DESTCLAS (PROVIDED)

EXPIRY (UNLIMITED)

© Copyright IBM Corporation 2015

Figure 7-17. Subscriptions: The three aspects (1 of 2)

WM3021.1

Notes:

Subscriptions might be one of the most confusing components of publish/subscribe. Some of the terminology that is used for subscriptions might appear to be contradictory, but breaking down the different aspects of subscription should prove helpful. Subscriptions have three distinct aspects:

- Creation and deletion. How was this subscription created? There are two considerations: either an application created it, or it was created administratively with an explicitly defined subscription object.
- Subscription queue management. How is the queue used by the subscription created or handled? If the subscription DESTCLASS attribute is set to MANAGED, the queue is taken care of by IBM MQ. These queues are dynamic queues that are prefixed with "SYSTEM.MANAGED." The next node in the queue name will be DURABLE or NONDURABLE, which brings you to the third subscription aspect.
- Subscription lifetime. There are two considerations. The first consideration is a durable subscription, which means that the life of the subscription does not depend on an application. The second type is nondurable; that is, an application controls the subscription's lifetime.

The three subscription aspects are summarized in the next slide.

Subscriptions: The three aspects (2 of 2)

- Subscription lifetime
 - Durable subscriptions: Lifetime is independent of applications
 - Nondurable subscriptions: Lifetime is driven by applications

Valid subscription combinations

Queue management →	Managed		Unmanaged	
Lifetime →	Durable	Nondurable	Durable	Nondurable
Creation and deletion ↓				
Administrative	Yes	No	Yes	No
Application	Yes	Yes	Yes	Yes

© Copyright IBM Corporation 2015

Figure 7-18. Subscriptions: The three aspects (2 of 2)

WM3021.1

Notes:

This slide summarizes the three aspects of subscriptions: creation and deletion, queue management, and lifetime. You identify these three aspects in the DISPLAY SUB output.

Subscription commands: DIS SUB (VEGET*) ALL

- Lists subscription details that include the topic string to which it subscribes

CSQM201I MQ11 CSQMDRTC DIS SUB
DETAILS 382
SUB (VEGETARIAN.SEARCH)
SUBID (C3E2D8D4D4D8F1F140404040404 ...
DURABLE (YES)
SUBTYPE (ADMIN)
DISTYPE (RESOLVED)
TOPICSTR (Store/Veggies)
DEST (SYSTEM.MANAGED.DURABLE.CDC4 ...)
DESTQMGR (MQ11)
DESTCORL (C3E2D8D4D4D8F1F14040404 ...
TOPICOBJ (FOODS.ANCHOR)
PSPROP (MSGPROP)
PUBACCT (00000000000000000000000000000000 ...
PUBAPPID ()
PUBPRTY (ASPB)
...
...

... continued

USERDATA()

SUBUSER(INGM000)

SUBLEVEL(1)

WSCHEMA(TOPIC)

SUBSCOPE(ALL)

DESTCLAS(MANAGED)

VARUSER(ANY)

SELECTOR()

SELTYPE(NONE)

EXPIRY(UNLIMITED)

REQONLY(NO)

CRDATE(2014-09-16)

CRTIME(16.07.06)

ALTDATE(2014-09-16)

ALTTIME(16.07.06)

END SUB DETAILS

© Copyright IBM Corporation 2015

Figure 7-19. Subscription commands: DIS SUB(VEGET*) ALL

WM3021.1

Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter commands in MQSC format. Equivalent capabilities are provided on IBM MQ Explorer.

The display for this slide shows the three aspects:

- Queue management: DESTCLASS = MANAGED, no explicit queue definition.
 - Lifetime: DURABLE. See the DEST queue name that is prefixed with SYSTEM.MANAGED.
 - Creation and deletion: SUBTYPE = ADMIN, an explicitly defined subscription object. A subscription that an MQSUB created would display SUBTYPE = API. Review the SUBTYPE attribute in the IBM MQ Knowledge Center for other possible MQSUB values.

An important attribute of a subscription object is PSPROP. PSPROP mitigates version-related behaviors that deal with the way that publish/subscribe message properties are added to messages sent to the subscription. You can prevent any properties from being added by setting PSPROP to NONE; however, there are other values applicable to PSPROP, depending on the version of

publish/subscribe and the applications used. Understand your publish/subscribe environment and applications, and review PSPROP attribute values in the knowledge center.

Subscription commands: DISPLAY PUBSUB ALL

- Shows how many topics subscribed to the local queue manager

```
MQ11 DIS PUBSUB ALL
CSQM201I MQ11 CSQMDRTC  DIS PUBSUB DETAILS 413
  TYPE (LOCAL)
  QMNAME (MQ11)
  STATUS (ACTIVE)
  SUBCOUNT (8)
  TPCOUNT (13)
  END PUBSUB DETAILS
CSQ9022I MQ11 CSQMDRTC ' DIS PUBSUB' NORMAL COMPLETION
```

© Copyright IBM Corporation 2015

Figure 7-20. Subscription commands: DISPLAY PUBSUB ALL

WM3021.1

Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

For IBM MQ Explorer, right-click the correct queue manager, and then select **Status > Publish/Subscribe**.



Subscription commands:

DISPLAY SUB (*) TOPICSTR DEST

- Shows all subscriptions with the focus on the topic strings they subscribe to and the associated queue

© Copyright IBM Corporation 2015

Figure 7-21. Subscription commands: DISPLAY SUB(*) TOPICSTR DEST

WM3021.1

Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

On IBM MQ Explorer, select **Subscriptions**, and then scroll to the right on the subscription display to find the destination type and information on the dynamic or local queue.

Subscription-related command: DISPLAY QLOCAL

- Displaying queues that are associated with subscriptions

```
dis q(SYSTEM.MANAGED.DURABLE.CDC456AF9B060E32) curdepth
CSQN205I COUNT=      3, RETURN=00000000, REASON=00000000
QUEUE (SYSTEM.MANAGED.DURABLE.CDC456AF9B060E32)
TYPE (QLOCAL)
QSGDISP (QMGR)
CURDEPTH (0)

-----
CSQ9022I MQ11 CSQMDRTS ' DIS QUEUE' NORMAL COMPLETION
CSQ9022I MQ11 CSQMDRTC ' DIS SUB' NORMAL COMPLETION
MQ11 DIS QLOCAL(VEGGIES) CURDEPTH
CSQM293I MQ11 CSQMDRTC 1 QLOCAL FOUND MATCHING REQUE
CSQM201I MQ11 CSQMDRTC DIS QLOCAL DETAILS 451
QUEUE (VEGGIES)
TYPE (QLOCAL)
QSGDISP (QMGR)
CURDEPTH (1)
```

© Copyright IBM Corporation 2015

Figure 7-22. Subscription related command: DISPLAY QLOCAL

WM3021.1

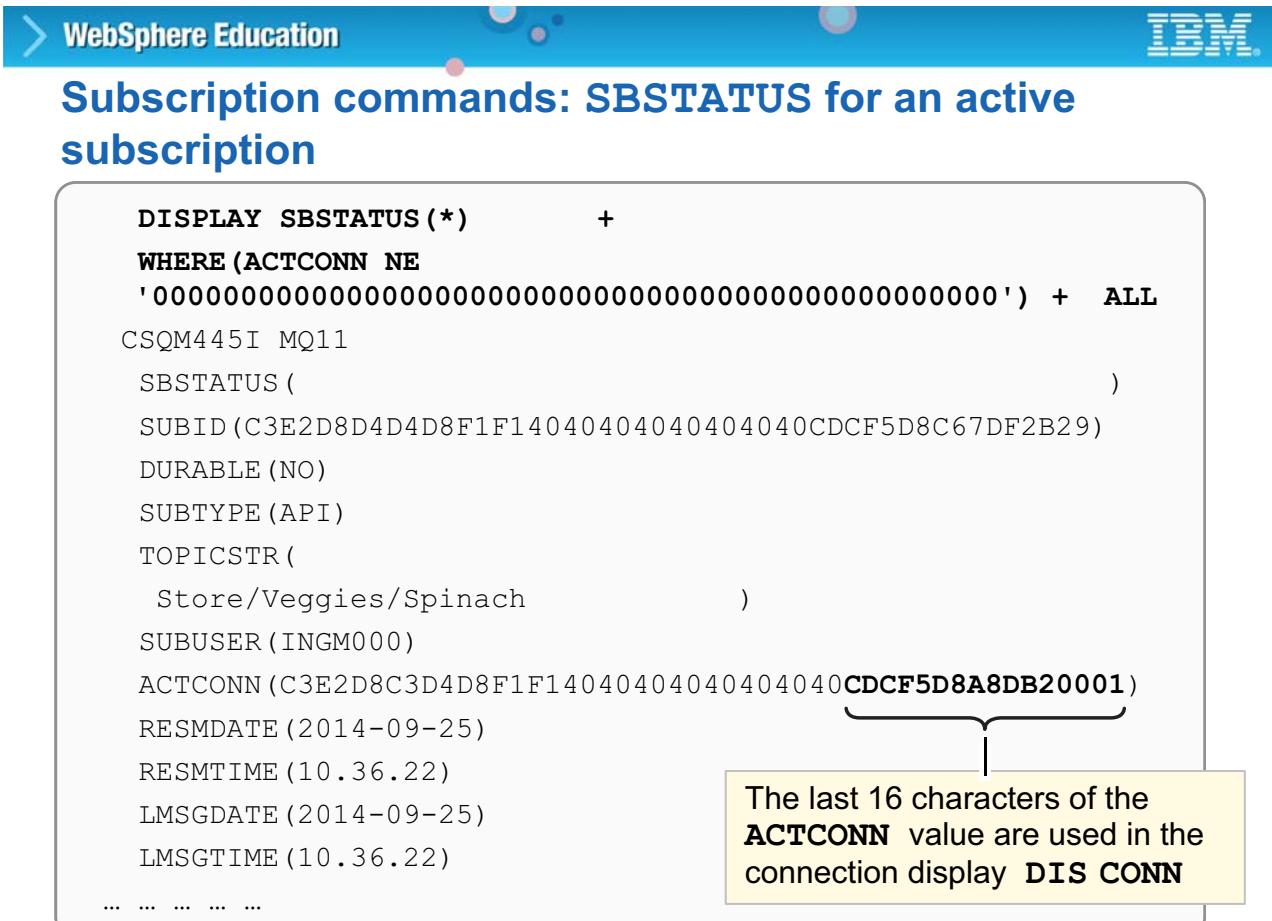
Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Equivalent capabilities are provided on IBM MQ Explorer.

To view the associated queues, select the Queues menu item in IBM MQ Explorer.

When looking for managed or dynamic queues on IBM MQ Explorer, make sure to select the **Show Temporary Queues** toggle switch at the upper right of the queue display panel, and then look for the SYSTEM.MANAGED.* named queues.



© Copyright IBM Corporation 2015

Figure 7-23. Subscription commands: SBSTATUS for an active subscription

WM3021.1

Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Equivalent capabilities are provided on IBM MQ Explorer.

To obtain the connection ID with IBM MQ Explorer, select the **Subscriptions** menu and scroll to the right.

Subscription-related commands: DIS CONN

- Provides additional information about the application

```

dis conn(CDCF5D8A8DB20001) all ... ... ... ...
CONN (CDCF5D8A8DB20001)
EXTCONN (C3E2D8C3D4D8F1F1404040404040404040)
TYPE (CONN)
URDISP (QMGR)
CONNOPTS (
    MQCNO_STANDARD_BINDING )
...
APPLTAG (MQ11CHIN)
ASID(0030)
APPLTYPE (CHINIT)
APPLDESC (WebSphere MQ Channel Initiator)
CHANNEL (MQ11.XP)
CONNNAME (10.4.127.184)
CSQ9022I MQ11 CSQMDRTS ' DIS CONN' NORMAL COMPLETION

```

© Copyright IBM Corporation 2015

Figure 7-24. Subscription related commands: DIS CONN

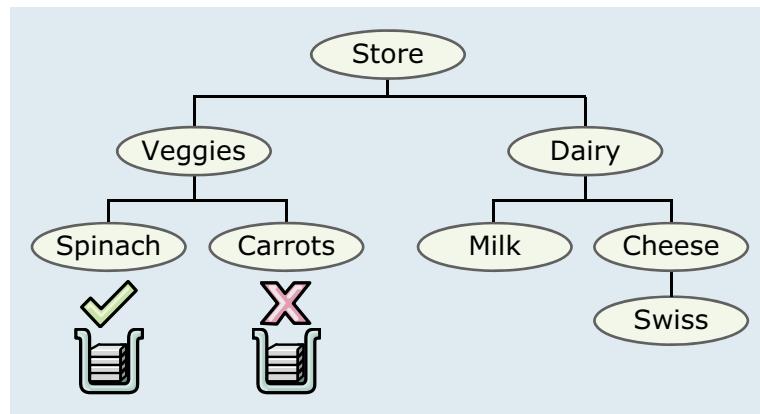
WM3021.1

Notes:

This set of slides shows the MQSC commands available to display subscriptions, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Publications

- What is considered a successful publication?
- IBM MQ default behavior depends on the persistence of the message and the durability of the subscription
- Persistent messages
 - If a publication cannot be delivered to one or more durable subscriptions: publish fails
 - If a publication cannot be delivered to a nondurable subscription: publish still succeeds
- Publications to nonpersistent messages are deemed successful regardless of the actual outcome
- Default behavior is controlled through `PMSGDLV` and `NPMSGDLV` topic attributes
- Publication ending up in the dead-letter queue is considered successful; controlled through the `USEDLQ` topic attribute



© Copyright IBM Corporation 2015

Figure 7-25. Publications

WM3021.1

Notes:

When starting to work with publish/subscribe, it is important to determine what constitutes a successful return code on a publication, as this feedback might be misleading. Always test that a subscription was indeed received.

Topic attributes influence the handling of publications. These attributes might be inherited from the `SYSTEM.BASE.TOPIC`. `PMSGDLV` and `NPMSGDLV` to determine how to treat persistent or nonpersistent messages during a publication.

- `ASARENT`: Behavior is inherited.
- `ALL`: Message must be delivered to all subscribers. If one subscriber fails, no other subscribers receive the message, and the call fails.
- `ALLAVAIL`: Message is delivered to all available subscribers regardless of whether any subscriber fails.
- `ALLDUR`: Delivery failures to nondurable subscribers do not send an error to the `MQPUT` call. Delivery failures to durable subscribers cause no subscribers to receive the message and the `MQPUT` call to fail.

USEDLQ can be set to YES or NO. If NO, or if the queue manager does not specify a DLQ value, the message is treated per PMSGDLV or NPMMSGDLV. If USDLQ is set to YES, then delivery to the dead-letter queue is treated as a successful publication.

Retained publications

- Applications that subscribe to a topic receive messages that are published to the topic at the time the message is published
- Applications that subscribe to the topic after the publication occurs do not get the messages
- Retained publications
 - The queue manager retains the most recent publication for each topic
 - Applications that subscribe after the publication takes place receive the most recent retained message
- The use of retained options is for specific use cases, such as infrequent publications



Using retained publications in a large multi-queue manager environment can lead to unanticipated problems: **Plan carefully** when introducing retained publications

© Copyright IBM Corporation 2015

Figure 7-26. Retained publications

WM3021.1

Notes:

The application uses the `MQPMO_RETAIN` PMO option to specify retained publications.

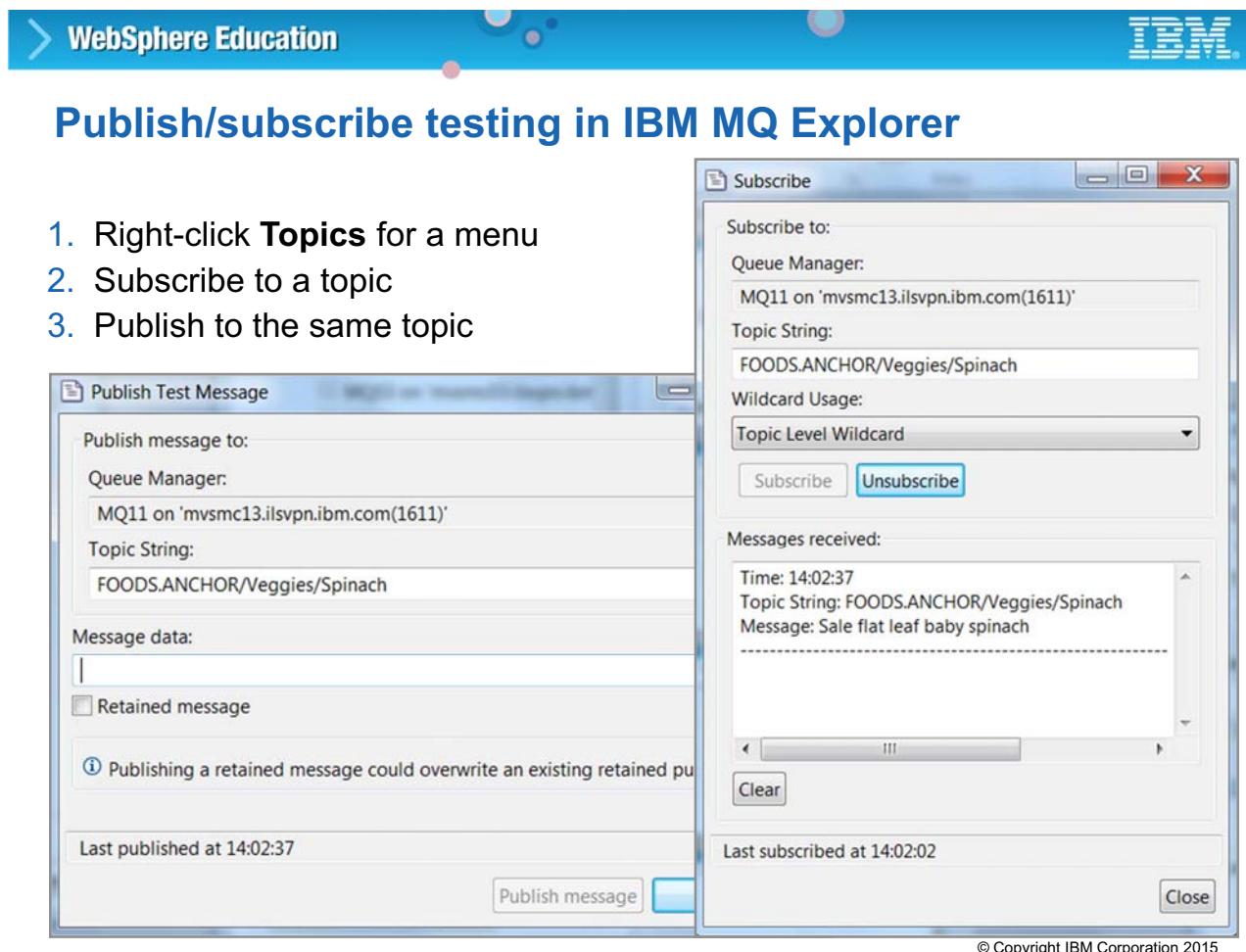


Figure 7-27. Publish/subscribe testing in IBM MQ Explorer

WM3021.1

Notes:

Here is another look at the IBM MQ Explorer publish/subscribe testing capabilities. When using these panels, always subscribe to the topic first.

Publish/subscribe lifecycle descriptions

- Provide a good understanding on how and why to select different attributes for the way publish/subscribe is implemented
- Aid in planning and design stages
- Three available lifecycle scenarios:
 - Managed, nondurable subscriber
 - Managed, durable subscriber
 - Unmanaged, durable subscriber

© Copyright IBM Corporation 2015

Figure 7-28. Publish/subscribe lifecycle descriptions

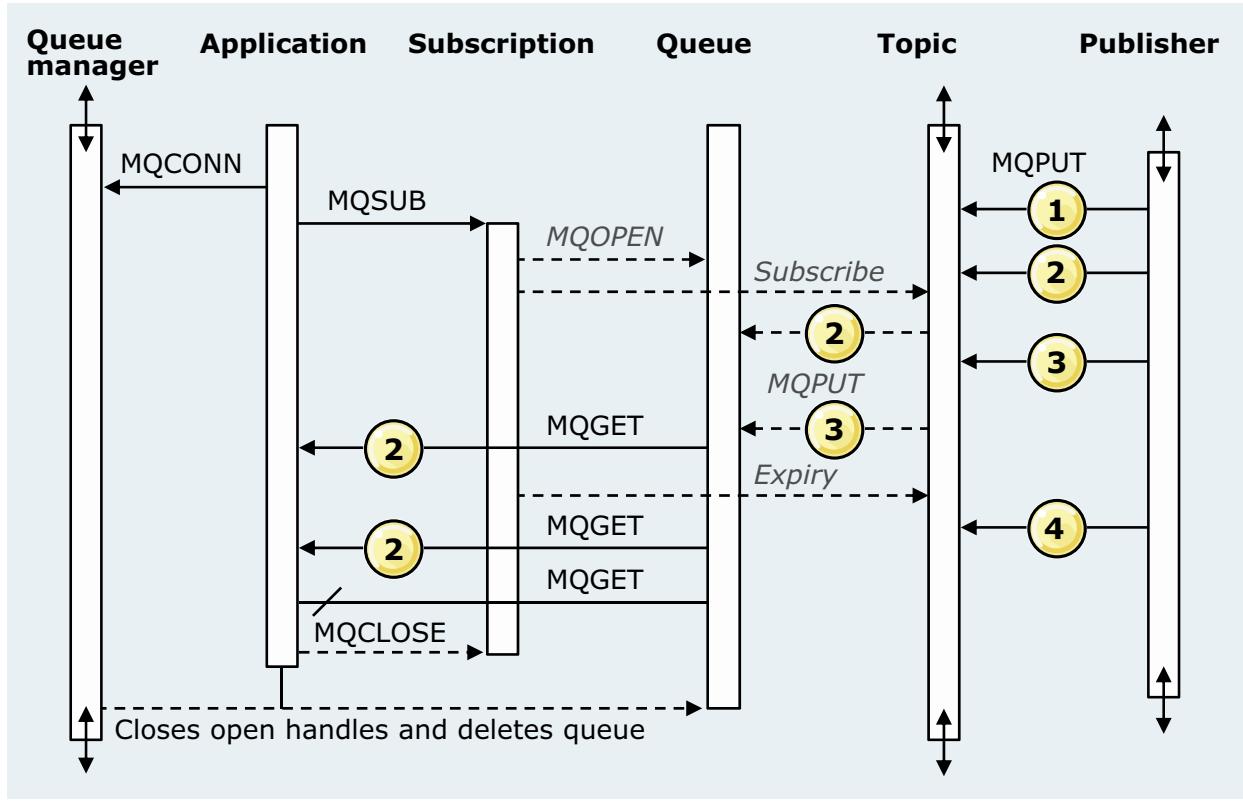
WM3021.1

Notes:

The publish/subscribe topic can fill one entire class to cover. While this class must adhere to the basic scope, the publish/subscribe lifecycle descriptions are introduced as a follow-up if you need to administer or support publish/subscribe applications. The lifecycle topics are helpful when designing and troubleshooting publish/subscribe applications.

There are three available lifecycle descriptions in the IBM MQ Knowledge Center. One of these lifecycle descriptions is shown as an illustration.

Publish/subscribe lifecycles: Managed nondurable subscriber



© Copyright IBM Corporation 2015

Figure 7-29. Publish/subscribe lifecycles: Managed nondurable subscriber

WM3021.1

Notes:

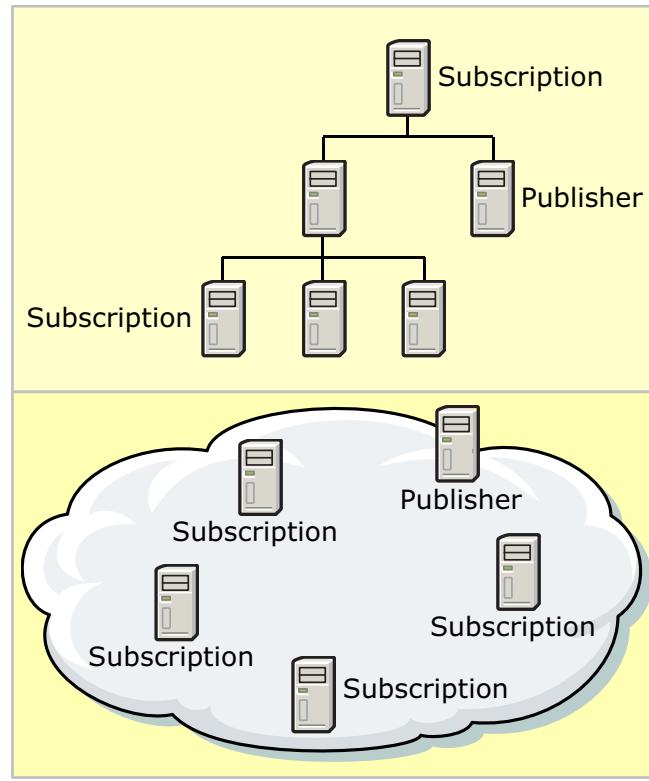
This slide is used to familiarize you with the lifecycle descriptions available in the IBM MQ Knowledge Center. **The text in this slide is copied directly from the IBM MQ Knowledge Center.**

1. The application creates a subscription on a topic was already published to twice. When the subscriber receives its first publication, it receives the second publication, which is the currently retained publication.
2. The queue manager creates a temporary subscription queue along with creating a subscription for the topic.
3. The subscription has an expiry. When the subscription expires no more publications on the topic are sent to this subscription, but the subscriber continues to get messages published before the subscription expired. Subscription expiry does not affect publication expiry.
4. The fourth publication is not placed on the subscription queue; consequently, the last MQGET does not return a publication.
5. Although the subscriber closes its subscription, it does not close its connection to the queue or the queue manager.

6. The queue manager cleans up shortly after the application terminates. Because the subscription is managed and nondurable, the subscription queue is deleted.

Distributed publish/subscribe

- Extends subscriptions and publications from a single queue manager to other queue managers in the IBM MQ infrastructure
- Uses distributed IBM MQ infrastructure
- Hierarchies:
 - Indirect many-to-many connectivity
 - Direct one-to-many connectivity
- Publish/subscribe clusters:
 - Many-to-many connectivity
 - Direct routing or topic host routing many-to-many connectivity
 - Based on IBM MQ clustering



© Copyright IBM Corporation 2015

Figure 7-30. Distributed publish/subscribe

WM3021.1

Notes:

As a review, so far you learned about the basics of publish/subscribe, but all the items so far – topics, subscriptions, and publications – might take place within the same queue manager. In this next set of slides, you learn about extending publish/subscribe to the IBM MQ infrastructure.

A publish/subscribe infrastructure can consist of:

- Publish/subscribe hierarchies
- Publish/subscribe clusters
- A combination of hierarchies and clusters

With IBM MQ V8, publish/subscribe clusters can be direct, or based on the topic host.

Publish/subscribe implicit structure: Proxy subscriptions

- When a queue manager subscribes to a topic string, the interest in this topic is propagated to remote queue managers in the publish/subscribe topology
- The remote queue managers create proxy subscriptions to indicate that the original queue manager is interested in the topic
- When a queue manager receives messages for that topic, a copy of the subscription is sent to proxy subscriptions
- Proxy subscriptions are handled differently according to publish/subscribe topology

```

MQ00 DIS SUB(*) SUBTYPE(PROXY)
CSQM201I MQ00 CSQMDRTC DIS SUB DETAILS 138
SUB(SYSTEM.PROXY.MQ11 Store/Veggies)
SUBID(C3E2D8D4D4D8F0F04040404040404040CDD0B4D2826486AA)
SUBTYPE(PROXY)
DISTYPE(RESOLVED)
END SUB DETAILS
...

```

© Copyright IBM Corporation 2015

Figure 7-31. Publish/subscribe implicit structure: Proxy subscriptions

WM3021.1

Notes:

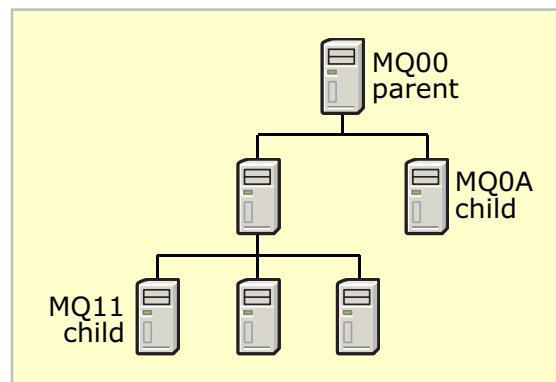
Distributed publish/subscribe introduces a new concept, proxy subscriptions. A queue manager creates proxy subscriptions for each topic string that is subscribed to in that one queue manager. The subscriptions are then propagated to all other queue managers.

The number of proxy subscriptions is a factor that influences how a publish/subscribe infrastructure is defined, by using hierarchies or either type of clustering.

Explanation of proxy subscription attributes is continued on a later slide.

Publish/subscribe hierarchies

- Parent-child relationship is established through the queue manager **PARENT** attribute
- Hierarchy proxy subscriptions are sent from a queue manager to every queue manager connected directly in the hierarchy
- Recipients in turn send proxy subscriptions to other relations
- Publications are sent down the path of proxy subscriptions



MQ11 ALTER QMGR PARENT (MQ00)

```

CSQ9022I MQ11 CSQMAMMS ' ALTER QMGR' NORMAL COMPLETION
+CSQT977I MQ11 CSQXDPSC Establishing Pub/Sub hierarchy relation, 454
(queue manager MQ00)
+CSQX500I MQ11 CSQXRCTL Channel MQ11.MQ00 started
+CSQX500I MQ00 CSQXRESP Channel MQ11.MQ00 started 456
connection 10.31.187.61
  
```

© Copyright IBM Corporation 2015

Figure 7-32. Publish/subscribe hierarchies

WM3021.1

Notes:

A publish/subscribe hierarchy is based on the use of the queue manager PARENT attribute. The display shows what happens in a queue manager, or at least what messages are displayed, after the PARENT attribute is updated to establish a hierarchy.

Hierarchy configuration view

MQ11 DIS PUBSUB ALL

CSQM293I MQ11 CSQMDRTC 2 PUBSUB FOUND MATCHING REQUEST CRITERIA

CSQM201I MQ11 CSQMDRTC DIS PUBSUB DETAILS 167

TYPE (PARENT)**QMNAME (MQ00)**

STATUS (ACTIVE)

SUBCOUNT (NONE)

TPCOUNT (NONE)

END PUBSUB DETAILS

CSQM201I MQ11 CSQMDRTC DIS PUBSUB DETAILS 168

TYPE (LOCAL)**QMNAME (MQ11)**

STATUS (ACTIVE)

SUBCOUNT (8)

TPCOUNT (15)

END PUBSUB DETAILS

© Copyright IBM Corporation 2015

Figure 7-33. Hierarchy configuration view

WM3021.1

Notes:

The hierarchy configuration can be displayed with the **DIS PUBSUB** command or with **IBM MQ Explorer Queue Manager > Status > Publish/subscribe**.

Topic attributes that influence all distributed publish/subscribe configurations

- **PUBSCOPE** topic object attribute
 - QMGR: *Local* publications are delivered to local queue manager only
 - ALL: *Global* publications are delivered to local and remote queue managers that are connected in a cluster or hierarchy
 - ASARENT: Inherit behavior from parent queue manager
 -  Applications might override the PUBSCOPE setting with the MQPMO_SCOPE_QMGR put message option
- **SUBSCOPE** topic object attribute
 - QMGR: Subscription is received from the local queue manager only; no proxy subscriptions are propagated to remote queue managers
 - ALL: Subscriber receives local and remote publications; a proxy subscription is propagated to all queue managers connected in a cluster or hierarchy
 -  If the scope is set to ALL, applications can override scope to use QMGR; if the scope is set to QMGR, applications cannot override scope to use ALL
- **PROXYSUB**
 - Determines whether a ***proxy subscription*** is forced to the rest of the publish/subscribe network
 - Broadcast versus propagation

© Copyright IBM Corporation 2015

Figure 7-34. Topic attributes that influence all distributed publish/subscribe configurations

WM3021.1

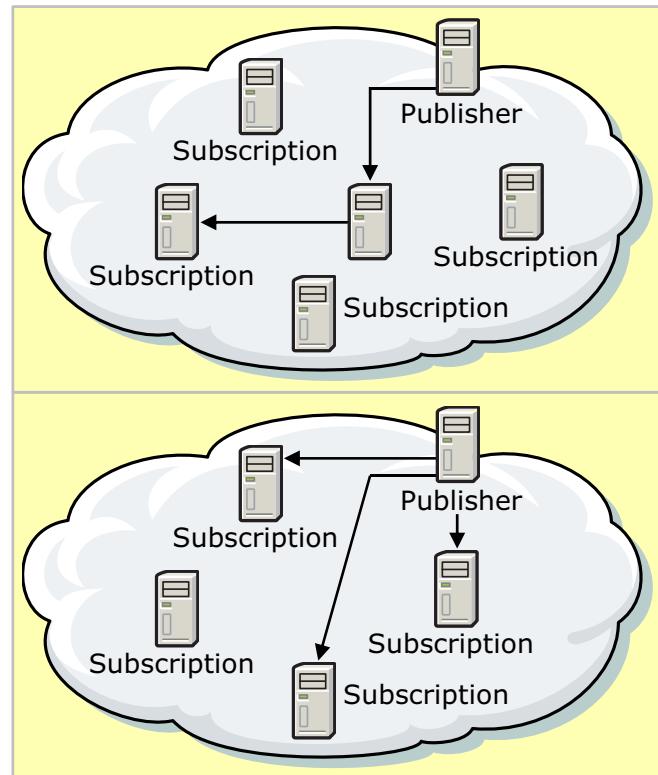
Notes:

PROXYSUB has two settings:

- FIRSTUSE: Proxy subscriptions are generated automatically in the queue manager when a user subscription is created for a topic.
- FORCE: Proxy subscriptions are created regardless whether a local subscription exists.

Publish/subscribe clusters: Direct and topic host routing

- Topic attribute **CLROUTE** determines the cluster routing type
- Direct-routed clusters:
 - All cluster queue managers are aware of other queue managers with matching subscriptions
 - CLROUTE (DIRECT)**
 - Default
- Topic host is routed
 - Queue managers at IBM MQ V8
 - Only those queue managers where the topic is defined are aware of other queue managers
 - Topic host queue managers are used as an interim hop on the way to queue managers with matching subscriptions
 - CLROUTE (TOPICHOST)**



© Copyright IBM Corporation 2015

Figure 7-35. Publish/subscribe clusters: Direct and topic host routing

WM3021.1

Notes:

To create a cluster-based publish/subscribe infrastructure, the TOPIC object is clustered.

With IBM MQ V8, the ability to have a topic host routing for clustered publish/subscribe was introduced. As you see in the next slide, there are advantages to using topic host routing. The attribute that controls the routing on a publish/subscribe clustering is the CLROUTE attribute of the topic object.

Clustered publish/subscribe proxy subscriptions

- Direct routed
 - Proxy subscriptions are sent from a subscribing queue manager to ***all other*** queue managers in the cluster
 - Publications from a queue manager are sent directly to the queue managers that sent proxy subscriptions
- Topic host is routed
 - Proxy subscriptions from subscribing queue managers are sent only to the ***topic hosts***, which consist of queue managers that host a definition of the clustered topic
 - Publications are always sent to one of the topic hosts, and the topic hosts then forward the messages to subscribing queue managers

© Copyright IBM Corporation 2015

Figure 7-36. Clustered publish/subscribe proxy subscriptions

WM3021.1

Notes:

Use of topic host routed clusters can significantly reduce the traffic, and in turn, improve performance in a large IBM MQ publish/subscribe cluster.

PSCLUS queue manager attribute

- Exclude a clustered queue manager from inclusion in the publish/subscribe topology
- Helps mitigate added load in the cluster due to publish/subscribe propagation activities
- Intent is to use PSCLUS in *all* the queue managers in the cluster
 - Setting PSCLUS to disabled on a partial subset of queue managers can lead to many error messages
 - Consider using topic host routing for publish/subscribe instead of direct-routed clusters

© Copyright IBM Corporation 2015

Figure 7-37. PSCLUS queue manager attribute

WM3021.1

Notes:

Another queue manager attribute controls whether the queue manager is, or is not, included in a publish/subscribe topology.

Values for the PSCLUS queue attribute are ENABLED and DISABLED.



Important

While setting PSCLUS to DISABLED effectively excludes the queue manager from participating in the publish/subscribe topology, there are considerations to be aware of before setting PSCLUS to DISABLED in one or more queue managers. Read the documentation in section “Inhibiting clustered publish/subscribe” in the IBM MQ Knowledge Center.



Publish/subscribe cluster administration: Direct routing

- Alter the **CLUSTER** attribute of the **TOPIC** object to include the cluster name:

```
/MQ11 ALTER TOPIC (FOODS.ANCHOR) CLUSTER (WMADMCLS)
```
- CLROUTE** defaults to DIRECT
- Topic information is now propagated across cluster queue managers

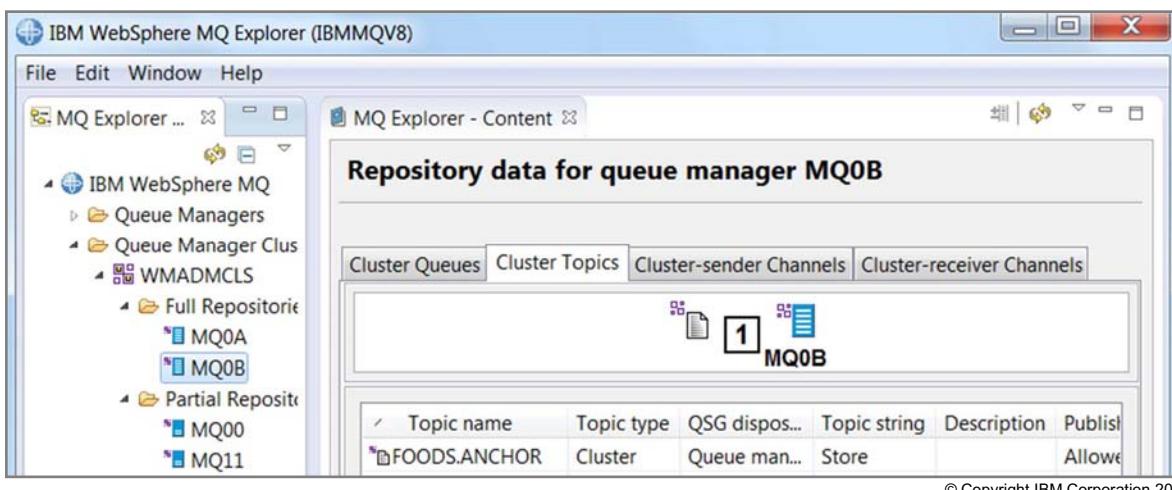


Figure 7-38. Publish/subscribe cluster administration: Direct routing

WM3021.1

Notes:

This set of slides shows the MQSC commands available to administer distributed publish/subscribe, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Equivalent capabilities are provided on IBM MQ Explorer.

Clustered publish/subscribe views

```

CSQU055I Target queue manager is
MQ11

DIS TOPIC(*) TYPE(CLUSTER)
TOPICSTR CLUSTER CLUSQMGR
CLROUTE CLSTATE
...
...
TOPIC(FOODS.ANCHOR)
TYPE(CLUSTER)
QSGDISP(QMGR)
CLUSTER(WMADMCLS)
TOPICSTR( Store )
CLROUTE(DIRECT)
CLSTATE(ACTIVE)
CLUSQMGR(MQ11)
CSQ9022I MQ11 CSQMDRTS ' DIS
TOPIC' NORMAL COMPLETION
...
...

```

MQ0B DIS PUBSUB TYPE(ALL)

```

CSQM293I MQ0B CSQMDRTC 1 PUBSUB
FOUND MATCHING REQUEST CRITERIA
CSQM201I MQ0B CSQMDRTC DIS
PUBSUB DETAILS 494
TYPE(LOCAL)
QMNAME(MQ0B)
STATUS(ACTIVE)
SUBCOUNT(6)
TPCOUNT(9}
END PUBSUB DETAILS

```

Figure 7-39. Clustered publish/subscribe views

WM3021.1

Notes:

This set of slides shows the MQSC commands available to administer distributed publish/subscribe, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Equivalent capabilities are provided on IBM MQ Explorer.



DISPLAY TCLUSTER view

```
MQ00 DIS TCLUSTER(*) ALL
CSQM293I MQ00 CSQMDRTC DIS TCLUSTER DETAILS 580
TOPIC(FOODS.ANCHOR)
...
CLUSTER(WMADMCLS)
TOPICSTR(Store)
...
PUBSCOPE(ALL)
SUBSCOPE(ALL)
PROXYSUB(FIRSTUSE)
WILDCARD(PASSTHRU)
CLROUTE(DIRECT)
CLSTATE(ACTIVE)
CLUSQMGR(MQ11)
QMID(MQ11.CDC1E6FC3BE17D2B)
```

© Copyright IBM Corporation 2015

Figure 7-40. DISPLAY TCLUSTER view

WM3021.1

Notes:

This set of slides shows the MQSC commands available to administer distributed publish/subscribe, which can be run in the console display, and CSQUTIL JCL. If using option 8 of the ISPF panels, the ISPF COMMAND option can also be used to enter the commands in MQSC format.

Use the Queue Manager Clusters menu item, Cluster Topics tab, on IBM MQ Explorer.

Changing cluster topic from direct to topic host routing

```
+CSQU055I Target queue manager is MQ11
ALTER TOPIC(FOODS.ANCHOR) CLUSTER('')
```

1 Remove from cluster

```
+CSQX500I MQ0B CSQXRCTL Channel WMADMCLS.MQ0A started
+CSQX500I MQ0A CSQXRCTL Channel WMADMCLS.MQ0B started
```

Change causes cluster activity

```
+CSQU055I Target queue manager is MQ11
ALTER TOPIC(FOODS.ANCHOR) CLROUTE(TOPICHOST) +
CLUSTER('WMADMCLS')
```

2 Change CLROUTE and recluster

```
/MQ11 DIS TCLUSTER(FOODS.ANCHOR) ALL
+CSQM201I MQ11 CSQMDRTC DIS TCLUSTER DETAILS 623
```

3 Check results

```
TOPIC(FOODS.ANCHOR)
TYPE(CLUSTER) ... ...
CLUSTER(WMADMCLS)
TOPICSTR(Store)
```

...

CLROUTE(TOPICHOST)

CLSTATE(ACTIVE)

CLUSQMGR(MQ11)

© Copyright IBM Corporation 2015

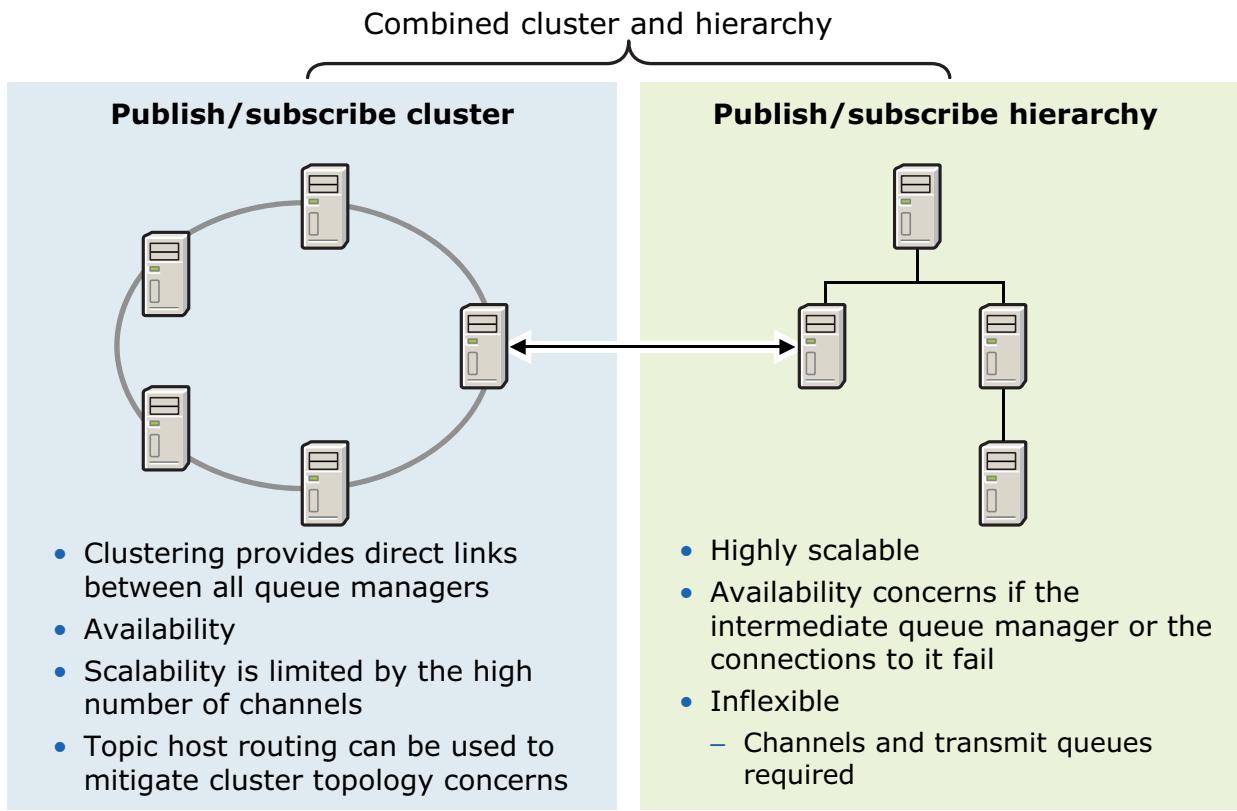
Figure 7-41. Changing cluster topic from direct to topic host routing

WM3021.1

Notes:

This slide explains the process of how to change a direct cluster topic to topic host routing.

Distributed publish/subscribe topologies considerations



© Copyright IBM Corporation 2015

Figure 7-42. Distributed publish/subscribe topologies considerations

WM3021.1

Notes:

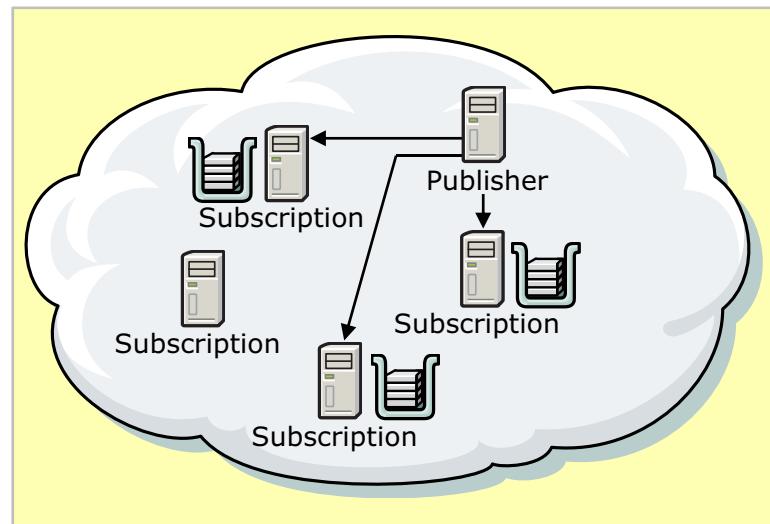
With the introduction of topic host routing in IBM MQ V8, clustered publish/subscribe might be the best option. As with any design exercise, adequate research and consideration must be given to the selection of a topology.

Before finalizing a publish/subscribe topology, a review of section *Designing publish/subscribe clusters* in the IBM MQ Knowledge Center should be completed. This section includes documentation on publish/subscribe best practices.

Highly available subscriptions: A different approach

Achieve the same resiliency as point-to-point clustered queues in a publish/subscribe environment:

1. Configure duplicate cluster queues
2. **Do not cluster** the topic
3. Define an administrative subscription for every queue manager where the publisher can connect, and point these subscriptions to the clustered queues
4. Publish to the topic



© Copyright IBM Corporation 2015

Figure 7-43. Highly available subscriptions: A different approach

WM3021.1

Notes:

This slide provides another interesting option to consider while designing a publish/subscribe topology. However, review of the publications that are listed in the previous slide should take precedence over completing the topology design.



Before you start implementing publish/subscribe (1 of 2)

- Has the architecture team had adequate training on publish/subscribe?
 - Impacts of the topic tree structure
 - Effects of publish/subscribe topology choices on queue managers and IBM MQ clusters
- Has a formal analysis of topics necessary to the organization been conducted?
- Have topic standards been set up for administrators and developers?
- Is publish/subscribe planning to use existing clusters?
- Is publish/subscribe planning to use existing or new queue managers?

© Copyright IBM Corporation 2015

Figure 7-44. Before you start implementing publish/subscribe (1 of 2)

WM3021.1

Notes:

This slide presents some questions to consider when implementing or administering a publish/subscribe infrastructure.



Before you start implementing publish/subscribe (2 of 2)

- Are there existing publish/subscribe applications?
 - What publish/subscribe implementation or version are they using?
 - What compatibility attributes are needed, such as PSMODE or PSPROP?
- Is the team familiar with the publish/subscribe best practices that are documented in IBM Knowledge Center?
- Are volume expectations known?
- Do any of the queue managers in the cluster need to be exempt from the publish/subscribe traffic?
- Do IBM MQ administrators have adequate knowledge of publish/subscribe?
- Has the team reviewed support packs MP0C and MP1H?

© Copyright IBM Corporation 2015

Figure 7-45. Before you start implementing publish/subscribe (2 of 2)

WM3021.1

Notes:

Unit summary

- Differentiate between publish/subscribe and point-to-point messaging
- Summarize the history of publish/subscribe and how it influences current functionality
- Identify the basic components of publish/subscribe
- Describe key properties of topics, subscriptions, and publications
- Summarize managed and unmanaged subscriptions
- Describe ways to administer publish/subscribe
- Summarize publish/subscribe life cycles
- Describe distributed publish/subscribe topologies and when to use them
- Identify factors to consider when implementing or maintaining publish/subscribe

© Copyright IBM Corporation 2015

Figure 7-46. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. True or false: The newest V7+ publish/subscribe implementation is referred to as *queued publish/subscribe* and *command-based publish/subscribe*.

2. You define a QALIAS to point to a topic. In addition to setting the TARGET attribute to the queue where the publication is sent, what other attribute needs to be set in the QALIAS definition?
 - a. DEFPSIST to NO
 - b. PUT to ENABLED
 - c. TARGTYPE to TOPIC
 - d. CUSTOM to the topic string

© Copyright IBM Corporation 2015

Figure 7-47. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

1.

2.

Checkpoint questions (2 of 2)

3. True or false: PSMODE in the queue manager is set to COMPAT. This setting indicates that a *queued publish/subscribe* interface is running.

4. How can you tell if a subscription was explicitly defined or created with MQSUB?
 - a. Check the displayed subscription attribute USERDATA
 - b. Check the displayed queue manager attribute PSMODE
 - c. Check the displayed subscription attribute PSPROP
 - d. Check the displayed subscription attribute SUBTYPE

5. True or false: Use of topic host-based publish/subscribe mitigates excessive use of resources in a cluster.

© Copyright IBM Corporation 2015

Figure 7-48. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

- 3.

- 4.

- 5.

Checkpoint answers (1 of 2)

1. True or false: The newest V7+ publish/subscribe implementation is referred to as *queued publish/subscribe* and *command-based publish/subscribe*.

Answer: False. The recent implementation is referred to as an integrated publish/subscribe or a publish/subscribe interface.

2. You define a QALIAS to point to a topic. In addition to setting the TARGET attribute to the queue where the publication is sent, what other attribute needs to be set in the QALIAS definition?
 - a. DEFPSIST to NO
 - b. PUT to ENABLED
 - c. TARGTYPE to TOPIC
 - d. CUSTOM to the topic string

Answer: c

© Copyright IBM Corporation 2015

Figure 7-49. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. True or false: PSMODE in the queue manager is set to COMPAT. This setting indicates that a *queued publish/subscribe* interface is running.

Answer: False. COMPAT indicates that the publish/subscribe engine is active, and queued publish/subscribe is stopped.

4. How can you tell if a subscription was explicitly defined or created with MQSUB?

- a. Check the displayed subscription attribute USERDATA
- b. Check the displayed queue manager attribute PSMODE
- c. Check the displayed subscription attribute PSPROP
- d. Check the displayed subscription attribute SUBTYPE

Answer: d

5. True or false: Use of topic host-based publish/subscribe mitigates excessive use of resources in a cluster.

Answer: True

© Copyright IBM Corporation 2015

Figure 7-50. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 6

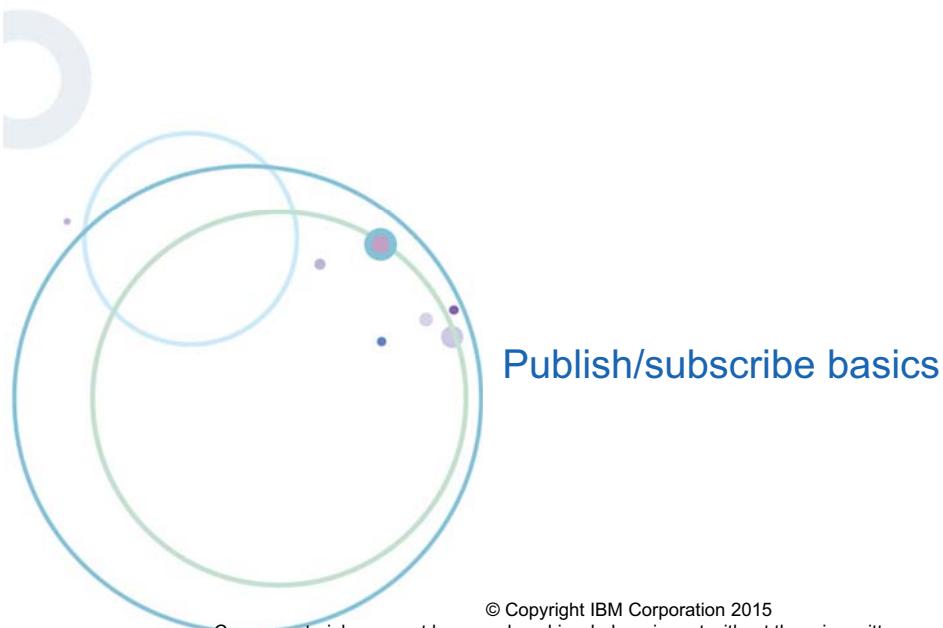


Figure 7-51. Exercise 6

WM3021.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Create a topic
- Create a managed subscription
- Publish messages
- Subscribe to messages without a managed subscription
- Display publish/subscribe subscription status
- Use IBM MQ Explorer to display publish/subscribe details

© Copyright IBM Corporation 2015

Figure 7-52. Exercise objectives

WM3021.1

Notes:

Unit 8. Queue sharing groups

What this unit is about

This unit covers the support in WebSphere MQ for shared queues in a z/OS sysplex environment.

What you should be able to do

After completing this unit, you should be able to:

- Describe the concept and use of queue-sharing groups
- Differentiate between non-shared and shared queues
- Identify coupling facility capabilities according to coupling facility level
- Differentiate among coupling facility offload and overflow options for queue sharing groups
- Identify commands to manage the queue sharing group environment
- Summarize distributed queuing support for queue sharing groups
- Describe Intra-group queuing
- Differentiate between queue sharing groups and cluster capabilities
- Describe how to join a queue sharing group to a cluster

Unit objectives

- Describe the concept and use of queue-sharing groups
- Differentiate between non-shared and shared queues
- Identify coupling facility capabilities according to coupling facility level
- Differentiate among coupling facility offload and overflow options for queue sharing groups
- Identify commands to manage the queue sharing group environment
- Summarize distributed queuing support for queue sharing groups
- Describe Intra-group queuing
- Differentiate between queue sharing groups and cluster capabilities
- Describe how to join a queue sharing group to a cluster

© Copyright IBM Corporation 2015

Figure 8-1. Unit objectives

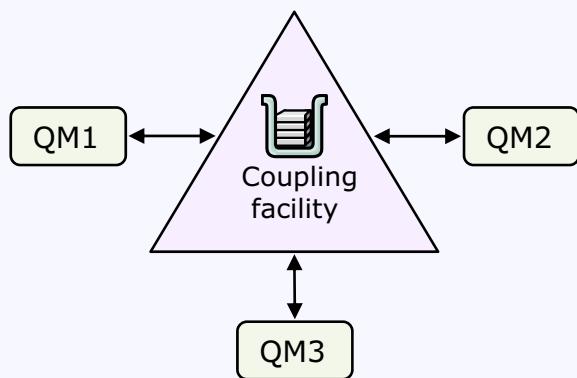
WM3021.1

Notes:

What are shared queues?

- Queues whose messages are kept in a coupling facility
- Queue managers in the same z/OS sysplex access the same shared queues by using connectivity via the coupling facility
- When several z/OS queue managers share the same queues, they are called a **queue-sharing group**
- Provide high availability in an IBM MQ infrastructure

Shared queues in a queue-sharing group



© Copyright IBM Corporation 2015

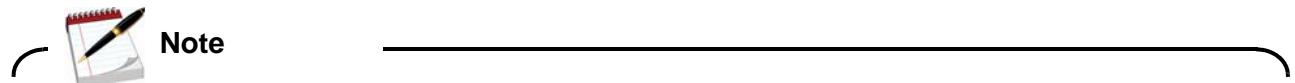
Figure 8-2. What are shared queues?

WM3021.1

Notes:

A shared queue is a special type of a local queue. The messages on a shared queue are stored in a list structure of the coupling facility (CF), and thus might be accessed from any queue manager in the sysplex.

The definition objects for the shared queues, together with other shareable definition objects, are stored in a DB2 shared database. Nonshared, regular local queues are still there, and the local page data sets for each queue manager continue to serve them.



Note

Developing applications for shared queues is similar to developing any other IBM MQ application. However, section *Application Programming with shared queues* in the IBM MQ Knowledge Center must be reviewed if you plan to use shared queues for a new application, or migrate an existing application to shared queues. Two key items to review are: special details on use of correlation or message ID, and dynamic queue consideration, where only permanent dynamic queues are allowed; however, the entire section should be reviewed.

Differences between nonshared and shared local queues

Storage	Nonshared local queues	Shared queues
Object definition storage	Stored on queue manager page set zero	Stored in DB2 OBJ_B_QUEUE table
Message storage	Page sets and buffer pools <ul style="list-style-type: none"> • Stored, or “offloaded” according to message size: <ul style="list-style-type: none"> — Small messages: Usually kept in the coupling facility — Large messages: Message data can be stored in a shared message data set (SMDS) or DB2 — Messages over 63 K: Always offloaded — Can also be set up to overflow to flash express cards, also called storage class memory (SCM) 	Represented by an entry in a coupling facility structure <ul style="list-style-type: none"> • Stored, or “offloaded” according to message size: <ul style="list-style-type: none"> — Small messages: Usually kept in the coupling facility — Large messages: Message data can be stored in a shared message data set (SMDS) or DB2 — Messages over 63 K: Always offloaded — Can also be set up to overflow to flash express cards, also called storage class memory (SCM)



For persistent messages, MQPUT and MQGET are logged to the log of the queue manager where the MQPUT or MQGET is done

© Copyright IBM Corporation 2015

Figure 8-3. Differences between nonshared and shared local queues

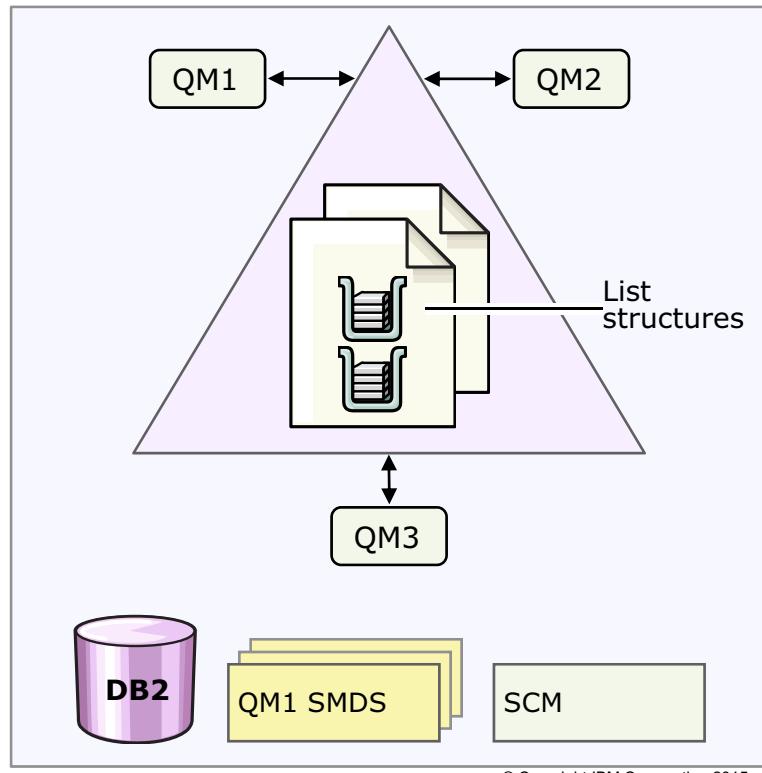
WM3021.1

Notes:

Basic differences between regular local queues and shared queues are shown on this slide.

Queue-sharing group components

- Coupling facility (CF)
- DB2 data sharing group
- IBM MQ CFSTRUCT objects
- Shared message data sets (SMDS)
- IBM MQ object definitions such as queues and channels
- Flash express cards normally referred to as storage-class memory (SCM)



© Copyright IBM Corporation 2015

Figure 8-4. Queue-sharing group components

WM3021.1

Notes:

The basic components of a queue-sharing group are:

- The coupling facility
- DB2
- IBM MQ object definitions
- A message offload or overflow option. SCM is considered overflow as it is part of the CF.
 - SCM overflow
 - SMDS offload
 - SMDS + SCM
 - DB2 offload

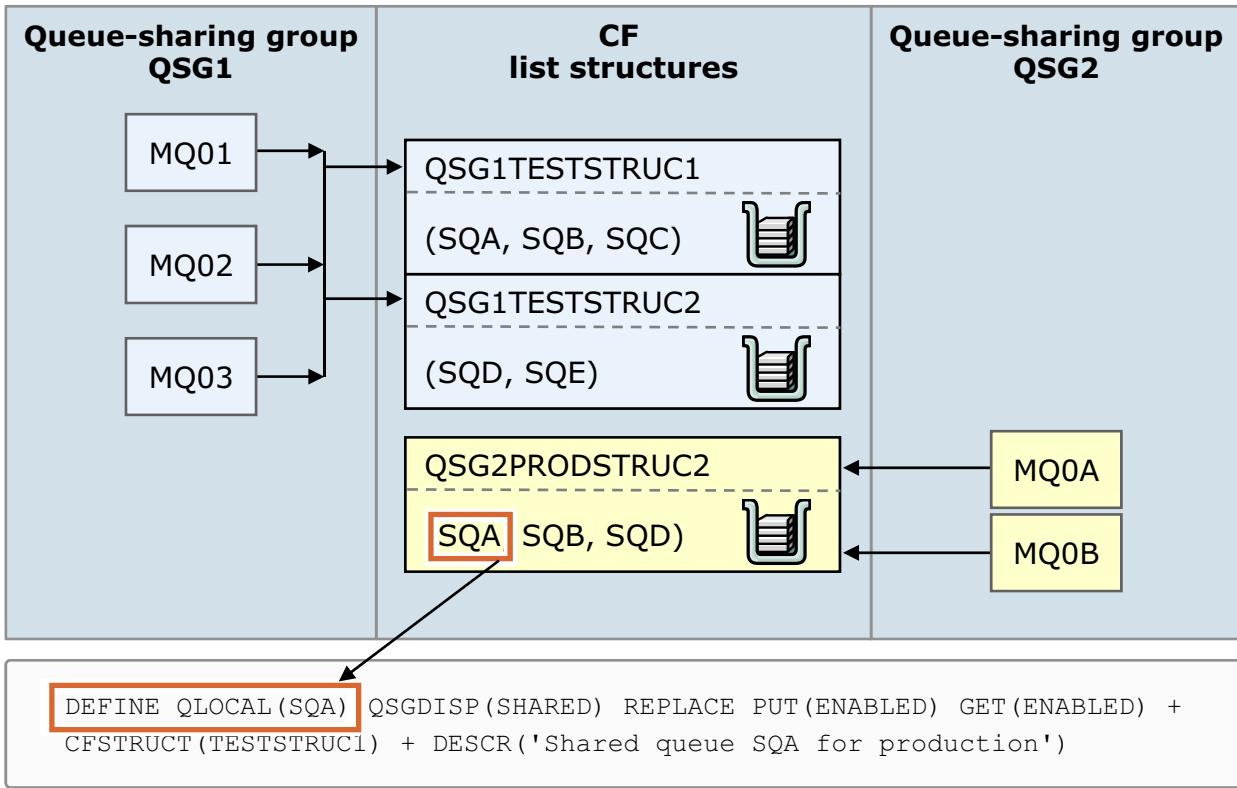
SMDS and SCM are the suggested resources.



Note

Consult the most recent *Capacity Planning and Tuning for z/OS* support pack for guidance.

Queue-sharing groups and CF list structures



© Copyright IBM Corporation 2015

Figure 8-5. Queue-sharing groups and CF list structures

WM3021.1

Notes:

A queue-sharing group (QSG) controls which queue managers can access which CF list structures, and hence which shared queues. Each queue manager can belong to only a single QSG. In effect, the queue-sharing group is the group of queue managers that can access the same shared queues.

Each CF list structure is owned by a QSG (by using a naming rule), and only the queue managers in that QSG can access it. Multiple CF list structures can belong to a particular QSG.

A shared queue is associated to a list structure at definition time.

Note the naming in the sample that is shown in the visual:

- The QSG name is up to 4 digits long.
- The structure name must start with the name of the owning QSG. Within different QSGs and structures, equal queue names can be used, which is a useful approach for migrating applications from test to production.

Coupling facility

- A coupling facility is a special type of LPAR available in z/OS
- Located outside the z/OS images in the sysplex and usually configured on a separate power supply
- Each message is referenced in the coupling facility's list structure, which is dedicated to the specific queue-sharing group
- Each coupling facility:
 - Can hold list structures for more than one queue-sharing group
 - Up to 32 queue managers can concurrently connect to a coupling facility list structure
- A coupling facility list structure can contain 512 shared queues
- The coupling facility capacity determines the amount of message data it can store



The real memory space that is used by the coupling facility is limited by the defined CF capacity. Keeping CF storage from being full is a key objective when using shared queues.

© Copyright IBM Corporation 2015

Figure 8-6. Coupling facility

WM3021.1

Notes:

The current maximum message size for shared queues is 100 MB.

Large message support and CF structure use history

- Availability of space in the CF structure is a key objective

Release	V5.2	V5.3	V6.0	V7.1	V8.0
Maximum message size	63 KB	63 KB	100 MB	100 MB	100 MB
Persistence	NP	Yes	Yes	Yes	Yes
Message store	CF	CF	CF, DB2	CF, DB2, SMDS	CF, DB2, SMDS, SCM CF overflow

- Three ways to mitigate CF overload:
 - Always active: Offload what is considered large (> 63 KB) messages to DB2 or SMDS, depending on what offload is configured
 - If configured: Offload smaller (< 63 KB) messages as the CF structure becomes full
 - CF structure extension with SCM; referred to as an *overflow* because SCM is part of the CF structure
- SCM can be used as the high performance solution to mitigate CF space constraints

© Copyright IBM Corporation 2015

Figure 8-7. Large message support and CF structure use history

WM3021.1

Notes:

This slide shows a historical perspective of how mitigating space constraints in the coupling facility were enhanced across IBM MQ releases.

SCM is an extension of the CF, an area where messages can overflow to maintain space in the CF.

Coupling facility capabilities and the CFLEVEL attribute

CFLEVEL	Capabilities
1-2	Nonpersistent messages under 63 KB
3	Persistent and nonpersistent under 63 KB
4	Persistent and nonpersistent up to 100 MB
5	<ul style="list-style-type: none"> • Persistent and nonpersistent up to 100 MB • Can be selectively offloaded to shared message data sets or DB2
19	<ul style="list-style-type: none"> • Persistent and nonpersistent up to 100 MB • Ability to overflow to SCM

- How a queue manager uses a coupling facility structure is defined in the **CFSTRUCT** IBM MQ object
- **CFSTRUCT** objects are stored in DB2

© Copyright IBM Corporation 2015

Figure 8-8. Coupling facility capabilities and the CFLEVEL attribute

WM3021.1

Notes:

The ability to use SCM is not dependent on the newest version of IBM MQ but rather on the ability to support a **CF 19 CFLEVEL** coupling facility.

SCM is part of the CF structure, not a true “offload.” When using SCM, you are extending the CF structure, or you are using SCM as an overflow area.

The IBM MQ CFSTRUCT object

```
DEFINE CFSTRUCT (structure-name)
  -CFCONLOS: ASQMGR, TERMINATE, TOLERATE
  -CFLEVEL: Must be at least 5 for offload to SMDS
  -OFFLOAD: SMDS or DB2

Offload rules. Different values for SMDS and DB2.
  SMDS default values shown:
  - OFFLD1SZ(32K/64K) OFFLD1TH(70)
  - OFFLD2SZ(4K/64K) OFFLD2TH(80)
  - OFFLD3SZ(0K/64K) OFFLD3TH(90)

SMDS database allocation attributes
  -DSGROUP, DSBLOCK, DSBUFFS, DSEXPAND(YES/NO)

Recovery attributes
  -RECOVER(YES/NO): states if CF recovery is supported in
    application
  -RECAUTO: determines if auto recovery upon detecting
    certain error conditions
```

© Copyright IBM Corporation 2015

Figure 8-9. The IBM MQ CFSTRUCT object

WM3021.1

Notes:

The CFSTRUCT object is used to define queue manager CF level capability, the message offload environment, and backup and recovery parameters for a coupling facility structure.

- The CFCONLOS attribute determines what action is taken when a queue manager loses connectivity to the CF structure.
 - ASQMGR takes direction from the queue manager attribute of the same name.
 - TERMINATE causes the queue manager to end.
 - TOLERATE results in the queue manager not ending.
- OFFLOAD indicates what resource is used to offload messages from the coupling facility. The offload rules are slightly different for SMDS than for DB2.
- SCM is an extension to the CF. The algorithm that is used to control overflow of the CF messages to CFM is the KEYPAIR1 algorithm in the CFRM policy. KEYPAIR1 is discussed in the SCM slide.

Enabling queue-sharing groups checklist (1 of 2)

- Before you start:
 - Ensure the objectives to be accomplished with an offload scheme are clear
 - Obtain expected volumes and arrival patterns, such as peak times
- Take time to understand the differences between SMDS and SCM
- Set up the required DB2 environment
- If using SMDS, create SMDS data set CSQ4SMDS member of SCSQPROC
- Add CF structure definition to CFRM policy and confirm outcome
- If using SCM, configure CFRM to use SCMALGORITHM



This checklist is for illustrative purposes only. Detailed steps to configure queue-sharing groups are documented in the IBM MQ Knowledge Center.

© Copyright IBM Corporation 2015

Figure 8-10. Enabling queue-sharing groups checklist (1 of 2)

WM3021.1

Notes:

Enabling queue-sharing groups checklist (2 of 2)

- Add QSG information to the CSQ6SYSP parameter macro
- Add the CSQ4INSS member to the CSQINP2 concatenation of the queue manager started task
- APF authorize and include the HLQ.SDSNLOAD library to the started task STEPLIB concatenation
- Define the CFSTRUCT MQ object
- More offload scenario and tuning information:
 - Support pack MP1H: IBM MQ for z/OS V7.1 performance report
 - Support pack MQ16: Capacity Planning and Tuning for IBM MQ for z/OS



This checklist is for illustrative purposes only. Detailed steps to configure queue-sharing groups are documented in the IBM MQ Knowledge Center.

© Copyright IBM Corporation 2015

Figure 8-11. Enabling queue-sharing groups checklist (2 of 2)

WM3021.1

Notes:

SMDS offload

- Requires CF level 5 and IBM MQ V7.1 or higher
- Offload algorithm based on the rule pairs that are defined in the CFSTRUCT object
- Rule pairs consist of message size + CF percentage availability
- Default settings are:
 - If the coupling facility structure is more than 70% full, offload data for messages that exceed 32 KB
 - If the coupling facility structure is more than 80% full, offload data for messages that exceed 4 KB
 - If the coupling facility structure is more than 90% full, offload data for messages that exceed 0 KB (all messages)

© Copyright IBM Corporation 2015

Figure 8-12. SMDS offload

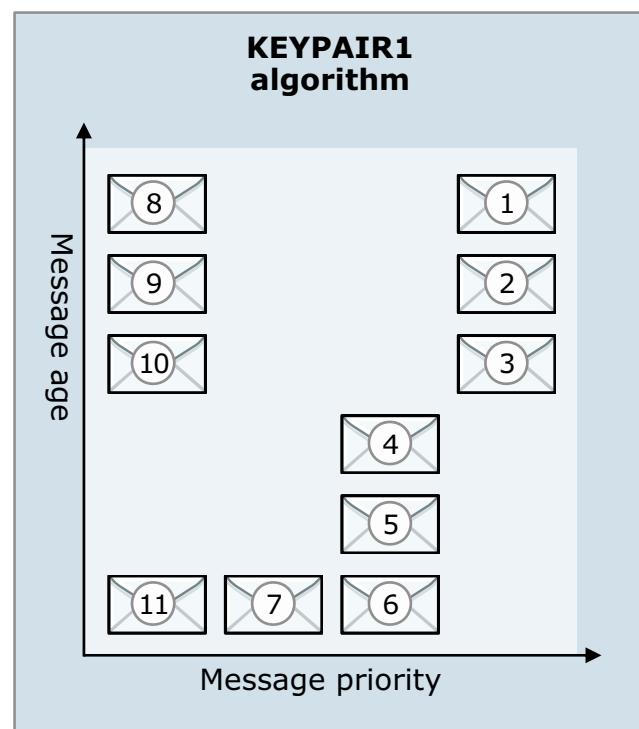
WM3021.1

Notes:

The SMDS offload rules are specified and can be altered in the CFSTRUCT document.

CF structure extension with SCM

- Requires CF level 19
- Configured by the **SCMALGORITHM** keyword in the CFRM policy for the structure
- Offload uses CF KEYPAIR1 algorithm based on message age and priority:
 - Pre-staging: When CF is 90% full, messages least likely to be accessed are moved to SCM
 - Prefetching: When 70% or more of the CF space is available, messages are brought back to CF



© Copyright IBM Corporation 2015

Figure 8-13. CF structure extension with SCM

WM3021.1

Notes:

The CFRM policy controls SCM overflow. As of the time this course is written, KEYPAIR1 is the only overflow algorithm for CF19 level.

What is the best option to mitigate CF space constraints?

- Increase the size of the CF structure?
- Use SMDS to offload messages?
- Extend the CF structure with SCM?
- Offload to DB2?
- Combination of SMDS offload and SCM overflow?

	DB2 offload	SMDS offload	SCM overflow	Bigger CF
Cost	Not considered	Less than SCM *	Less than CF	Most expensive
Performance	Very slow	Slower than SCM	High performance option	Best
Tradeoff	No longer advised for offload	Offload performance IBM MQ version must be MQV7.1	Requires CF 19	Real storage availability?
 No “one size fits all” answer.				* While in some cases SCM hardware costs might be higher, administrative costs might be lower, so entire solution cost must be assessed.

© Copyright IBM Corporation 2015

Figure 8-14. What is the best option to mitigate CF space constraints?

WM3021.1

Notes:

MQSC commands to manage shared queue environment

Manage CFSTRUCT options

DISPLAY CFSTRUCT
DEFINE CFSTRUCT
ALTER CFSTRUCT
DELETE CFSTRUCT

Check CFSTRUCT offload status

DISPLAY CFSTATUS

Manage SMDS data set options

DISPLAY SMDS
ALTER SMDS

Manage status and availability of QSG data sets

DISPLAY CFSTATUS TYPE(SMDS)
RESET SMDS

Review SMDS data set details:

DISPLAY USAGE TYPE(SMDS)

Manage status and availability of data set connections:

DISPLAY SMDSCCONN
START SMDSCCONN
STOP SMDSCCONN

Manage backup and recover of shared messages

BACKUP CFSTRUCT
RECOVER CFSTRUCT



It is critical to schedule frequent backups to facilitate recovery.

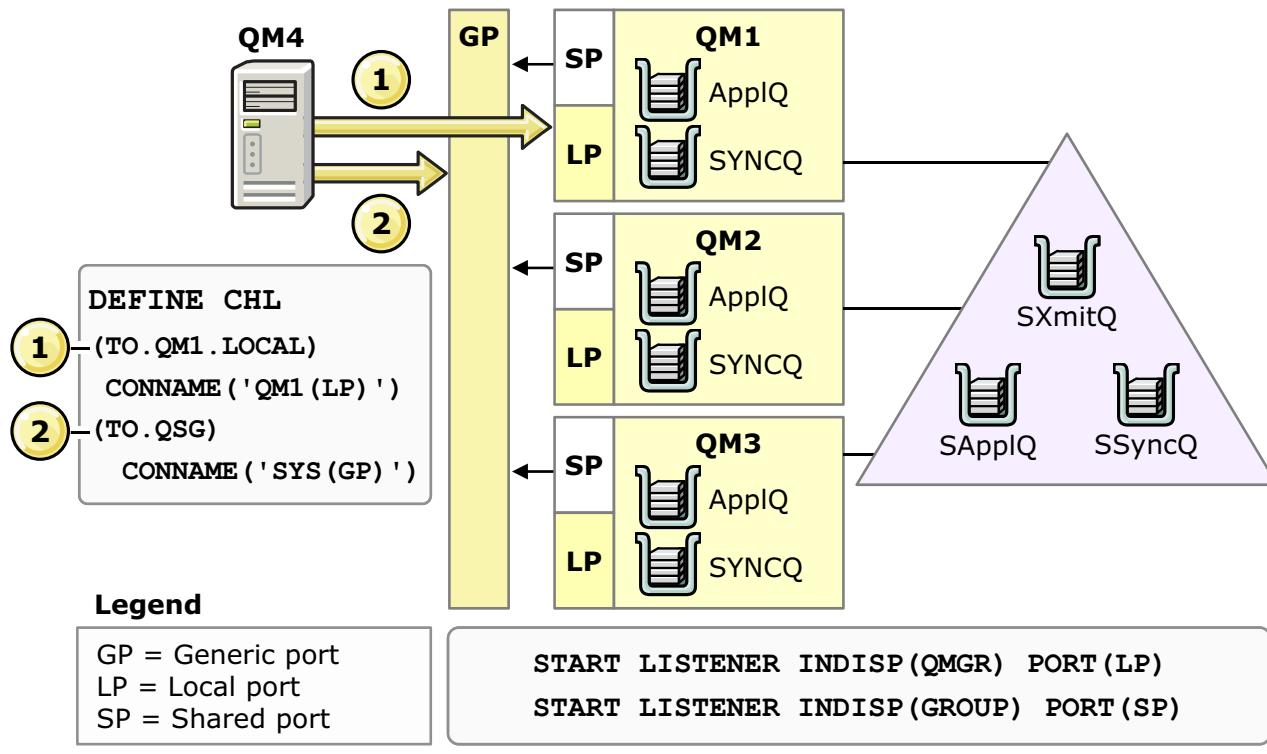
© Copyright IBM Corporation 2015

Figure 8-15. MQSC commands to manage shared queue environment

WM3021.1

Notes:

DQM support for shared queues: Inbound



© Copyright IBM Corporation 2015

Figure 8-16. DQM support for shared queues: Inbound

WM3021.1

Notes:

To support sysplex-wide addressability of shared queues, there is a concept of multiple listeners. The traditional listener is the one that is started for the local port (LP), which is unique for each queue manager. Connecting to the local port gives behavior as before. However, existing channels can also put messages on a shared target queue, giving high back-end availability.

There is a second listener, which is started for the shared port (SP).

- The shared ports of all queue managers of a QSG are mapped to a generic port (GP) by network configuration: VTAM generic resources or TCP/IP DNS.
- A remote sending MCA definition names the generic port.
- An inbound session request to the GP results in establishing a session with one of the queue managers in the group.
- The sending queue manager can resynchronize with any member of the queue-sharing group; thus, high front-end availability is given by using the generic port.
- The generic addressing network support provides automatic workload balancing and increased capacity.

- Connections to be made directly to the shared port (SP) are not expected, but they are not stopped, either.

DQM support for shared queues: Outbound

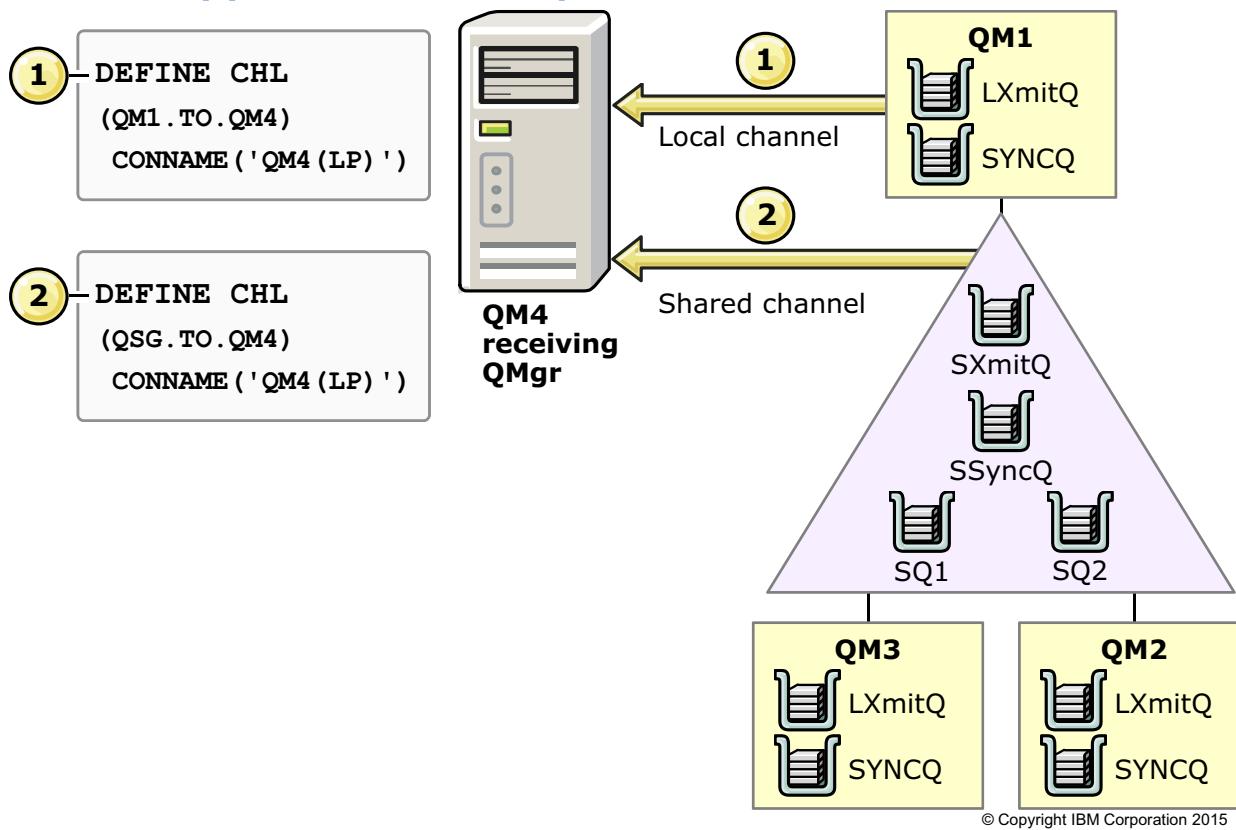


Figure 8-17. DQM support for shared queues: Outbound

WM3021.1

Notes:

An outbound channel is considered to be a shared channel if it is taking messages from a shared transmission queue. If the channel is shared, it holds synchronization information at queue-sharing group level, in a shared synchronization, or SYNC queue.

As a result, the channel can be restarted on a different queue manager and channel initiator instance within the queue-sharing group if the communications subsystem, channel initiator, or queue manager fails.

Restarting failed channels in this way is a feature of shared channels called peer channel recovery.

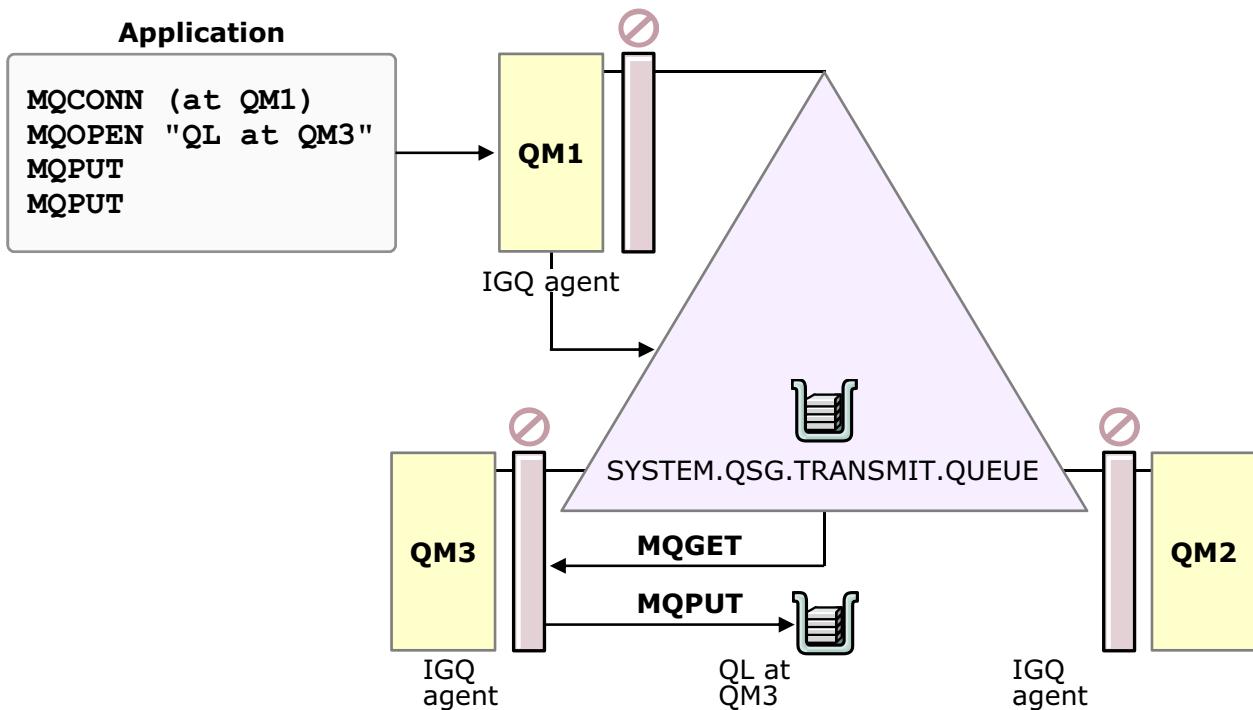
Typically, there is only one (sender) channel for a queue-sharing group to a particular remote queue manager. The members of the QSG are identified at the receiving queue manager by exchanging the QSG name, so logically they appear as the same instance.

Shared channel starts are workload-balanced across all the systems in the QSG.

All queue managers are triggered, but only the one that accesses the transmission queue first starts the channel and does the transmission.

Increased capacity is achieved by having more queue managers available to access transmission queues and to deliver the messages on to target systems.

Intragroup queuing



© Copyright IBM Corporation 2015

Figure 8-18. Intragroup queuing

WM3021.1

Notes:

There is a special feature to transfer messages between queues that are stored on different queue managers within a queue-sharing group, without the use of channel-based distributed queuing. This feature is called intra-group queuing (IGQ).

Eligible messages are MQPUT and MQGET with NO_SYNCPOINT to and from a single shared SYSTEM.QSG.TRANSMIT.QUEUE.

On the MQPUT, the CorrelId is set to the target queue manager.

IGQ agents serve this single shared transmission queue, one per queue manager in the group. The queue manager does an MQGET with the CorrelId set to the queue manager name and delivers the messages to the target queue.

IGQ is ENABLED or DISABLED by using an ALTER QMGR command, and thus might be activated automatically and run permanently.

If there is a problem with delivering the message and there is no dead-letter queue, then the messages are discarded.

The channel initiator can also use IGQ:

- An inbound shared channel that receives a message that is destined for a queue on a different queue manager in the queue-sharing group uses intra-group queuing to hop the message to the correct destination.
- This action provides high front-end availability by using receiver channels.

How do shared queues compare with clusters?

- Channel initiator usage
 - When messages are sent across shared queues, no channels are used
 - This scenario requires the queue managers to be on the same QSG
- Availability
 - If a clustered queue manager fails, the message is not accessible unless configured for failover with shared disks
 - A shared queue continues to be processed by another available queue manager
- Capacity
 - Storage in a coupling facility is more expensive than disk
 - CF storage can be mitigated by using the offload techniques
 - It is less expensive to use a local queue
- Workload balancing
 - A coupling facility does its workload balancing based on actual load
 - Cluster weighing distributes according to its configuration, regardless of actual load

© Copyright IBM Corporation 2015

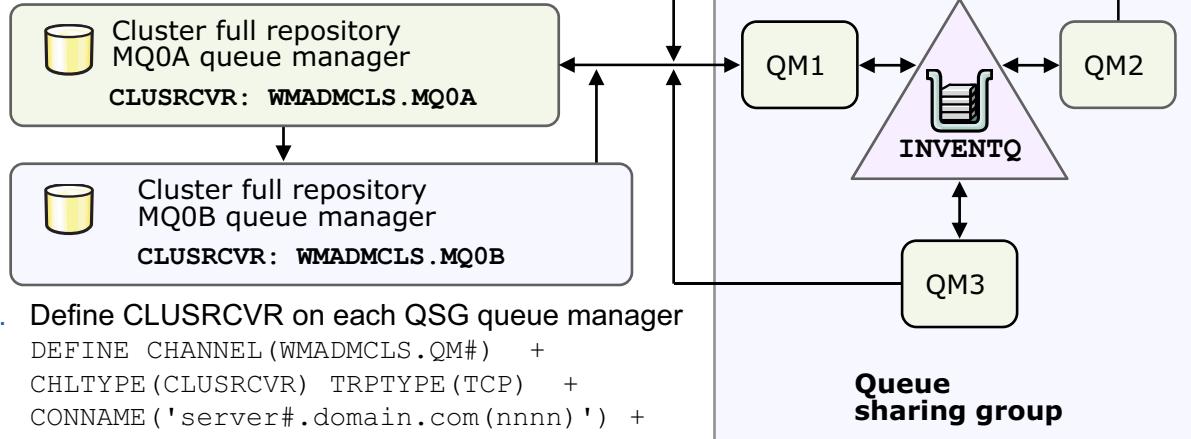
Figure 8-19. How do shared queues compare with clusters?

WM3021.1

Notes:

WebSphere Education 

Incorporating a queue-sharing group into an existing cluster



1. Define CLUSRCVR on each QSG queue manager

```
DEFINE CHANNEL (WMADMCLS.QM#) +
CHLTYPE (CLUSRCVR) TRPTYPE (TCP) +
CONNNAME ('server#.domain.com(nnnn)') +
CLUSTER (WMADMCLS)
```

2. Define one CLUSSDR for the QSG: using QM1

```
DEFINE CHANNEL (WMADMCLS.MQ0A) +
CHLTYPE (CLUSSDR) TPTYPE (TCP) +
CONNNAME ('mvsmc13.ilsvpn.ibm.com(1621)') +
CLUSTER (WMADMCLS) QSGDISP(GROUP)
```

3. Define the local queue on one of the QSG queue managers: using QM1

```
DEFINE QLOCAL (INVENTQ) CLUSTER (WMADMCLS) QSGDISP (SHARED) +
CFSTRUCT (TESTSTRUC1)
```

© Copyright IBM Corporation 2015

Figure 8-20. Incorporating a queue-sharing group into an existing cluster

WM3021.1

Notes:

When incorporating a queue-sharing group into an existing cluster, there are some details to consider:

- Each queue manager in the QSG requires the CLUSRCVR definition as in non-QSG queue managers.
- Only one queue manager in the QSG has a CLUSSDR definition.
- One clustered queue is defined in only one of the QSG queue managers.

Unit summary

- Describe the concept and use of queue-sharing groups
- Differentiate between non-shared and shared queues
- Identify coupling facility capabilities according to coupling facility level
- Differentiate among coupling facility offload and overflow options for queue sharing groups
- Identify commands to manage the queue sharing group environment
- Summarize distributed queuing support for queue sharing groups
- Describe Intra-group queuing
- Differentiate between queue sharing groups and cluster capabilities
- Describe how to join a queue sharing group to a cluster

© Copyright IBM Corporation 2015

Figure 8-21. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. What statements are correct when describing a coupling facility (CF), a queue-sharing group, and the way messages in the coupling facility are processed?
 - a. Each queue-sharing group has a dedicated CF list structure
 - b. Each message is referenced in the coupling facility's list structure
 - c. Coupling facilities can hold list structures for more than one queue-sharing group
 - d. Each list structure has an internal reference to the dedicated page set
2. True or False: Queue-sharing group object definitions are stored in the queue manager page set zero.

© Copyright IBM Corporation 2015

Figure 8-22. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.

Checkpoint questions (2 of 2)

3. Assuming use of the newest capabilities, what are the methods currently available to mitigate coupling facility storage constraints?
 - a. Shared message data set (SMDS) offload
 - b. Storage class memory (SCM) overflow
 - c. Page set offload
 - d. DB2 offload
4. True or False: The prerequisite to use SCM overflow is to use IBM MQ V8 for z/OS.
5. True or False: An advantage of a shared queue over an IBM MQ cluster is the ability to access messages in a queue even if a queue manager is unavailable.

© Copyright IBM Corporation 2015

Figure 8-23. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

5.

Checkpoint answers (1 of 2)

1. What statements are correct when describing a coupling facility (CF), a queue-sharing group, and the way messages in the coupling facility are processed?
 - a. Each queue-sharing group has a dedicated CF list structure
 - b. Each message is referenced in the coupling facility's list structure
 - c. Coupling facilities can hold list structures for more than one queue-sharing group
 - d. Each list structure has an internal reference to the dedicated page set

Answer: a, b, c. Page sets are not used for shared queues.

2. True or False: Queue-sharing group object definitions are stored in the queue manager page set zero.

Answer: False. Objects are stored in DB2.

© Copyright IBM Corporation 2015

Figure 8-24. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. Assuming use of the newest capabilities, what are the methods currently available to mitigate coupling facility storage constraints?
 - a. Shared message data set (SMDS) offload
 - b. Storage class memory (SCM) overflow
 - c. Page set offload
 - d. DB2 offload

Answer: a, b, d

4. True or False: The prerequisite to use SCM overflow is to use IBM MQ V8 for z/OS.

Answer: False. Use of SCM overflow requires CF level 19.

5. True or False: An advantage of a shared queue over an IBM MQ cluster is the ability to access messages in a queue even if a queue manager is unavailable.

Answer: True, shared queues mitigate “marooned messages.”

© Copyright IBM Corporation 2015

Figure 8-25. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Unit 9. Using IBM MQ events and the dead-letter queue utility

What this unit is about

This unit describes the monitoring and auditing capabilities that IBM MQ events facilitate. It also introduces the dead-letter queue handler utility.

What you should be able to do

After completing this unit, you should be able to:

- Describe the use of IBM MQ event types
- List the queues that are associated with each event
- Summarize use of the dead-letter queue handler



Unit objectives

- Describe the use of IBM MQ event types
- List the queues that are associated with each event
- Summarize use of the dead-letter queue handler

© Copyright IBM Corporation 2015

Figure 9-1. Unit objectives

WM3021.1

Notes:

Approaches to monitoring IBM WebSphere MQ

- Event monitoring
 - Detecting occurrences of instrumentation events
- Message monitoring
 - Identifying the route that a message has taken
- Accounting and statistics messages
 - Information about MQI operations
- Real-time monitoring
 - Determine the current state of queues and channels

© Copyright IBM Corporation 2015

Figure 9-2. Approaches to monitoring IBM WebSphere MQ

WM3021.1

Notes:

There are various approaches to monitoring IBM MQ. Each approach is applied and used in a different way, and each approach returns monitoring information in a different form. Depending on your needs, you use one, or a combination of the following approaches.

- Event monitoring
- Message monitoring
- Accounting and statistics messages
- Real-time monitoring

This unit refers to the event messages that are used for event monitoring, tracing and tracking messages and dead-letter queue messages.

Event queues: Overview

- SYSTEM.ADMIN.QMGR.EVENT
 - Authority, Inhibit, Local, Remote
 - Start-Stop (IBM WebSphere MQ for z/OS – start only)
- SYSTEM.ADMIN.PERFM.EVENT
 - Queue High/Low/Full
 - Service Interval
- SYSTEM.ADMIN.COMMAND.EVENT
 - Commands issued and processed
- SYSTEM.ADMIN.CONFIG.EVENT
 - Define, change, delete objects
- SYSTEM.ADMIN.CHANNEL.EVENT
 - Activated, Not Activated, Started, Stopped
 - Conversion Error
 - SSL and IMS Bridge related
- SYSTEM.ADMIN.LOGGER.EVENT
 - Write a new log extent if linear logging
 - Available for distributed platforms only

© Copyright IBM Corporation 2015

Figure 9-3. Event queues: Overview

WM3021.1

Notes:

You can request that IBM WebSphere MQ create *event messages* when certain conditions occur.

- These event messages have a predefined message format (**MQFMT_EVENT**) and are put on queues with fixed names.
- IBM MQ generates and writes these messages. It is *up to you* to provide an event message monitor capable of reading and interpreting the event messages and optionally, act on them.
- If an event queue is not available (maybe not defined, full, or **PUT**-inhibited), the appropriate event message is lost. It is not written to the dead-letter queue.

Event queues can be defined as remote queues. In this case, event messages are processed by the channels just as any other messages, which might include putting messages to the dead-letter queue if they cannot be written to the specified target queue.

Queue manager event



SYSTEM.ADMIN.QMGR.EVENT controlled by ALTER QMGR . . .

- Inhibit (INHIBITEV)
 - GET and PUT
- Local (LOCALEV)
 - MQOPEN fails to Local Object
 - Alias base queue type error
 - Queue type error
 - Unknown alias base queue
 - Unknown object name
- Start and Stop QMGR (STRSTPEV)
 - For z/OS, Start only
- Remote (REMOTEV)
 - Events relating to remote objects

© Copyright IBM Corporation 2015

Figure 9-4. Queue manager event

WM3021.1

Notes:

These exceptional conditions that can cause event messages to be written to **SYSTEM.ADMIN.QMGR.EVENT**.

- Conditions and events within this class are grouped into four categories.
- You can select or clear them by setting the appropriate queue manager attributes to **ENABLED** or **DISABLED** with the **ALTER QMGR** command.
- Remote events can be used to report the following conditions:
 - Remote queue unknown
 - Remote queue name error
 - Transmission queue unknown
 - Transmission queue type error
 - Transmission queue usage error



Performance events in ISPF panels

- Enabled if queue manager **PERFMEV** is also enabled

```

Display a Local Queue - 5

Press F7 or F8 to see other fields, or Enter to refresh details.
More: - +
Queue name . . . . . : MQ00
Disposition . . . . . : QMGR      MQ11
Event Control
  Queue full . . . . . : E  E=Enabled, D=Disabled

  Upper queue depth . . . : D  E=Enabled, D=Disabled
  Threshold . . . . . : 80    0 - 100 %

  Lower queue depth . . . : D  E=Enabled, D=Disabled
  Threshold . . . . . : 40    0 - 100 %

  Service interval . . . . : N  H=High, O=OK, N=None
  Interval . . . . . . : 999999999  0 - 999999999 milliseconds

```

© Copyright IBM Corporation 2015

Figure 9-5. Performance events in ISPF panels

WM3021.1

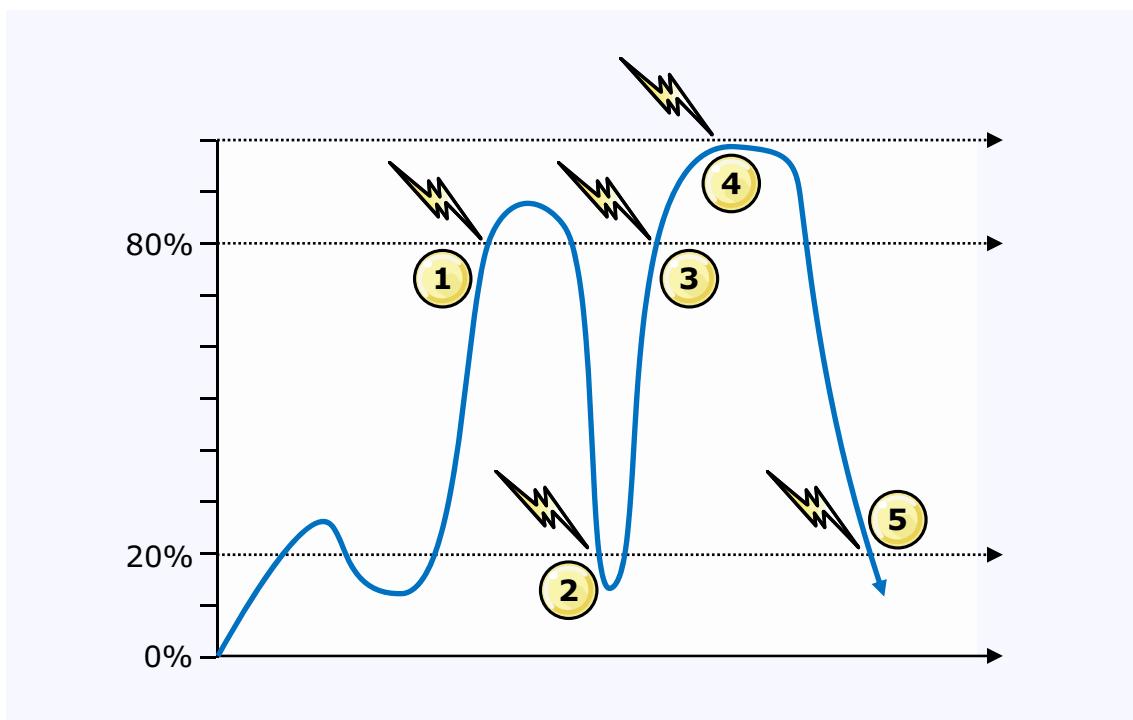
Notes:

Performance events are created and maintained per (local) queue.

- The figure shows the Event Control section (page) of the ISPF queue definition panels.
- Displayed are some of switches and values you can set to request events that help prevent a queue from becoming full:
 - An *upper queue threshold*, as a percentage value of the *maximum queue depth*. This event can be used to start an (extra) server task that processes the queue.
 - A *lower queue threshold*, as a percentage value of the *maximum queue depth*. This event signals that the queue depth is no longer critical.
 - The *queue full* event message indicates that, regardless of actions, the queue reached its *maximum queue depth* and therefore can no longer be written to.

Performance event messages are created when the QMGR attribute **PERFMEV** is **ENABLED**.

Depth event sample



© Copyright IBM Corporation 2015

Figure 9-6. Depth event sample

WM3021.1

Notes:

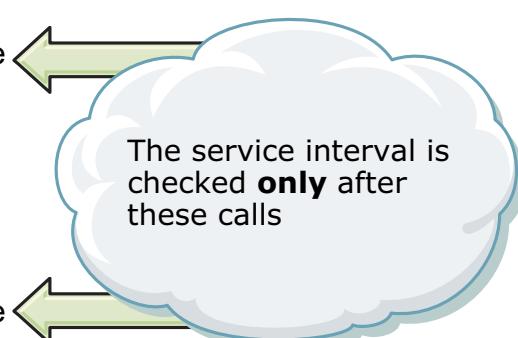
IBM MQ automatically maintains the **ENABLE** and **DISABLE** switches of the lower and upper threshold events.

- If the upper threshold is reached and the appropriate event message is written, the upper threshold event is disabled, and the lower threshold event is enabled.
- A second *upper threshold event* message does not exist, unless the lower threshold limit is reached first.

There must be a process to consume the event messages. IBM MQ as a system creates the event messages and puts them on the **SYSTEM.ADMIN.PERFM.EVENT** queue.

Consumption and clearing of the event queues is done with either a packaged utility or user written application. IBM MQ reports the events, but does not clear them; they are for the user to consume.

Service interval events

- Service *interval* is user-defined
 - In milliseconds
 - Queue Service Interval OK
 - Must be enabled
 - MQPUT or MQGET leaving nonempty queue
 - Followed by MQGET within service interval
 - Enables Queue Service Interval High
 - Queue Service Interval High
 - Must be enabled
 - MQPUT or MQGET leaving nonempty queue
 - Followed by MQGET outside service interval
 - Enables Queue Service Interval OK
- 

© Copyright IBM Corporation 2015

Figure 9-7. Service interval events

WM3021.1

Notes:

With the **Service Interval High** and **OK** events, you can monitor the time within a certain queue when it is serviced.

- The time that is specified as the *service interval* is checked, when a previous **MQGET** or **MQPUT** call does not empty the queue.
- A **Service Interval High** event message is written if a **GET** operation is not performed within the time specified.
- The **High** event is automatically disabled and the **OK** event is enabled.

Subsequent access to the queue will be checked as well, but the next service interval message that is generated is a **Service OK** message when the queue is accessed within the service interval for the first time.

Command events



SYSTEM.ADMIN.COMMAND.EVENT

Controlled by:

ALTER QMGR CMDEV(ENABLED | DISABLED | NODISPLAY)

- If enabled, an event message is written for each command that has been accepted and executed
 - Contains a time stamp, application name and type information, user ID, and the complete command syntax
 - Command reply is not included
- NODISPLAY prevents event message generation for DISPLAY commands

© Copyright IBM Corporation 2015

Figure 9-8. Command events

WM3021.1

Notes:

All commands that are successfully run cause an event message to be generated. In other words, the command input that is rejected because of incorrect syntax or missing authorization is not recorded, but the latter causes a queue manager local event, if enabled.

Configuration events



SYSTEM.ADMIN.CONFIG.EVENT

- Controlled by: **ALTER QMGR CONFIGEV (ENABLED | DISABLED)**
 - One message for each object create or delete command
 - Two messages for each object change command (even if no net change)
 - Tracks group / copy objects where command runs (not where entered)
 - No messages for temporary dynamic queues
 - No messages for internal interfaces or changes to volatile attributes
- Command **REFRESH QMGR TYPE (CONFIGEV)**
 - Generates an event for every object with specified type, generic name, and the duration of time since the last change

© Copyright IBM Corporation 2015

Figure 9-9. Configuration events

WM3021.1

Notes:

- Configuration events are reported when objects are created or modified
- A configuration event message contains information about the attributes of an object
- Operations that do not cause an event message being written are:
 - The command or **MQSET** call fails
 - Any operations on temporary dynamic queues
 - Changes to trigger state by internal interfaces
 - Changes to volatile queue attributes such as **CURDEPTH**, **IPPROCS**, and **OPPROCS**
 - Changes to the **SYSTEM.ADMIN.CONFIG.EVENT** queue
 - Clustering changes by the commands **REFRESH CLUSTER** or **RESET CLUSTER** and **RESUME CLUSTER** or **SUSPEND CLUSTER**
 - Creating or deleting a queue manager

The **REFRESH QMGR TYPE(CONFIGEV)** command can be used to take an (initial) snapshot of all the existing object definitions.

Channel events



SYSTEM.ADMIN.CHANNEL.EVENT

- Controlled by:
ALTER QMGR CHLEV (ENABLED | DISABLED | EXCEPTION)
- Event messages generated on:
 - Channel Start/Stop commands
 - Channel starting and stopping
 - Channel Conversion Error
 - Channel auto-definition attempts
- *Exception* option suppresses non-error event messages:
ALTER QMGR SSLEV (ENABLED | DISABLED)
 - Event messages for SSL errors
- **ALTER QMGR BRIDGEEV (ENABLED | DISABLED)**
 - Event messages for IMS bridge start and stop

© Copyright IBM Corporation 2015

Figure 9-10. Channel events

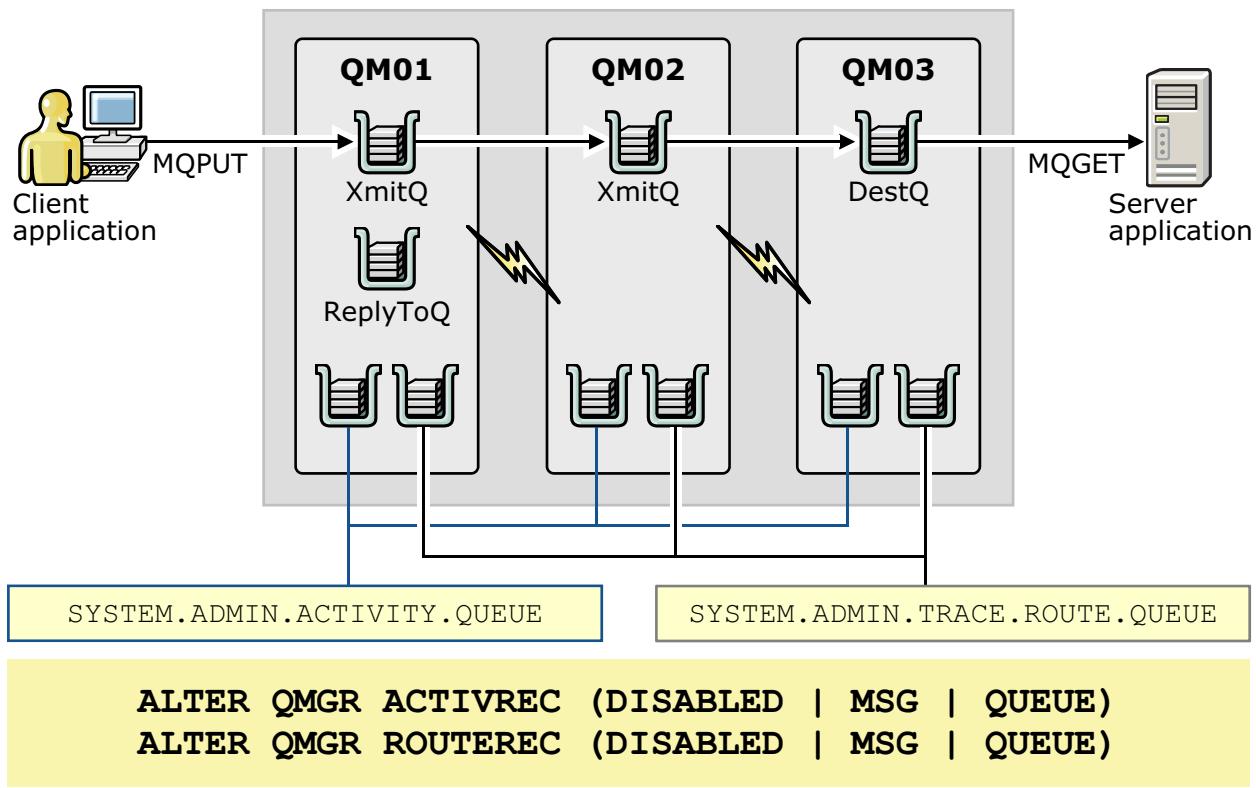
WM3021.1

Notes:

The **SYSTEM.ADMIN.CHANNEL.EVENT** is used to capture channel event situations. These events are captured by configuring the queue manager CHLEV attribute.

- Channel events can be limited to exceptional conditions, so no events are generated for channel starts and regular channel stops.
- Error events for SSL-protected channels and IMS bridge start and stop event messages also go to the channel event queue, if enabled.

Activity reporting and route tracking



© Copyright IBM Corporation 2015

Figure 9-11. Activity reporting and route tracking

WM3021.1

Notes:

Activity reporting and route tracking traces and tracks the route messages take through the IBM MQ network, which might be useful to determine what happened to messages that did not reach their intended destination.

Activity reporting is a per-message option and started by using the **ACTIVITY** report option in the MQMD.

- If this option is set, the queue manager generates a report message for every activity that takes place with this message, such as *put on a queue* or *sent down a channel*.
- The destination of this message is determined by the queue manager **ACTIVREC** attribute; it can be the application reply-to queue or the **SYSTEM.ADMIN.ACTIVITY.QUEUE**.

Trace-route messaging is based on dedicated diagnostic messages with a special format that allows for checking and recording the route that the message takes.

The queue manager **ROUTEREC** attribute directs the final report message either to the route message reply-to queue or to the **SYSTEM.ADMIN.TRACE.ROUTE.QUEUE**.

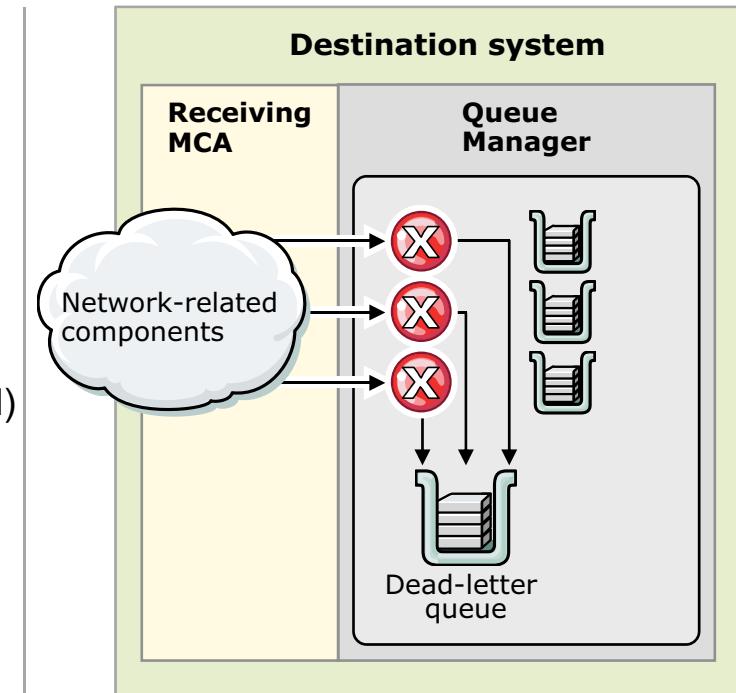
When messages go to the dead-letter queue

MCA's **MQPUT(1)** attempt to the destination queue returned a reason code of **MQRC_**

```
NOT_AUTHORIZED
PUT_INHIBITED
Q_FULL
PERSISTENT_NOT_ALLOWED
MSG_TOO_BIG_FOR_Q
Q_TYPE_ERROR
UNKNOWN_OBJECT_NAME
STORAGE_CLASS_ERROR
....
```

Dead-letter header (MQDLH) fields:

- Reason (**MQRC_*** or **MQFB***)
- DestQName
- DestQMGrName
- PutDate
- PutTime



© Copyright IBM Corporation 2015

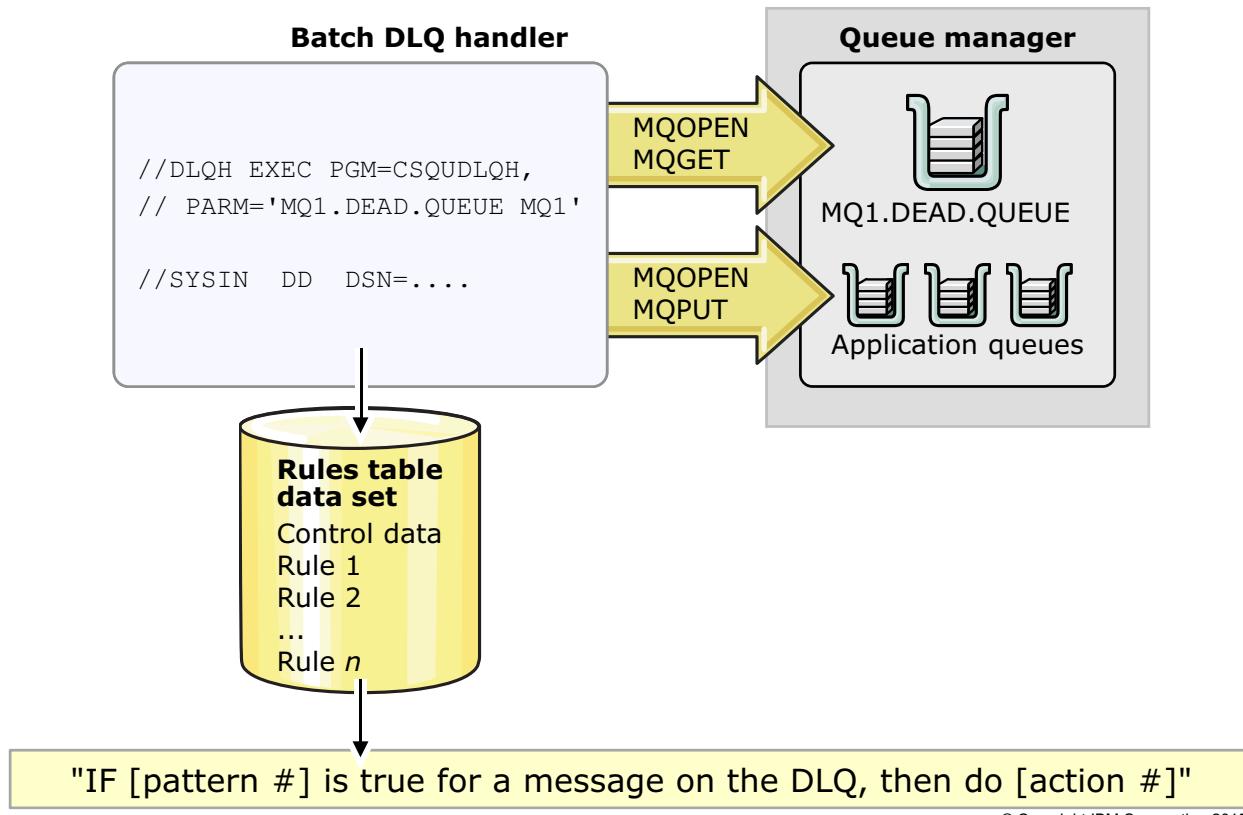
Figure 9-12. When messages go to the dead-letter queue

WM3021.1

Notes:

- The dead-letter queue (DLQ) is a holding queue for messages that cannot be delivered to their destination queues.
- The one local queue that takes the role of the DLQ is identified by the queue manager **DEADQ** attribute, which can be displayed and altered at any time.
- The most common condition for using the DLQ is when a receiving MCA cannot, for whatever reason, deliver a message that is sent by an adjacent queue manager.
- If no queue is named with the queue manager **DEADQ** attribute, then there is no dead-letter queue. If messages prove to be undeliverable and there is no dead-letter queue for the queue manager, in most cases the channel is stopped.
- When a message is placed on the DLQ, the queue manager includes an additional header to the message, and the message format is changed to **MQDLH**. The dead-letter header contains the original destination queue of the message and a reason code that tells why the message was put to the DLQ.

Dead-letter queue handler utility



© Copyright IBM Corporation 2015

Figure 9-13. Dead-letter queue handler utility

WM3021.1

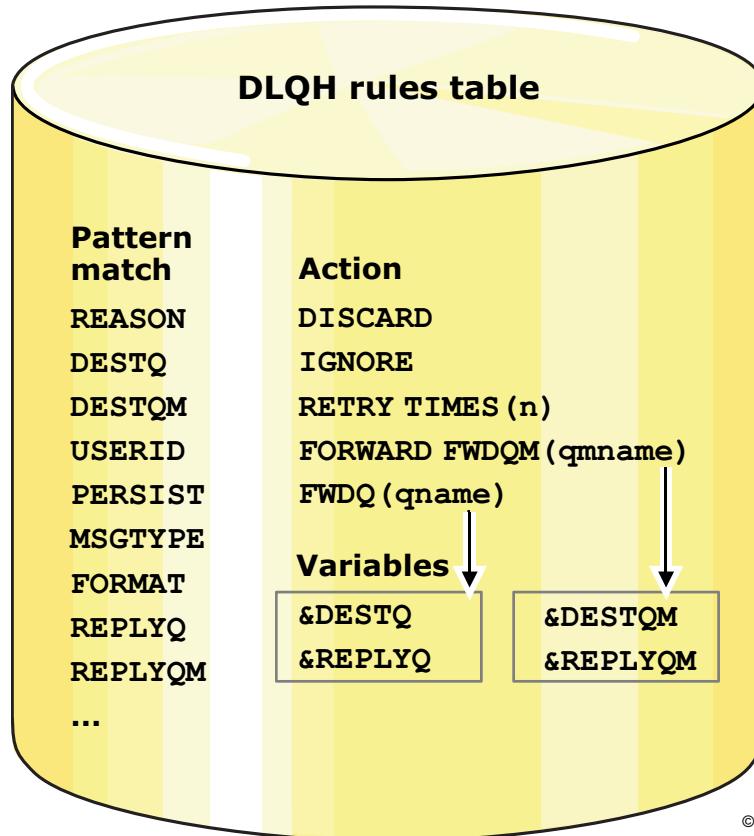
Notes:

If a dead-letter queue is identified, IBM MQ administrators must determine what to do with the messages that go there. Handling these messages can be done automatically with a dead-letter queue handler.

- The batch program CSQUDLQH is a dead-letter queue (DLQ) handler.
- It can be run as a never-ending job with a specified time interval for rescheduling, be started externally at certain points of time, or be started by using the IBM WebSphere MQ trigger mechanism through a self-written trigger monitor.
- A user-written *rules table* supplies instructions to the DLQ handler utility, for processing messages on the DLQ. The table is a simple sequential file that can be edited with ISPF.
- One *control data* record defines the overall processing options for the utility, such as the name of the queue manager and the dead-letter queue, and the time values for the time-dependent functions.

An arbitrary number of *pattern match* keywords that are associated to a particular *action* specifies what to do with the messages on the dead-letter queue.

Dead-letter queue handler patterns and actions



© Copyright IBM Corporation 2015

Figure 9-14. Dead-letter queue handler patterns and actions

WM3021.1

Notes:

- The **WAIT** entry in the control data section specifies whether the DLQ handler should wait for further messages to arrive on the DLQ when it detects that there are no further messages that it can process. The valid options are provided here:
 - **WAIT(YES)**: The DLQ handler waits indefinitely.
 - **WAIT(NO)**: The DLQ handler terminates when it detects that the DLQ is either empty or contains no messages that it can process.
- WAIT(nnN)**: The DLQ handler waits for *nnN* seconds for new work to arrive after it detects that the queue is either empty or contains no messages that it can process before terminating.
- The pattern-matching keywords of a single rule can refer to one or more of the fields in the dead-letter header, or of the original message descriptor of the message, or both. The parameters can be specified with their full string or generically.
- Each entry specifies an **ACTION** to be taken by the utility, which can be:
 - **DISCARD**: Delete the message
 - **IGNORE**: Leave the message on the DLQ

- **RETRY**: Try to put the message on its original destination queue
- **FWD and FWDQ**: Put the message on the queue specified
- If more than one keyword is associated with the **ACTION**, they take effect as a logical **AND** condition.
 - **REASON(MQRC_Q_FULL)**
As the only parameter, it would select all messages that contain this reason code in their **MQDLH_REASON** field.
 - **REASON(MQRC_UNKNOWN_OBJECT_NAME) USERID(JAMES)**
This setting selects only those messages with RC=2085 in the **MQDLH_REASON** field that, at the same time, contain “JAMES” in their **MQMD_USERIDENTIFIER** context field.
 - **DESTQ(TARGET.*) REPLYQM(alpha*)**
This condition would select all messages that are destined for queue names starting with **TARGET** and reply-to queue manager (probably the originating QMGR) names starting with **alpha**.
- The DLQ handler utility matches every message on the DLQ against entries in the rules table, going through the table from top to bottom.

When a message matches an entry in the rules table, the utility performs the action that is associated with that entry. To take effect, the more distinctive rules must be coded in front of the common ones.
- To ensure that all messages are processed and that the DLQ handler really empties the dead-letter queue, an ACTION statement without a rule pattern should be coded at the end of the rule table.

Detailed documentation of the dead-letter queue handler utility and all its options with the rules syntax reference can be found in the *IBM WebSphere MQ for z/OS System Administration Guide*.

Unit summary

- Describe the use of IBM MQ event types
- List the queues that are associated with each event
- Summarize use of the dead-letter queue handler

© Copyright IBM Corporation 2015

Figure 9-15. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. The company audit needs to track any additions, deletions, or changes to IBM MQ channels in the production environment. How can this activity be tracked?
 - a. Instrumentation event
 - b. Command event
 - c. Configuration event
 - d. Queue manager event

2. True or False: To activate channel events for a critical production channel, the IBM MQ administrator issues the
ALTER CHANNEL(<channel name>) CHLEV(ENABLED)
command.

© Copyright IBM Corporation 2015

Figure 9-16. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.

- 2.

- 3.

- 4.

Checkpoint questions (2 of 2)

3. To be alerted when the number of messages in a queue is reaching a critical level, what actions does the IBM MQ administrator needs to take? (Select the correct answers.)
 - a. Sets the queue attributes to the required thresholds in the specific queue
 - b. Enable the required event in the specific queue
 - c. Enable queue events in the queue manager
 - d. Enable performance events in the queue manager
4. True or False: The event messages generated by IBM MQ are consumed by the IBM MQ event processor and displayed in the system log.

© Copyright IBM Corporation 2015

Figure 9-17. Checkpoint questions (2 of 2)

WM3021.1

Notes:



Checkpoint answers (1 of 2)

1. The company audit needs to track any additions, deletions, or changes to IBM MQ channels in the production environment. How can this activity be tracked?
 - a. Instrumentation event
 - b. Command event
 - c. Configuration event
 - d. Queue manager event

Answer: c, configuration event.

2. True or False: To activate channel events for a critical production channel, the IBM MQ administrator issues the
ALTER CHANNEL (<channel name>) CHLEV(ENABLED)
command.

Answer: False; alter the queue manager.

© Copyright IBM Corporation 2015

Figure 9-18. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. To be alerted when the number of messages in a queue is reaching a critical level, what actions does the IBM MQ administrator needs to take? (Select the correct answers.)
 - a. Sets the queue attributes to the required thresholds in the specific queue
 - b. Enable the required event in the specific queue
 - c. Enable queue events in the queue manager
 - d. Enable performance events in the queue manager

Answers: a, b, d. There are no event types called queue events.

4. True or False: The event messages generated by IBM MQ are consumed by the IBM MQ event processor and displayed in the system log.

Answer: False. The user is responsible for event processing and removal. There is no such thing as the MQ event processor.

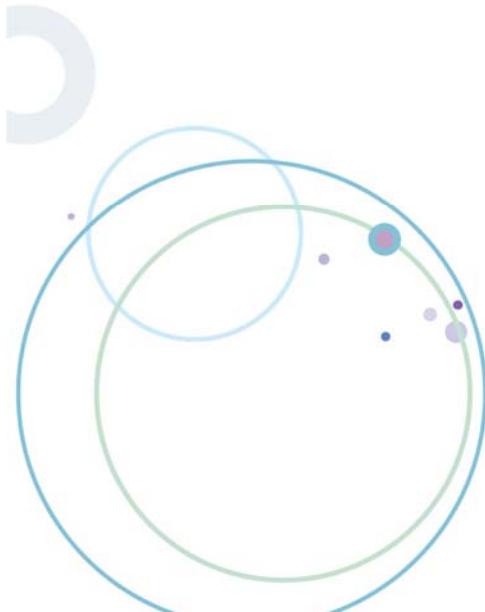
© Copyright IBM Corporation 2015

Figure 9-19. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 7



Working with IBM MQ events

© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 9-20. Exercise 7

WM3021.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Explain the uses of performance events
- Configure the queue manager and queues to generate events
- Use publish/subscribe to redirect event messages

© Copyright IBM Corporation 2015

Figure 9-21. Exercise objectives

WM3021.1

Notes:

Unit 10. Security considerations

What this unit is about

This unit identifies the security capabilities that are available with IBM MQ for z/OS and sets the base for designing security standards in the organization.

What you should be able to do

After completing this unit, you should be able to:

- Contrast security areas as they apply to IBM MQ
- Differentiate between link level and application level security
- Summarize the way security is implemented in IBM MQ for z/OS
- Describe how to implement connection authentication
- Describe how to use channel authentication records
- Summarize the mechanisms that are available to secure IBM MQ for z/OS



Unit objectives

- Contrast security areas as they apply to IBM MQ
- Differentiate between link level and application level security
- Summarize the way security is implemented in IBM MQ for z/OS
- Describe how to implement connection authentication
- Describe how to use channel authentication records
- Summarize the mechanisms that are available to secure IBM MQ for z/OS

© Copyright IBM Corporation 2015

Figure 10-1. Unit objectives

WM3021.1

Notes:

Terminology baseline: Security areas

- **Identification:** Determine the identity of a person that is using a system or resource
- **Authentication:** Proving the person or resource that is being identified is who they claim to be
- **Access control/authorization***: Limiting access to resources to only those people who need it
- **Confidentiality:** Protection of sensitive information
- **Data integrity:** Ability to detect tampering with critical data
- **Auditing:** Ability to determine whether any unauthorized access was done or attempted

Note: *The terms *access control* and *authorization* are used interchangeably.

© Copyright IBM Corporation 2015

Figure 10-2. Terminology baseline: Security areas

WM3021.1

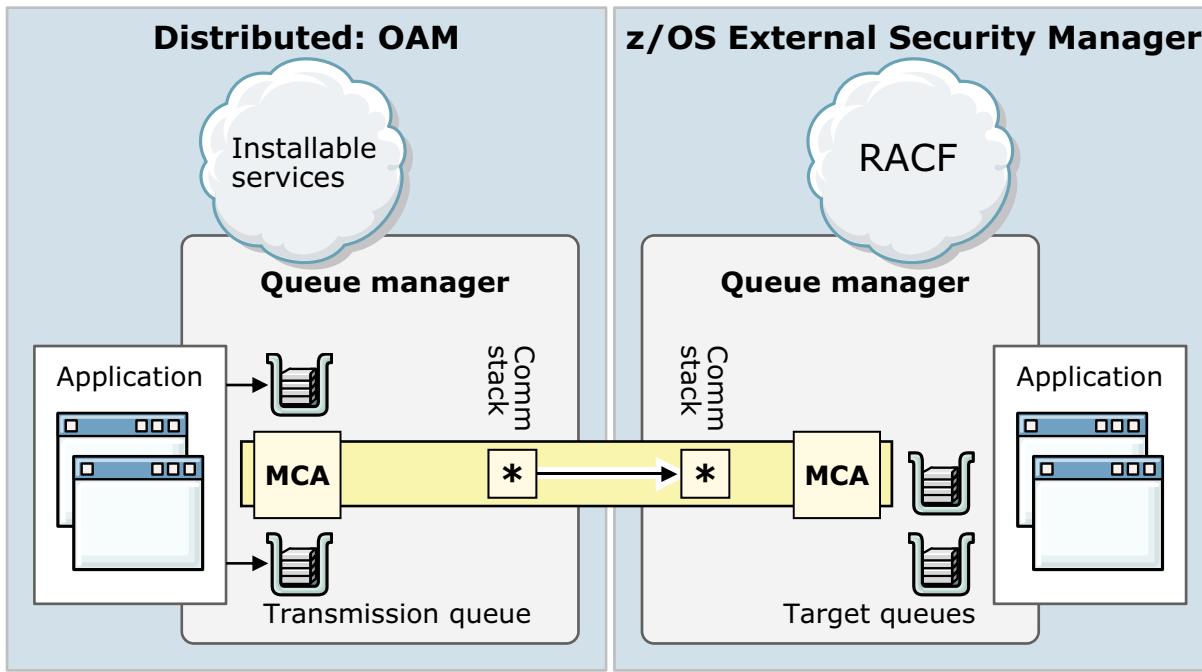
Notes:

You do a baseline of security terminology before proceeding.

Access control mechanisms

Two equivalent mechanisms:

- Secure administration
- Work with IBM MQ objects



© Copyright IBM Corporation 2015

Figure 10-3. Access control mechanisms

WM3021.1

Notes:

Access control for MQI calls is granted to users or groups with the OAM for distributed platforms, or External Security Manager (ESM) mechanisms for z/OS. For this course, the ESM is RACF.

Examples of access control are:

- Application issues an MQCONN; queue manager checks if the ID is allowed to connect to the queue manager.
- Application issues an MQOPEN; queue manager queries the OS for the ID associated with the application; queue manager checks if the ID is allowed to open the queue.

When an application issues an MQSUB, a security check is made to ensure that the application has rights to access the topic.

The ability to change to an alternative user ID needs to be authorized.

Security considerations for IBM MQ

- Architectural considerations
 - Application-level security
 - Link-level security
- IBM MQ library security
 - Installation data sets
 - Data sets used by IBM MQ for queuing and logging
- IBM MQ resources to be secured fall into three areas:
 - **Administer IBM MQ:** Create queue managers, queues, channels, administer security, work with IBM MQ libraries, and access log files
 - **Use of the MQI:** Connect to a channel, open a queue, and publish or subscribe to a topic
 - **Channel security:** Connect to a queue manager, open a transmit queue or target queues, and administer channel initiators and listeners

© Copyright IBM Corporation 2015

Figure 10-4. Security considerations for IBM MQ

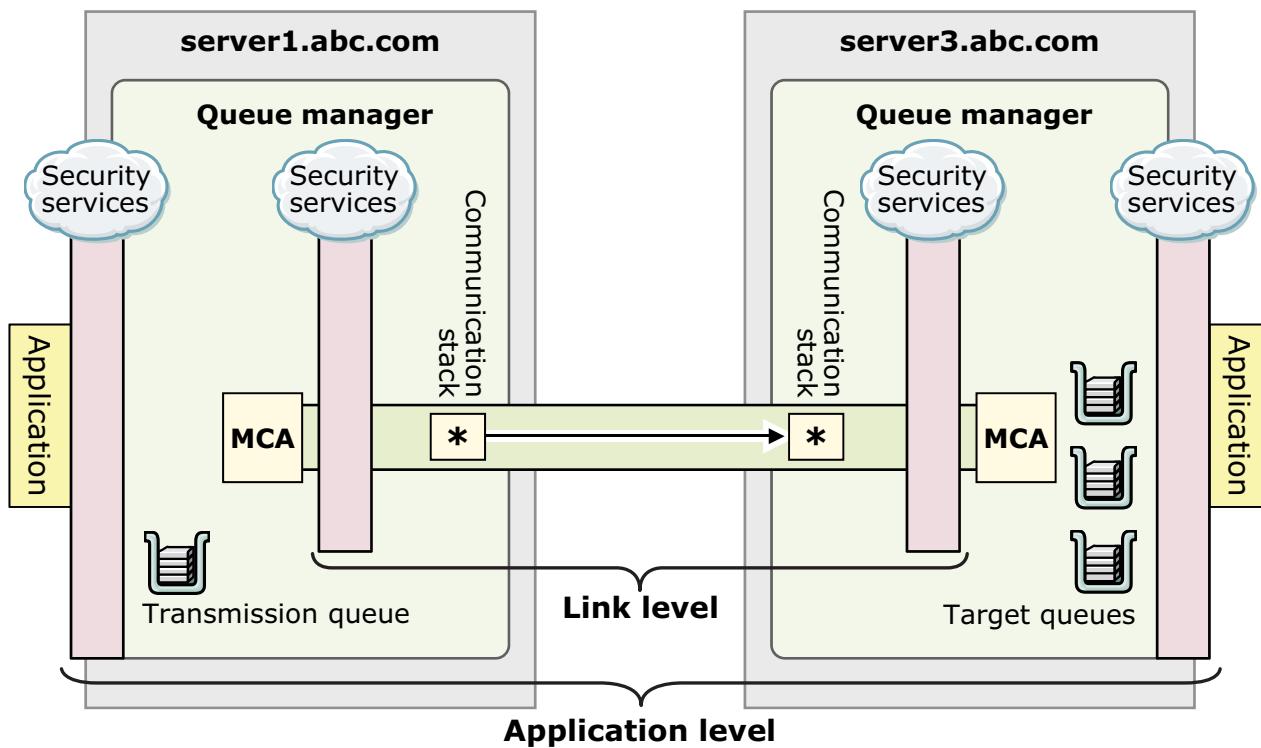
WM3021.1

Notes:

Security for IBM MQ on z/OS and distributed platforms can be interpreted differently depending on the work role. An architect might refer to application-level or link-level security. A systems programmer and IBM MQ administrator would consider the product and configuration data sets.

This unit looks at architectural considerations, data sets, and IBM MQ resources, such as queues, channels, logs, and the MQI application interface.

Link-level and application-level security



© Copyright IBM Corporation 2015

Figure 10-5. Link-level and application-level security

WM3021.1

Notes:

This visual shows how link-level security applies from MCA to MCA, while application-level security starts at the application boundary.

Access required for IBM MQ z/OS libraries

Library files	MQM s/sys	MQM admin	TSO/batch	ISPF panels	IMS/CICS	Application development
• thlqual.SCSQAUTH • thlqual.SCSQANLx	READ	READ	READ	READ	READ	
• thlqual.SCSQLINK • thlqual.SCSQCICS	READ				READ	
• thlqual.SCSQLOAD		READ				READ
• thlqual.SCSQMSGE/K • thlqual.SCSQPMLE/K • thlqual.SCSQTBLE/K • thlqual.SCSQEXEC • thlqual.SCSQPNLS				READ	READ	READ
• thlqual.SCSQPROC • thlqual.SCSQINST		READ	READ			

© Copyright IBM Corporation 2015

Figure 10-6. Access required for IBM MQ z/OS libraries

WM3021.1

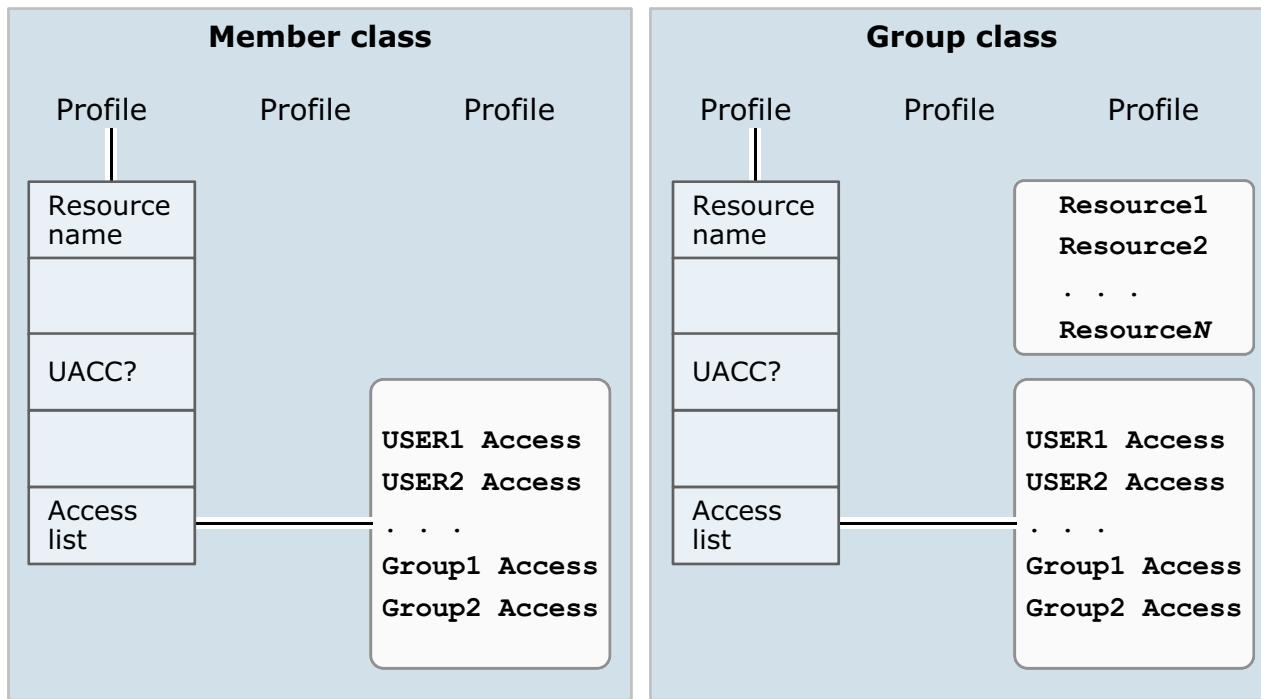
Notes:

This table summarizes the different kinds of authority that is needed by classes of user to the IBM MQ for z/OS resources that are set up at the installation and customization stages.

- As with other subsystems, IBM MQ application *users* do not need to have access to any system data sets.
In an IMS or CICS environment, only the *system user ids* that are associated with the IMS or CICS regions require (READ) access to the libraries referenced in their startup JCL.
- Access to the IBM MQ for z/OS command and control panels might be controlled by protecting the appropriate IBM WebSphere MQ ISPF libraries.

External Security Manager: RACF terminology

General resource classes



© Copyright IBM Corporation 2015

Figure 10-7. External Security Manager: RACF terminology

WM3021.1

Notes:

This figure shows some of the RACF terminology.

- General Resources: All but data sets
- User ids: Defined to RACF for each user
 - Group IDs, for simplicity
- Profiles: Identify resources
- Use of classes

Authorization: IBM MQ for z/OS resource classes

Class	Group class	Used for:
MQADMIN MXADMIN	GMQADMIN GMXADMIN	Administration function profiles
MQCONN		Connection security profiles
MQCMDS		Command security profiles
MQQUEUE MXQUEUE	GMQUEUE GMXQUEUE	Resource profiles: Queues
MQPROC MXPROC	GMQPROC GMXPROC	Resource profiles: Processes
MQNLIST MXNLIST	GMQNLIST GMXNLIST	Resource profiles: Namelists
MXTOPIC	GMXNTOPIC	Publish/Subscribe topics

© Copyright IBM Corporation 2015

Figure 10-8. Authorization: IBM MQ for z/OS resource classes

WM3021.1

Notes:

IBM WebSphere MQ for z/OS uses several resource classes to hold different types of profiles.

Classes are defined in the z/OS *class descriptor table* (CDT):

The queue manager parameter **SCYCASE** (Security Case) determines whether UPPER or MIXED case profiles are used by the queue manager:

- UPPER: Only uppercase profiles are used
- MIXED: Mixed case profiles are used, in addition to uppercase profiles

Changes to this parameter are only effective after a successful `REFRESH SECURITY(*) TYPE(CLASSES)` command or after the queue manager is recycled. This parameter is only valid on z/OS.

Profiles are always in uppercase.

There are some profiles, or parts of profiles, that are required by IBM WebSphere MQ to be in uppercase.

- All high-level qualifiers (HLQ) for subsystem identifiers and queue-sharing group identifiers
- Profiles for **SYSTEM** objects, including the default objects

- The **MQCMDS** class, as all command profiles are uppercase only
- The **MQCONN** class, as all connection profiles are uppercase only
- Switch profiles
- **RESLEVEL** profiles
- The command type in command resource profiles, for example, `hlq.QUEUE.queueusername`
- Dynamic queue profiles that are used by IBM WebSphere MQ administration tools, such as `hlq.CSQOREXX.*`, `hlq.CSQUTIL.*`, and `CSQXCMD.*`
- The “CONTEXT” part of the `hlq.CONTEXT.resourceName` profiles

Mixed case profiles

The following RACF commands illustrate the advantage of using mixed case profiles when mixed case object names exist in the queue manager.

The mixed case class **MXQUEUE** can be used to explicitly protect a queue that is called **PAYROLL.Dept1** on a queue manager called QMHQ:

```
RDEFINE MXQUEUE QMHQ.PAYROLL.Dept1
```

The uppercase class **MQQUEUE** can only be used to protect this queue by referring to the high-level qualifier in the queue name:

```
RDEFINE MQQUEUE QMHQ.PAYROLL.**
```

This profile also protects all other queues that begin with PAYROLL, which might be an undesirable situation for the queue manager security.

Authorization: MQADMIN class switch profiles

MQADMIN class switch profile name	Description
ssid.NO.SUBSYS.SECURITY	Turns OFF ALL security checking
ssid.NO.CONNECT.CHECKS	Turns OFF connection security
ssid.NO.QUEUE.CHECKS	Turns OFF queue resource checks
ssid.NO.PROCESS.CHECKS	Turns OFF process resource checks
ssid.NO.NAMELIST.CHECKS	Turns OFF namelist resource checks
ssid.NO.CONTEXT.CHECKS	Turns OFF context security
ssid.NO.ALTERNATE.USER.CHECKS	Turns OFF alternate user security
ssid.NO.CMD.CHECKS	Turns OFF command security
ssid.NO.CMD.RESC.CHECKS	Turns OFF command resource checks
ssid.NO.TOPICS.CHECKS	Turns OFF TOPICS security

© Copyright IBM Corporation 2015

Figure 10-9. Authorization: MQADMIN class switch profiles

WM3021.1

Notes:

- If ssid.NO.SUBSYS.SECURITY is defined, there is no checking
- During start, IBM MQ issues the following message:

```
CSQH021I +QM09 CSQHINSQ SUBSYSTEM security switch set off, profile  
'QM09.NO.SUBSYS.SECURITY' found
```

Otherwise, checking depends on the presence or absence of the other specific profiles, and an appropriate console message is issued for each of the individual sections and the switch profile that is associated with it.

Authorization: Connection security profiles

- The profile used in the MQCONN class for connection security checks depends on the environment in which the application runs:

Environment	Profile name	READ access required for:
Batch	ssid.BATCH	Batch and TSO user ID
CICS	ssid.CICS	CICS region user ID
IMS	ssid.IMS	IMS system and task user
MCA	ssid.CHIN	Channel initiator STC user ID

- For all types of connections, the same user ID is checked against the **ssid.RESLEVEL** profile in the MQADMIN class

© Copyright IBM Corporation 2015

Figure 10-10. Authorization: Connection security profiles

WM3021.1

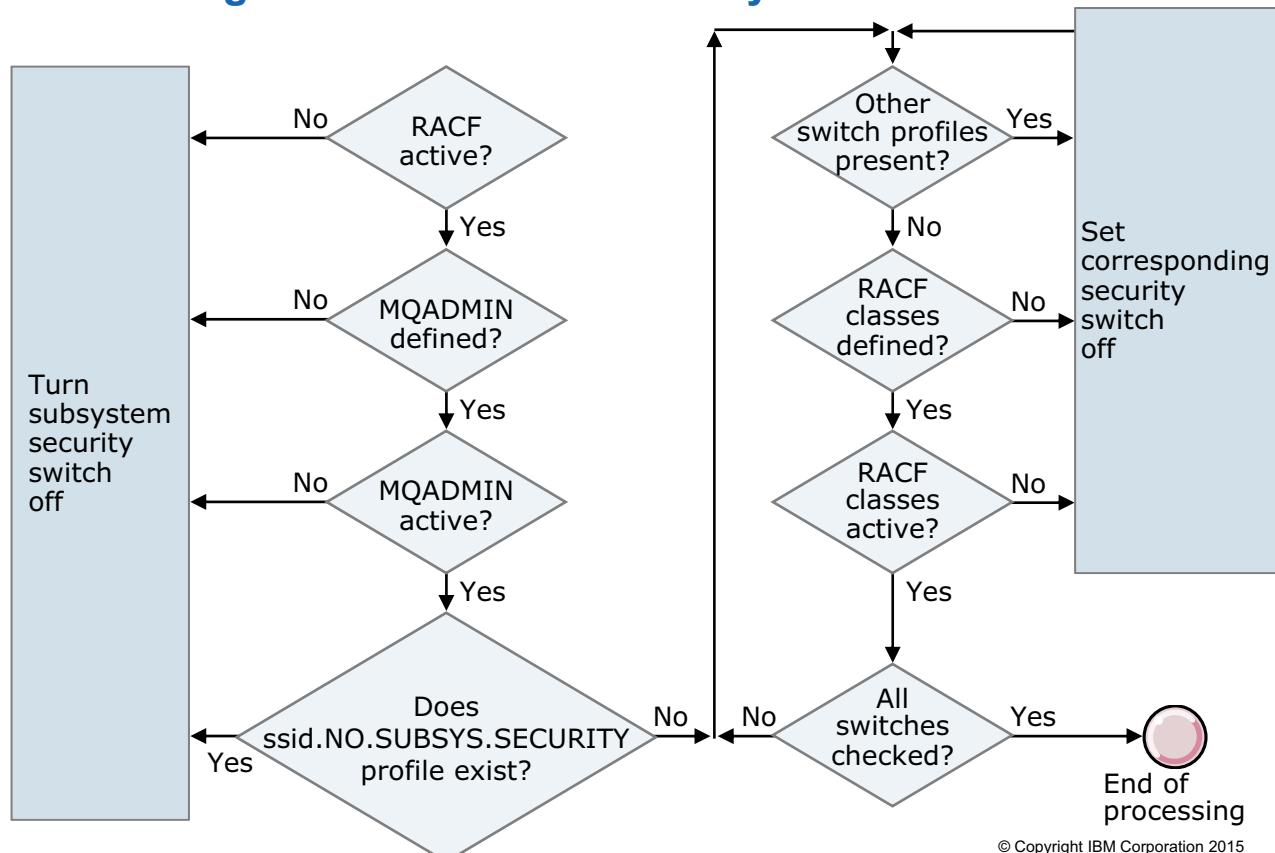
Notes:

- Based on the application environment from which the MQCONN call is issued, one of three profiles is checked against the application user ID
- Batch includes TSO
- MCA user IDs require access to the ssid.CHIN profile

As connection security must decide about whether to allow or reject the connect request, it is a matter of convention that IBM WebSphere MQ always checks for READ authorization, not higher.

WebSphere Education

Enabling IBM MQ for z/OS security



© Copyright IBM Corporation 2015

Figure 10-11. Enabling IBM MQ for z/OS security

WM3021.1

Notes:

IBM WebSphere MQ for z/OS subsystem security requires the following condition, at a minimum:

- RACF is active
- **MQADMIN** class is installed and activated
- If any of the earlier conditions are not present or active, *no security checking is performed* (the subsystem security switch is turned off)
- If all is true, *the subsystem security switch profile* determines whether a particular queue manager runs secured or not

IBM WebSphere MQ for z/OS then checks the security switches in the MQADMIN class to decide which types of security checks are performed.

Authorization: RESLEVEL profile

Userids checked?	NONE	READ	UPDATE	CONTROL/ALTER
TSO/job userid (or alternative)	Yes	Yes	Yes	No checking
CICS address space	Yes	Yes	Yes	No checking
CICS task userid (or alternative)	Yes	No	As RESSEC	No checking
IMS address space	Yes	Yes	Yes	No checking
IMS second userid (or alternative)	Yes	Yes	No	No checking

© Copyright IBM Corporation 2015

Figure 10-12. Authorization: RESLEVEL profile

WM3021.1

Notes:

A special security option allows control of the level of resource checking to be performed for a particular connection.

- On successful check of the connection profile, the same user ID is checked against the profile **ssid.RESLEVEL** in the **MQADMIN** class.
This process then sets the level of resource checking for all subsequent API calls on that connection that are affected by *resource security*, which means access to queues, processes, and namelists.
- CONTROL or ALTER access to the RESLEVEL profile exempts the appropriate user completely from resource checking.
- For CICS and IMS environments, the RESELVEL profile might be used to perform resource checking to the system (region) user IDs only, and not to the task user IDs. For CICS, the task user ID checking for resources might be based on the CICS option for resource security that is active for a particular task.
- RESLEVEL checking is activated by the connect security switch profile.

Authorization: Access levels for resource checks

Option	RACF access level
MQOPEN option	
MQOO_INQUIRE	READ
MQOO_BROWSE	READ
MQOO_INPUT_EXCLUSIVE	UPDATE
MQOO_INPUT_SHARED	UPDATE
MQOO_INPUT_AS_Q_DEF	UPDATE
MQOO_OUTPUT (and MQPUT1 calls)	UPDATE
MQOO_SET	ALTER
MQCLOSE option	
MQCO_DELETE	ALTER
MQCO_DELETE_PURGE	ALTER

© Copyright IBM Corporation 2015

Figure 10-13. Authorization: Access levels for resource checks

WM3021.1

Notes:

- All resource checks are carried out on the **MQOPEN** (or **MQPUT1**) call except when **MQCO_DELETE_*** is specified on **MQCLOSE** for a permanent dynamic queue. (**MQCO_DELETE_*** is only valid for permanent dynamic queues.)
- **MQOO_INQUIRE** is the only OPEN option that requires a security check for Processes and Namelists.

MQOO_OUTPUT is implied for all **MQPUT1** calls.

Resolving queue names on MQOPEN

- Queue name resolution performed at every queue manager
- MQOPEN performs a name resolution for the queue
- Local queue names resolve to local queue
- Alias queue names resolve to target queue
- Model queue names resolve to newly-created queue
- Remote queue names resolve to:
 - Destination queue
 - Remote queue manager
 - Transmission queue

Note: Security checks **only the name provided** on the MQOPEN.

© Copyright IBM Corporation 2015

Figure 10-14. Resolving queue names on MQOPEN

WM3021.1

Notes:

Queue name resolution was introduced in an earlier unit. You now revisit the topic.

- Queue name resolution occurs at every queue manager each time a queue is opened.
- Its purpose is to identify the target queue, target queue manager, and the route to the queue manager (if it is not local).
- When an application issues **MQOPEN** to gain access to a queue, the application must specify the name of the queue as the **ObjectName** in the Object Descriptor (MQOD).
- Normally, the **ObjectQMgrName** that might be coded in the MQOD is left blank.

In this case, IBM WebSphere MQ must know the named queue by a local definition; otherwise, **MQOPEN** completes with RC=2085.

- If the named queue is a *local queue*, the object handle points to this queue.
- If the named queue is an *alias queue*, the object handle points to the target queue specified in the QALIAS definition.
- If the named queue is a *model queue*, IBM WebSphere MQ creates a dynamic queue, which is based on the model definition, and the object handle points to this newly created dynamic queue.

- If the named queue is a *remote queue*, no alias definitions are referenced. The object handle points to name of the destination queue, the remote queue manager, and the transmission queue.

However, security access for an object is only checked against the name that is provided in the **MQOPEN**.

Authorization: Alias queue considerations

```

OPTIONS = MQOO_OUTPUT
MQOD_OBJECTNAME = 'REQ1A'
MQOPEN

DEFINE QALIAS (REQ1A)
TARGQ (APPL1.REQUEST)

```

```

OPTIONS = MQOO_OUTPUT
MQOD_OBJECTNAME = 'APPL1.REQUEST'
MQOPEN

DEFINE QLOCAL (APPL1.REQUEST)

```



QALIAS

RACF profile ssid.REQ1A
checked in class MQQUEUE

No check against
ssid.APPL1.REQUEST



QLOCAL

RACF profile ssid.APPL1.REQUEST
checked in class MQQUEUE

No check against
ssid.REQ1A

© Copyright IBM Corporation 2015

Figure 10-15. Authorization: Alias queue considerations

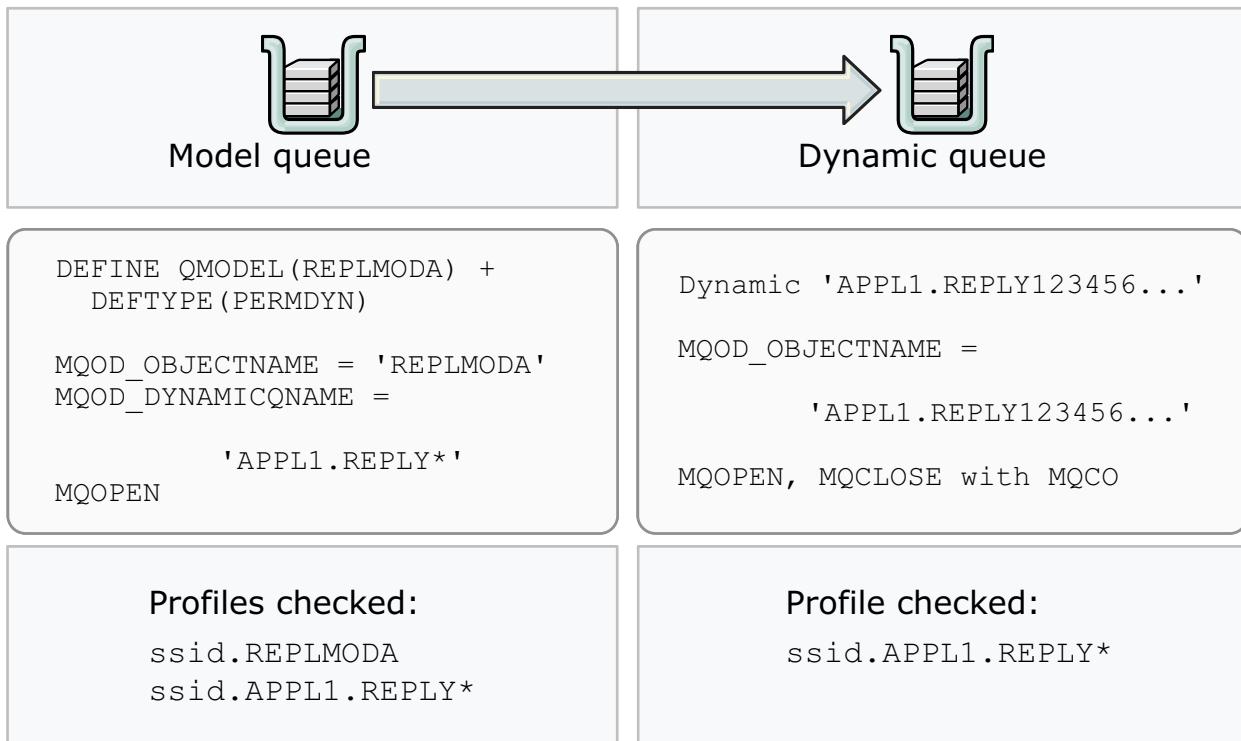
WM3021.1

Notes:

Security checking of *alias* queues is performed on the alias name only. The *target* queue is not checked.

Remember to restrict who can define or alter alias queues.

Authorization: Model queues and dynamic queues



© Copyright IBM Corporation 2015

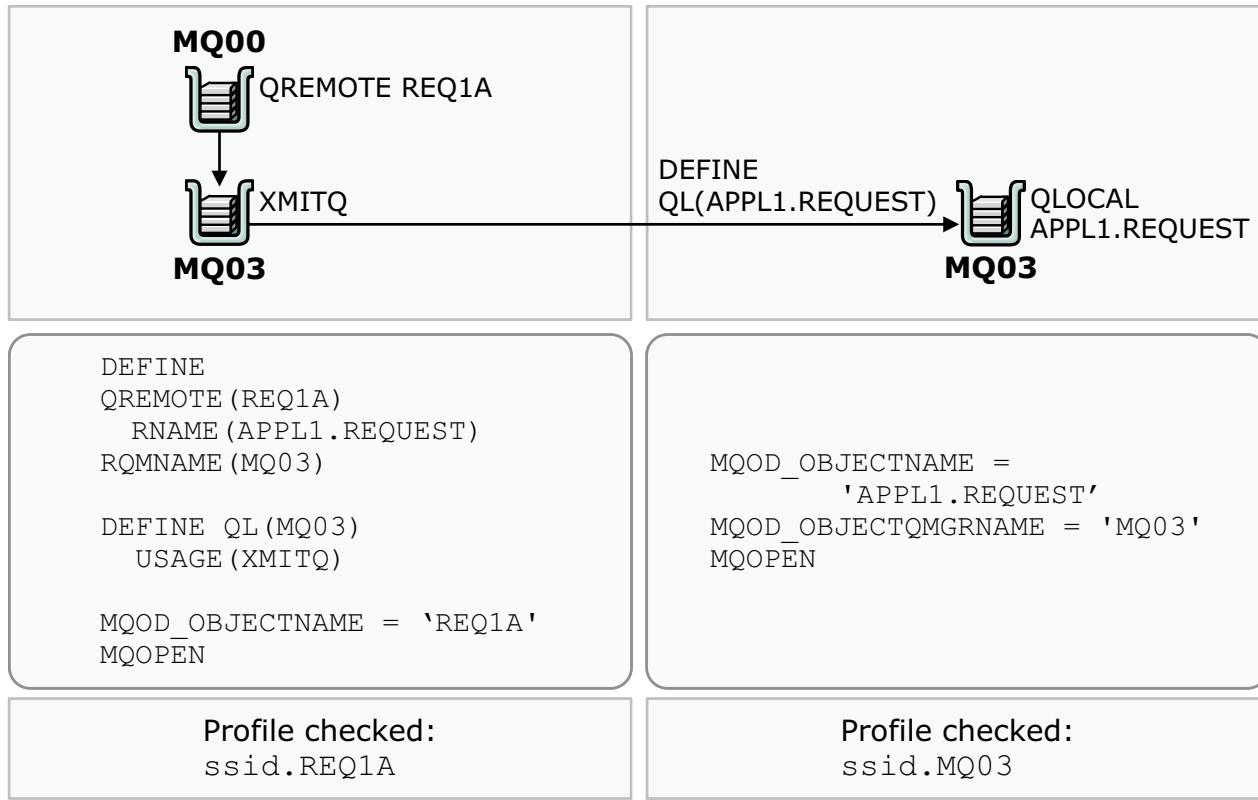
Figure 10-16. Authorization: Model queues and dynamic queues

WM3021.1

Notes:

- When a *model queue* is opened, access checks are made against the model queue name, and the resolved dynamic queue name.
- UPDATE access is required for both names.
Although the dynamic queue is newly created as part of the **MQOPEN** processing, only UPDATE (not ALTER) permission is required at **MQOPEN** time.
- If the dynamic queue name specified by the application contains a trailing asterisk (*), an appropriate generic profile must be defined.
- Processes that access the dynamic queue, typically, the servers that put responses to the reply-to queue, are checked against the queue name they use on **MQOPEN**.
- Permanent dynamic queues are also checked at **CLOSE** time if **MQCO_DELETE** or **MQCO_DELETE_PURGE** is specified, which requires ALTER access to the appropriate queue name.

Authorization: Controlling access to remote queues



© Copyright IBM Corporation 2015

Figure 10-17. Authorization: Controlling access to remote queues

WM3021.1

Notes:

If the remote queue being opened was defined with a `DEFINE QREMOTE` command, checking is limited to the locally defined queue name and not the target queue (as for *alias queues*).

If the **ObjectQMgrName** is specified in the object descriptor and does not resolve to the local queue manager, the queue name that is checked is that of the transmission queue of the same name.

If the **ObjectQMgrName** is specified and is an alias of the local queue manager, the alias name is checked and *not* the queue manager name.

Overview of context data

Message descriptor context fields	
Identity context	Origin context
UserIdentifier	PutAppIType
Accounting Token	PutAppName
ApplIdentityData	PutDate
	PutTime
	ApplOriginData

© Copyright IBM Corporation 2015

Figure 10-18. Overview of context data

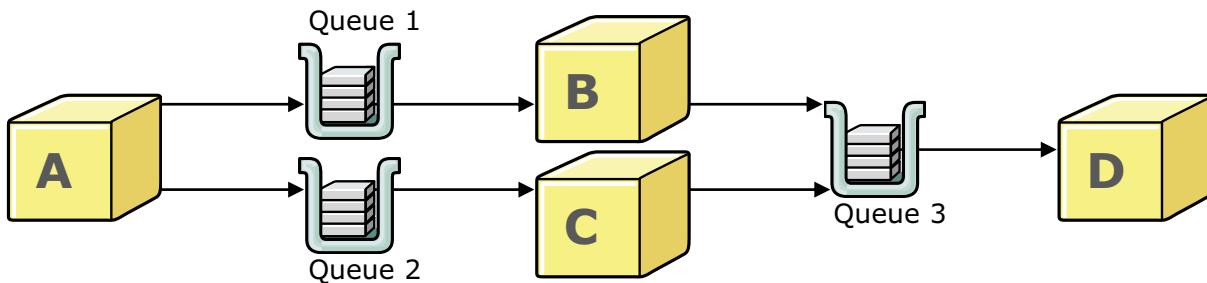
WM3021.1

Notes:

Context data consists of two parts:

- *Identity context* gives information about the *owner* of the message, which normally is the user who first created the message and put it on a queue.
- *Origin context* gives information about the application from which this message originated.

How context data is used



- Information about the source of a message:
 - Identity section: User-related
 - Origin section: Program-related
- Part of Message Descriptor
- Can be passed in related message

© Copyright IBM Corporation 2015

Figure 10-19. How context data is used

WM3021.1

Notes:

When used, the *identity* portion will normally be associated with the user or application, which created the message that is read by the first (or only) process making up an application, and passed to the next process (if any) in the outgoing message.

The *origin* portion will normally be associated with the application that actually puts this message on the queue (or last *changed* it when moving it to the *next* queue).

Authorization: Context options and permissions required

`MQPUT (MQPUT1) option` `MQOPEN option` ————— Permission required to
ssid.CONTEXT.qname in MQADMIN

- Retrieve message context on MQGET and pass it unchanged on a subsequent MQPUT:

<code>MQPMO_PASS_IDENTITY_CONTEXT</code>	—————	No check
<code>MQOO_SAVE_ALL_CONTEXT</code>	—————	READ
<code>MQPMO_PASS_ALL_CONTEXT</code>	—————	No check
<code>MQOO_SAVE_ALL_CONTEXT</code>	—————	READ

- Specify context data directly in the MQMD structure on MQPUT1

<code>MQPMO_SET_IDENTITY_CONTEXT</code>	—————	UPDATE
<code>MQOO_SET_IDENTITY_CONTEXT</code>	—————	
<code>MQPMO_SET_ALL_CONTEXT</code>	—————	CONTROL
<code>MQOO_SET_ALL_CONTEXT</code>	—————	

© Copyright IBM Corporation 2015

Figure 10-20. Authorization: Context options and permissions required

WM3021.1

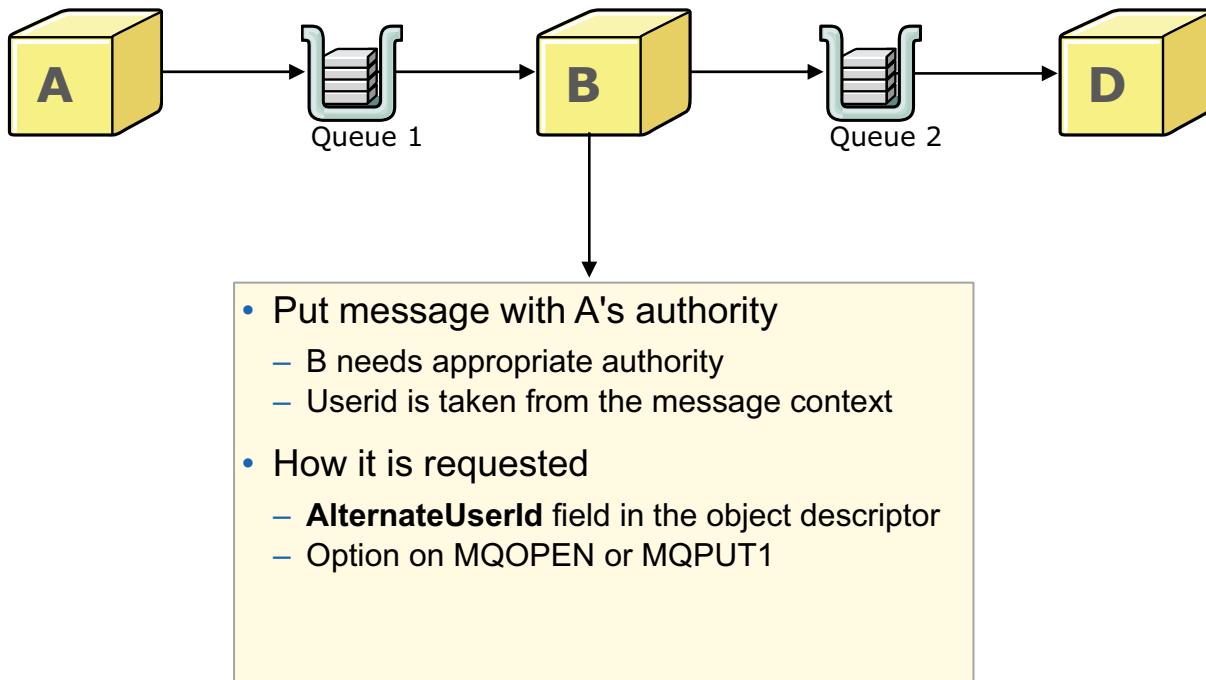
Notes:

A separate security switch profile controls the context handling security check. Context manipulation can be secured at the queue name level.

- `MQPMO_DEFAULT_CONTEXT` must be specified explicitly if required. No special authorization is required for this option.
 - The first time a message is **PUT**, specify `MQPMO_DEFAULT_CONTEXT` if you have any security active, or use context for auditing, and other tasks.
 - It must be specified when writing to the **SYSTEM.COMMAND.INPUT** queue if command or command resource security is active.
- `MQPMO_PASS_IDENTITY_CONTEXT` and `MQPMO_PASS_ALL_CONTEXT` require a higher security authorization, so the corresponding *open* option must be set.
 - `MQOO_SAVE_ALL_CONTEXT` must be specified when opening the *input* queue from where context data is passed. The handle for this queue is then specified in the context field of the *put message options*.

Only the context of the last message (for each input handle) is saved.

Authorization: Use of alternate user security



© Copyright IBM Corporation 2015

Figure 10-21. Authorization: Use of alternate user security

WM3021.1

Notes:

- *Alternate user authority* allows one user, if given the authority, to open a queue (or other objects for inquiry) by using a different user ID. The user then has the same access to resources as the alternate user ID.
- The alternate user ID is used for resource checking. For batch and TSO users, this ID is the only user ID checked. For CICS and IMS users, it replaces the second (task) user ID (the first user ID being the user ID for the system).

The object must be opened with option `MQOO_ALTERNATE_USER_AUTHORITY` and the alternate user ID provided in the **AlternateUserId** field of the object descriptor.

Authorization: RACF access for an alternate user ID

Profiles for an alternate user authority:

- MQOPEN (or MQPUT1) options:
 - MQOO_ALTERNATE_USER_AUTHORITY
 - (MQPMO_ALTERNATE_USER_AUTHORITY for MQPUT1)
- Require UPDATE access to profile in MQADMIN class:
 - ssid.ALTERNATE.USER.alternateuserid
 - or
 - ssid.ALTERNATE.USER.*

© Copyright IBM Corporation 2015

Figure 10-22. Authorization: RACF access for an alternate user ID

WM3021.1

Notes:

- Alternate user authority checks are performed independently of the normal resource checks against the specific object
- The original user ID is used for the alternate user security check, and any queue-independent context checks if applicable
- To use the alternate user, a user must have UPDATE authority to a profile in MQADMIN class of the form: ssid.ALTERNATE.USER.alternateuserid, where alternateuserid is the user ID specified in the object descriptor



Reminder

Remember the specific switch profile that would disable this check.

Authorization: Checking IBM MQ commands

Origin of the command	Userid checked:
Initialization input data sets CSQINP1 or CSQINP2	No security checks performed
SYSTEM.COMMAND.INPUT queue	Useridentifier from MQMD identity context field
MVS Console	User ID that is signed in, or CSQOPR
SDSF/TSO Console	TSO or job userid
CSQUTIL utility program	Job userid

© Copyright IBM Corporation 2015

Figure 10-23. Authorization: Checking IBM MQ commands

WM3021.1

Notes:

Commands that are put on the **SYSTEM.COMMAND.INPUT** queue should have the option **MQPMO_DEFAULT_CONTEXT** set to ensure that the user ID is included in the Message Descriptor if command security is activated.

The CSQUTIL batch utility program also works in the manner described.

Authority to issue commands

Command	Profile in MQADMIN class	Access level
ALTER pkw	ssid.ALTER.pkw	
DEFINE pkw	ssid.DEFINE.pkw	ALTER
DELETE pkw	ssid.DELETE.pkw	
DISPLAY pkw	ssid.DISPLAY.pkw	READ
MOVE QLOCAL	ssid.MOVE.QLOCAL	ALTER
ARCHIVE LOG	ssid.ARCHIVE.LOG	CONTROL
RECOVER BSDS	ssid.RECOVER.BSDS	CONTROL
REFRESH CLUSTER/QMGR/SECURITY	ssid.REFRESH.*	ALTER
RESET CHANNEL/CLUSTER/...	ssid.RESET.*	CONTROL
RESOLVE CHANNEL/INDOUBT	ssid.RESOLVE.*	CONTROL
START pkw	ssid.START.pkw	
STOP pkw	ssid.STOP.pkw	CONTROL

© Copyright IBM Corporation 2015

Figure 10-24. Authority to issue commands

WM3021.1

Notes:

Profiles for checking authority to issue *all* IBM WebSphere MQ for z/OS commands are held in the **MQCMDS** class and are of the general form: ssid.verb.pkw

Where:

- **verb:** Such as DEFINE, DISPLAY, ARCHIVE
- **pkw:** Primary keyword, or the keyword following the *verb*, which normally specifies the type of resource that is affected by the command such as QLOCAL, NAMELIST, or LOG

Access to command resources

Command	Profile in MQADMIN class	Access level
ALTER NAMELIST DEFINE NAMELIST DELETE NAMELIST	ssid.NAMELIST.namelist	ALTER
ALTER PROCESS DEFINE PROCESS DELETE PROCESS	ssid.PROCESS.process	ALTER
ALTER QALIAS etc. DEFINE QALIAS etc. DELETE QALIAS etc.	ssid.QUEUE.queue	ALTER
DISPLAY NAMELIST DISPLAY PROCESS DISPLAY QUEUE	No command resource checks	

© Copyright IBM Corporation 2015

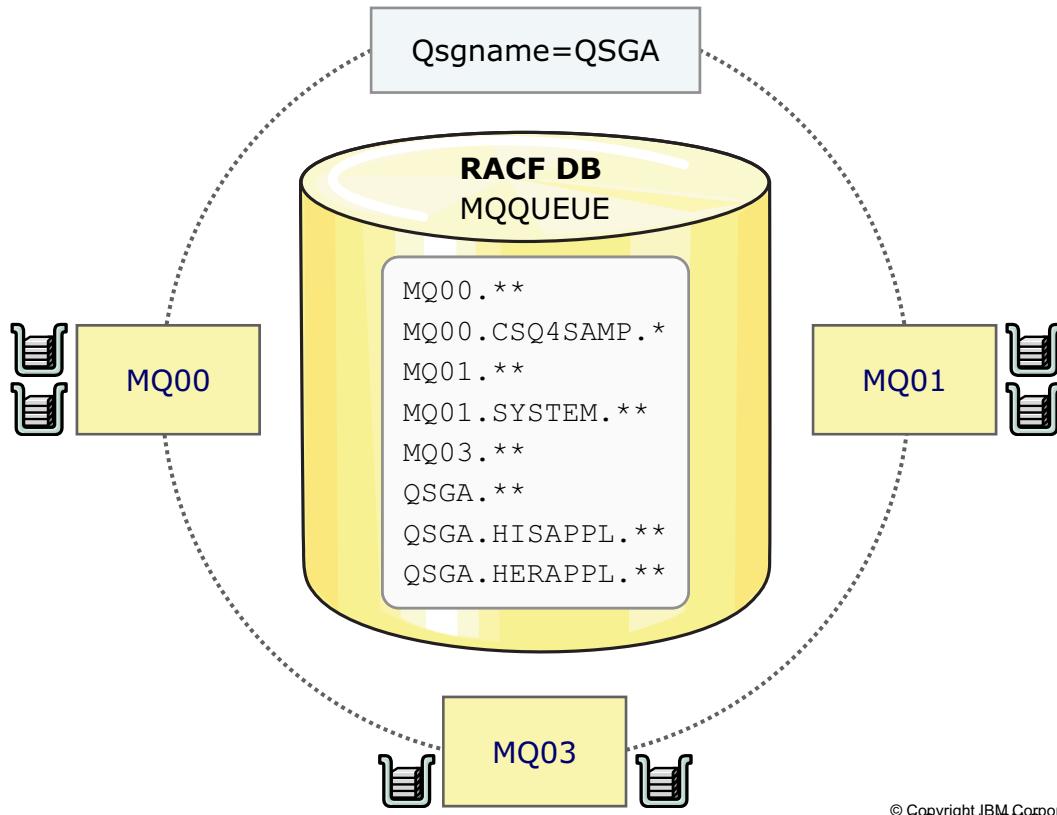
Figure 10-25. Access to command resources

WM3021.1

Notes:

- If you require to control the use of commands that are based on the real resources' names, you should consider *resource command security*, which is a separate security switch option.
- Profiles for command resource checks are held in the **MQADMIN** class and are of the general form:
 - ssid.QUEUE.queuename
 - ssid.CHANNEL.chlname
- Refer to the section on profiles for command security in the *IBM WebSphere MQ for z/OS System Setup Guide* for a comprehensive table of all possible resource command security options for all commands and all types of resources.

Queue-sharing group level security



© Copyright IBM Corporation 2015

Figure 10-26. Queue-sharing group level security

WM3021.1

Notes:

- Security can be implemented at the queue-sharing group level, so that all the queue managers in the group share profiles.
This means that there are fewer profiles to define and maintain, making security management easier.
- The QSG name must be used as the prefix for profile names.
- Queue-sharing group level security profiles can be used to protect all types of resource, whether local or shared.

Determine the level of security

MQADMIN class profile name	Description
qmgr-name.NO.QMGR.CHECKS	No QMGR level checks for this queue manager
qsg-name.NO.QMGR.CHECKS	No QMGR level checks for this queue-sharing group
qmgr-name.YES.QMGR.CHECKS	Queue manager level checks override for this queue manager
qmgr-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue manager
qsg-name.NO.QSG.CHECKS	No queue-sharing group level checks for this queue-sharing group
qmgr-name.YES.QSG.CHECKS	Queue-sharing group level checks override for this queue manager

© Copyright IBM Corporation 2015

Figure 10-27. Determine the level of security

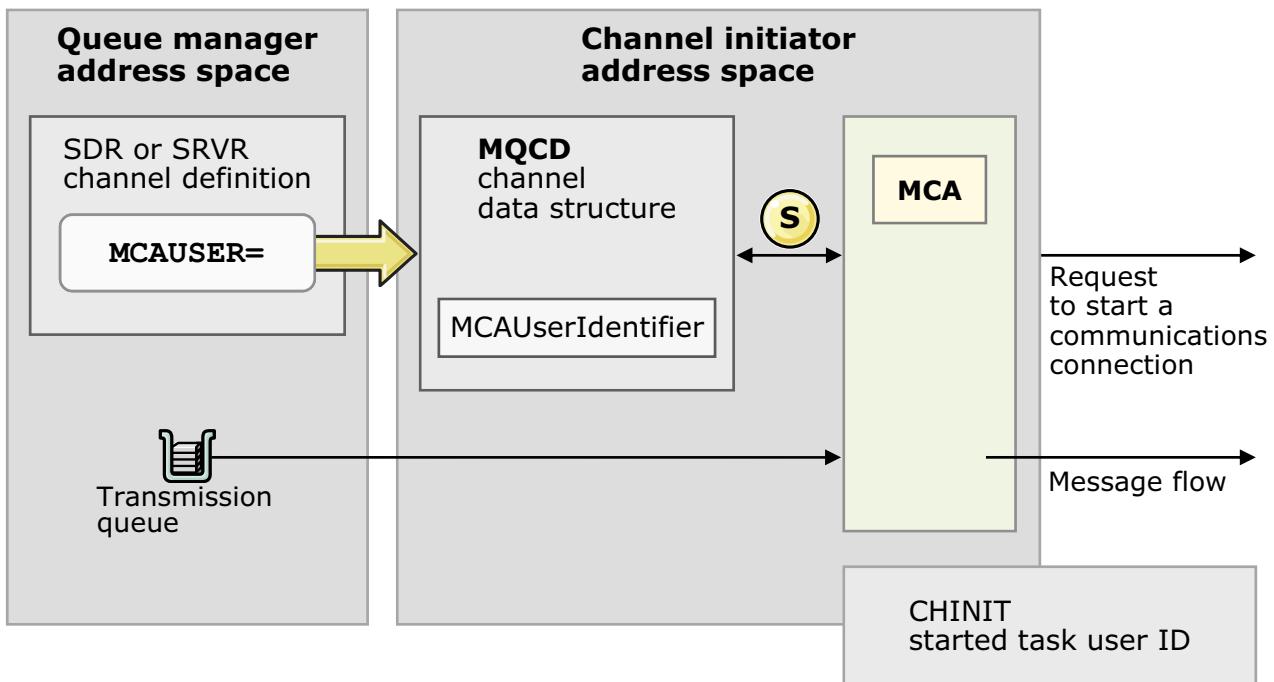
WM3021.1

Notes:

- For members of a queue-sharing group, all security checks might be performed at the following levels:
 - At the queue manager level
 - At the queue-sharing group level
 - At both levels or not at all
- In addition to the profiles shown in the figure, the subsystem security is controlled in the same way:
 - qmgr-name.NO.SUBSYS.SECURITY: Switches off security for this queue manager
 - qsg-name.NO.SUBSYS.SECURITY: Switches off security for this queue-sharing group
 - qmgr-name.YES.SUBSYS.SECURITY: Overrides the subsystem security switch for this queue manager

This principle is applicable to all switch profiles. This means that, for example, context security and alternate user security might be enabled or disabled either at the queue manager level or at the queue-sharing group level.

Securing transmission queues



Legend



= Security exit SSL option

© Copyright IBM Corporation 2015

Figure 10-28. Securing transmission queues

WM3021.1

Notes:

- The internal tasks in the channel initiator address space that make use of queue manager services regularly connect (**MQCONN**) to the queue manager at CHINIT start time, with the channel initiator address space (started task) user ID being the one that is checked against the `ssid.CHIN` profile in the **MQCONN** resource class.
 - All accesses to DQM system queues, such as **SYSTEM.CHANNEL.INITQ** or the **SYSTEM.CLUSTER.REPOSITORY.QUEUE**, are performed under authority of the channel initiator started task user ID.
 - For most of the queue accesses, UPDATE authority is sufficient. For some queues, whose attributes might be changed, ALTER permission is required.
- The user ID checked for UPDATE permission when a sending MCA opens a transmission queue is determined by the **MCAUSER** parameter of the channel definition:
 - If **MCAUSER** is specified, then this user ID is used to check the access authority.
 - If none is specified, then the channel initiator started task user ID is checked.

The channel initiator started task user ID requires ALTER permission for all transmission queues, as it might change its attributes during operation.

Securing the remote queue access

- LU6.2: The CHINIT process Userid of the remote QMGR
- TCP/IP using SSL: Userid associated with a digital certificate

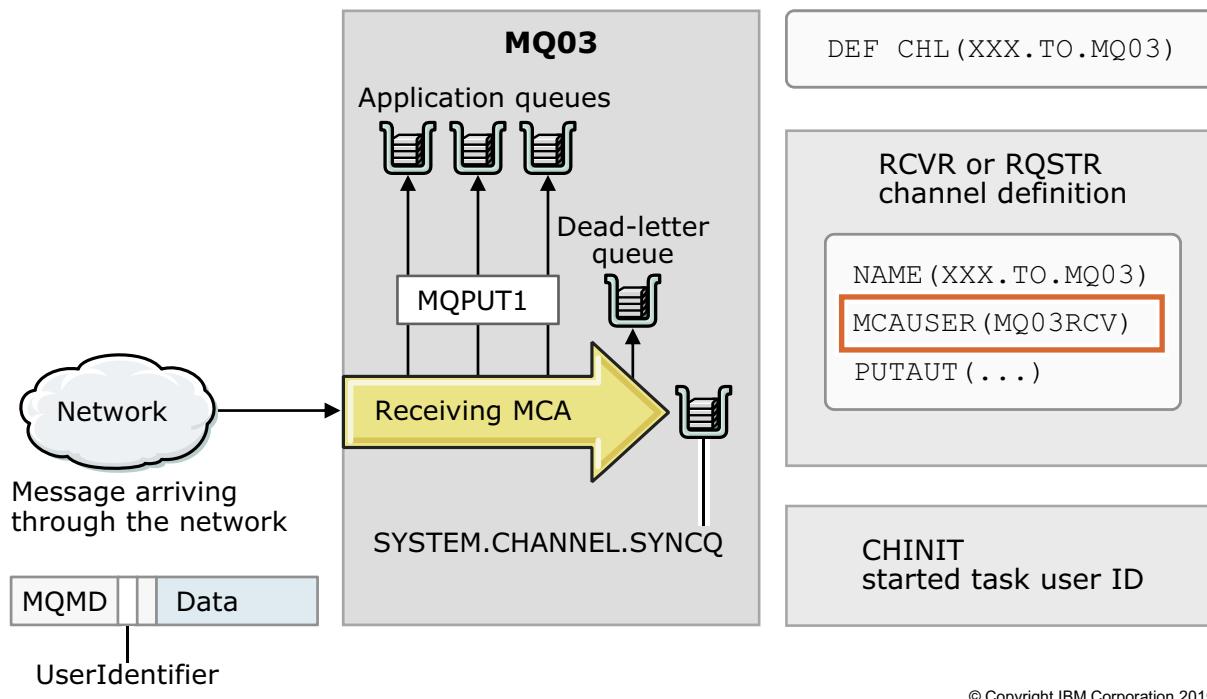


Figure 10-29. Securing the remote queue access

WM3021.1

Notes:

The user ID that is checked for writing a message to the destination queue after its transmission through the network depends on the following criteria:

- The type of the receiving channel (whether receiver or requester)
- The network type, whether TCP/IP or LU62
- The definition for receiving channel, namely the specifications for the **MCAUSER** and the **PUTAUT** parameters
- The RESLEVEL permission level that is defined for the channel initiator user ID
- Three user IDs might be involved in the access to destination queues:
 - The *MCA user*, which is the user ID specified in the **MCAUSER** channel definition parameter, for the receiver or the requester. If the parameter is not used, it defaults to the channel initiator started task user ID.

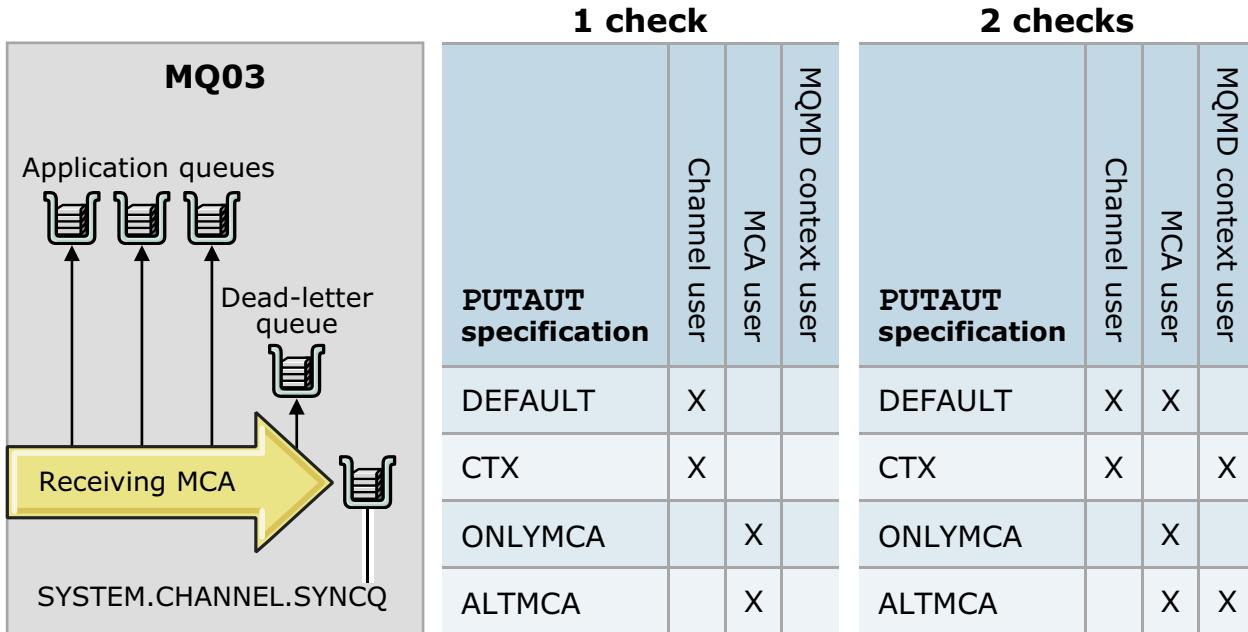
Any channel without an **MCAUSER** value allows user impersonation and anonymous administrative authority.

- Security exits might implement any arbitrary authentication at the link level. The user ID is then placed by the exit into the **MCAUSER** field of the channel where it is used for API authorization.

- The *channel user*, which:
 - For LU62 receivers, is the user ID flowed across the network: usually *the remote channel initiator started task user ID*.
 - For TCP/IP channels that use SSL, it is the user ID that is associated with the partner Digital Certificate.
 - For TCP/IP channels do not use SSL and LU62 requesters, it is the *local* channel initiator started task user ID.
- The *AlternateUser*, which is the user ID contained in the MQMD context field (**MQMD-USERIDENTIFIER**) for the incoming message.
- Understand the two-fold use offered by SSL:
 - Based on the digital certificate of the partner and the SSL-related channel definitions, the start channel (attach) request that is issued by the partner might be honored or rejected.

If the channel is started, the user ID derived from the digital certificate of the partner might be checked when the destination queues are being opened for message delivery.

User IDs to be checked



© Copyright IBM Corporation 2015

Figure 10-30. User IDs to be checked

WM3021.1

Notes:

When an API-resource security check is made by the channel initiator, one or two user IDs are checked to see if access is allowed to the resource, depending on the access level that is granted to the ssid.RESLEVEL profile for the one user ID that acts as the primary user for the particular channel.

Use this method to read the table that is shown in the figure, based on the option that is provided by the **PUTAUT** parameter in the requester and receiver channel definition:

- DEFAULT, referred to as **PROCESS** by the ISPF panels:
 - The *channel user ID* is the primary user ID checked for access to the destination queue.
 - If the RESLEVEL profile results in two checks, then the *MCA user ID* is also checked.
 - Accesses to all destination queues and for all messages are checked for the same user IDs, which normally are system user IDs, not the individual user ids.
- CTX, meaning **context**:
 - The *channel user ID* is the primary user ID checked for access to the destination queue.
 - If the RESLEVEL profile results in two checks, then the MCA takes the user ID from the **UserIdentifier** context field of the incoming messages and uses the

MQOO_ALTERNATE_USER_AUTHORITY option for opening the destination queues. Thus, this user ID is also checked.

- In this last case, the individual user ids that created the incoming messages at some remote location must be authorized for access to the destination queues.
- ONLYMCA
 - Regardless of the RESLEVEL settings, only the *MCA user ID* is checked for access to the destination queue.
- ALTMCA
 - Behaves just like the CTX option, but with the *MCA user ID* used as the primary ID instead of the channel user ID.



Reminder

Remember that you can specify a security exit routine to perform further validation before any messages are exchanged, by using a conversation with a *matching* exit at the remote location.

Security controls

- Update in-storage profiles per class:

```
REFRESH SECURITY MQADMIN | MQCMDS | MXQUEUE...
```

- Update in-storage profiles per user:

```
RVERIFY SECURITY UID1,UID2,...
```

- To define system-wide security options, such as the behavior for **rverify**:

```
ALTER SECURITY TIMEOUT(720) INTERVAL(1440)
```

© Copyright IBM Corporation 2015

Figure 10-31. Security controls

WM3021.1

Notes:

IBM WebSphere MQ for z/OS provides security-related commands:

- REFRESH SECURITY: To cause the deletion or refresh of in-storage RACF profiles for which IBM WebSphere MQ for z/OS keeps a cache, either for a particular class or for all eligible classes.
- RVERIFY SECURITY: To cause user-related profiles, which are cached as well, to be deleted and refreshed on a per-user base. The user is reverified the next time that security is checked for that user.
- ALTER SECURITY: To define system-wide security options.

When a user accesses an IBM WebSphere MQ resource, the queue manager authenticates this user to RACF and caches their profiles. At specified intervals, the table of signed-on users is checked. Users that have been inactive for a longer period than specified by the TIMEOUT keyword are discarded from the table, thus being *signed off*. The next time that this user requests service again, the appropriate profiles must be reloaded.

If the default values are not suitable, the ALTER SECURITY command might be stored in the CSQINP1 initialization commands to always guarantee the setting that is required.

Time values are in minutes.

Authentication: Security exits

- IBM MQ provides exit points to extend the functionality of IBM MQ for messages and channels
- IBM MQ security exits:
 - Main objective is to allow MCA to authenticate its partner
 - Is called after the initial data negotiation but before messages flow
 - Work in pairs at the sending and receiving of queue managers
- The link-edited exits names are:
 - Placed in the CSQXLIB DD statement of the channel initiator procedure
 - Included in the channel definition.
- z/OS exit examples:
 - CSQ4BAX0: Assembler
 - CSQ4BCX1 and CSQ4BCX2: C language, and shows parameter access
 - CSQ4BCX3 and CSQ4BAX3: C and assembler



Consider the use of CONNAUTH and CHLAUTH before attempting to implement a security exit.

© Copyright IBM Corporation 2015

Figure 10-32. Authentication: Security exits

WM3021.1

Notes:

Security exits involve user written code that should be considered only as a last resort. After the introduction of CHLAUTH and CONNAUTH, the use of a security exit should not be necessary in most cases.



Note

This slide on security exits was included for completeness only. It does not imply that the use of an exit is warranted particularly with the availability of configurable alternatives that do not require development or maintenance.

Security inquiry

/MQ00 DIS SECURITY

```
CSQH015I MQ00 Security timeout = 54 minutes
CSQH016I MQ00 Security interval = 12 minutes
CSQH037I MQ00 Security using uppercase classes
CSQH030I MQ00 Security switches ...
CSQH034I MQ00 SUBSYSTEM: ON, 'MQ00.NO.SUBSYS.SECURITY' not found
CSQH034I MQ00 CONNECTION: ON, 'MQ00.NO.CONNECT.CHECKS' not found
CSQH034I MQ00 COMMAND: ON, 'MQ00.NO.CMD.CHECKS' not found
CSQH034I MQ00 CONTEXT: ON, 'MQ00.NO.CONTEXT.CHECKS' not found
CSQH034I MQ00 ALTERNATE USER: ON, 'MQ00.NO.ALTERNATE.USER.CHECKS'
not found
CSQH031I MQ00 PROCESS: OFF, 'MQ00.NO.PROCESS.CHECKS' found
CSQH031I MQ00 NAMELIST: OFF, 'MQ00.NO.NLIST.CHECKS' found
CSQH034I MQ00 QUEUE: ON, 'MQ00.NO.QUEUE.CHECKS' not found
CSQH034I MQ00 TOPIC: ON, 'MQ00.NO.TOPIC.CHECKS' not found
CSQH031I MQ00 COMMAND RESOURCES: OFF, 'MQ00.NO.CMD.RESC.CHECKS'
found
CSQH040I MQ00 Connection authentication ...
CSQH041I MQ00 Client checks: OPTIONAL
CSQH042I MQ00 Local bindings checks: OPTIONAL
CSQ9022I MQ00 CSQHPDTC 'DIS SECURITY' NORMAL COMPLETION
```

© Copyright IBM Corporation 2015

Figure 10-33. Security inquiry

WM3021.1

Notes:

The DISPLAY SECURITY command displays the setting of the security switches, and optionally, the time settings for internal profile administration.

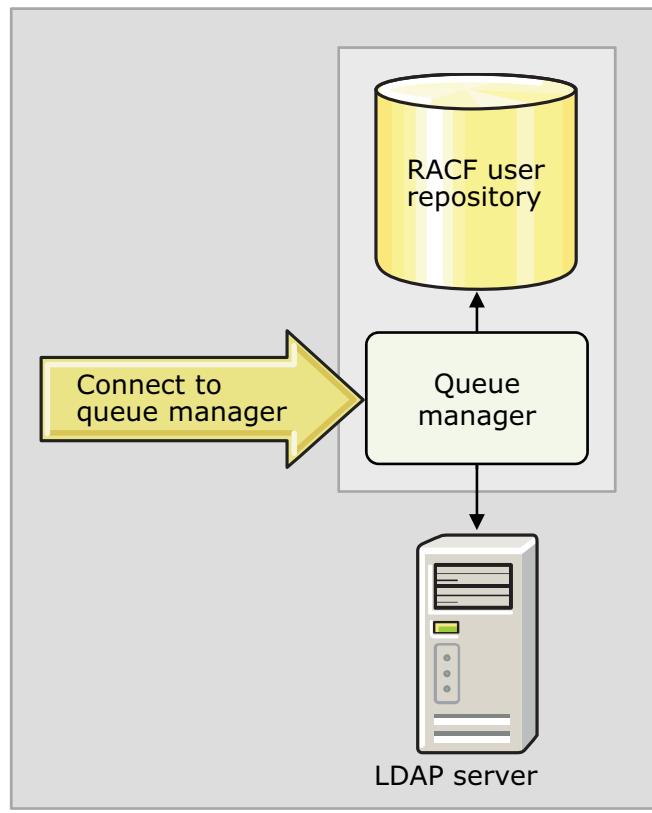
This command is also available in:

- ISPF: Select **option 1 (display)** for object type SECURITY.
- IBM MQ Explorer: Right-click the z/OS queue manager name and then click **Configuration > Security**.

The last four lines in the display show information on how connection authentication is set for this queue manager.

Link-level security: Connection authentication

- Queue manager attribute CONNAUTH
- Two new types of authentication objects:
 - OS repository
 - LDAP
- Defaults
 - New queue managers default to using OS system authentication
 - Migrated queue managers have CONNAUTH disabled
- Use of MQSAMP_USER_ID environment variable with sample programs sets ID and allows password entry
- On the application side:
 - Code change is required to add security structure
 - Option to use mqccred exit



© Copyright IBM Corporation 2015

Figure 10-34. Link-level security: Connection authentication

WM3021.1

Notes:

IBM MQ V8 introduced the Connection Authentication feature, which allows applications to provide a user ID and password that the queue manager can validate before allowing an application to connect. On z/OS, you can have your password validated by RACF, or the External Security Manager in use.

When working with IBM MQ applications in server or client bindings on the distributed side, it is possible to test the CONNAUTH settings with the usual IBM MQ sample programs.

- amqsput, amqsputc, amqsget, amqsgetc, amqsbcg, amqsbcgc:

Use the environment variable to set the user before running

```
set MQSAMP_USER_ID=youruserID
```

You are prompted for a password.

- New connectivity test sample amqscnxc:

Use -u to provide the ID. The program then prompts for password.

```
amqscnxc -x server1.ibm.com(1625) -c WM302.SVRCONN -u yourUserID MQ00
```

- JMS: JmsProducer and JmsConsumer have `-u` and `-w` flags added so they can be used.

```
set CLASSPATH= mqinst\java\lib\com.ibm.mqjms.jar;mqinst\tools\jms\samples
```

where `mqinst` is the location of your IBM MQ installation

```
java JmsProducer -m MQ00 -d TESTCNQ -u yourUserID -w y0ourpwd
```

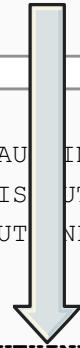
These programs are test programs only.

CONNAUTH settings: Defaults

```
/MQ00 DIS QMGR CONNAUTH
CSQM201I MQ00 CSQMDRTC  DIS QMGR DETAILS
QMNAME (MQ00)
CONNAUTH (SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
END QMGR DETAILS
```

MQ00 DIS AUTHINFO(*) ALL

```
CSQM293I MQ00 CSQMDRTC 3 AU INFO FOUND MATCHING
CSQM201I MQ00 CSQMDRTC  DIS UTHINFO DETAILS 090
AUTHINFO (SYSTEM.DEFAULT.AUTHINFO.CRLLDAP)
AUTHTYPE (CRLLDAP)
...
AUTHINFO (SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
AUTHTYPE (IDPWOS)
QSGDISP (QMGR)
ADOPTCTX (NO)
CHCKCLNT (OPTIONAL)
CHCKLOCL (OPTIONAL)
FAILDELAY (1)
```



© Copyright IBM Corporation 2015

Figure 10-35. CONNAUTH settings: Defaults

WM3021.1

Notes:

When a queue manager is created, the default is to have CONNAUTH enabled with optional authentication for remote (CHCKCLNT) or local(CHCKLOCL) users.

The display shows how to issue DIS QMGR CONNAUTH to determine the connection authentication setting for the queue manager. This information is used to check the AUTHINFO record that is associated with the connection authentication record to determine the CHCKCLNT and CHCKLOCL attribute settings.

However, the optional setting means that if no user id is provided, it continues without authenticating. If a user ID is provided, then it must have the correct password or the authentication fails.

If you must disable CONNAUTH, use the following commands:

```
ALTER QMGR CONNAUTH( )
REFRESH SECURITY TYPE (CONNAUTH)
```

However, it would be in the best interest of the organization to include CONNAUTH in the security plans.

CHCKCLNT and CHCKLOCL values (1 of 3)

- **NONE:**
Switches off checking
- **OPTIONAL:**
If an ID is provided, the ID and password must be correct
- **REQUIRED:**
All applications must provide valid credentials

- CHCKCLNT=OPTIONAL, amqsputc without ID

```
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
target queue is ORDERS.AT.MQ00
xyz
Sample AMQSPUTO end
```

```
+CSQX511I MQ00 CSQXRESP Channel MQ00.CL
started 178
connection 10.4.127.184
```

- CHCKCLNT=OPTIONAL, amqsputc with ID but a bad password

```
C:\>set MQSAMP_USER_ID=INGMUSR
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
Enter password: abc123
MQCONNX ended with reason code 2035
```

```
ICH408I USER(INGMUSR ) GROUP(SYS1 ) NAME(WM30 INSTRUCTOR ) 20
LOGON/JOB INITIATION - INVALID PASSWORD
```

© Copyright IBM Corporation 2015

Figure 10-36. CHCKCLNT and CHCKLOCL values (1 of 3)

WM3021.1

Notes:

This set of slides display the outcome of the possible CHCKCLNT and CHCKLOCL attribute values for the selected scenarios.

- NONE means no checking. This scenario is skipped in the test displays.
- OPTIONAL allows connections if no passwords are used, but requires a valid password if an ID is supplied.
- REQUIRED requires all applications to provide credentials.

The slides show the distributed (Windows) side of the test, followed by the outcome on the z/OS log.

When the `MQSAMP_USER_ID` variable is set, the user ID is passed to the program.

This first test uses CHCKCLNT set to optional.

- If you test without an ID, the test works as expected.
- If you provide an ID but no password or an invalid password, you receive a 2035 security error in the client side, and a RACF error in the z/OS log.

CHCKCLNT and CHCKLOCL values (2 of 3)

- CHCKCLNT=REQUIRED, amqsputc without credentials

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) +
    CHCKLOCL(OPTIONAL) CHCKCLNT(REQUIRED)
REFRESH SECURITY TYPE(CONNAUTH)
...
CSQH040I MQ00 Connection authentication ...
CSQH041I MQ00 Client checks: REQUIRED
CSQH042I MQ00 Local bindings checks: OPTIONAL
```

```
C:>set MQSAMP_USER_ID=
C:>amqsputc ORDERS.AT.MQ00
Sample AMQSPUTO start
MQCONNX ended with reason code 2035
```

```
+CSQX511I MQ00 CSQXRESP Channel MQ00.CL started 232
connection 10.4.127.184
CSQH045E MQ00 CSQHNSIG MQ00.CL/10.4.127.184 did not provide a password
+CSQX512I MQ00 CSQXRESP Channel MQ00.CL no longer active
```

© Copyright IBM Corporation 2015

Figure 10-37. CHCKCLNT and CHCKLOCL values (2 of 3)

WM3021.1

Notes:

On the z/OS side, you now set CHCKCLNT to REQUIRED, and refresh security so the change takes effect.

On the client side, the user ID is cleared the test by setting **MQSAMP_USER_ID** to blanks.

The test without sending a user ID now fails with a 2035 in the client side.

In the z/OS log, the error message indicates the error.

CHCKCLNT and CHCKLOCL values (3 of 3)

- CHCKCLNT=REQUIRED, amqsputc with valid credentials

```
C:\>set MQSAMP_USER_ID=INGMUSR
C:\>amqsputc ORDERS.AT.MQ00
Sample AMQSPUT0 start
Enter password: VALIDPWD
target queue is ORDERS.AT.MQ00
Check message that the application now works
Sample AMQSPUT0 end
```

```
+CSQX511I MQ00 CSQXRESP Channel MQ00.CL started 235
connection 10.4.127.184
+CSQX512I MQ00 CSQXRESP Channel MQ00.CL no longer active
... ... ... ... ISPF Option H ... ... ...
----- IBM WebSphere MQ for z/OS -- Sample Programs --- Row 1 to 3 of 3
Queue Manager      : MQ00
Queue             : ORDERS.AT.MQ00
Message number    01 of 03          Total Msgs : 000
  Msg  Put Date   Put Time Format     User       Put Application
  01  10/09/2014  21:28:23 MQSTR    NOMAD789  00000011 C:\mqv8\bin64\amqs
  02  10/09/2014  23:57:20 MQSTR    NOMAD789  00000011 C:\mqv8\bin64\amqs
... ... ... ...
```

© Copyright IBM Corporation 2015

Figure 10-38. CHCKCLNT and CHCKLOCL values (3 of 3)

WM3021.1

Notes:

On the client side, the user ID is set up for the test.

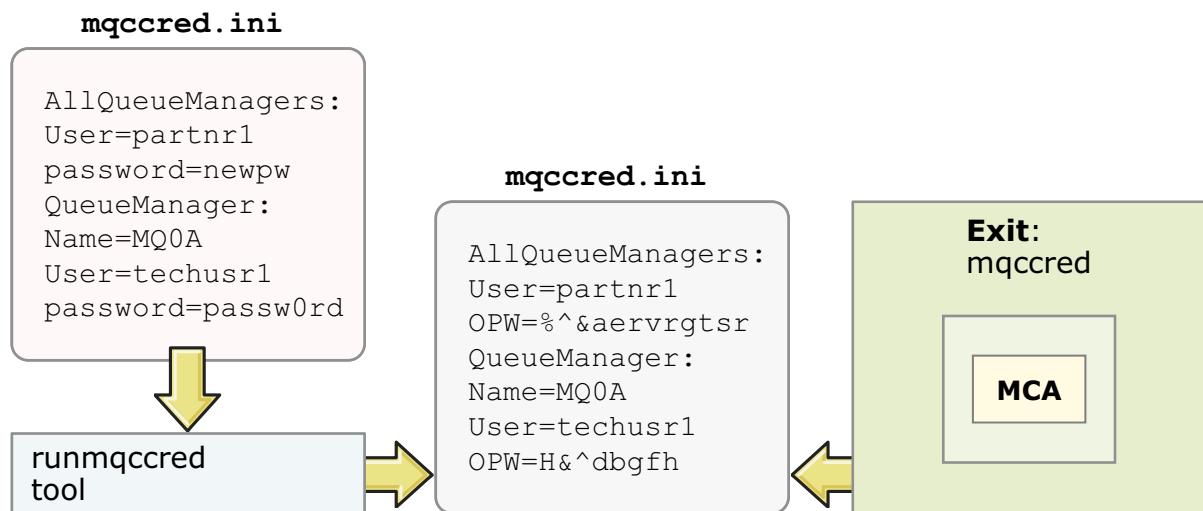
When amqsputc starts, it requests a password.

On the z/OS side, you can see the channel start, then end without error when the message is received.

You can use the message handler program at ISPF option H in the lab environment to see the messages.

For testing, the password is being sent in the clear, however, for production the passwords need to be obfuscated.

Client-side security exit: Distributed client side



- New file for distributed client platforms: `mccred.ini`
- IBM MQ V8 clients use the `runmqccred` tool to obfuscate the password
- Exit can be imported to the CLNTCONN side V7.0.1 or later
- If the queue manager or client is earlier than V8, the password is clear text

© Copyright IBM Corporation 2015

Figure 10-39. Client-side security exit: Distributed client side

WM3021.1

Notes:

To mitigate the time it takes applications to update their code to include name and password exchange, the distributed platforms provide:

- The `mqccred` exit that can be installed to process the ID and password
- The `runmqccred` tool to obfuscate the password

While the exit can be installed starting with IBM MQ version 7.0.1 or later queue managers, the ability to obfuscate the password is only available to IBM MQ V8 clients and queue managers.

Link-level security: Identification and authentication

Channel authentication records:



A security mechanism that enables setting rules to determine which connections are allowed and that are excluded from accessing the queue manager.

- Rules can be set to allow or ban connections, privileged users, or a combination of attributes
- Precedence of rules is important and can be listed with the **DIS CHLAUTH** option
- Rules can be set to use different details of the incoming connection, such as an IP address or a remote queue manager name

© Copyright IBM Corporation 2015

Figure 10-40. Link-level security: Identification and authentication

WM3021.1

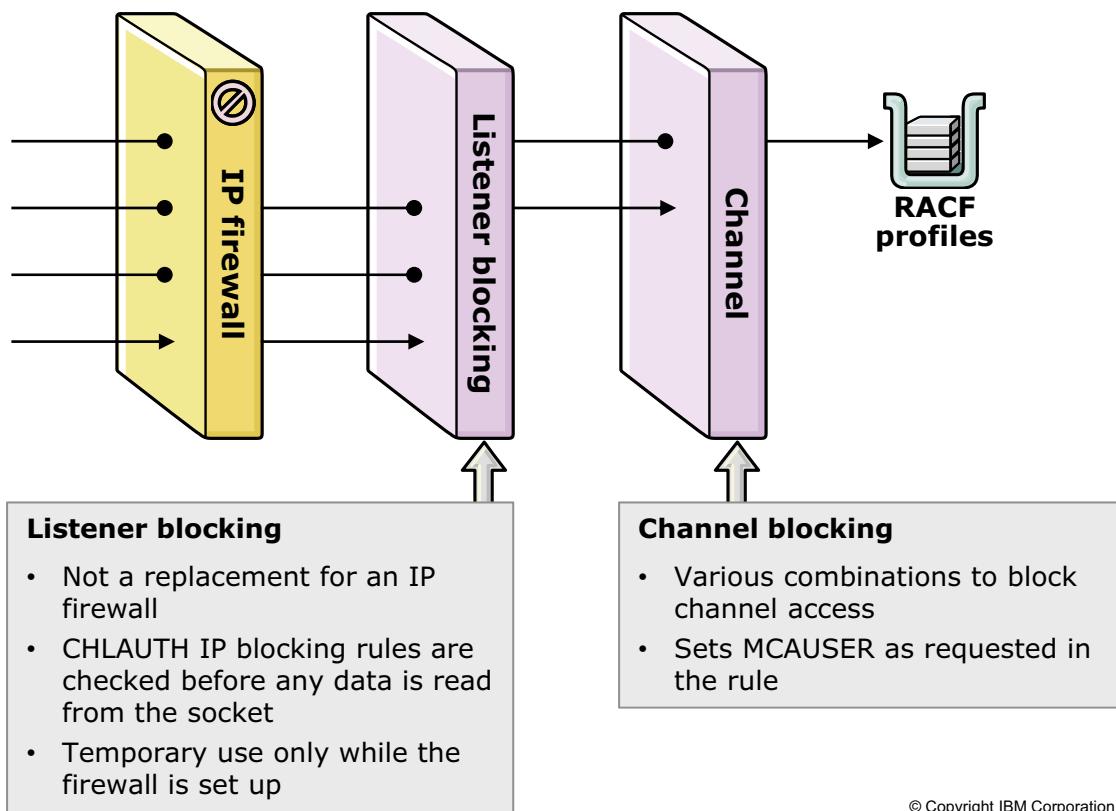
Notes:

A queue manager is faced with numerous inbound connections such as other IBM MQ server applications, IBM MQ client applications, and IBM MQ Explorer user. While most of these connections are valid, you need a way to block out certain connections based on where they come from.

Channel Authentication records provide a way to define rules as to what connections can access the queue manager, and what should be done with the connection.

Rules can be set for different types of connections and can be based on IP address, host name, SSL details, and user ID. The actions that are taken by a rule might be to block access, or to map the inbound user ID to another user ID.

Channel blocking points



© Copyright IBM Corporation 2015

Figure 10-41. Channel blocking points

WM3021.1

Notes:

An incoming connection must clear several points before getting to a queue.

- The IP firewall should be at the forefront of these connections.
- The next point that can be blocked is the listener. Listener blocking should not be a replacement for a firewall, but it is possible to use CHLAUTH to block at the listener as a temporary measure while the firewall rules are in place.
- Listener blocking is accomplished with a BLOCKADDR type of CHLAUTH rule. The different rule types are presented later in this unit.
- After clearing the listener, CHLAUTH might be used at the channel level. The outcome of the CHLAUTH:
 - Can block a connection, or
 - Behave in a positive manner such as when the rule is used to map a user id and allow the connection

Replacing and removing CHLAUTH rules

Key fields						
Channel name	Type	Values for type	Address (restrictor)	Source of ID	Mapped ID	User ID and password
SET CHLAUTH(APP1.*) TYPE(SSLPEERMAP) SSLPEER('CN=OrdApp') ADDRESS('1.2.3.4')					USERSRC(MAP)	MCAUSER('Ordmgr')
SET CHLAUTH(ADMIN.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*.ibm.com')					USERSRC(CHANNEL)	CHKCLNT(REQUIRED)
<pre>SET CHLAUTH('APP1.*') TYPE(SSLPEERMAP) SSLPEER('CN=OrdApp') ADDRESS('1.2.3.4') ACTION(REMOVE) SET CHLAUTH(ADMIN.SVRCONN) TYPE(ADDRESSMAP) ADDRESS('*.ibm.com') USERSRC(MAP) MCAUSER('IBMUSER') ACTION(REPLACE) SET CHLAUTH('APP1.*') TYPE(SSLPEERMAP) ACTION(REMOVEALL)</pre>						

All the key fields

© Copyright IBM Corporation 2015

Figure 10-42. Replacing and removing CHLAUTH rules

WM3021.1

Notes:

CHLAUTH records are “SET” with an ACTION attribute to either ADD, REPLACE, REMOVE, or REMOVEALL”.

When you start working with CHLAUTH rules, there are subtleties to be aware of:

- ACTION(ADD) is the default.
- When using ACTION(REPLACE) and ACTION(REMOVE), it is critical to use all required key fields to qualify the change or removal. If all the required fields are not used, the expected behavior is that the rule is added instead because a rule matching the attributes was not found.
- When a rule gets inadvertently added with an ACTION(REMOVE) and now you have two rules with a similar name. Sometimes, the best route is to issue an ACTION(REMOVEALL) by using a partial wildcard name for the rules, then add the rule as intended.



Note

Be careful to use quotation marks, particularly for IDs, wildcards, and host names.

CHLAUTH types

CHLAUTH type	Description	Required attribute
BLOCKUSER	Prevents specified users from connecting	USERLIST
BLOCKADDR	<ul style="list-style-type: none"> • Prevents connections from specified IPs • Works at the listener level 	ADDRLIST
SSLPEERMAP	Maps SSL/TLS distinguished names (DNs) to MCAUSER values	SSLPEER
ADDRESSMAP	Maps IP addresses to MCAUSER values	ADDRESS
USERMAP	Maps user IDs to MCAUSER values	CLNTUSER
QMGRMAP	Maps remote queue manager names to MCAUSER values	QMNAME

© Copyright IBM Corporation 2015

Figure 10-43. CHLAUTH types

WM3021.1

Notes:

This slide shows the different types of CHLAUTH rules that might be created. The records take attributes other than the required attributes. Due to the different functions performed, some of these records also require other parameters. There are different interactions among some of the CHLAUTH records and its attributes. For that reason, it is good to consult the IBM MQ Knowledge Center when creating CHLAUTH rules, or after discovering a rule resulted in an error. CHLAUTH does provide good feedback for most errors that are committed when creating rules.

Some rules also work different in distributed platforms than on z/OS. For instance, on distributed platforms it is possible to override a CONNAUTH setting by using a CHLAUTH type ADDRESSMAP with CHCKCLNT, such as:

```
SET CHLAUTH( 'CONNECT' ) TYPE(ADDRESSMAP) ADDRESS( '192.167.3.4' ) USERSRC(CHANNEL)
CHCKCLNT(ASQMGR) ACTION(ADD)
```

In z/OS, use of CHCKCLNT and CHCKLOCL with CHLAUTH is not valid.

CHLAUTH rules using IP addresses and host names

Rules	Using IP address	Using host names
Listener blocking list	SET CHLAUTH('*') + TYPE(BLOCKADDR) + ADDRESS('10.5*', '192.168.2.3') + USERSRC(NOACCESS)	Not allowed
Channel-based blocking	SET CHLAUTH('MQ00.CL.*') + TYPE(ADDRESSMAP) + ADDRESS('10.5*', '192.168.2.3') + USERSRC(NOACCESS)	SET CHLAUTH('MQ00.CL.*') + TYPE(ADDRESSMAP) + ADDRESS('*.ibm.com') + USERSRC(NOACCESS)
Channel allowed	SET CHLAUTH('* .SVRCONN') + TYPE(ADDRESSMAP) + ADDRESS('10.5-6*') + MCAUSER(INGUSR)	SET CHLAUTH('* .SVRCONN') + TYPE(ADDRESSMAP) + ADDRESS('ac.ibm.com') + MCAUSER(INGUSR)
Qualification of another rule type with an IP address or a host name	SET CHLAUTH('*') + TYPE(USERMAP) + CLNTUSER('NOMAD123') + ADDRESS('10.4.*') MCAUSER('INGUSR')	SET CHLAUTH('*') + TYPE(USERMAP) + CLNTUSER('NOMAD123') + ADDRESS('a*.ibm.*') + MCAUSER('INGUSR')

© Copyright IBM Corporation 2015

Figure 10-44. CHLAUTH rules using IP addresses and host names

WM3021.1

Notes:

You now look at some examples of CHLAUTH rules that use IP addresses and host names. Host names can be used in most places, except with the BLOCKADDR type rule.

When starting to use CHLAUTH, as a rule of thumb, it is better to start with a rule to block everything, then add rules to allow valid connections. You discuss this approach later in this unit.

One frequent question is what type of CHLAUTH rule to use. What is the difference between a BLOCKADDR and an ADDRESSMAP with USERSRC(NOACCESS)? The answer has two parts:

- BLOCKADDR works at the listener level. This type of CHLAUTH rule is applied before the connection sends any data and IBM MQ does not have the incoming channel name. However, the appropriate place for this rule is the firewall. The BLOCKADDR CHLAUTH rule can be used to mitigate the risk of the connection while the firewall rule is in place. Care should be taken to follow up and remove any such rules after the firewall rule is in place.
- ADDRESSMAP with USERSRC(NOACCESS) is the correct method to set IP address rules. When an inbound connection is blocked as a result of one of these rules, the error message that is written to your error log, and the event message that is written to the SYSTEM.ADMIN.CHANNEL.EVENT queue, if events are enabled, contains the details about the blocked connection, not available with a BLOCKADDR type rule.

CHLAUTH rule precedence

Channel name	
SSL distinguished name	CHLAUTH ('MQ00. ') TYPE(SSLPEERMAP) SSLPEER('O="IBM US") MCAUSER(INGUSR)
Client user ID	CHLAUTH ('MQ00.*') TYPE(USERMAP) CLNTUSER('NOMAD123') MCAUSER(TSM0000)
Queue manager name	CHLAUTH ('MQ00.*') TYPE(ADDRESSMAP) ADDRESS('10.6.187.184') MCAUSER(TSM0000)
IP address	CHLAUTH ('MQ00.*') TYPE(ADDRESSMAP) ADDRESS('*.*.ibm.com') MCAUSER(IBMUSER)
Hostname	

- Channel authentication record selection criteria
 - Channel authentication with a full channel name takes precedence over a name that uses a wildcard
 - SSL DN or TLS DN takes precedence over a user ID, a queue manager name, or an IP address
 - Client ID or queue manager name takes precedence over an IP address
- Within each category, the most specific entry takes precedence
- Precedence also within SSL rules

© Copyright IBM Corporation 2015

Figure 10-45. CHLAUTH rule precedence

WM3021.1

Notes:

CHLAUTH rules follow precedence rules based on two factors:

- The type of rule, where SSL type rules have the highest weight, and host names have the least.
- Specificity rules, that is, the more specific a name, the higher in the precedence rank. For example, if you had an IP address of '10.6.187.184' and another address that uses a wildcard such as '10.6.*', the full IP address would precede the partial wildcard address.

Precedence also exists within SSL types. The SSL precedence list can be found under topic *Interaction between channel authentication records* in the IBM MQ Knowledge Center.

CHLAUTH initial settings

```
MQ00 DIS QMGR CHLAUTH
```

```
CSQM201I MQ00 CSQMDRTC DIS QMGR DETAILS 163
```

```
QMNAME (MQ00)
```

```
CHLAUTH (ENABLED)
```

```
END QMGR DETAILS
```

```
MQ00 DIS CHLAUTH ('*')
```

```
CSQM293I MQ00 CSQMDRTC 3 CHLAUTH FOUND MATCHING REQUEST CRITERIA
```

```
CSQM201I MQ00 CSQMDRTC DIS CHLAUTH DETAILS 168
```

```
CHLAUTH(*) DESCRIPTOR(Default rule to disallow privileged users)
```

```
TYPE (BLOCKUSER) USERLIST (*MQADMIN)
```

```
CSQM201I MQ00 CSQMDRTC DIS CHLAUTH DETAILS 169
```

```
CHLAUTH(SYSTEM.*) DESCRIPTOR(Default rule to disable all SYSTEM channels)
```

```
TYPE (ADDRESSMAP) ADDRESS (*)
```

```
USERSRC (NOACCESS)
```

```
CSQM201I MQ00 CSQMDRTC DIS CHLAUTH DETAILS 170
```

```
CHLAUTH(SYSTEM.ADMIN.SVRCONN) DESCRIPTOR(Def rule to allow MQ Explorer access)
```

```
TYPE (ADDRESSMAP) ADDRESS (*)
```

```
USERSRC (CHANNEL)
```

```
END CHLAUTH DETAILS
```

© Copyright IBM Corporation 2015

Figure 10-46. CHLAUTH initial settings

WM3021.1

Notes:

The initial settings for CHLAUTH in a queue manager are found in two places:

- First, in the queue manager object CHLAUTH attribute. Default is ENABLED.
- There are three default rules set in the queue manager. Each rule is explained in the DESCRIPTOR attribute of the display.

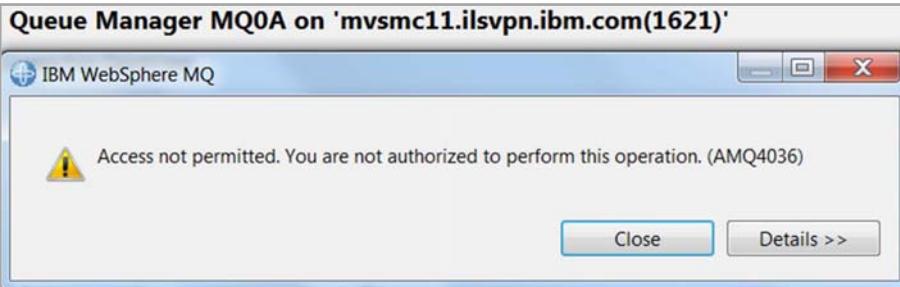
Getting started with CHLAUTH (1 of 5)

- Start with the “back-stop” rule to keep everyone out



Back-stop rule can be initially implemented in warning mode.

```
SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS) +
DESCR('Back-stop rule')
```



If there are active users, soon there will be numerous connection failures following the back-stop rule

```
+CSQX511I MQ0A CSQXRESP Channel WM302.SVRCONN started 181
connection 10.4.127.184
+CSQX777E MQ0A CSQXRESP Channel WM302.SVRCONN from 10.4.127.184 182
(10.4.127.184) has been blocked due to USERSRC(NOACCESS), Detail:
CLNTUSER(TSMUSER)
+CSQX512I MQ0A CSQXRESP Channel WM302.SVRCONN no longer active
```

© Copyright IBM Corporation 2015

Figure 10-47. Getting started with CHLAUTH (1 of 5)

WM3021.1

Notes:

The suggestion is to start with what is referred to as the “back-stop rule”. This rule stops all access. Soon after you set this rule you see numerous blocking messages in the system log. If you have channel events that are configured, along with the messages in the log there will also be event messages that are placed in the **SYSTEM.ADMIN.CHANNEL.EVENT** queue.

Slide two of this set shows many other rejected users. If this environment is a development environment with many active users as evidenced in slide 2, the inbox or cell phone for IBM MQ administrators is busy. But how else are you going to determine who is connecting to the queue manager to implement the correct rules?

CHLAUTH can be set in warning mode by using attribute **WARN(YES)**. With **WARN(YES)**, CHLAUTH. However, blocked messages do not display in the logs, you must have events that are enabled and check the channel events to determine what channels are blocked. **WARN(YES)** does allow the connections to proceed. In this manner, you can implement the rules, starting with the back-stop rule, and better organize the rules that need to be set for the users. You now identify the known users by review of the event messages, and plan the rules without disrupting access to users. After the rules are in place and tested and no longer in warning mode, the number of rejected connections should disappear or diminish to a manageable volume.

Getting started with CHLAUTH (2 of 5)

The billing client cannot connect ...

```
+CSQX511I MQ0A CSQXRESP Channel BILLING.MASTER started 192
  connection 10.4.127.184
+CSQX777E MQ0A CSQXRESP Channel BILLING.MASTER from 10.4.127.184
  (10.4.127.184) has been blocked due to USERSRC(NOACCESS), Detail:
    CLNTUSER(nomad789)
```

```
+CSQX512I MQ0A CSQXRESP Channel BILLING.MASTER no longer active
```

the orders application cannot connect ...

```
+CSQX777E MQ0A CSQXRESP Channel QMGR1.MQ0A from 10.4.127.184 203
  (10.4.127.184) has been blocked due to USERSRC(NOACCESS), Detail:
    QMNAME (QMGR1)
```

```
+CSQX599E MQ0A CSQXRESP Channel QMGR1.MQ0A ended abnormally 204
  connection 10.4.127.184
```

... and the cluster channels are also challenged

```
+CSQX191I MQ0B CSQXRCTL Channel WMADMCLS.MQ0A beginning message 2
  reallocation
```

```
+CSQX777E MQ0A CSQXRESP Channel WMADMCLS.MQ0A from mvsmc11 225
  (10.31.187.59) has been blocked due to USERSRC(NOACCESS), Detail:
    QMNAME (MQ0B)
```

```
+CSQX599E MQ0A CSQXRESP Channel WMADMCLS.MQ0A ended abnormally 226
  connection 10.31.187.59
```

© Copyright IBM Corporation 2015

Figure 10-48. Getting started with CHLAUTH (2 of 5)

WM3021.1

Notes:

Using IBM MQ events to check CHLAUTH rejections.

In this sample, you use the message handler sample program, option H in the lab environment for this course, to look at the event record.

The return code is at offset X'1C' into the message data: x'0A11' or 2577, Channel Blocked.

```
Queue Manager      : MQ00
Queue             : SYSTEM.ADMIN.CHANNEL.EVENT
Message Content  :
PutApplName       : 'MQ00CHIN <=the CHIN region put the event message'
PutDate           : '20141013'
PutTime           : '13111657'
Message Buffer   : 200 byte(s)
00000000 : 0000 0007 0000 0024 0000 0001 0000 002E
00000010 : 0000 0001 0000 0001 0000 0001 0000 0A11
```

You can scroll down the rest of the message data for more details.

Getting started with CHLAUTH (3 of 5)

- Add rules to allow known valid connections:

```

SET CHLAUTH(QMGR1.MQ0A) TYPE(QMGRMAP) ADDRESS('10.4.127.184') +
DESCR('Access for ORDERS application') +
MCAUSER(TSMUSER) QMNAME(QMGR1)

SET CHLAUTH('WMADMCLS.*') TYPE(QMGRMAP) ADDRESS('10.31.*') +
DESCR('Access for WMADMCLS cluster channels') +
MCAUSER(TSMUSER) QMNAME(MQ*)

SET CHLAUTH(BILLING.MASTER) TYPE(USERMAP) +
ADDRESS(10.4.127.184) +
CLNTUSER('nomad789') MCAUSER(TSMUSER) +
DESCR('Access for BILLING client')

SET CHLAUTH(WM302.SVRCONN) +
TYPE(ADDRESSMAP)
ADDRESS('10.4.127.*') MCAUSER('TSMUSER') +
DESCR('Allow MQ Explorer users from 10.4.127.*')

```

© Copyright IBM Corporation 2015

Figure 10-49. Getting started with CHLAUTH (3 of 5)

WM3021.1

Notes:

This slide shows the rules that are implemented for the rejected applications. CHLAUTH records are a good place to make use of the DESCRIPTOR field to document what a rule is for.

A few months after the records are created, or if different IBM MQ administrators need to maintain the queue manager, the description of why the rule was created might be indispensable.

Getting started with CHLAUTH (4 of 5)

- Use of **DIS CHLAUTH** and **MATCH(RUNCHECK)** for rule checking:

```
DIS CHLAUTH('BILLING.MASTER') MATCH(RUNCHECK) +
  ADDRESS('10.4.127.184') +
  CLNTUSER('nomad789')
CSQN205I COUNT=            3, RETURN=0000000,
REASON=00000000
CSQM454I MQOA
  CHLAUTH(BILLING.MASTER)
  TYPE(USERMAP)
  ADDRESS(10.4.127.184)
  CLNTUSER(nomad789)
  MCAUSER(TSMUSER)
  USERSRC(MAP)
```

© Copyright IBM Corporation 2015

Figure 10-50. Getting started with CHLAUTH (4 of 5)

WM3021.1

Notes:

Use of the DIS CHLAUTH with MATCH(RUNCHECK) option returns the record that is matched by an inbound channel at run time if it does connect to this queue manager. The command can be used for the following tests:

- Check if DNS is properly set up
- Determine whether hostname-based rules work

Reverse DNS lookup can be disabled at the queue manager level by setting REVDNS(DISABLED).

The MATCH(RUNCHECK) command needs the IP address. The queue manager calls the Domain Name Server (DNS) as if it was an actual inbound connection to find a host name.

- If it finds a host name, CHLAUTH works with host name rules.
- If RUNCHECK does not return a host name, hostname-based rules will not work.
- If reverse DNS lookup is disabled with REVDNS(DISABLED) in the queue manager object, an error displays if a rule that uses hostname runs.

Getting started with CHLAUTH (5 of 5)

Hints and tips:

- Implement the back-stop rule in warning mode if you need to minimize impact
- Monitor the outcome of the back-stop rule
 - Check the system log for blocked channels if WARN(NO)
 - Enable channel events and review event messages
 - Blocked connection messages are **not** generated on the system log for rules with WARN(YES). Must enable channel events to view rejected connections
- Add one **CHLAUTH** rule at a time and test it
- Make sure to understand the attributes required with each type of CHLAUTH rule and use correct attributes for all ACTION types, particularly ALTER and REMOVE
- Use the **DESCR** attribute of the **CHLAUTH** agent to identify why each rule is in place
- After you have implemented and tested your rules
 - Run a backup of the rules with CSQUTIL MAKEDEF for the CHLAUTH objects
 - Confirm that your organization's scheduled object backup includes CHLAUTH rules

© Copyright IBM Corporation 2015

Figure 10-51. Getting started with CHLAUTH (5 of 5)

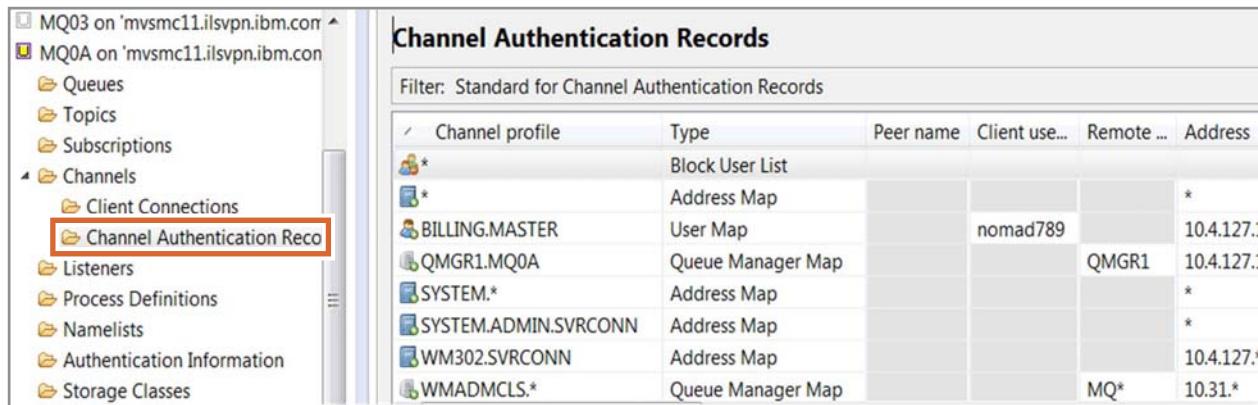
WM3021.1

Notes:

This slide contains suggestions to get started with CHLAUTH rules.

 WebSphere Education 

Use of MQ Explorer for CHLAUTH (1 of 2)



The screenshot shows the IBM MQ Explorer interface. On the left is a tree view of resources under 'MQ03 on 'mvsme11.ilsvpn.ibm.com''. The 'Channels' node is expanded, and its 'Channel Authentication Reco' child node is selected and highlighted with a red box. To the right is a table titled 'Channel Authentication Records' with the following data:

Channel profile	Type	Peer name	Client use...	Remote ...	Address
*	Block User List			*	
*	Address Map		nomad789		10.4.127.1
BILLING.MASTER	User Map			QMGR1	10.4.127.1
QMGR1.MQOA	Queue Manager Map				*
SYSTEM.*	Address Map				*
SYSTEM.ADMIN.SVRCONN	Address Map				*
WM302.SVRCONN	Address Map				10.4.127.*
WMADMCLS.*	Queue Manager Map		MQ*		10.31.*

© Copyright IBM Corporation 2015

Figure 10-52. Use of MQ Explorer for CHLAUTH (1 of 2)

WM3021.1

Notes:

IBM MQ Explorer is updated to process CHLAUTH rules. The CHLAUTH rules are found in the **Channels** menu.

Click the **Channel Authentication** menu for a list of the channel authentication records.

Right-click in the **Channel Authentication Records** selection for the CHLAUTH wizard.

Use of MQ Explorer for CHLAUTH (2 of 2)

- Wizard walks through the definition and present a preview of the corresponding MQSC
SET CHLAUTH
command

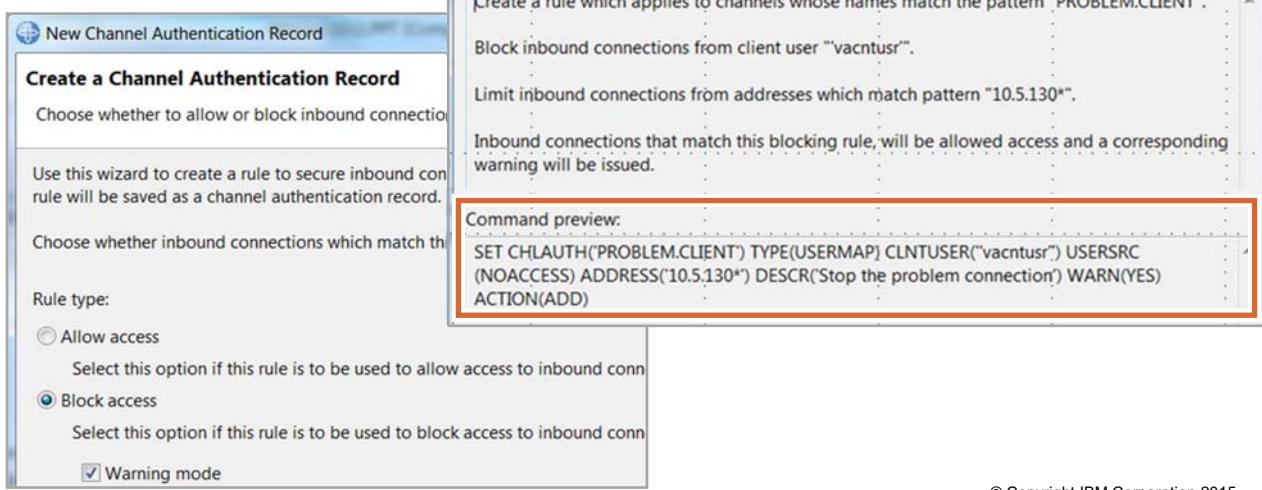


Figure 10-53. Use of MQ Explorer for CHLAUTH (2 of 2)

WM3021.1

Notes:

The channel authentication wizard presents the necessary pop-up panels to create a CHLAUTH record.

On the last panel, the wizard displays the command in its MQSC form as it would be run in CSQUTIL.

SSL/TLS: Confidentiality, data integrity, and authentication

**Secure Sockets Layer (SSL)
Transport Level Security (TLS)**



- When SSL/TLS is used with IBM MQ, CA certificates are added to the z/OS key ring file
- SSL and TLS are configured using the same process but selecting different cipher suites
- After the certificate is set up, it can be added to a channel by altering the channel attributes
- For MQ V8 and later a queue manager can have multiple certificates:
 - Single certificate: `ALTER QMGR CERTLBL('Certificate name')`
 - Per channel certificate:
`ALTER CHANNEL CERTLBL('This certificate')`
 - `QSGCERTLABEL` attribute is used for a QSG level certificate
- **CHLAUTH** rules can be used on certificates:
 - `SSLPEER` matches the subject's DN
 - `SSPCERTI` matches the issuer's DN

© Copyright IBM Corporation 2015

Figure 10-54. SSL/TLS: Confidentiality, data integrity, and authentication

WM3021.1

Notes:

Before IBM MQ V8, queue managers might be configured with one certificate. With IBM MQ V8, queue managers can now use different SSL certificates per channel. Label requirements have changed. For IBM MQ V8, an organization can provide their label name for the queue manager or IBM MQ Client. New attributes have been added to handle the ability to use multiple certificates:

- `CERTLBL` queue manager attribute
- `CERTQSQL` for queue-sharing groups
- `CERTLBL` channel attribute
- `SSLCERTI` attribute for CHLAUTH records to match the issuer's DN

If a channel needs to use a specific certificate, the label name can be specified in the channel's `CERTLBL` attribute. If the channel uses the queue manager certificate, leave `CERTLBL` blank and it uses the queue manager's default certificate.

For clients:

- Use SSL stanza in `mqclient.ini`
- Use `MQCERTLBL`
- Provide certificate label in `MQSCO` structure

Dead-letter queue considerations

- Adequate planning needs to take place to ensure that access to the dead-letter queue does not become challenged by security
- Who can read the dead-letter queue if the message is encrypted?
- Who is authorized to put messages to the dead-letter queue?
- Unauthorized attempts to place a recoverable message in the dead letter queue might cause a channel to fail

© Copyright IBM Corporation 2015

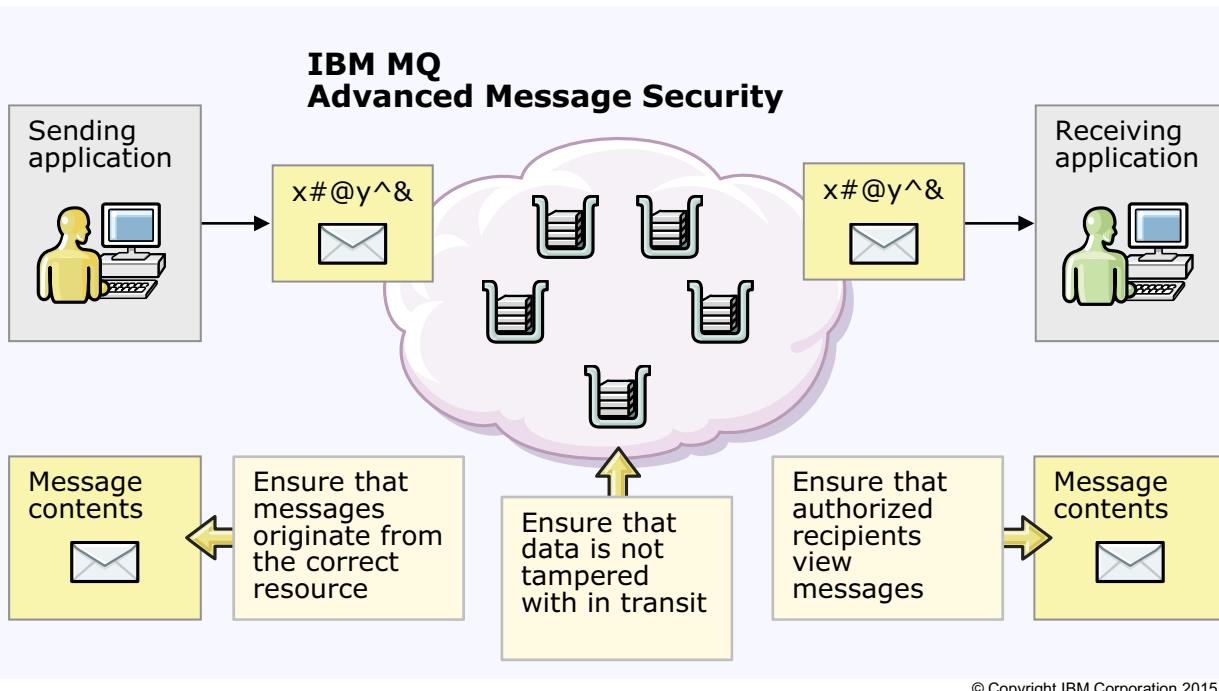
Figure 10-55. Dead-letter queue considerations

WM3021.1

Notes:

Confidentiality and data integrity: Application-level security

- Application-level security pertains to the safety of messages while **not** in transit, also referred to as “data at rest”



© Copyright IBM Corporation 2015

Figure 10-56. Confidentiality and data integrity: Application-level security

WM3021.1

Notes:

Application level security was introduced earlier in this unit. Advanced Message Security (AMS) is the mechanism that provides the functions necessary to address application level security concerns.

Advanced Message Security provides the following capabilities:

- End-to-end data protection that uses either encryption or digitally signing messages
- Uses Public Key Infrastructure to provide authentication, authorization, confidentiality, and data integrity function for messages
- Provides security policy administration point

AMS is separately licensed. Other business partners and customized solutions can be used to achieve these results; however, AMS provides these facilities without the need to write complex security code or modify existing applications.

Auditing

- Queue manager events
- Security failures such as:
 - Connection not authorized
 - Open not authorized
 - Subscription not authorized
- Captured as messages sent to the SYSTEM.ADMIN.QMGR.EVENT queue
- Other events can also be checked, such as using a configuration event to capture “someone defined a channel in production”
- Use external security manager facilities such as RACF to capture:
 - Failed access to resources
 - Successful access to resources
 - Any access or changes to security profiles
- Need to be enabled:
 - **RACROUTE REQUEST=AUDIT**
 - **ZPARM: RESAUDIT (YES | NO)**

© Copyright IBM Corporation 2015

Figure 10-57. Auditing

WM3021.1

Notes:

Auditing is another consideration of the overall security solution. When auditing is implemented, you can follow up on any possible intrusions, or review reports to determine if any unauthorized activities need follow up, possibly a new CHLAUTH rule that is implemented, or a different action.

IBM MQ security summary

- Identification
 - OS user ID's
 - Context
 - Channel Authentication Records
 - Channel Security Exits
- Authentication
 - Original OS Logon
 - MQCONNX MQI call
 - Channel Security Exits
 - Channel Authentication Records (CHLAUTH)
 - Secure Sockets Layer (SSL)
 - Connection authentication
- Access control
 - RACF switches
 - Put authority
 - Channel authentication records
 - Firewalls
- Confidentiality
 - Advanced Message Security (AMS)
 - Secure Sockets Layer (SSL)
 - Channel exits
- Data Integrity
 - Advanced Message Security (AMS)
 - Secure Sockets Layer (SSL)
- Auditing
 - Events

© Copyright IBM Corporation 2015

Figure 10-58. IBM MQ security summary

WM3021.1

Notes:

This summary has emphasis on IBM MQ for z/OS security. Distributed systems use OAM instead of RACF.

Unit summary

- Contrast security areas as they apply to IBM MQ
- Differentiate between link level and application level security
- Summarize the way security is implemented in IBM MQ for z/OS
- Describe how to implement connection authentication
- Describe how to use channel authentication records
- Summarize the mechanisms that are available to secure IBM MQ for z/OS

© Copyright IBM Corporation 2015

Figure 10-59. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. True or false: IBM WebSphere MQ for z/OS makes use of RACF for system security.

2. The MCA needs to consider the security profiles of:
 - a. Connections
 - b. Resources
 - c. Commands
 - d. Data sets
 - e. All of the above

© Copyright IBM Corporation 2015

Figure 10-60. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.

- 2.

Checkpoint questions (2 of 2)

3. True or false: The MQADMIN class must be enabled before any security checking is performed.

4. The CHLAUTH “back-stop rule” is implemented by:
 - a. SET CHLAUTH('*) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
 - b. Set queue manager CHLAUTH attribute to ENABLED
 - c. SET CHLAUTH('*) TYPE(BLOCKADDR) ADDRESS('*') USERSRC(NOACCESS)
 - d. a and c
 - e. a and b

5. How do you check how connection authentication is configured for a queue manager?

© Copyright IBM Corporation 2015

Figure 10-61. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

5.



Checkpoint answers (1 of 2)

1. True or false: IBM WebSphere MQ for z/OS makes use of RACF for system security.

Answer: True. Can also use CA ACF/2 or Top Secret.

2. The MCA needs to consider the security profiles of:

- a. Connections
- b. Resources
- c. Commands
- d. Data sets
- e. All of the above

Answer: e, although it is not necessary to secure all resources depending on the environment.

© Copyright IBM Corporation 2015

Figure 10-62. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. True or false: The MQADMIN class must be enabled before any security checking is performed.

Answer: True

4. The CHLAUTH “back-stop rule” is implemented by:
- a. SET CHLAUTH('*) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
 - b. Set queue manager CHLAUTH attribute to ENABLED
 - c. SET CHLAUTH('*) TYPE(BLOCKADDR) ADDRESS('*') USERSRC(NOACCESS)
 - d. a and c
 - e. a and b

Answer: e

5. How do you check how connection authentication is configured for a queue manager?

Answer: (1) DIS QMGR CONNAUTH for active AUTHINFO, then (2) DIS AUTHINFO (shown in #1) to determine CHCKCLNT and CHCKLOCL settings

© Copyright IBM Corporation 2015

Figure 10-63. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 8



© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 10-64. Exercise 8

WM3021.1

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Determine the queue manager connection authentication (CONNAUTH) settings
- Test initial client connection authentication settings
- Change client connection authentication settings to require full credentials
- Use available tools to test connection authentication changes
- Adjust IBM MQ Explorer to work with client connection authentication enabled
- Display the initial channel authorization (CHLAUTH) rules
- Add a rule to allow selected administrative users
- Implement the back-stop rule in full blocking mode
- Add CHLAUTH rules that allow valid connections
- Change and delete CHLAUTH rules
- Interpret event messages that are related to CHLAUTH
- Back up CHLAUTH rules
- Explain how to set rules in warning mode and where to check results of rules that are running in warning mode

© Copyright IBM Corporation 2015

Figure 10-65. Exercise objectives

WM3021.1

Notes:

Unit 11. Problem determination

What this unit is about

This session augments the troubleshooting concepts that were presented in earlier units with topics such as how to handle internal failures and analyze performance.

What you should be able to do

After completing this unit, you should be able to:

- Differentiate between queue manager and IBM MQ problems
- Identify the cause of channel problems
- Examine performance-related data

Unit objectives

- Differentiate between queue manager and IBM MQ problems
- Identify the cause of channel problems
- Examine performance-related data

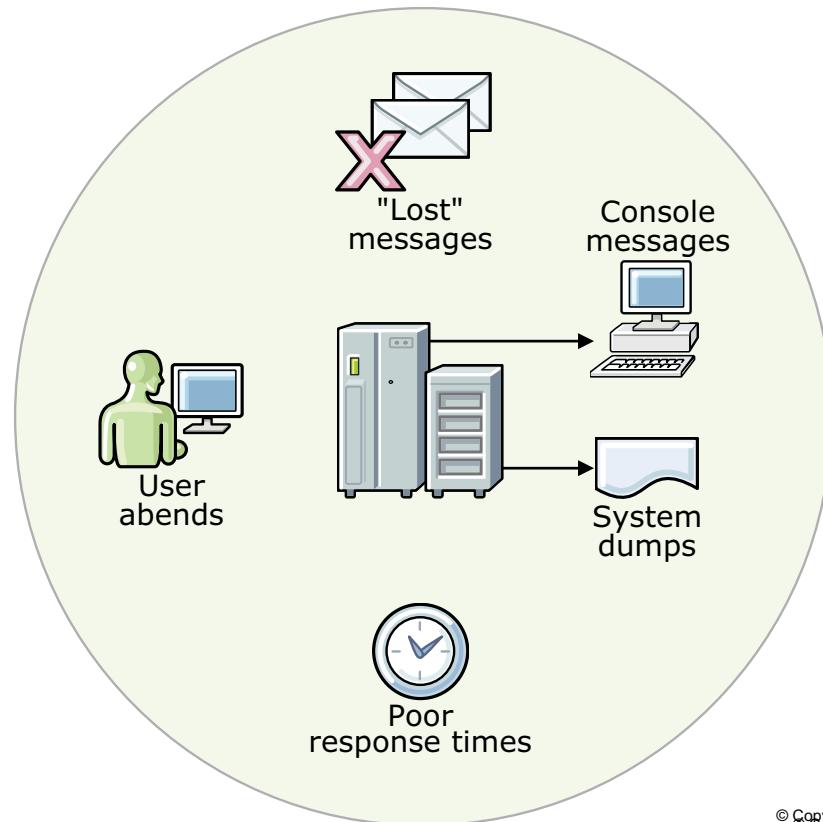
© Copyright IBM Corporation 2015

Figure 11-1. Unit objectives

WM3021.1

Notes:

Initial problem identification



© Copyright IBM Corporation 2015
© Copyright IBM Corporation 2015

Figure 11-2. Initial problem identification

WM3021.1

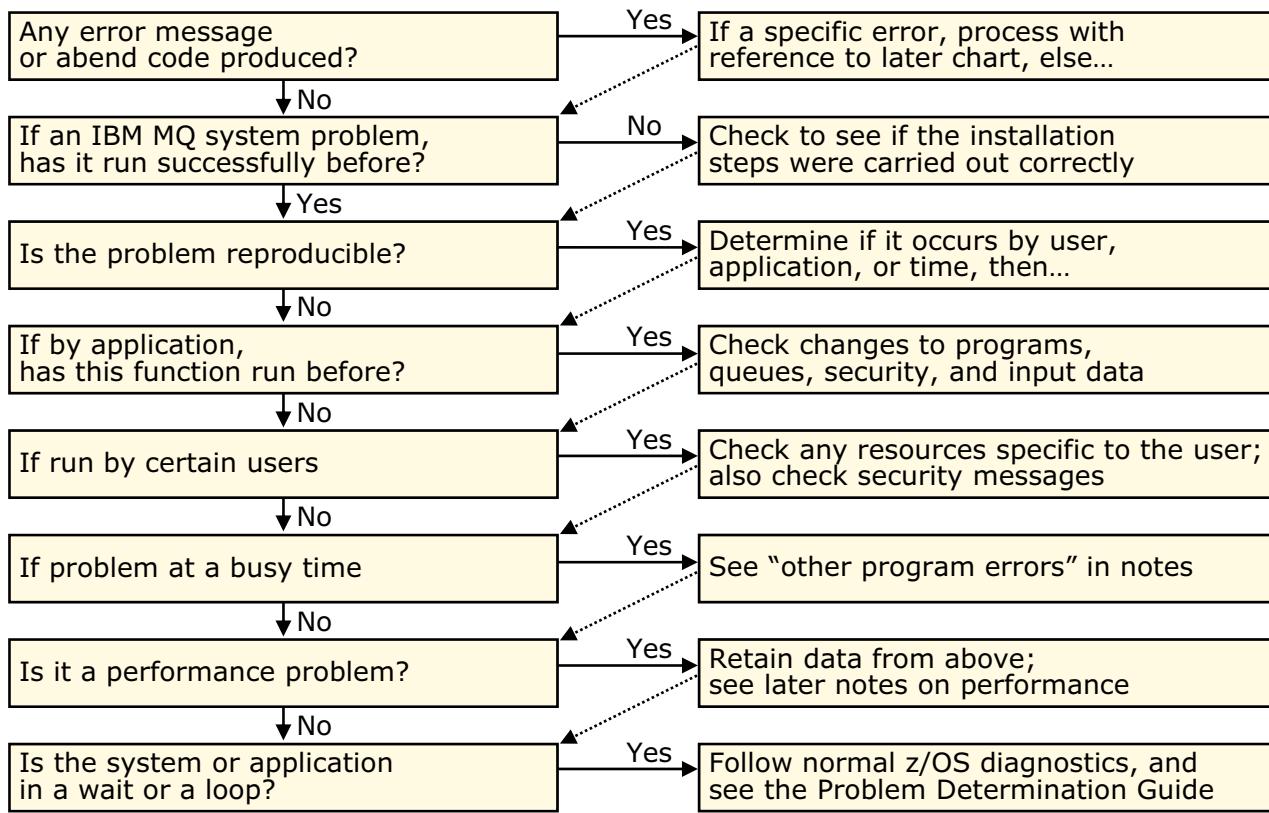
Notes:

This unit describes the basic approaches to problem determination in an IBM MQ for z/OS environment. This information includes the system facilities that provide information that might help identify the causes of errors and failures, be suitable to send to the IBM software support for investigation, or both.

Initially, problems are identified in one of several ways, such as:

- An application or the subsystem terminates unexpectedly
- An unexpected error message is received
- IBM MQ for z/OS messages or replies are not received
- Incorrect output is received
- The system or application might not respond due to a wait or loop
- The system performs badly for other reasons

Identifying programming errors



© Copyright IBM Corporation 2015

Figure 11-3. Identifying programming errors

WM3021.1

Notes:

Perform the preliminary checks for types of error, as contained in the left column, and act according to the right column. If the problem cannot be solved by the hints that are provided, use information from the earlier checks to help problem determination, which is discussed in the following notes, according to the symptoms of the problem.

The following common programming errors can cause X'0C4' or similar abends:

- Linking a program with the wrong or no stub
- Passing the wrong number, incorrect parameters, or the wrong order
- Using incorrect addresses or variables with incorrect lengths
- Not initializing storage before use

Other program errors causes are provided here:

- Failure to check completion and reason codes from MQI requests
- Failing to initialize MsgID and CorrelId before each call
- Not designing applications to use the default share options correctly with the share and noshare queue attributes

IBM MQ for z/OS diagnostics

- Applications
 - Completion and reason codes
- Queue manager
 - CSQ+ console messages
 - abends
 - **SYS1.LOGREC**
- CICS
 - **CSMT** log
 - CICS dump data set
- IMS log
- Batch and TSO
 - **SYSUDUMP**

© Copyright IBM Corporation 2015

Figure 11-4. IBM MQ for z/OS diagnostics

WM3021.1

Notes:

If IBM MQ for z/OS detects an error:

- User errors (for example, bad parameters) are reflected back to the user with completion and reason codes.
- For more serious errors, a console message can be put on the z/OS console and reflected in the job output. All IBM MQ for z/OS messages are prefaced CSQ followed by a single character component ID.
- If caused by an IBM MQ for z/OS-related condition (for example, page set full) or other system problem from which IBM MQ for z/OS can recover, completion and reason codes are sent to the application.
- The IBM MQ for z/OS CICS adapter, or distributed queuing component, might issue a transaction abend code. IBM MQ for z/OS abend reason codes begin with the letter queue.
- If any of the adapters detect an `MQRC_UNEXPECTED_ERROR` reason code, a snap dump is taken for the application region if the JCL specifies a DD statement for `CSQSNAP`.
- If IBM MQ for z/OS is unable to recover, it terminates with a specific abend reason code and usually an SVC dump is taken.

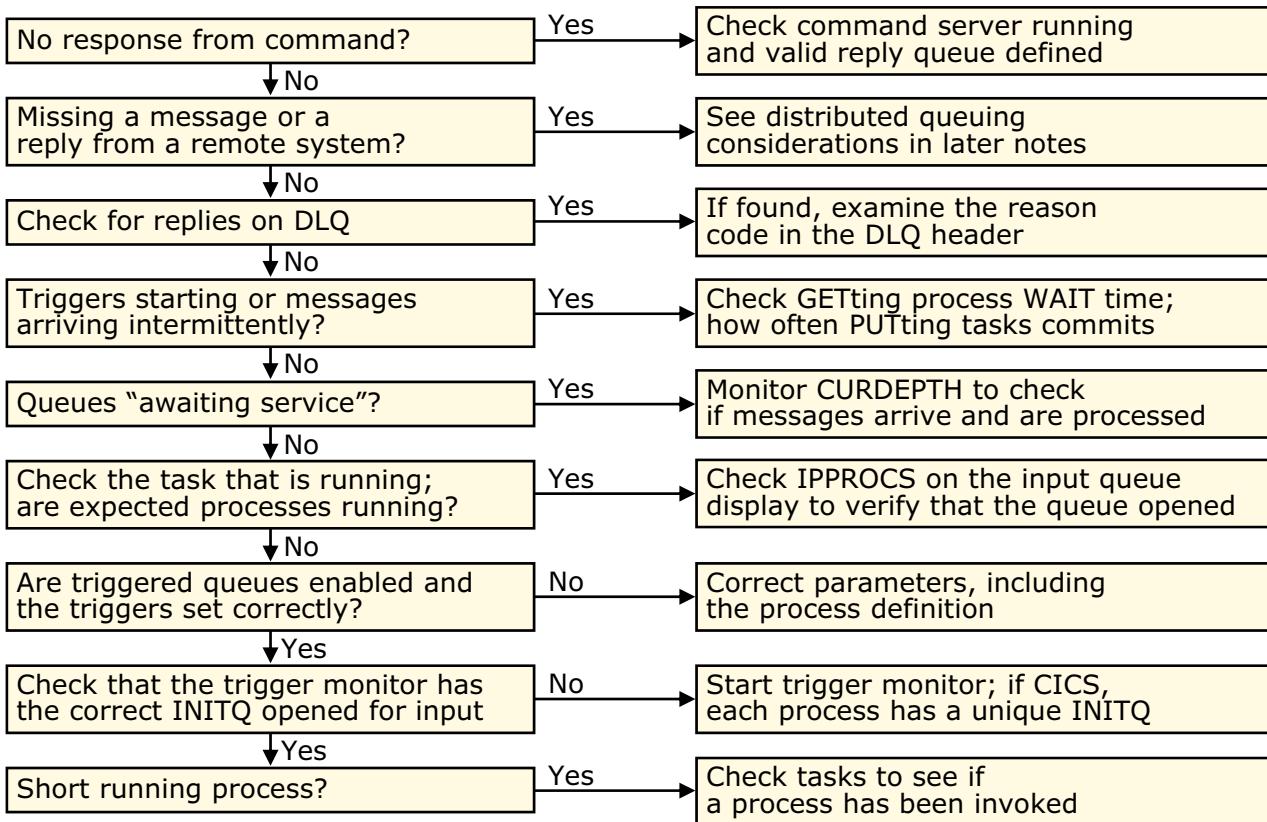
If z/OS detects an error:

- Normal z/OS (MVS) diagnostic messages are produced, which might be followed by further IBM MQ for z/OS messages or abends.
- If a system abend occurs, normal MVS system abend completion codes are used.

If a user detects an error (for example, a nonzero return code after completion of an IBM MQ for z/OS call).

Further information can be obtained by rerunning the application with the IBM MQ for z/OS user parameter trace activated.

Investigating missing or unexpected output



© Copyright IBM Corporation 2015

Figure 11-5. Investigating missing or unexpected output

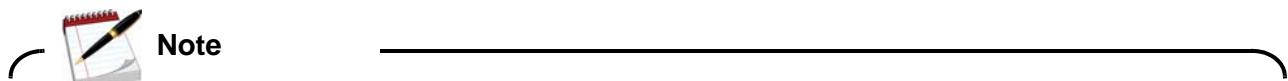
WM3021.1

Notes:

This flowchart shows further checks that can be performed when messages do not arrive or replies are not received when expected.

If they do not help you to solve the problem, use IBM MQ for z/OS trace to display API calls and user parameters as described in later notes to further analyze the problem.

Information available from the DISPLAY QUEUE command (or Operations and Control panels) for earlier checks: CURDEPTH, IPPROCS, and OPPROCS show whether messages exist on the queue and if they are being processed.



If messages are put on a queue under syncpoint control, they are not available until a syncpoint is taken.

- Check whether the triggering options are correct for this queue, and that a trigger monitor transaction is running against the INITQ specified
- Check that the queue is enabled for **GETs** and **PUTs** as appropriate
- Check whether the queue can be shared, and whether another process already has it open for input

If the problem being investigated involves remote queues, refer to the *Distributed Queuing* unit for this course.

Activity reports and route tracking

Activity reports

- PCF messages that are generated after work has been performed on behalf of a message
- Controlled by **MQRO_ACTIVITY** report field in the MQMD
- **QMGR ACTIVREC** attribute determines where the Activity Report message is placed:
 - **MSG** → to the message Reply-To-Queue
 - **Q** → to **SYSTEM.ADMIN.ACTIVITY.QUEUE**
 - **DISABLED** → no message

Route tracking

- Based on dedicated diagnostic messages
- Queue manager **ROUTEREC** controls the accumulation of activity information within traceroute messages and the generation of traceroute reply messages for applications that are connected to it

© Copyright IBM Corporation 2015

Figure 11-6. Activity reports and route tracking

WM3021.1

Notes:

The routes of messages can be monitored. This monitoring can be helpful in checking the configuration of the queue manager network and in diagnosing configuration errors.

There are three ways for activity information to be recorded for a message:

1. In an activity report, generated after each activity has taken place. Typical activities in this sense are **MQPUT** and **MQGET** operations, and the sending and receiving of messages by MCAs. This process is called *activity recording*.
2. Internally accumulated within a trace route message as it is routed through a queue manager network. This process is called *trace route messaging*.
3. In a *trace route reply message*, generated when the message has reached its destination. This process can take place only if point 2 has also taken place.

On distributed platforms, a sample program that is named DSPMQRTE is provided as a Display Route Application that can configure, generate, and then put a trace-route message. This program can connect to z/OS QMGRs as a client application.

Scheduled abend codes

- 5C6
 - Error during normal operation
 - Task is terminated unexpectedly
 - Reason code in console message or dump title

- 6C6
 - Fatal error
 - Queue manager terminates
 - May be preceded by 5C6 abends or other messages

© Copyright IBM Corporation 2015

Figure 11-7. Scheduled abend codes

WM3021.1

Notes:

ABEND CODE X'5C6'

- Indicates that IBM MQ for z/OS has detected an *internal* error and has abnormally terminated an IBM MQ for z/OS internal task (TCB), or user-connected task, with an SVC dump.
- 5C6 abends might be preceded by a z/OS system code or IBM MQ for z/OS internal error messages.
- The IBM MQ for z/OS abend reason code is displayed as part of the dump title that is generated by z/OS, and is often sufficient enough to identify the cause of the error without looking at the dump.
- Dump titles can be displayed from IPCS or alternatively from a z/OS console (or authorized NETVIEW user) by using the z/OS DISPLAY DUMP,TITLE or D D,T.

ABEND CODE X'6C6'

- Indicates that IBM MQ for z/OS has detected a severe error and has abnormally terminated the entire IBM MQ for z/OS subsystem

- 6C6 abends might be preceded by other messages and abends:
 - One or more X'5C6' abend completion codes
 - Error message CSQV086E (IBM MQ for z/OS abnormal termination)
 - Error driven by a z/OS system error

Formatting system dumps using IPCS

----- IBM WebSphere MQ for z/OS - DUMP ANALYSIS -----
 COMMAND ===>

- 1 Display all dump titles 00 through 99
- 2 Manage the dump inventory
- 3 Select a dump

- 4 Display address spaces active at time of dump
- 5 Display the symptom string
- 6 Display the symptom string and other related data
- 7 Display LOGREC data from the buffer in the dump
- 8 Format and display the dump

- 9 Issue IPCS command or CLIST

(C) Copyright IBM Corporation 1993,2005. All rights reserved.

F1=Help F2=Split F3=Exit F9=Swap F12=Cancel

© Copyright IBM Corporation 2015

Figure 11-8. Formatting system dumps using IPCS

WM3021.1

Notes:

IPCS should be used to format any SVC dumps from IBM MQ for z/OS, or in the case of a stand-alone dump that is written to disk or tape. IBM MQ for z/OS provides a set of panels to process IBM MQ for z/OS dumps, which are accessed through the IPCS Primary Option Menu in ISPF. Check that these options are installed for your installation and proceed as follows:

- Select **option 2, ANALYSIS**: Analyze dump contents, from the IPCS menu
- Select **option 6, COMPONENT**: z/OS Component data, from this panel
- Select **CSQMAIN**: IBM MQ z/OS dump formatter panel interface
- The panel that is shown in the figure should now be displayed

Select the dump to process (if you have not viewed it before):

- Option 1 displays all dump titles in the specified range for the SYS1.DUMPxx data sets.
- If the dump is in one of the SYS1.DUMPxx data sets, use option 2 to check that the data set is not already in the dump inventory showing details from a previous dump.
- Use option 3 to specify the dump data set name to work with in the **Source** field, and ensure that the **NOPRINT** and **TERMINAL** options are set (to view dump at the terminal).

The remaining options display different areas of the dump and are explained later.

Analyzing dump data using IPCS

The dump title has the following format:

```
ssnm,ABN=compltn-reason,U=userid,C=compid.release.com-function,
M=module,LOC=loadmod.csect+csect_offset
```

The last keyword and value can be replaced with:

```
PSW=psw_contents,ASID=address_space_id
```

A real sample:

```
MQ0 ,ABN=5C6-00E90202,U=SYSOPR ,C=13700.114.IPC -CSQYSIRM,
M=CSQYECTE,LOC=CSQZTDDM.CSQZTGET+017C
```

© Copyright IBM Corporation 2015

Figure 11-9. Analyzing dump data using IPCS

WM3021.1

Notes:

The following areas of the dump can be displayed directly from the IBM MQ for z/OS Dump Display Menu:

- Dump titles (showing IBM MQ for z/OS abend reason code) - **option 1**
- List of address spaces active at the time of the dump - **option 4**
- Symptom String only - **option 5**; contains keywords that can be used for searching RETAIN database
- Symptom String and System Diagnostic Work Area fields (SDWA) - **option 6**; formats data from the unformatted SDWA, which describes the subsystem status at the time of the error
- In-storage LOGREC buffer - **option 7**

LOGRECs containing a snapshot of the information from the SDWA are written out by the error recovery routines (by using the LOGREC buffer).

Displaying error diagnostics

```
PIDS/5655W9700 RIDS/CSQJL002#L RIDS/CSQJB005 AB/S05C6 PRCS/00D10120 REGS/0E0
REGS/0404E RIDS/CSQJB003#R
Symptom Description
-----
PIDS/5655W9700 Program id: 5655W9700
RIDS/CSQJL002#L Load module name: CSQJL002
RIDS/CSQJB005 Csect name: CSQJB005
AB/S05C6 System abend code: 05C6
PRCS/00D10120 Abend reason code: 00D10120
REGS/0E02A Register/PSW difference for R0E: 02A
REGS/0404E Register/PSW difference for R04: 04E
RIDS/CSQJB003#R Recovery routine csect name: CSQJB003
OTHER SERVICEABILITY INFORMATION
Date Assembled: 20140502
Module Level: 12.27GA
Subfunction: RLMC BSDS
SERVICEABILITY INFORMATION NOT PROVIDED BY THE RECOVERY ROUTINE
Time of Error Information
PSW: 077C0000 960DE89C Instruction length: 02 Interrupt code: 000D
Failing instruction text: 58F0C040 0A0D5810 C03858F0
Registers 0-7
GR: 00000002 045C6000 009C6C68 009C6FF0 160DE84E 7F4D19C0 1417F928 ... ...
AR: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 ... ...
Registers 8-15
```

© Copyright IBM Corporation 2015

Figure 11-10. Displaying error diagnostics

WM3021.1

Notes:

The example in the figure is similar to what you might see if you displayed the contents of SYS1.LOGREC record. This example is for format only; the contents are from a different abend.

After this content, is the full unformatted SDWA (at the time the logrec was created) including the VRA entries.

The format of the displays for *symptom string*, SDWA fields, and in-storage LOGREC displays is the same as used for the relevant parts of the SYS1.LOGREC display.

Formatting and displaying entire dump

----- IBM WebSphere 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_ 1 Yes
 2 No

Subsystem name —
 (required if summary dump not used)

Address space identifier or ALL. ALL_

F1=Help F2=Split F3=Exit F9=Swap F12=Cancel

© Copyright IBM Corporation 2015

Figure 11-11. Formatting and displaying entire dump

WM3021.1

Notes:

Option 8 on the IBM MQ for z/OS Dump Display Menu gives the panel in the figure so you can format and display the SVC dump data set specified on the earlier panel.

- Select whether to display control blocks only, trace data only, or both
 - The dump title
 - The VRA diagnostic information report
 - The save area trace report
 - The control block summary
 - The trace table
- You should use the summary dump in normal circumstances unless the dump was not requested by IBM MQ for z/OS (such as by the operator)

If you specify no summary dump (or there is no summary data), you have to specify the subsystem name.

- You can limit the address space IDs if you only want to look at a certain task in the IBM MQ for z/OS address space (the default is ALL).



Using the IBM WebSphere MQ for z/OS trace

```
USRD9 5E9 ASCB 00FB5600          JOBN MQ03MSTR
CSQW072I ENTRY: MQ user parameter trace
CSQW075I WARNING - data was truncated at 256 bytes
GETMSG
      Thread... 00000000  Userid...           Hobj...
      pMsgDesc. 7EC2EE98  pGMO..... 7EC2F008  Buffer
      pBuffer.. 5C3B5035  DataL..... 00000000  pECB..
      RSV1..... 00000000  RSV2..... 00000000  RSV3..
      ORPLX.... 00000000
```

```
START TRACE(GLOBAL)
    DEST(GTF)
    CLASS(03)
    IFCID(*)

RMID(component)
USERID(user)
```

20:17:00 GMT-01/14/2015 01:17:03.239364 LOC-01/13/2015

```
USRD9 5E9 ASCB 00FB5600                      JOBN MQ03MSTR
CSQW072I ENTRY: MQ user parameter trace
+0000  D4C44040  00000001  00000040  00000001
+0010  00000BB8  00000000  00000311  000001F4
+0020  D4D8C1C4  D4C9D540  00000005  00000000
+0030  00000000  00000000  00000000  00000000
+0040  00000000  00000000  00000000  00000000
+0050  00000000  00000000  00000000  00000000
+0060  00000000  C1D4D84B  D4D8C5E7  D7D3D6D9
```

© Copyright IBM Corporation 2015

Figure 11-12. Using the IBM WebSphere MQ for z/OS traces

WM3021.1

Notes:

To trace API calls and user parameters, you must start GTF trace on your z/OS system, and the IBM MQ for z/OS GLOBAL trace.

Start GTF trace and specify the **USRP** option:

- If you want to limit trace to certain jobs, also specify the **JOBNAMEP** option, and specify the job name of the CICS or IMS system, TSO user, or batch job when requested
 - When requested, specify the event identifiers that are used by IBM MQ for z/OS:
 - **USR=(5E9 , 5EA , 5EB)**
 - **5E9:** Control blocks on entry to IBM MQ
 - **5EA:** Control blocks on exit to IBM MQ
 - **5EB:** Information internal to IBM MQ: applies to class(04), which is for IBM service only

Start IBM MQ for z/OS trace for each subsystem to be traced:

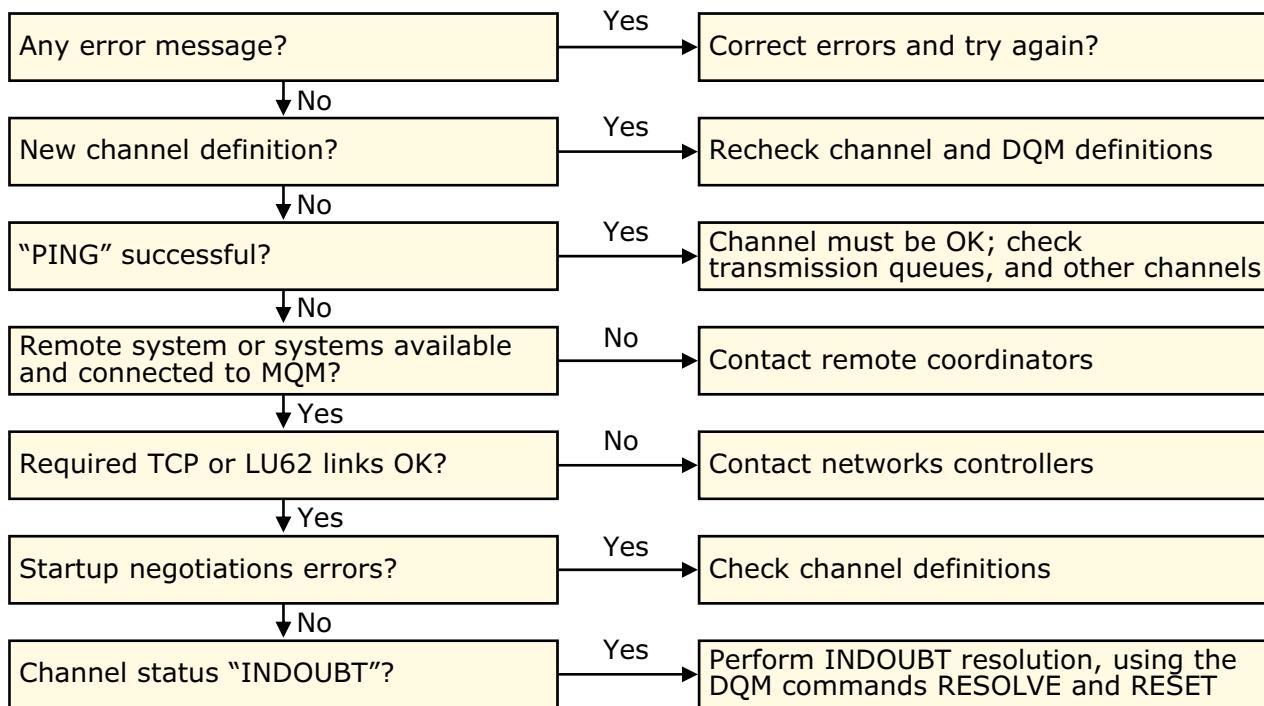
- To trace all API calls and parameters on entry and exit to IBM MQ for z/OS:

```
START TRACE(GLOBAL) DEST(GTF) CLASS(3)
```

- This command will also trace syncpoint flows such as **PREPARE** and **COMMIT**
- To trace only API calls and parameters when an error is detected, specify CLASS(2) instead of CLASS(3)
- QRPLTRAC records, if active, use the same trace identifiers, and are output on entry to and exit from **MQOPEN**

The CICS adapter automatically writes trace records to the appropriate CICS trace data set if a trace number was specified when the adapter was started, and CICS internal or external trace is enabled.

Channel problems



© Copyright IBM Corporation 2015

Figure 11-13. Channel problems

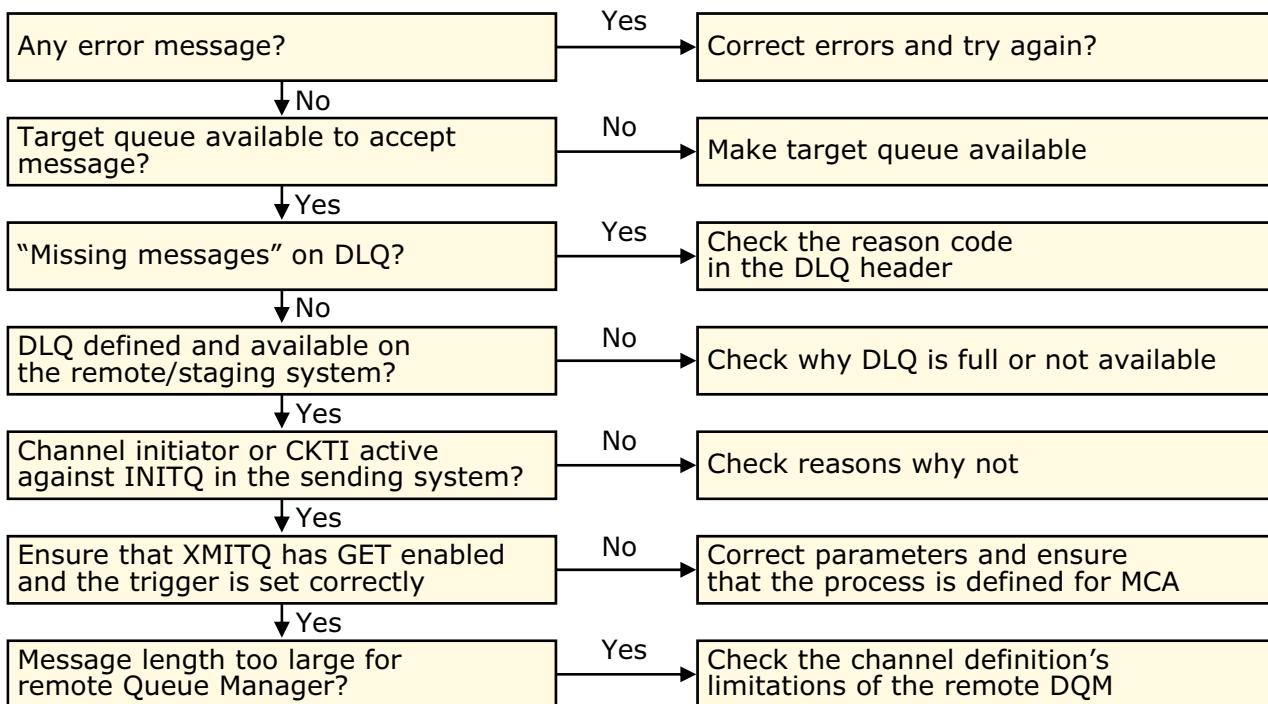
WM3021.1

Notes:

The following areas need to be investigated during distributed queuing problems.

- If this is a new channel definition, or a new implementation of DQM, you need to ensure:
 - That the channel definitions match, and DQM was correctly installed on local, remote, and staging systems.
 - That the links between the two systems are available, and that one or more channels have started successfully (and not stopped).
 - That triggering is active on each system, and that trigger definitions are set up correctly for transmission queues.
 - That the dead-letter queues are defined correctly for each system, and that some process is in place to monitor these (see DLQ considerations).
- In-doubt channel status, if not resolved automatically, should be resolved by using DQM functions, not the IBM MQ for z/OS RESOLVE INDOUBT command.
- Errors in user exits cause the MCA to stop. Check error messages and return codes from user exits and ensure that they conform to the requirements for each user exit.

DQM queue and trigger problems



© Copyright IBM Corporation 2015

Figure 11-14. DQM queue and trigger problems

WM3021.1

Notes:

If the target queues cannot accept the messages being sent, they are redirected to the dead-letter queue, if available.

Some dead-letter queue considerations are provided here.

- If the channel program cannot put messages on to the dead-letter queue, either because it is full or the messages are too large:
 - The channel is STOPPED and suitable console messages are issued.
 - Triggering is DISABLED for the transmission queue.
 - The current unit of work is backed out, and messages replaced on the transmission queue.

Therefore, you should ensure that your dead-letter queue is large enough to accept the largest messages in your system.

- It is advisable to have an application that processes the dead-letter queue, either triggered or set to run at regular intervals, to ensure that it does not fill up and to identify the reasons for the nondelivery of messages.

If a channel should be triggered:

- Check to see that the TRIGGER option on the transmission queue has not been reset. Channel errors sometimes set this option back to NOTRIGGER.
- Ensure that the transmission queue is enabled for **GETs**.

If you are using the CICS mover, enable CICS tracing and use CICS trace records and log messages produced by DQM for further analysis.

In-doubt resolution with DQM: ISPF and MQSC

Action	2
Object type . .	CHANNEL
Name	MQ00.M*

MQSC equivalent

RESOLVE CHL(MQ00.MQ0A) ACTION(xxx)

where xxx is COMMIT or BACKOUT

List Channels - MQ00 Row 1 of 1
Type action codes, then press Enter. Press F11 to display connection status.
1=Display 2=Define like 3=Alter 4=Manage 5=Perform
6=Start 7=Stop

Name	Type	Disposition	Status
<> MQ00*	CHANNEL	QMGR	MQ00
<u>5</u> MQ00.MQ0A	SENDER	QMGR	MQ00 INACTIVE

Perform a Channel Function

Select function type, complete fields, then press Enter.

Function type	1. Reset	3. Resolve with commit
	2. Ping	4. Resolve with backout

Channel name : MQ00.MQ0A

Channel type : SENDER

Description :

© Copyright IBM Corporation 2015

Figure 11-15. In-doubt resolution with DQM: ISPF and MQSC

WM3021.1

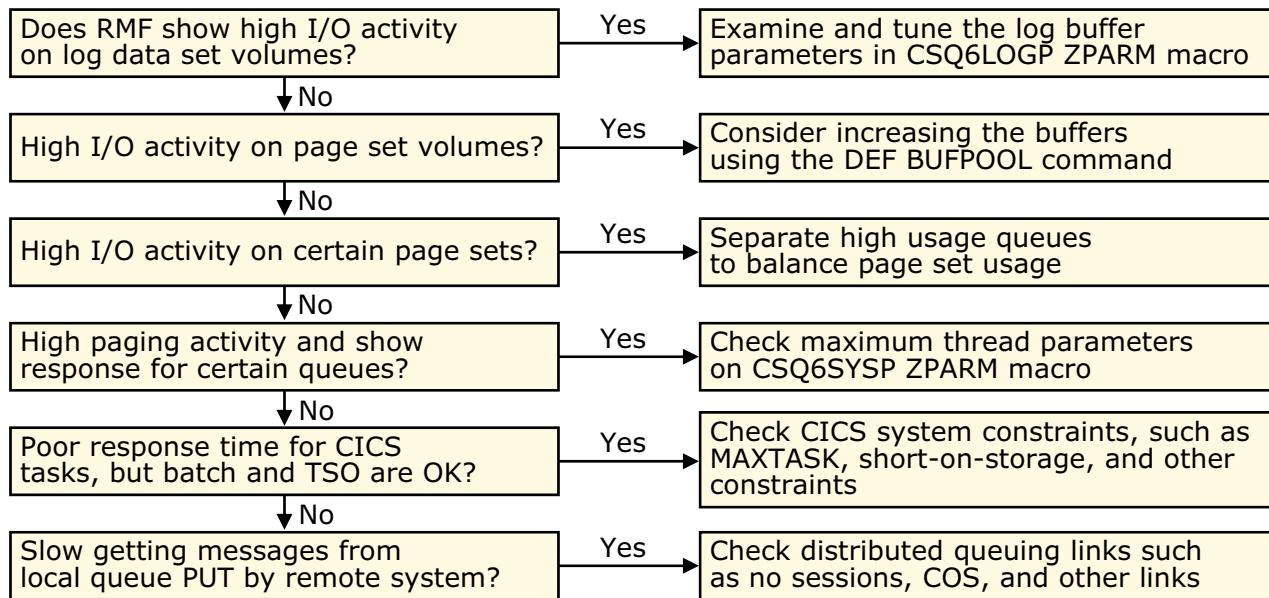
Notes:

If an in-doubt situation is not resolved automatically when the channel is started, you might need to use the DQM *Resolve channel* function (from the Sender/Server end):

- Ensure that any other errors, which might have stopped resynchronization at start time, have been resolved and try again by using Start or Resync
- Use the **DISPLAY CHANNEL** function from the control panels, then press F11 for channel status against the channel in-doubt to check the last *sequence number* that is sent and last *unit of work* that was committed
- Determine the same values from administrators at the remote end of the channel
- If the remote channel has committed a number or unit of work that has not yet been committed at this end, use *Resolve channel* with **Commit** specified; otherwise, use *Resolve channel* with **Backout** specified

If all in-doubts have been resolved and the *Sequence numbers* still do not agree, the sender should use the *Reset* function and specify the sequence number from the remote end.

Overall performance considerations



Become familiar with the z/OS performance support pack MP1J

© Copyright IBM Corporation 2015

Figure 11-16. Overall performance considerations

WM3021.1

Notes:

- First look at the overall system, by using a standard monitoring tool such as Resource Monitoring Facility (RMF). In particular, look at Total processor usage - DASD activity - Channel path usage - Paging.
- Use normal z/OS (MVS) tuning methods to address any overall problems.
- Use normal CICS monitoring facilities to identify any IBM MQ for z/OS system tasks or applications that are affected, especially where distributed queuing problems occur.
- Where monitoring highlights abnormal activity in certain areas, or the problems have previously been identified as confined to certain queues, transactions, and other areas, the flowchart in the figure gives an indication of where to look next.
- Where problems with log buffers or page set I/O, and so forth, are indicated, use IBM MQ for z/OS trace to produce performance statistics.

Queue manager statistics and accounting data

- Started when Queue Manager starts, through **ZPARM CSQ6SYSP** options:
 - Start data collection for default class 01
 - Time in minutes between consecutive gatherings of statistics

- Started by command:
 - SMF 115 and 116 records
 - Accounting data that is written at the end of each task
 - Statistical data that is written at the specified interval, or at SMF broadcast
 - Macros defining the record structures
 - Sample formatting utility

SMFACT (NO | YES)
SMFSTAT (NO | YES)
STATIME (0 . . . 1440)

START TRACE (STAT | ACCT)
DEST (SMF)
CLASS (01 | 03)
RMID (rmid)

Figure 11-17. Queue manager statistics and accounting data

© Copyright IBM Corporation 2015

WM3021.1

Notes:

- Accounting and statistics data are gathered through the IBM MQ for z/OS trace function.
- IBM MQ performance statistics data is written as SMF type 115 records; accounting data is written as SMF subtype 116 records.
- All records contain a self-defining section and can be accessed and formatted by use of assembler macros that are supplied in the product SCSQMACS library.
- Formatting is done with the MP1B utility. From the IBM MQ support and download Web pages, download support pack MP1B to format and print these SMF records

For more information about the detailed syntax and the options of the `START TRACE` command, see the *IBM MQ Script (MQSC) Command Reference*.

Channel initiator statistics and accounting data

```
MVCA, MQPV, 2014/03/18, 13:00:00, VRM:800,
From 2014/03/18, 12:45:00.015222 to 2014/03/18, 13:00:00.083630
duration
900.068408 seconds
Peak number used of current channels..... 4
Peak number used of active channels ..... 0
MAXCHL. Max allowed current channels..... 9999
ACTCHL. Max allowed active channels..... 9999
TCPCHL. Max allowed TCP/IP channels..... 9999
LU62CHL. Max allowed LU62 channels..... 200
Storage used by Chinit..... 436MB
```

- Use the MP1B statistics and accounting support pack
- Ability to extract various key information such as statistics, SSL details, accounting, and other data that can be used for tuning and troubleshooting
- Set the trace to collect channel statistics START TRACE (STAT) CLASS (04)
- For more information about trace settings, see *Table 3 of the START TRACE command in the IBM MQ V8 Knowledge Center*

© Copyright IBM Corporation 2015

Figure 11-18. Channel initiator statistics and accounting data

WM3021.1

Notes:

Unit summary

- Differentiate between queue manager and IBM MQ problems
- Identify the cause of channel problems
- Examine performance-related data

© Copyright IBM Corporation 2015

Figure 11-19. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. What are some ways to troubleshoot a lost message?
 - a. If you are able to reproduce the “message loss,” enable MONQ in the queue manager and queues along the path by issuing DIS QSTATUS
 - b. Checking the channel status along the message path
 - c. Check the dead-letter queue
 - d. Confirm with the application that they did not receive an IBM MQ error code
 - e. If new definitions, verify the new IBM MQ objects
2. True or false: To collect channel initiator and statistics information, you enable the trace using START TRACE(STAT) CLASS(04) and format results using the MP1B support pack.

© Copyright IBM Corporation 2015

Figure 11-20. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.

Checkpoint questions (2 of 2)

3. Which of the following resources and actions are available to the IBM MQ administrator to develop an ongoing performance review plan?
 - a. Review the IBM MQ V8 for z/OS performance support pack MP1J
 - b. Collect and analyze queue manager and channel initiator statistics
 - c. Review older versions of the IBM MQ performance support packs
 - d. Conduct regular calls with application developers and architects to ensure that good design and coding practices are being followed
 - e. Understand security needs and implement those security capabilities that are required

© Copyright IBM Corporation 2015

Figure 11-21. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

Checkpoint answers (1 of 2)

1. What are some ways to troubleshoot a lost message?
 - a. If you are able to reproduce the “message loss,” enable MONQ in the queue manager and queues along the path by issuing DIS QSTATUS
 - b. Checking the channel status along the message path
 - c. Check the dead-letter queue
 - d. Confirm with the application that they did not receive an IBM MQ error code
 - e. If new definitions, verify the new IBM MQ objects

Answers: All answers are correct

2. True or false: To collect channel initiator and statistics information, you enable the trace using START TRACE(STAT) CLASS(04) and format results using the MP1B support pack.

Answer: True

© Copyright IBM Corporation 2015

Figure 11-22. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. Which of the following resources and actions are available to the IBM MQ administrator to develop an ongoing performance review plan?
 - a. Review the IBM MQ V8 for z/OS performance support pack MP1J
 - b. Collect and analyze queue manager and channel initiator statistics
 - c. Review older versions of the IBM MQ performance support packs
 - d. Conduct regular calls with application developers and architects to ensure that good design and coding practices are being followed
 - e. Understand security needs and implement those security capabilities that are required

Answer: All answers are correct. Some security capabilities such as encryption of data at rest and SSL channels can also impact performance, and any security that is applied should be justified.

© Copyright IBM Corporation 2015

Figure 11-23. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Unit 12. IBM MQ Managed File Transfer

What this unit is about

This unit introduces the IBM MQ Managed File Transfer component, describes installation and configuration, explains ways of initiating a file transfer, and summarizes auditing capabilities and other transfer options.

What you should be able to do

After completing this unit, you should be able to:

- Summarize the capabilities of IBM MQ Managed File Transfer
- List the key components of IBM MQ Managed File Transfer
- Describe how to configure an IBM MQ Managed File Transfer agent and supporting environment
- List the mechanisms that are available to initiate a file transfer
- Describe facilities that are available to display transfer status
- Explain how to perform basic problem determination
- Describe the IBM MQ File Transfer logger component

Unit objectives

- Summarize the capabilities of IBM MQ Managed File Transfer
- List the key components of IBM MQ Managed File Transfer
- Describe how to configure an IBM MQ Managed File Transfer agent and supporting environment
- List the mechanisms that are available to initiate a file transfer
- Describe facilities that are available to display transfer status
- Explain how to perform basic problem determination
- Describe the IBM MQ File Transfer logger component

© Copyright IBM Corporation 2015

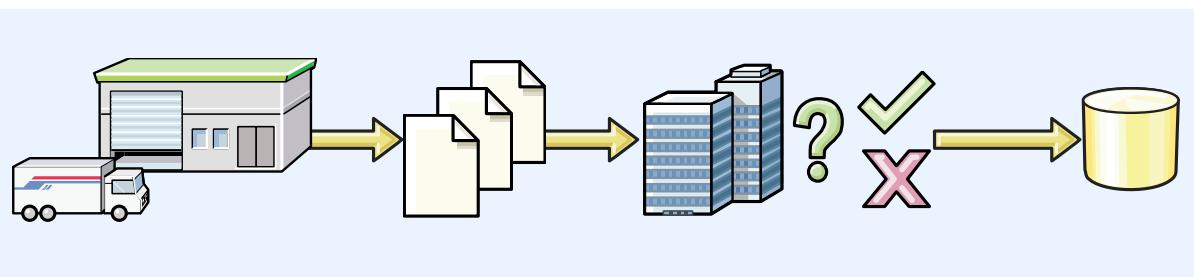
Figure 12-1. Unit objectives

WM3021.1

Notes:

Introducing IBM MQ Managed File Transfer

- Reliable, auditable text and binary file transfer capabilities
- Transfer files over WebSphere MQ network or FTP, FTPS, or SFTP
- Different choices to initiate file transfers, such as:
 - Set up a transfer schedule with WebSphere MQ managed file transfer mechanisms
 - Manually on demand by command line or WebSphere MQ Explorer
 - Placing specially formatted message in agent command queue
 - Scripts



© Copyright IBM Corporation 2015

Figure 12-2. Introducing IBM MQ Managed File Transfer

WM3021.1

Notes:

IBM MQ Managed File Transfer adds the required capabilities to the infrastructure. Originally purchased from a Business Partner, IBM MQ Managed File Transfer was acquired by IBM and offered as a separate product. With IBM MQ V7.5, it was incorporated into IBM MQ, although it remains a differently priced option with separate SMP/E installation.

With IBM MQ Managed File Transfer organizations can exchange files over IBM MQ queues, or if required over FTP through the protocol bridge.

Some examples of the flexibility that is provided by IBM MQ Managed File Transfer are:

- Organizations can stop maintaining elaborate scripts and defer monitoring and scheduling of transfers to IBM MQ Managed File Transfer.
- A transfer can be manually initiated with various options.
- Products such as IBM Integration Bus (formerly WebSphere Message Broker) can place special XML messages in the agent's command queue, which contain the transfer specification, and initiate a transfer from a message flow without using an IBM Integration Bus or WebSphere Message Broker native IBM MQ Managed File Transfer node.

Main components and product options

- WebSphere MQ File Transfer Agent
 - Includes file transfer agent
 - Transfers files over WebSphere MQ messages
 - Installed on a server with or without a local WebSphere MQ server
- WebSphere MQ File Transfer Service
 - Includes all capabilities of the WebSphere MQ File Transfer Agent
 - Is able to use the protocol bridge to exchange files with FTP, FTPS, or SFTP
 - Requires installation with a co-located WebSphere MQ server
- WebSphere MQ File Transfer Logger to maintain audit records on a database or flat file
- WebSphere MQ File Transfer Tools
- WebSphere MQ File Transfer basic base installation (UNIX platforms only)

© Copyright IBM Corporation 2015

Figure 12-3. Main components and product options

WM3021.1

Notes:

If you worked with IBM MQ Managed File Transfer in the past, you might find some changes in terminology. Here are the main IBM MQ Managed File Transfer components:

- IBM MQ or WebSphere MQ File Transfer Agent: In versions of IBM MQ Managed File Transfer before IBM MQ V7.5, the File Transfer Agent was an WebSphere MQ File Transfer Edition Client.
- IBM MQ or WebSphere MQ File Transfer Service is the full WebSphere MQ File Transfer Edition pre-IBM MQ V7.5 WebSphere MQ File Transfer Edition server.
- IBM MQ or WebSphere MQ File Transfer Logger is the former database logger.
- IBM MQ or WebSphere MQ File Transfer Tools have kept the same name.

Sample WebSphere MQ managed file transfer architecture

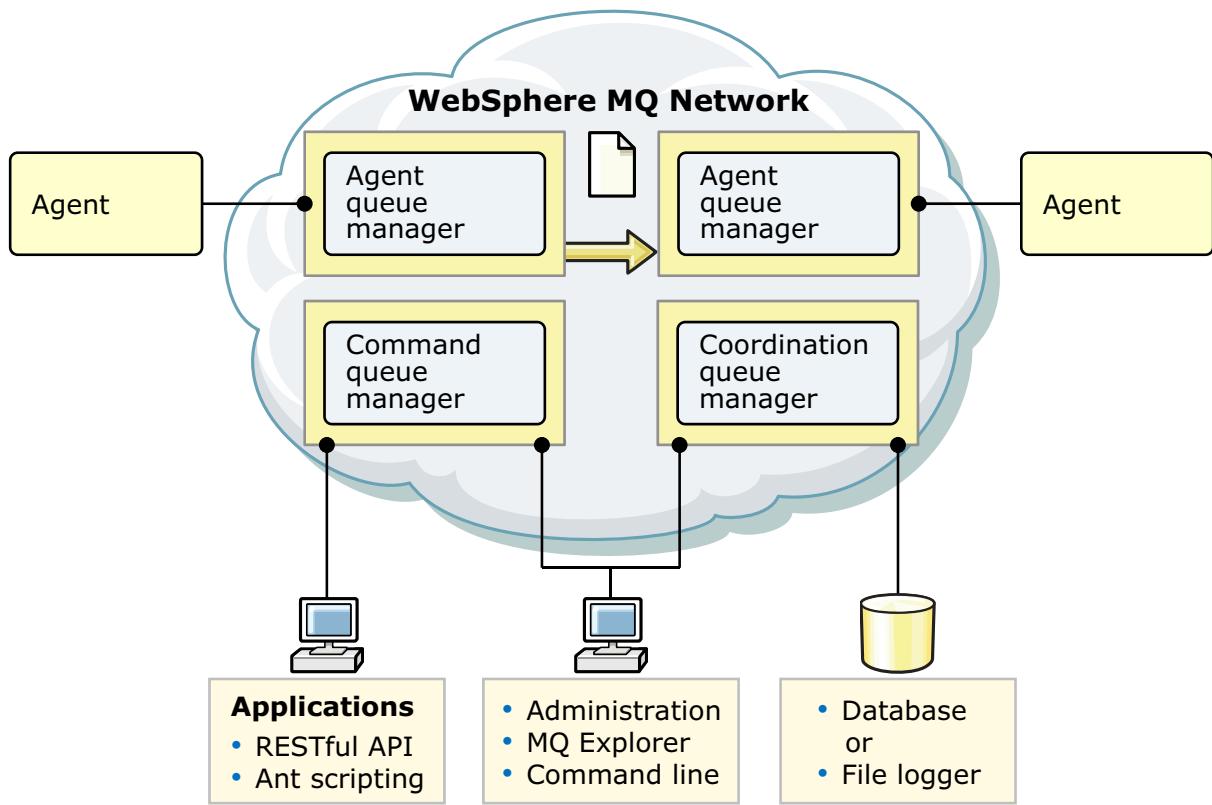


Figure 12-4. Sample WebSphere MQ managed file transfer architecture

WM3021.1

Notes:

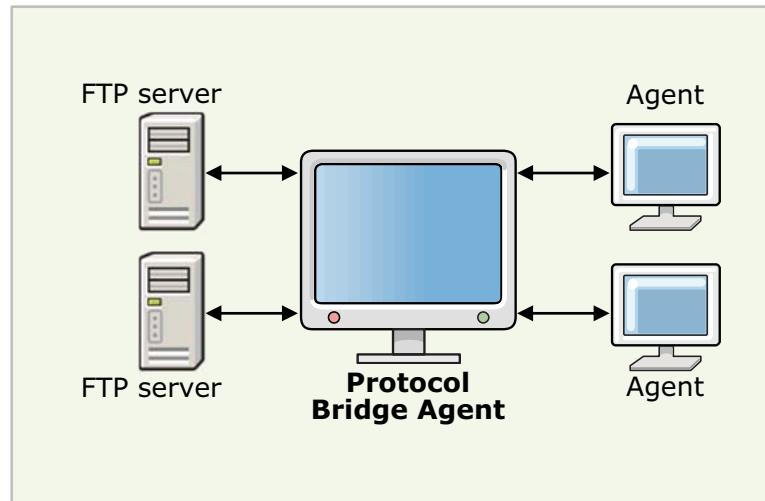
IBM MQ Managed File Transfer uses a queue manager to serve three distinct roles. These roles can be served by the same queue manager, or by three different queue managers. When starting to work with IBM MQ Managed File Transfer, it is best to start with one queue manager until the basic concepts are understood. The roles are:

- Coordination queue manager is a single point for the collection of agent transfer information. The coordination queue manager might be temporarily unavailable for a transfer to take place. When you complete your IBM MQ Managed File Transfer configuration, you see how the agents “line up” under, or belong to, a particular coordination configuration. The configuration name carries the same name as the coordination queue manager. It is possible for agents in other IBM MQ Managed File Transfer servers to be configured to a single coordination queue manager.
- Command queue manager is used to send commands to an agent, or to a coordination queue manager to obtain information.
- The agent queue managers register with the coordination queue manager and use publish/subscribe to publish details to the coordination queue manager.

By default, file transfers are carried by segmenting the file contents and sending them over IBM MQ queues. IBM MQ Managed File Transfer agent segments the files, then restores the file from the messages upon arrival to the transfer destination. IBM MQ Managed File Transfer can use FTP or FTPS if the protocol bridge is configured.

Protocol Bridge

- Enables the Managed File Transfer exchange files with FTP, SFTP, and FTPS servers
- Upload files from the Managed File Transfer network to a remote server with FTP, FTPS, or SFTP
- Download files from a remote server to the Managed File Transfer network by using FTP, FTPS, or SFTP
- Managed Transfer Agent is always required to work with the Protocol Bridge Agent



© Copyright IBM Corporation 2015

Figure 12-5. Protocol bridge

WM3021.1

Notes:

If you installed the IBM MQ Managed File Transfer Managed Transfer Agent, or in the older terminology a full WebSphere MQ File Transfer Edition server, you can configure the protocol bridge to transfer files by using FTP, SFTP, or FTPS instead of sending the segmented files over the IBM MQ network. This option is good for large files, and might be a requirement for exchanging information with servers that only have FTP.

A protocol bridge agent is always between an FTP, SFTP, or FTPS server and an IBM MQ Managed File Transfer agent. The exchanges can be:

- Download from an FTP, FTPS, or SFTP server to the IBM MQ Managed File Transfer infrastructure.
- Download from the IBM MQ Managed File Transfer infrastructure to an FTP, FTPS, or SFTP server.

Protocol bridge can support servers that allow access to files by the absolute file path. Protocol bridge tries to convert a relative path to an absolute path by using the home directory that is used to log on to the protocol server. Servers that only allow access to the current directory are not supported.

Required IBM MQ connectivity paths (1 of 2)

- Agent communicating to another agent
 - Agents that exchange files need bidirectional communication
 - This path carries the file data and should be selected carefully as far as speed and cost depending on organizational
- Agent to coordination
 - The agents involved in a transfer need bidirectional communication with the coordination queue manager

© Copyright IBM Corporation 2015

Figure 12-6. Required IBM MQ connectivity paths (1 of 2)

WM3021.1

Notes:

Bindings connections refer to sender-receiver connections.

The context of this slide acknowledges that the IBM MQ Managed File Transfer configuration can use one queue manager for all three roles, or different queue managers.

The first IBM MQ Managed File Transfer agent that you configure in the lab uses the same queue manager for all three roles. When you need a different configuration, these connectivity requirements can be reviewed.

Required IBM MQ connectivity paths (2 of 2)

- Command to agent
 - Applications queue managers and IBM MQ Explorer need to be able to send commands to the agent queue managers
 - To receive responses, a bidirectional connection must be used
- Before attempting to use IBM MQ Managed File Transfer, bidirectional Sender and Receiver channels should be set up and tested between all queue managers that use IBM MQ Managed File Transfer

Terminology clarification:



- The terms “bind” and “bindings” appear in different contexts in IBM MQ
- IBM MQ Managed File Transfer command displays and documentation use the term “bindings mode” to refer to message channels, and “client mode” to refer to MQI channels

© Copyright IBM Corporation 2015

Figure 12-7. Required IBM MQ connectivity paths (2 of 2)

WM3021.1

Notes:

Steps to configure an agent on z/OS

1. Complete and test required IBM MQ channel definitions noted in connectivity paths slide
2. Review Program Directory for IBM MQ Managed File Transfer for z/OS V8.0.0
 - UNIX System Services configuration
 - Installation user ID requirements
3. Review *Configuration tasks for Managed File Transfer* in IBM MQ Knowledge Center
4. Complete SMP/E process
5. Copy files from SMP/E to a working partitioned data set (PDS)
6. Confirm all values that are needed for BFGCUSTM job
7. Run BFGCUSTM to generate customized PDS work members
8. Set up coordination queue manager: BFGCFCR
9. Set up command queue manager
10. Create agent
11. Confirm that the agent-related queues were created before starting the agent



- Step 1 testing is critical, particularly having correct permissions
- Most IBM MQ Managed File Transfer configuration problems are caused by IBM MQ security and object definition issues

© Copyright IBM Corporation 2015

Figure 12-8. Steps to configure an agent on z/OS

WM3021.1

Notes:

IBM MQ Managed File Transfer uses the UNIX System Services (USS). Instructions on setup of the z/OS UNIX System Services are in the program directory that is referenced in this slide. There are other setup details, which are omitted for brevity, that must be confirmed in the program directory.

This unit picks up the process on step 6, copying the IBM MQ Managed File Transfer installation files to a working directory.

During the creation of the IBM MQ Managed File Transfer environment and through this unit, you alternate between z/OS and z/OS UNIX System Services environments.

Copy the IBM MQFT libraries

```
//IEBCOPY EXEC PGM=IEBCOPY,REGION=0M  
//SYSPRINT DD SYSOUT=*  
//SYSIN    DD DUMMY  
//SYSUT1   DD DSN=MQ8000.SBFGCMDS,DISP=SHR  
//SYSUT2   DD DSN=D62WW.WM302V1.MQMFT,DISP=(,CATLG),  
=====
```

© Copyright IBM Corporation 2015

Figure 12-9. Copy the IBM MQFT libraries

WM3021.1

Notes:

You run a standard IEBCOPY to create the working IBM MQ Managed File Transfer PDS. Before proceeding to the next steps, confirm that all prerequisite work with the z/OS UNIX System Services file system, the IBM MQ infrastructure, and authorizations is completed.

Partial list of jobs that are generated by BFGCUSTM

Name	Description
BFGCOPY	Job to create a copy of this library
BFGCUSTM	Job to customize this library for agent or logger
BFGCFCR	<code>fteSetupCoordination</code> : Sets up a coordination queue manager configuration
BFGCMCR	<code>fteSetupCommands</code> : Sets up the command queue manager
BFGAGCR	<code>fteCreateAgent</code> : Creates the agent
BFGAGST	<code>fteStartAgent</code> : Start agent job
BFGAGSTP	<code>fteStartAgent procedure</code> : Start agent started procedure
BFGAGPI	<code>ftePingAgent</code>
BFGAGSP	<code>fteStopAgent</code> : Stops the agent
BFGAGCL	<code>fteCleanAgent</code> : Used sometimes if there are problems
BFGAGDE	<code>fteDeleteAgent</code> : Deletes an agent
BFGPRSH	<code>fteDisplayVersion</code>
BFGAGLI	<code>fteListAgents</code> : Display agents and status (used after starting agent)
BFGTRCAS	<code>fteCancelTransfer sample</code> : Cancel a transfer
BFGTRCRS	<code>fteCreateTransfer sample</code> : Create a transfer

© Copyright IBM Corporation 2015

Figure 12-10. Partial list of jobs that are generated by BFGCUSTM

WM3021.1

Notes:

Before you start, make a backup of the BFGCUSTM PDS member.

BFGCUSTM is a critical job and holds all the parameters that go into your IBM MQ Managed File Transfer configuration.

- Do not proceed to run this job until you have confirmed the required information.
- BFGCUSTM overwrites the PDS that it is a member of and requires exclusive use when running.
- Read the JCL in-line documentation for some detailed caveats.
- After the job runs, refrain from altering the generated members; if individual members are customized and you need to rerun BFGCUSTM (as permitted by the caveats that are documented in the JCL), any customizations would be overwritten **unless any changed member is renamed**.

The JCL members are documented in section *z/OS agent and logger command JCL scripts* in the IBM MQ Knowledge Center. You use a set of these members in the lab.

BFGCUSTM (1 of 2)

```
# Java installation directory USS path
BFG_JAVA_HOME=/usr/lpp/java/J7.0
# Use if queue sharing group or leave blank
BFG_GROUP_NAME=
# USS path to IBM MQ MFT product installation
PGM=/usr/lpp/mqmfte/V8R0M0/bin/
# PDSE library for IBM MQ MFT JCL and other members
LIBRARY=D62WW.WM302V1.MQMFT
#
TMPDIR=/tmp
#AGENT or LOGGER build type variable
SERVICE_TYPE=AGENT
# Name of agent or logger - this will usually take the queue manager name
NAME=MQ0B
# USS root path for product installation
BFG_PROD=/usr/lpp/mqmfte/V8R0M0
# USS configuration directory
BFG_DATA=/var/mqmft
```

© Copyright IBM Corporation 2015

Figure 12-11. BFGCUSTM (1 of 2)

WM3021.1

Notes:

These two slides show the parameters that need to be provided to the BFGCUSTM job. The parameters that you use in the lab are slightly different from the parameters that are shown on this slide.

BFGCUSTM (2 of 2)

```

#
QMGR=MQ0B
#
# MQ_PATH variable
#
MQ_PATH=/usr/lpp/mqm/V8R0M0
#
# MQ_HLQ variable
#
MQ_HLQ=MQ8000
#
# MQ_LANG variable
MQ_LANG=E
#
FTE_CONFIG=
#CREDENTIAL_PATH Optional variable USS path credential file for migration
CREDENTIAL_PATH=
# DB_PROPS_PATH Optional variable
DB_PROPS_PATH=

```

© Copyright IBM Corporation 2015

Figure 12-12. BFGCUSTM (2 of 2)

WM3021.1

Notes:

The name that you use for the QMGR parameter becomes your configuration name in the z/OS UNIX System Services file structure.

These parameters are documented in *Before you start* in the section that is called *Configuration tasks for Managed File Transfer on z/OS*.

**Hint**

This next set of slides walk through the setup you are completing in the IBM MQ Managed File Transfer lab.

Partial output of BFGCUSTM job

```
%BFGZCMD STDIN,TMPDIR=/tmp, PGM=/usr/lpp/mqmfte/V8R0M0/bin/fteCustom
READY
END
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2014. ALL RIGHTS RESERVED
Creating command script: BFGCROBS from template: BFGXCROB
Creating command script: BFGMNCRS from template: BFGXMNCR
Creating command script: BFGMNDES from template: BFGXMNDE
Creating command script: BFGPRANS from template: BFGXPRAN
Creating command script: BFGSTDDES from template: BFGXSTDE
Creating command script: BFGTMCRS from template: BFGXTMCR
Creating command script: BFGTMDES from template: BFGXTMDE
Creating command script: BFGTRCAS from template: BFGXTRCA
Creating command script: BFGTRCRS from template: BFGXTRCR
Creating command script: BFGAGSTP from template: BFGYAGST
Creating command script: BFGAGCL from template: BFGZAGCL
Creating command script: BFGAGCR from template: BFGZAGCR
Creating command script: BFGAGDE from template: BFGZAGDE
Creating command script: BFGAGLI from template: BFGZAGLI
Creating command script: BFGAGMG from template: BFGZAGMG
...
...
```

© Copyright IBM Corporation 2015

Figure 12-13. Partial output of BFGCUSTM job

WM3021.1

Notes:

When BFGCUSTM runs, you see the different JCL members being created. You are after three members to configure the IBM MQ Managed File Transfer environment, then you look at two other members to start and display the agent.

1. Set up the coordination queue manager: BFGCFCR
2. Set up the command queue manager: BFGCMCR
3. Create the agent: BFGAGCR
4. Start the agent: You are using the JCL: BFGAGST
5. Agent started procedure: BFGAGSTP
6. List agent: BFGAGLI
7. Manually start a transfer: BFGTRCRS. Back up before using.
8. Display all information about the agent: BFGAGSH
9. Stop the agent: BFGAGSP

Set up coordination: BFGCFCR

```
//*****
//* Create Configuration
//BFGCMD EXEC PGM=IKJEFT01,REGION=0M
//SYSEXEC DD DSN=D62WW.WM302V1.MQMFT,DISP=SHR
//SYSTSIN DD *
%BFGCMD CMD=fteSetupCoordination +
-coordinationQMgr MQ0B +
//*****
//* Update properties
//BFGPRP EXEC PGM=IKJEFT01,REGION=0M,COND=(0,NE)
//SYSTSIN DD *
%BFGCMD CMD=fteCustom -updateProps +
D62WW.WM302V1.MQMFT +
COORDINATION +
MQ0B +
coordinationQMgrHost +
coordinationQMgrPort +
coordinationQMgrChannel +
coordinationQMgrAuthenticationCredentialsFile
```

© Copyright IBM Corporation 2015

Figure 12-14. Set up coordination: BFGCFCR

WM3021.1

Notes:

The job to set up the coordination queue manager invokes the `fteSetupCoordination` command. The only parameter that is used, `-coordination QMgr`, is the name of the coordination queue manager, MQ0B.

The name of the configuration queue manager is used to create the configuration directory structure for this environment.

You notice that in this definition no additional information is provided for the queue manager host, port, or channel. The reason is because this component uses “bindings” or sender-receiver connections rather than client MQI connections.

After running the JCL with “COND CODE 0” completion, you check z/OS UNIX System Services.

**Note**

You are using the JCL members to create the configuration. If you are familiar with z/OS UNIX System Services commands, you can use the various fte-prefixed commands that are shown on the slides to create the configuration.

Set up coordination results

OMVS ...

```
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>ls -l
total 40
-rw-r--r--  1 INGM000  OMVSGRP      607 Sep 22 20:48 MQ0B.mqsc
drwsr-xr-x  3 INGM000  OMVSGRP    8192 Sep 22 22:12 agents
-rw-r--r--  1 INGM000  OMVSGRP      50 Sep 22 22:12
  command.properties
-rw-r--r--  1 INGM000  OMVSGRP     52 Sep 22 22:12
  coordination.properties
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>cat coord*
#Mon Sep 22 22:12:17 GMT 2014
coordinationQMgr=MQ0B
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>
```

© Copyright IBM Corporation 2015

Figure 12-15. Set up coordination results

WM3021.1

Notes:

If you go into z/OS UNIX System Services, you see the results of BFGCFCR:

- A directory structure is created under <xxxx>/var/mqmft/mqft/config/MQ0B, where MQ0B is the name of your coordination queue manager. This name is also the configuration name.
- A file called MQ0B.mqsc contains definitions for IBM MQ objects that support the coordination queue manager. In distributed platforms, you must manually create these object definitions with runmqsc, but in z/OS the definitions are created as part of BFGCFCR.
- A file is named coordination.properties. This setting configures the default IBM MQ Managed File Transfer configuration for this server. If you run certain IBM MQ Managed File Transfer commands, such as fteListAgents, if you want to list information pertinent to the MQ0B configuration, it is not necessary to specify the configuration name by use of the -p parameter.
- There can be more than one IBM MQ Managed File Transfer configuration in the same server. It would be listed as a separate directory under the config directory.

This output on the slide was captured after all the jobs for coordination, command, and agent were run, so you can see all the files. Normally you would only see MQ0B.mqsc, and the coordination.properties file. You now set up the command queue manager.

Set up commands: BFGCMCR

```

//*****  

//BFGCMD EXEC PGM=IKJEFT01,REGION=0M  

... ... ... ...  

//SYSTSIN DD *  

%BFGCMD CMD=fteSetupCommands +  

  -connectionQMgr MQ0B +  

  -p MQ0B  

/*  

/* Add -f parameter to force overwrite of existing configuration  

//*****  

//BFGCMD EXEC PGM=IKJEFT01,REGION=0M  

//SYSEXEC DD DSN=D62WW.WM302V1.MQMFT,DISP=SHR  

//SYSTSPRT DD SYSOUT=*  

//STDOUT DD SYSOUT=*  

//STDERR DD SYSOUT=*  

//SYSTSIN DD *  

%BFGCMD CMD=fteSetupCommands +  

  -connectionQMgr MQ0B +  

  -p MQ0B  

/*

```

© Copyright IBM Corporation 2015

Figure 12-16. Set up commands: BFGCMCR

WM3021.1

Notes:

The job looks similar. You can see the `-p` parameter pointing to the MQ0B configuration. This job runs the `fteSetupCommands`.

Set up commands results

```
OMVS ...
```

```
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>cat command*
#Mon Sep 22 22:12:17 GMT 2014
connectionQMgr=MQ0B
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>
```

© Copyright IBM Corporation 2015

Figure 12-17. Set up commands results

WM3021.1

Notes:

You now look at the `command.properties` file. It is showing that MQ0B is the queue manager to connect to.

Create agent: BFGAGCR

```
//BFGCMD EXEC PGM=IKJEFT01,REGION=0M
//SYSTSIN DD *
%BFGCMD CMD=fteCreateAgent +
-agentName MQ0B +
-agentQMgr MQ0B +
-p MQ0B
/*
//BFGPRP EXEC PGM=IKJEFT01,REGION=0M,COND=(0,NE)
//SYSTSIN DD *
%BFGCMD CMD=fteCustom -updateProps +
D62WW.WM302V1.MQMFT +
AGENT MQ0B +
agentName +
agentDesc +
agentQMgr +
agentQMgrHost +
agentQMgrPort +
agentQMgrChannel +
authorityChecking +
agentQMgrAuthenticationCredentialsFile
```

© Copyright IBM Corporation 2015

Figure 12-18. Create agent: BFGAGCR

WM3021.1

Notes:

You now look at the job to create the agent. The agent name can be other than the configuration name, for instance it might be BILLAG1, but this screen display used MQ0B as the agent name.

No queue manager host, port, or any other information is provided because it is not using client connections, it is using bindings, or message channels.

Create agent results

OMVS ...

```
INGM001:/MC14/var/mqmft/mqft/config/MQ0B:>cd agents
INGM001:/MC14/var/mqmft/mqft/config/MQ0B/agents:>ls -l
total 16
drwsr-xr-x  3 INGM000  OMVSGRP      8192 Sep 22 22:12 MQ0B
INGM001:/MC14/var/mqmft/mqft/config/MQ0B/agents:>cd M*
INGM001:/MC14/var/mqmft/mqft/config/MQ0B/agents/MQ0B:>ls -l
total 56
-rw-r--r--  1 INGM000  OMVSGRP     2550 Sep 22 22:12 MQ0B_create.mqsc
-rw-r--r--  1 INGM000  OMVSGRP      841 Sep 22 22:12 MQ0B_delete.mqsc
-rw-r--r--  1 INGM000  OMVSGRP    6553 Sep 22 22:12 UserSandboxes.xml
-rw-r--r--  1 INGM000  OMVSGRP       60 Sep 22 22:12 agent.properties
drwsr-xr-x  2 INGM000  OMVSGRP      8192 Sep 22 22:12 exits
INGM001:/MC14/var/mqmft/mqft/config/MQ0B/agents/MQ0B:>cat agent.*
#Mon Sep 22 22:12:17 GMT 2014
agentQMgr=MQ0B
agentName=MQ0B
```

© Copyright IBM Corporation 2015

Figure 12-19. Create agent results

WM3021.1

Notes:

After BGCAGCR runs the command `fteCreateAgent`, a subdirectory to hold the components pertinent to the agent is created in the `agents` subdirectory. So in the directory structure, you have a first MQ0B to represent the configuration name, and a second MQ0B under the `agents` subdirectory that is the agent name.

There is also another `MQ0B_create.mqsc` script file to create IBM MQ objects that support the agent. If you displayed some of the system queues, you would see queues such as:

`SYSTEM.FTE.COMMAND.MQ0B`, `SYSTEM.FTE.DATA.MQ0B`, `SYSTEM.FTE.EVENT.MQ0B`

The `SYSTEM.FTE` queues that support the agent are suffixed with the agent name, `MQ0B`. You also see a `SYSTEM.FTE` topic, which was created with the coordination queue manager.

The `agent.properties` file shows the name of agent's queue manager, and agent name.

So now you have the three components, coordination, command, and agent. It is now time to start the agent.

Start agent: BFGAGST

```
//*****  
//BFGCMD EXEC PGM=IKJEFT01,REGION=0M,  
//          PARM=' %BFGCMD CMD=fteStartAgent -p MQ0B -F MQ0B'
```

- Normally uses the started task in member **BGAGSTP**

© Copyright IBM Corporation 2015

Figure 12-20. Start agent: BFGAGST

WM3021.1

Notes:

In an application environment, you use the started task procedure. For the lab, the start JCL is used. The command that is issued is `fteStartAgent`.

Display agent and status fteListAgent: BFGAGLI

```
//*****
//BFGCMD EXEC PGM=IKJEFT01,REGION=0M
... ... ...
//SYSTSIN DD *
%BFGCMD CMD=fteListAgents +
-v -p MQ0B
```

Results:

Agent Name:	Queue Manager Name:	Transfers:	Status:
(Source/Destination)			
MQ0B	MQ0B	0/0	READY

© Copyright IBM Corporation 2015

Figure 12-21. Display agent and status fteListAgent: BFGAGLI

WM3021.1

Notes:

After you start the agent, you must display it to check its status. The JCL invokes the fteListAgents command. A good start should show a status of READY for the agent. If the status is other than READY, you need to look at the IBM MQ Managed File Transfer logs to determine the problem.

Agent logs (1 of 2)

```
/var/mqmft/mqft/logs/MQOB/agents/MQOB/logs:>cat output0.log
***** Start Display Current Environment *****
Build level: V8.0.0.0 f000-20140506-1626
Java runtime version:
    JRE 1.7.0 IBM J9 2.6 z/OS s390-31 20131013_170512 (JIT enabled, AOT
enabled)
        J9VM - R26_Java726_SR6_20131013_1510_B170512
        JIT   - r11.b05_20131003_47443
        GC    - R26_Java726_SR6_20131013_1510_B170512
        J9CL - 20131013_170512
The maximum amount of memory that the Java virtual machine will attempt
to use is: '512'MB ... ... CU4J version: 52.1.0.0
Properties:
    agentName=MQOB, agentQMgr=MQOB, agentType=STANDARD,
    coordinationQMgr=MQOB, defaultProperties=MQOB
Install Locations:
    com.ibm.wmqfte.product.root=/V2R1/usr/lpp/mqmft/V8R0M0/mqft
WebSphere MQ Components:
    WebSphere MQ classes for Java / / unknown
    Common Services for Java Platform, Standard Edition / 8.0.0.0 / p000-
L140429.1
```

© Copyright IBM Corporation 2015

Figure 12-22. Agent logs (1 of 2)

WM3021.1

Notes:

This agent started and went to READY state without problems, but you continue to look at the log to see the normal messages that display when the agent starts.

The log is in a different directory path than the config files. You look in directory /var/mqmft/mqft/logs/MQOB/agents/MQOB/logs, substituting your configuration name for MQOB. Then, view the file output0.log.

- Information about the product and the Java run time are shown.
- The memory message.
- The properties of the agents are listed. More information about the logs on the next slide.

Agent logs (2 of 2)

```
***** End Display Current Environment *****
Ý22/09/2014 22:24:55:376 GMT" 00000001 IFAEDJRegWrap I
BFGLM0016I: The MVS Product Registration Service reports that the agent
is permitted to start.
Ý22/09/2014 22:24:58:047 GMT" 00000001 CredentialsUs W BFGPR0127W: No
credentials file has been specified to connect to WebSphere MQ.
Therefore, the assumption is that WebSphere MQ authentication has been
disabled.
Ý22/09/2014 22:24:58:484 GMT" 00000001 Agent I BFGAG0115I:
Relative path transfer root directory: /u/wm30/ingm000
Ý22/09/2014 22:24:58:487 GMT" 00000001 Agent W BFGAG0125W: The
maximum size to which the java heap can grow is '512'MB, which is the
default value. This value may be too low dependent on the agent's work
load.
Ý22/09/2014 22:24:58:489 GMT" 00000001 AgentRuntime I BFGAG0058I: The
agent has successfully initialized.
Ý22/09/2014 22:24:58:571 GMT" 00000001 AgentRuntime I BFGAG0059I: The
agent has been successfully started.
```

© Copyright IBM Corporation 2015

Figure 12-23. Agent logs (2 of 2)

WM3021.1

Notes:

- You now see other interaction with the z/OS system.
- Next, information on credentials. Credentials files are not used in the class environment.
- Information about the relative transfer root directory.
- Finally, the two messages you need, BFGAG0058I for successful initialization, followed by BFGAG0059I for a successful start.

You are now ready to transfer a file.

Create a transfer using `fteCreateTransfer`: BFGTRCRS

- Transfer request issued by JCL member **BFGTRCRS**
- Checking results with IBM MQ Explorer
- For more information about the transfer attributes that are available, see the IBM MQ Knowledge Center

```
//SYSTSIN DD *
  %BFGCMD CMD=fteCreateTransfer -h

  %BFGCMD CMD=fteCreateTransfer +
    -p MQ0B +
    -sa MQ0B +
    -da MQ0B +
    -ds "'//INGM000.SAVE.CSQUTIL1'" +
      "'//INGM000.CSQUTIL.COMMANDS'"
/*
```

	MQ0B	MQ0B	Successful
	//'INGM000.CSQUTIL.COMMANDS'	//'INGM000.SAVE.CSQUTIL1'	Successful

© Copyright IBM Corporation 2015

Figure 12-24. Create a transfer using `fteCreateTransfer`: BFGTRCRS

WM3021.1

Notes:

This is a simple transfer. Assumption is that you used the ISPF panels command option 8, and have some MQSC scripts in the xxxx.CSQUTIL.COMMANDS file you need to save.

First, back up the JCL member BFGTRCRS. Now change the data set name. The example that is shown is not using data set attributes. To specify data set attributes use the syntax that is shown in the following code. Pay special attention to the use of semi-colons and single quotation marks and double quotation marks:

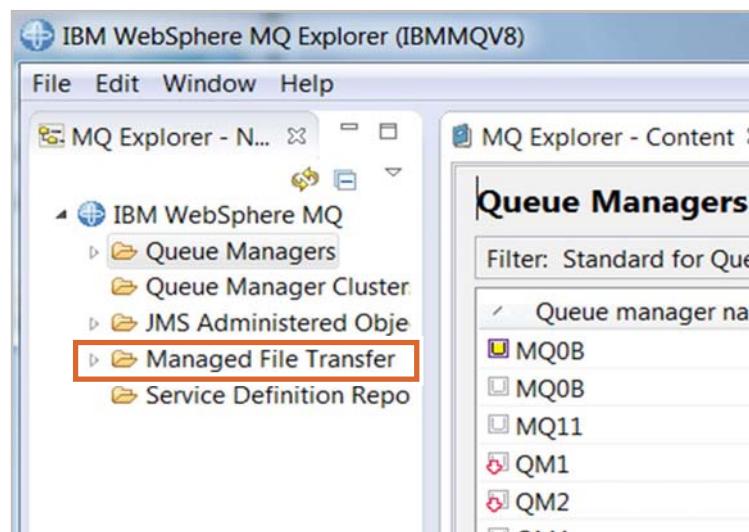
```
%BFGCMD CMD=fteCreateTransfer +
  -p MQ0B +
  -sa MQ0B +
  -da MQ0B +
  -ds "'//INGM000.SAVE.CSQUTIL1';RECFM(F,B);LRECL(80);BLKSIZE(0)" +
    "'//INGM000.CSQUTIL.COMMANDS'"
```

However, because you are only moving a file around in the same server, you can manually check the results, IBM MQ Explorer is used for this purpose. You now look at how to set up IBM MQ Explorer.

Adding an existing configuration to IBM MQ Explorer (1 of 2)

- Add Managed File Transfer queue manager to IBM MQ Explorer

- Place the cursor on the Managed File Transfer (MFT) menu item



- When the MFT welcome page is displayed, scroll down to view the **Add an IBM MQ Managed File Transfer Configuration to WebSphere MQ Explorer** link

© Copyright IBM Corporation 2015

Figure 12-25. Adding an existing configuration to IBM MQ Explorer (1 of 2)

WM3021.1

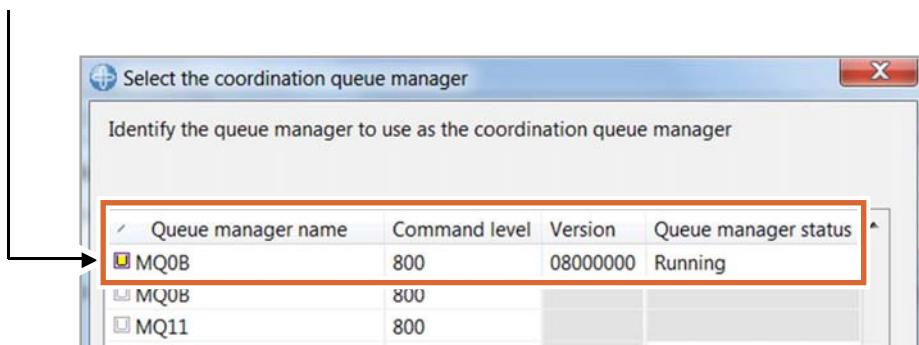
Notes:

To add the IBM MQ Managed File Transfer configuration to IBM MQ Explorer, first add the coordination queue manager to IBM MQ Explorer and ensure that you are able to see channels and queues in that queue manager. For more information, see the next slide.



Adding an existing configuration to IBM MQ Explorer (2 of 2)

- Select the coordination queue manager for the configuration



- Provide the queue manager with information for the command queue manager
- Use the defaults from the last panel and click **Finish**

© Copyright IBM Corporation 2015

Figure 12-26. Adding an existing configuration to IBM MQ Explorer (2 of 2)

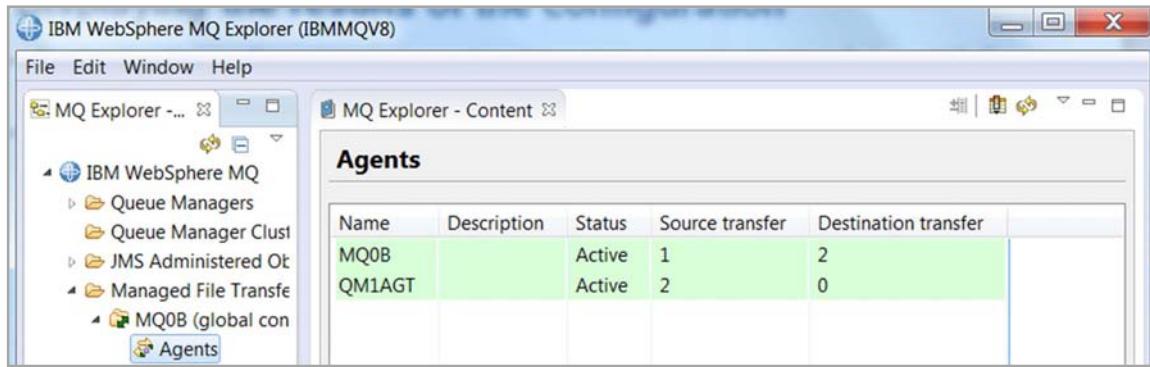
WM3021.1

Notes:

This screen capture shows the queue manager selection view.

Displaying the results of the configuration

- To display the configuration and results of the start request with WebSphere MQ Explorer, connect to the configuration by selecting the configuration name and selecting **Connect**
- To use the command line, type: **fteListAgents**



```
C:\IBM\WebSphere MQ\mqft\config\MQ0B\agents\QM1AGT>fteListAgents
5655-MFT, 5724-H72 Copyright IBM Corp. 2008, 2014. ALL RIGHTS ...
...
Agent Name: Queue Manager Name: Status:
MQ0B MQ0B ACTIVE
QM1AGT QMGR1 ACTIVE
```

© Copyright IBM Corporation 2015

Figure 12-27. Displaying the results of the configuration

WM3021.1

Notes:

After you add the queue manager, right-click the queue manager that you added under the **Managed File Transfer** menu and click **Connect**. You should then see your agent. At the time this course was created, there is a pending fix to correct a problem in this area:

http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.pro.doc/q004060_.htm?lang=en

The fix is due for release as indicated in the link provided, however, it was applied and tested in the lab environment for this course. There is also a workaround that is documented at the end of the notes for this slide.

The screen display shows one coordinator queue manager managing two agents. One of the agents is using a different agent and command queue manager.

If you need to configure IBM MQ Managed File Transfer with separate queue managers, that is a dedicated coordination queue manager, and a separate queue manager to handle the agent and command roles, consult **pages 3-7, IBM techdoc: 7023933, Installation, Configuration and First File Transfer of WebSphere MQ File Transfer Edition 7.0.4 in Windows and Linux** at <http://www.ibm.com/support/docview.wss?uid=swg27023933>

The referenced document is for an older IBM MQ Managed File Transfer edition in distributed platforms, but what you need to understand is the IBM MQ connectivity requirements, which are covered well in this paper. The document also provides a good exercise in testing connectivity between two queue managers. When using this document for z/OS, substitute the z/OS PUT/GET sample programs that are used in the labs for this course, for the distributed amqsput and amqsbcg.



Information

Workaround to use IBM MQ Explorer before application of the documented fix.

Have at least the IBM MQ Managed File Transfer Agent component installed in the same host as IBM MQ Explorer.

Create a partial client configuration on the IBM MQ Explorer server as shown changing the information to reflect your environment. This command reflects the course lab environment values:

Proceed to the Windows Command prompt. Enter the command:

```
fteSetupCoordination -coordinationQMgr MQ##  
-coordinationQMgrHost mvsmc++.ilsvpn.ibm.com  
-coordinationQMgrPort 16##  
-coordinationQMgrChannel TSM00##.SVRCONN
```

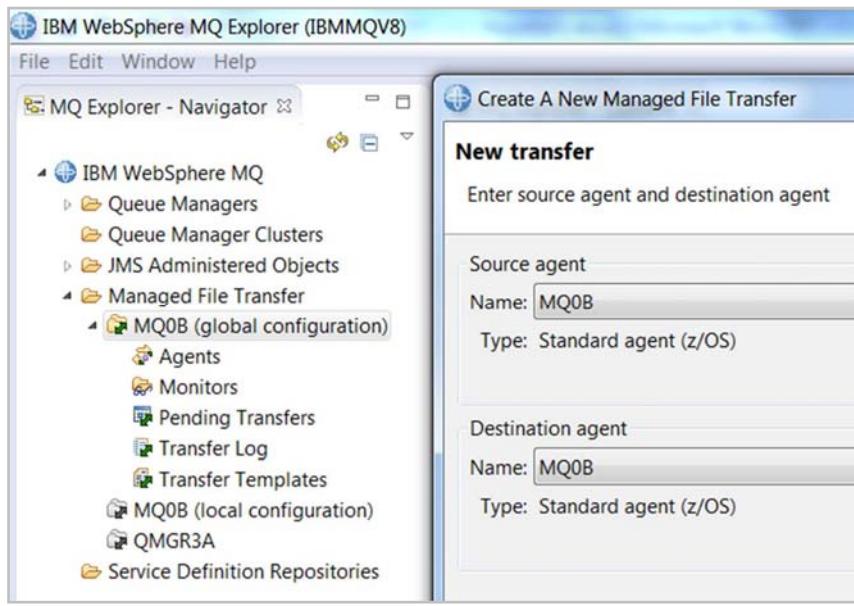
Return to IBM MQ Explorer, you now see a Global denomination for this configuration. Either configuration for the same configuration queue manager should now connect without any problems.

There is a difference when you use the Local configuration or add the Global configuration with the fteSetupCoordination command.

- The local configuration uses the IBM MQ Explorer configuration information. There are no MQFT configurations set up in the IBM MQ Explorer host.
- IBM MQ Managed File Transfer agent component must be installed in the same host as IBM MQ Explorer. The global configuration sets up a coordination queue manager *directory structure* in the IBM MQ Explorer server.

Create a transfer with IBM MQ Explorer (1 of 2)

- From the coordinator queue manager name, right-click and click **New Transfer**
- Select the source and target agents



© Copyright IBM Corporation 2015

Figure 12-28. Create a transfer with IBM MQ Explorer (1 of 2)

WM3021.1

Notes:

You can also do a file transfer by using IBM MQ Explorer. If agents are associated with two queue managers and the IBM MQ connectivity is defined across two queue managers, you can test a transfer across the two queue managers, as shown in the figure.

To run the transfer:

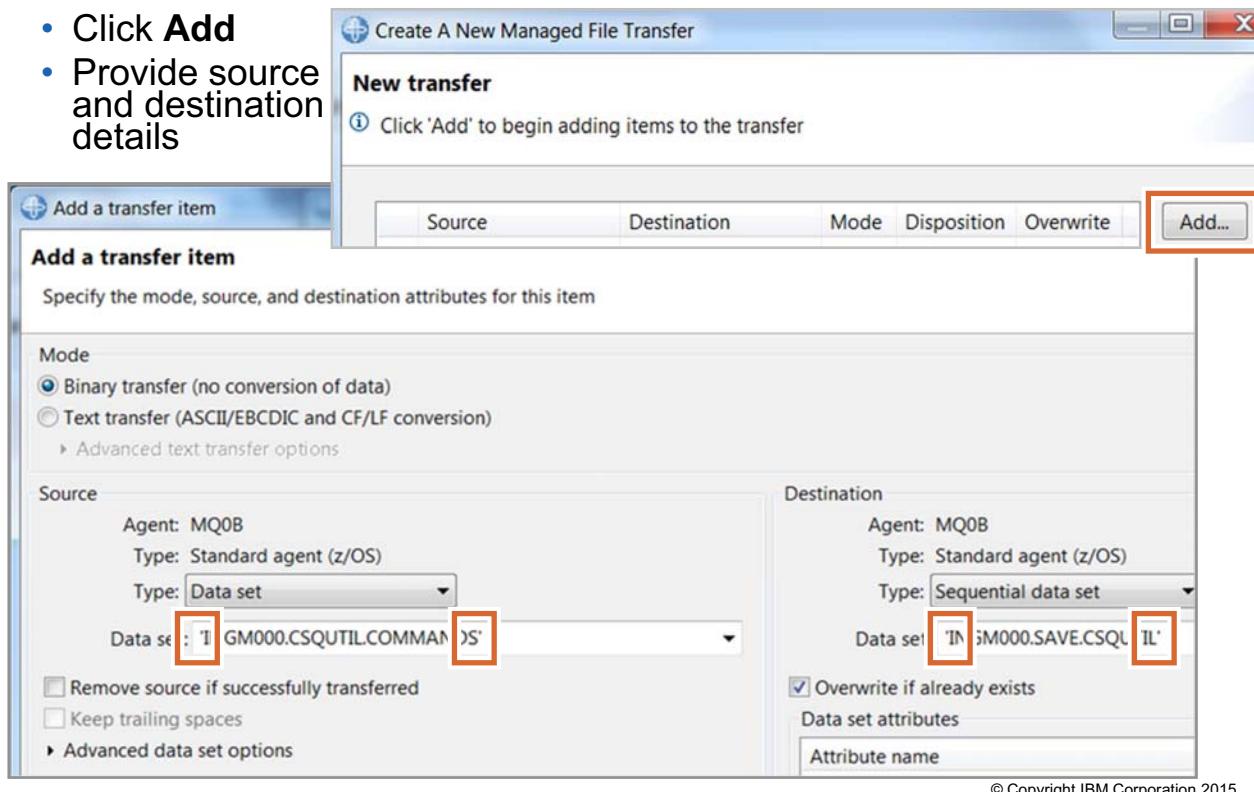
- Right-click the coordinator queue manager name in the **Managed File Transfer** menu, and click **New Transfer**.
- Select the agent names for the originating and target agents.
- Go to the next page by clicking **Next**.

For the remaining steps, see the next slide.

WebSphere Education 

Create a transfer with IBM MQ Explorer (2 of 2)

- Click **Add**
- Provide source and destination details



© Copyright IBM Corporation 2015

Figure 12-29. Create a transfer with IBM MQ Explorer (2 of 2)

WM3021.1

Notes:

- Select the type of data set.
- Enter data set names including quotation marks.



Important

The quotation marks are important.

Checking results (1 of 2)

The screenshot shows the IBM WebSphere MQ Explorer interface. On the left, the navigation tree is expanded to show 'MQ Explorer - N...' and 'IBM WebSphere MQ' with its sub-options like 'Queue Managers', 'JMS Administered Objects', and 'Managed File Transfer'. Under 'Managed File Transfer', 'MQ0B (global config)' is selected, revealing 'Agents', 'Monitors', 'Pending Transfers', 'Transfer Log', and 'Transfer Template'. The main window is titled 'MQ Explorer - Content' and contains a table titled 'Transfer Log'. The table has three columns: 'Source', 'Destination', and 'Completion Status'. The data is as follows:

Source	Destination	Completion Status
MQ0B	MQ0B	Failed
MQ0B	MQ0B	Failed
MQ0B	MQ0B	Successful
MQ0B	QM1AGT	Cancelled
MQ0B	QM1AGT	Starting
MQ0B	MQ0B	Failed
//INGM000.CSQUTIL.COMMANDS'	//'INGM000.SAVE.CSQUTIL'	Failed - BFGTR
MQ0B	MQ0B	Successful
//INGM000 CSQUTIL COMMANDS'	//'INGM000 SAVE CSQUTIL 1'	Successful

© Copyright IBM Corporation 2015

Figure 12-30. Checking results (1 of 2)

WM3021.1

Notes:

After you complete the transfer, select **Transfer Log** from **Managed File Transfer > MQ##** and check the results.

WebSphere Education

Checking results (2 of 2)

MQ Explorer - Content

Transfer Log			
	Source	Destination	Completion State
1	MQ0B /u/wm30/ingm000/INGM000.CSQUTIL.COMMANDS	MQ0B INGM000.SAVE.CSQUTIL	Failed Failed - BFGIO0001E: Fi
2	MQ0B //INGM000.CSQUTIL.COMMANDS'	MQ0B //INGM000.SAVE.CSQUTIL'	Successful Successful
3	MQ0B //INGM000.CSQUTIL.COMMANDS'	MQ0B //INGM000.SAVE.CSQUTIL'	Failed Failed - BFGTR0072E: T
	MQ0B //INGM000.CSQUTIL.COMMANDS'	MQ0B //INGM000.SAVE.CSQUTIL1'	Successful Successful

Completion State

- 1 - BFGIO0001E: File "/u/wm30/ingm000/INGM000.CSQUTIL.COMMANDS" does not exist.
- 2 successful
- 3 - BFGTR0072E: The transfer failed to complete due to the exception : BFGIO0006E: File "//INGM000.SAVE.CSQUTIL" already exists

© Copyright IBM Corporation 2015

Figure 12-31. Checking results (2 of 2)

WM3021.1

Notes:

You get good information on the results of a transfer in the Transfer Log menu.

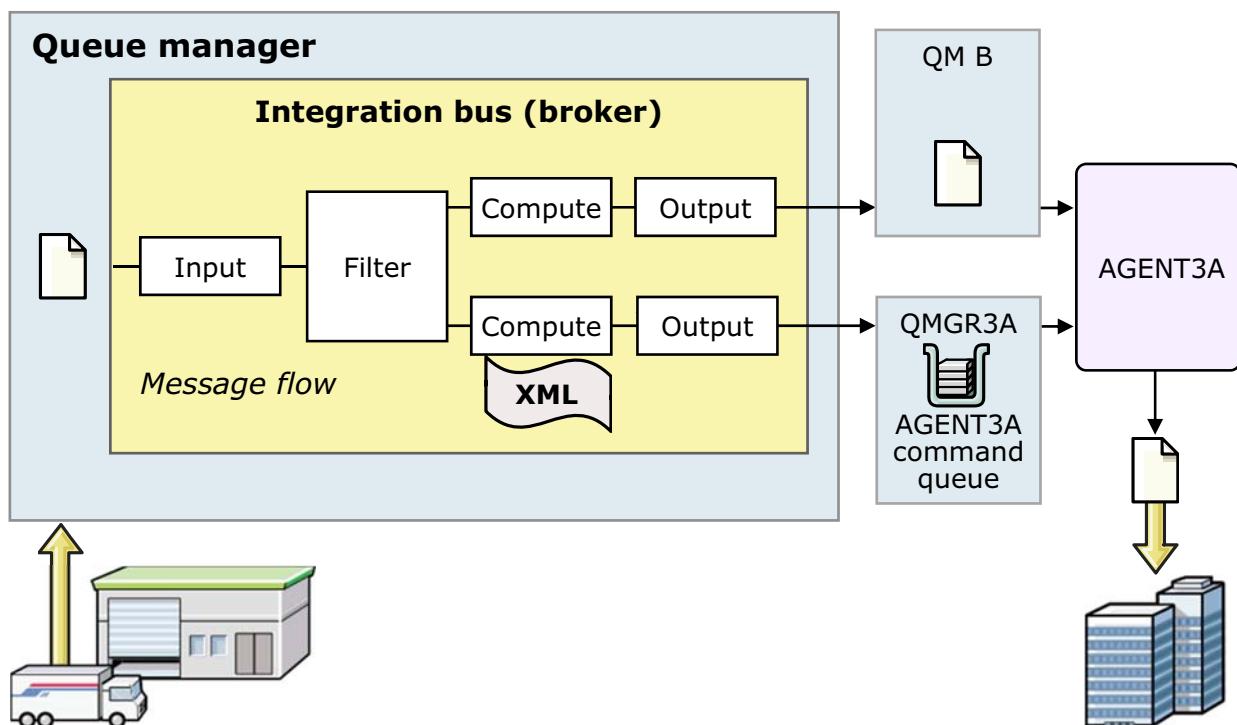
You can also select an individual transfer and double-click. XML information with more details is displayed in a pop-up window. After the double-click, you need to scroll down to see the extra information displayed. An example of the data provided:

```
# IN PROGRESS
<?xml version="1.0" encoding="UTF-8"?><transaction
ID="414d5120514d47523341202020202020b26b5e5320068404" agentRole="sourceAgent"
version="6.00" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="TransferLog.xsd">

<action time="2014-05-06T18:45:39.933Z">progress</action>

<sourceAgent QMgr="QMGR3A" agent="AGENT3A" agentType="STANDARD">
...
<destinationAgent QMgr="QMGR3B" agent="AGENT3B" agentType="STANDARD">
<... ...
<supplement>BFGIO0001E: File "C:\bdir\toofile.txt" does not exist.</supplement>
```

Starting a transfer by placing a specially formatted message in the agents command queue



© Copyright IBM Corporation 2015

Figure 12-32. Starting a transfer by placing a specially formatted message in the agents command queue

WM3021.1

Notes:

You can initiate a file transfer by placing specially formatted XML in the agent's command queue. This use is common for applications, and integration products such as IBM Integration Bus, where a formatted message gets sent to a queue, however, in this case the queue is the agent's command queue. Information about the transfer is included in the XML.

The XML must adhere to the `FileTransfer.xsd` schema and have the `<request>` element as the root element. A sample schema can be found in the z/OS UNIX System Services directory at `<install>/mqft/samples/schema` directory.

There can be three different root elements in the file transfer XML messages:

- `<request>`: For new file transfer requests, managed call requests, or deleting scheduled transfers that are pending
- `<cancel>`: For canceling file transfers in progress
- `<transferSpecifications>`: For specifying multiple transfer file groups, as used by the `fteCreateTransfer` command

IBM MQ Managed File Transfer use of publish/subscribe

```
CSQM201I MQ0B CSQMDRTC DIS TPSTATUS DETAILS 128
TPSTATUS(SYSTEM.FTE/Agents/QM1AGT)
TYPE(TOPIC)
DEFPRESP(SYNC) READY
```

```
CSQM201I MQ0B CSQMDRTC DIS TPSTATUS DETAILS 129
TPSTATUS(SYSTEM.FTE/Scheduler)
TYPE(TOPIC)
DEFPRESP(SYNC)
```

```
CSQM201I MQ0B CSQMDRTC DIS TPSTATUS DETAILS 130
TPSTATUS(SYSTEM.FTE/Scheduler/MQ0B)
TYPE(TOPIC)
DEFPRESP(SYNC)
DEFPSIST(NO)
```

© Copyright IBM Corporation 2015

Figure 12-33. IBM MQ Managed File Transfer use of publish/subscribe

WM3021.1

Notes:

IBM MQ Managed File Transfer uses publish/subscribe to accomplish some exchanges. A sample of the SYSTEM topics that are created to support IBM MQ Managed File Transfer is shown in the screen capture. You can see how some of the topics are done per agent, so the QM1AGT agent and the MQ0B agents each have an IBM MQ topic object.

These topics are defined along with other agent IBM MQ support objects when the agent was created by the `fteCreateAgent` command, BFGAGCR job.



Auditing loggers

- In the z/OS platform, a stand-alone database logger is available
 - Requires installation of logger Java application
 - Requires type 2 JDBC Driver to supported database
 - Logger helps save transfer results
 - Results are not formatted, must be obtained with SQL requests from the database
 - Separate application, not required for IBM MQ Managed File Transfer to function

© Copyright IBM Corporation 2015

Figure 12-34. Auditing loggers

WM3021.1

Notes:

A logging component is available for IBM MQ Managed File Transfer to save the file transfer history. This logger must be configured just like an IBM MQ Managed File Transfer configuration, by using a copy of BFGCUSTM and setting the SERVICE_TYPE parameter to LOGGER instead of AGENT. The logger requires DB2.

Transfer information can be checked by using IBM MQ Explorer. The logger is more suitable for historical activities, such as audits.

In this course, you are configuring an agent only.

Locating FFDCs

BFGUT0002E: An internal error has occurred. Product failure data was captured in file "FFDC.FTE.20140923142411419.log".

Your OMVS home directory:

```
INGM000:/u/wm30/ingm000:>ls -l
total 208
-rw-rw-rw-    1 INGM000    SYS1   2897 Sep 23 13:51 FFDC.FTE.20140923135137861
.log
-rw-rw-rw-    1 INGM000    SYS1   2897 Sep 23 14:01 FFDC.FTE.20140923140139159
...
```

© Copyright IBM Corporation 2015

Figure 12-35. Locating FFDCs

WM3021.1

Notes:

If the product issues an FFDC, these logs can be found in the z/OS UNIX System Services home directory of the IBM MQ Managed File Transfer user.

BFGAGSH fteShowAgentDetails output (1 of 2)**Agent Information:**

Name:	MQ0B
Type:	Standard
Description:	
Operating System:	z/OS
Host Name:	mvsmc14.ilsvpn.ibm.com
Time Zone:	Greenwich Mean Time
Product Version:	8.0.0.0
Build Level:	f000-20140506-1626
Trace Level:	No trace specified
Trace FFDC:	No FFDC specified

Agent Controller Information:

Controller Type:	z/OS Automatic Restart Manager (ARM)
Registered with ARM	No
Restarted	n/a

Agent Availability Information:

Status:	READY 475
---------	----------------

Status Details: The agent is running and is publishing its status at regular intervals. The last update was received within the expected time period. The agent is ready to process transfers, but none are currently in progress.

© Copyright IBM Corporation 2015

Figure 12-36. BFGAGSH fteShowAgentDetails output (1 of 2)

WM3021.1

Notes:

To extract detailed information about an agent, use the BFGAGSH job. This information is good to review the configuration details, and can be used if a call to software support is required. The display is continued on the next page.



BFGAGSH fteShowAgentDetails output (2 of 2)

...

Queue Manager Information:

Name:	MQ0B
Transport:	Bindings
Last Status Reported:	AVAILABLE
Status Details:	The queue manager is available.

Maximum Number of Running Source Transfers: 25

Maximum Number of Queued Source Transfers: 1000

Source Transfer States:

No current transfers

Maximum Number of Running Destination Transfers: 25

Destination Transfer States:

No current transfers

© Copyright IBM Corporation 2015

Figure 12-37. BFGAGSH fteShowAgentDetails output (2 of 2)

WM3021.1

Notes:

Unit summary

- Summarize the capabilities of IBM MQ Managed File Transfer
- List the key components of IBM MQ Managed File Transfer
- Describe how to configure an IBM MQ Managed File Transfer agent and supporting environment
- List the mechanisms that are available to initiate a file transfer
- Describe facilities that are available to display transfer status
- Explain how to perform basic problem determination
- Describe the IBM MQ File Transfer logger component

© Copyright IBM Corporation 2015

Figure 12-38. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. What are the three configuration scripts that need to be invoked when creating a new IBM MQ Managed File Transfer configuration?
 - a. fteSetupCoordination
 - b. fteSetupAgent
 - c. fteSetupCommand
 - d. fteCreateAgent
2. True or False: After you customize BFGCUSTM, any changes you make to an individual PDS member will not be overwritten if you need to run BFGCUSTM again.

© Copyright IBM Corporation 2015

Figure 12-39. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.

Checkpoint questions (2 of 2)

3. Choose the best correct answer. Where is your new configuration stored?
 - a. In the command queue manager directory in the UNIX System Services file system
 - b. In the `/var/TSM00##/mqft /config` directory for your coordination queue manager as specified in the BFGCUSTM job
 - c. In the location that is specified by the `fteSetupCoordination` script
 - d. In a directory of the same name as the coordination queue manager in the path that is specified by the BGF_DATA variable of the BFGCUSTM job
4. True or False: An application can initiate a transfer by placing a specially formatted XML message in the `SYSTEM.FTE.<agentname>.COMMAND` queue

© Copyright IBM Corporation 2015

Figure 12-40. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

Checkpoint answers (1 of 2)

1. What are the three configuration scripts that need to be invoked when creating a new IBM MQ Managed File Transfer configuration?
 - a. fteSetupCoordination
 - b. fteSetupAgent
 - c. fteSetupCommand
 - d. fteCreateAgent

Answer: a, c, and d

2. True or False: After you customize BFGCUSTM, any changes you make to an individual PDS member will not be overwritten if you need to run BFGCUSTM again.

Answer: False. You must copy and rename any members of your IBM MQ Managed File Transfer PDS that you have customized if you want to keep the changes else they are overwritten.

© Copyright IBM Corporation 2015

Figure 12-41. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. Choose the best correct answer. Where is your new configuration stored?
 - a. In the command queue manager directory in the UNIX System Services file system
 - b. In the /var/TSM00##/mqft /config directory for your coordination queue manager as specified in the BFGCUSTM job
 - c. In the location that is specified by the fteSetupCoordination script
 - d. In a directory of the same name as the coordination queue manager in the path that is specified by the BGF_DATA variable of the BFGCUSTM job

Answer: d

4. True or False: An application can initiate a transfer by placing a specially formatted XML message in the SYSTEM.FTE.<agentname>.COMMAND queue.

Answer: True

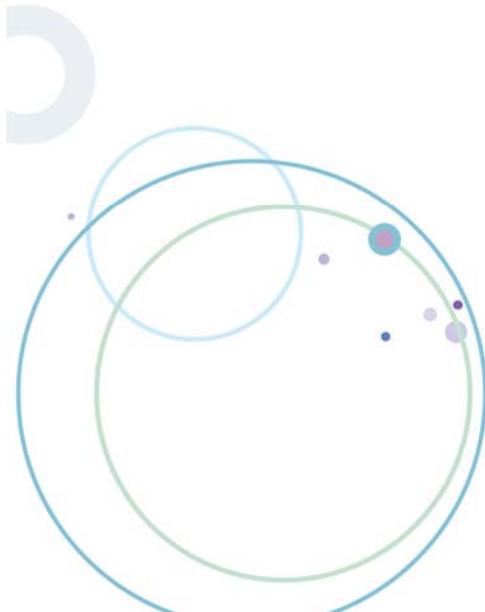
© Copyright IBM Corporation 2015

Figure 12-42. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 9



IBM MQ Managed File Transfer
configuration for z/OS

© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 12-43. Exercise 9

WM3021.1

Notes:



Exercise objectives

After completing this exercise, you should be able to:

- Examine the parameters that are required to configure IBM MQMFT on z/OS
- Configure and execute the IBM MQMFT customization job
- Create an IBM MQMFT configuration and agent
- Review changes to the UNIX System Services file structure during the IBM MQMFT configuration
- Start the IBM MQMFT agent
- Display the health of the agent
- Perform a basic file transfer by using batch JCL
- Add the MQMFT configuration to IBM MQ Explorer
- Check transfer results with IBM MQ Explorer
- Modify an existing MQMFT configuration

© Copyright IBM Corporation 2015

Figure 12-44. Exercise objectives

WM3021.1

Notes:



IBM MQ Managed File Transfer exercise reminders

- Be careful to create the `/var/TSM00##` directory as documented
- If you need help creating this directory, request assistance
- Take your time with the required changes to BFGCUSTM, the accuracy of this configuration makes the difference in the success of your lab
- Do not make any changes to the JCL members created by BFGCUSTM (unless indicated by the instructor)

© Copyright IBM Corporation 2015

Figure 12-45. IBM MQ Managed File Transfer exercise reminders

WM3021.1

Notes:

Unit 13. IBM MQ for z/OS backup, recovery, and related file tasks

What this unit is about

This unit covers critical tasks and utilities that the z/OS IBM MQ administrator needs to master, such as restart and recovery.

What you should be able to do

After completing this unit, you should be able to:

- Describe the contents and use of the IBM MQ for z/OS log and bootstrap data sets
- Describe and use the options to dynamically adjust the number and sizes of logs, page data sets, and buffers
- Determine the appropriate log and archive parameter module options according to requirements
- Describe how IBM MQ for z/OS maintains consistency with IMS, CICS, and Resource Recovery Services (RRS)
- Use the appropriate commands to display and control affected units of recovery
- Describe how IBM MQ for z/OS cleans up inconsistent data on restart
- Perform queue and page set management functions, such as expanding page data sets and restoring unloaded data

Unit objectives

- Describe the contents and use of the IBM MQ for z/OS log and bootstrap data sets
- Describe and use the options to dynamically adjust the number and sizes of logs, page data sets, and buffers
- Determine the appropriate log and archive parameter module options according to requirements
- Describe how IBM MQ for z/OS maintains consistency with IMS, CICS, and Resource Recovery Services (RRS)
- Use the appropriate commands to display and control affected units of recovery
- Describe how IBM MQ for z/OS cleans up inconsistent data on restart
- Perform queue and page set management functions, such as expanding page data sets and restoring unloaded data

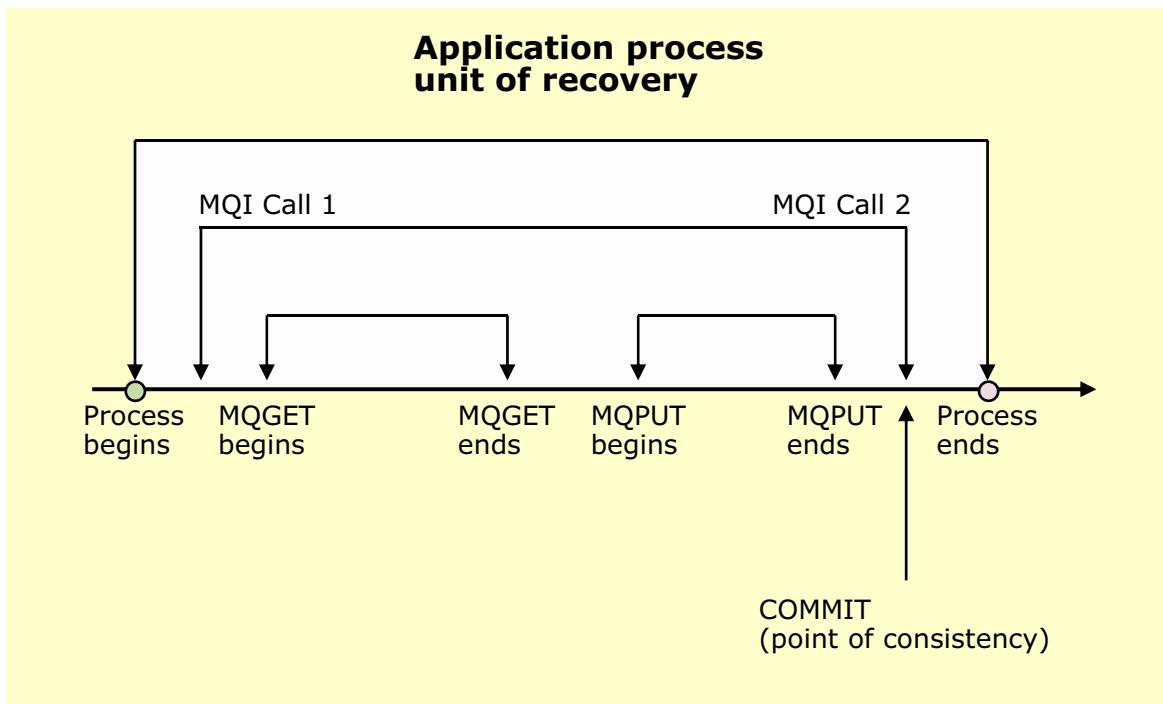
© Copyright IBM Corporation 2015

Figure 13-1. Unit objectives

WM3021.1

Notes:

Unit of recovery (UR)



© Copyright IBM Corporation 2015

Figure 13-2. Unit of recovery (UR)

WM3021.1

Notes:

Some terminology:

- A *unit of recovery* (UR) is the processing that is done by a single IBM WebSphere MQ subsystem for an application program that changes data from one point of consistency to another.
- A *point of consistency* is a point in time when all the recoverable data that an application program accesses is consistent. The final point of consistency is also called the *syncpoint* or *commit point*.
- An application can process one unit of recovery at a time.
- A UR begins with the first change (**MQGET** or **MQPUT**) to data after the beginning of the program, and by default ends when the program ends or disconnects from the queue manager.
- Applications can define more *commit points* during execution, thus splitting their activity into multiple units of recovery.
- IBM WebSphere MQ UR processing ensures that all changes of messages within the UR:

- Are locked until the commit point is reached, and that they are freed and made durable thereafter.
 - Are reset to their former state (“backed out”), if the application ends unexpectedly before it reaches the commit point or if the commit processing fails.
- The MQI offers the choice to decide for each **MQGET** or **MQPUT** operation, whether the action is included in the current UR or not by use of appropriate specifications within the MQGMO (Get Message Options) or MQPMO (Put Message Options) structures.
 - Operations that are performed with the **NO_SYNCPOINT** option take effect immediately, and are not backed out if the application fails.
 - In the case where the application does not specify syncpoint options, the default is to operate within syncpoint.

Interaction of buffers, logs, and page sets

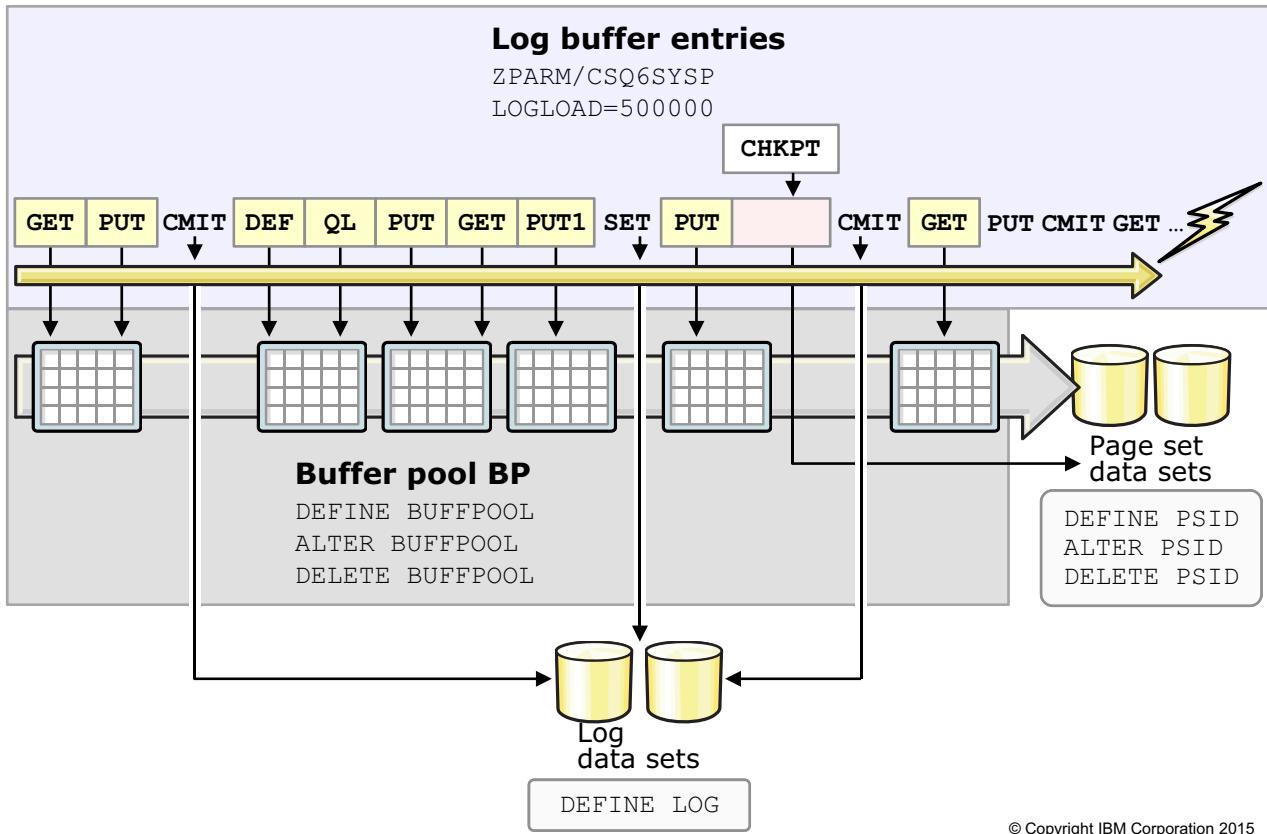


Figure 13-3. Interaction of buffers, logs, and page sets

WM3021.1

Notes:

- All message read/write activity that requested by applications, at the first instance, is done within pages of the queue manager *buffer pools*.
- At the same time, changes to recoverable (persistent) messages are written to log records that are placed in main storage log buffer pages.
- When a unit of recovery ends and thus initiates a commit point, changes must be *hardened* to ensure their durability. Hardening is done by writing the log data that is accumulated so far to one of the *active log data sets*.
- *Buffer pool contents* are flushed to page sets by an asynchronous process that uses a *lazy-write technique*.
 - At *checkpoint* time, those buffer pages that are marked changed for longer than two checkpoint intervals are written to page sets.
 - Buffer pool pages are also written to page sets to free and reuse buffer pool space, if it runs short.
 - Thus, short-lived messages might never go on page sets.
- Checkpoint frequency is determined by the **LOGLOAD** start parameter of the CSQ6SYSP macro, which specifies the number of regular log records that are written by the queue

manager between the start of one checkpoint and the next checkpoint. Thus, checkpoints occur more often during busy times.

A console message is issued for every checkpoint that includes information about the activity that is performed for each affected page set, so the effective checkpoint interval can easily be determined.

However, the **LOGLOAD** value cannot be changed during queue manager execution.

On controlled shutdown, IBM WebSphere MQ for z/OS writes all current buffer pool pages to page sets; when the system crashes, page sets are recovered to a consistent state when IBM WebSphere MQ restarts.

Logging overview

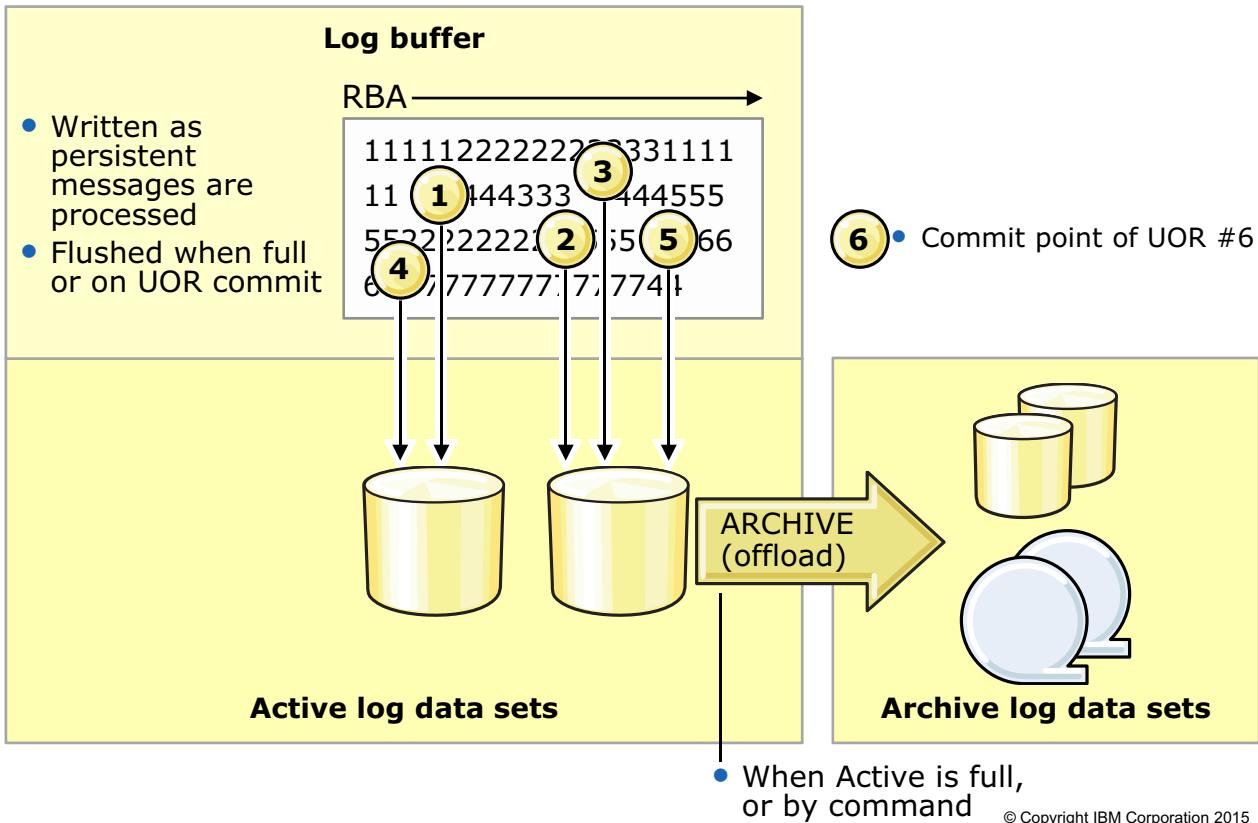


Figure 13-4. Logging overview

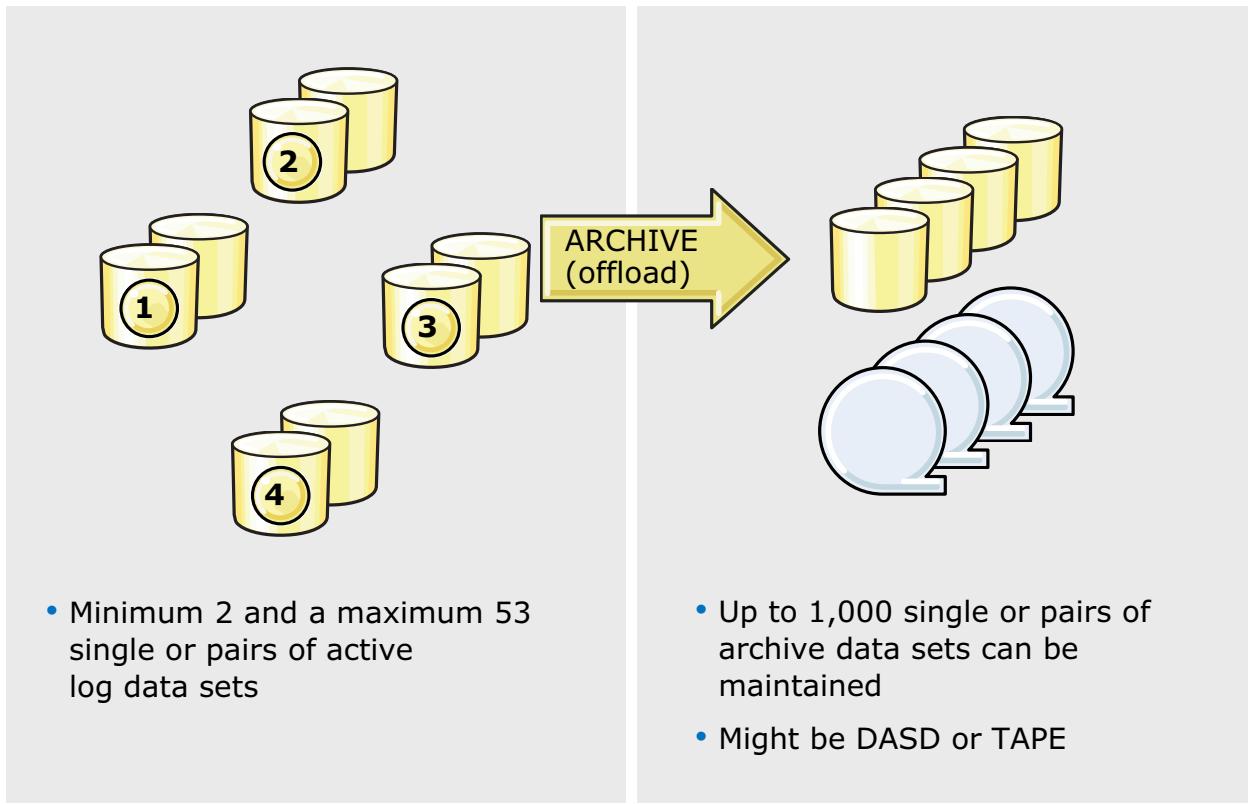
WM3021.1

Notes:

- On UR processing (mainly **GET** and **PUT** persistent messages), log records are created that reflect any data changes, and written to a *log output buffer* in virtual storage.
- Each log record is assigned a *relative byte address (RBA)*, which is the offset of the record's first byte from the beginning of the IBM WebSphere MQ log.
 - The RBA uniquely identifies a record that starts at a particular point in the log.
 - The log can contain up to 280 million * million bytes.
- Log records are packed into log buffer pages (4 KB each). The buffer is written ("flushed") to the active log data set when:
 - The log buffer becomes full.
 - The write threshold (specified in CSQ6LOGP system parameter **WRTHRSH**) is reached.
 - A UR reaches a commit point.

When the active log data sets are full, they are copied to *archive log data sets*. The process is called *archiving* or *offloading*.

Active and archive logs



© Copyright IBM Corporation 2015

Figure 13-5. Active and archive logs

WM3021.1

Notes:

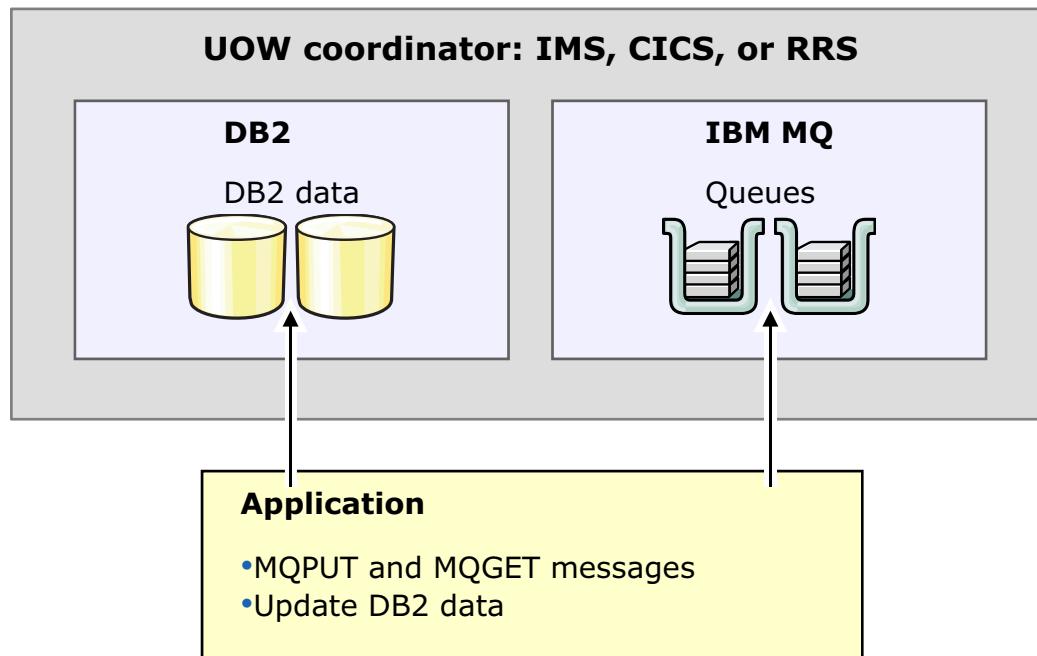
You must define and create at least two active log data sets for IBM MQ for z/OS, which are used cyclically.

- In production, it is suggested to have a minimum of four logs. The logs should be placed on separate disk/head assemblies in such a way that the read accesses performed to archive the old log does not interfere with the I/O to the new log that processes the current application activities.
- Log definition is performed as part of the IBM MQ system setup.
- More logs can be defined when system use increases. Defining more logs requires a restart of the queue manager.

If specified by the CSQ6SYSP parameter, IBM MQ for z/OS automatically performs dual logging, which means that identical log information is written to two different active log data sets. IBM MQ automatically archives full active logs to archive data sets, which can be on DASD or TAPE units.

- Archive logs can be written in SINGLE or DUAL mode as well, independent from the active logs.
- Default number of logs is 500. If this number is exceeded, recording goes back to the start of the BSDS. Specify number of logs to 1000, which is the limit so BSDS can continue archiving.

Consistency with other resource managers



© Copyright IBM Corporation 2015

Figure 13-6. Consistency with other resource managers

WM3021.1

Notes:

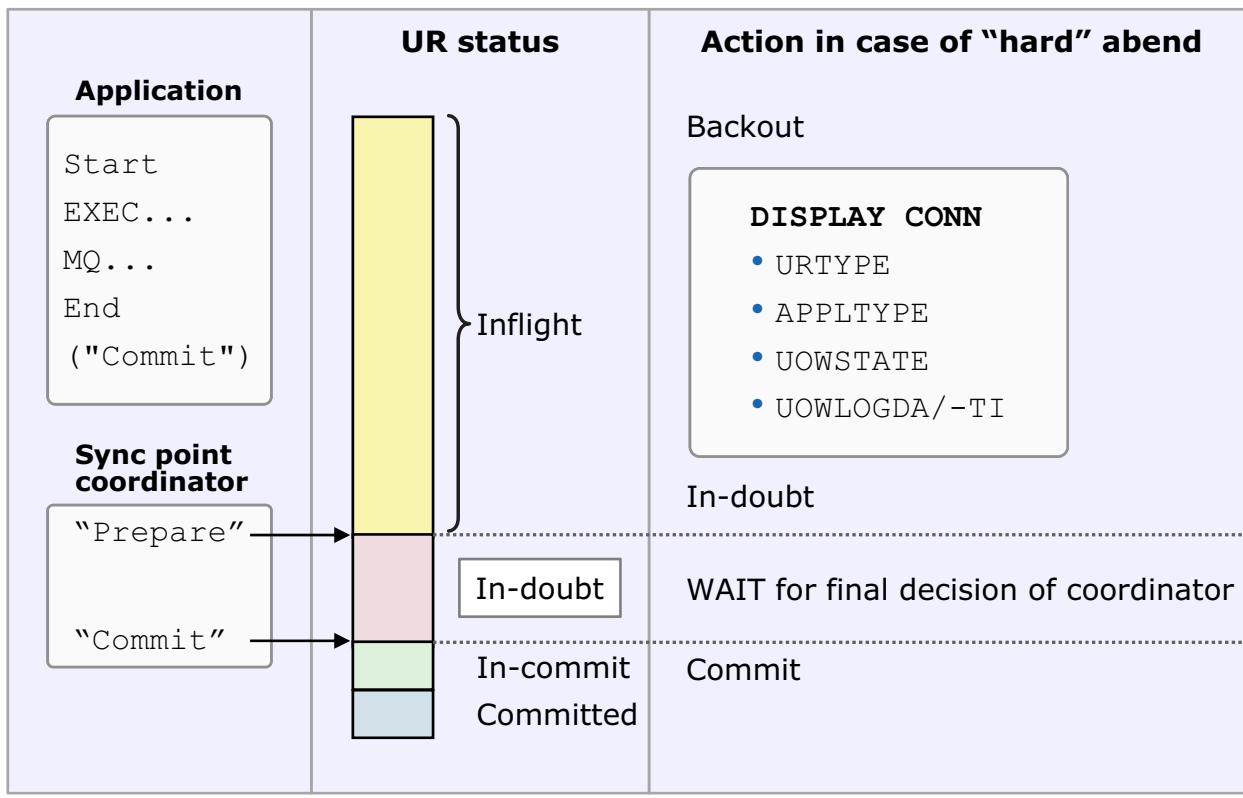
A typical IBM MQ server application handles message traffic and also applies changes to other resource manager data, such as DB2.

When an application spans resources outside of IBM MQ, coordination to ensure the consistency of updates across the resource managers is required.

- Coordination might be done by a transaction manager, such as CICS, IMS, RRS, or WebSphere Application Server. These products provide a complete application environment including a syncpoint service.
- For applications operating directly under the control of z/OS, such as batch and TSO processes, z/OS Restart Recovery Management Services (RRMS or RRS) takes the part of this unit-of-work control instance.

Applications must use the services of the application environment to request commit or backout processing. Committing or backing out causes the unit of work to be controlled by the transaction manager, by using the two-phase commit protocol at syncpoint.

Two-phase commit and in-doubt unit of recovery



© Copyright IBM Corporation 2015

Figure 13-7. Two-phase commit and in-doubt unit of recovery

WM3021.1

Notes:

IBM MQ acts as a participant or agent during syncpoint processing on an application environment.

The process that is followed during unit of recovery is:

1. The first request to change queue manager resource, an MQGET or MQPUT for a message, starts a unit of recovery in the IBM MQ for z/OS system.
2. When the application reaches a logical point of consistency, usually at task end, and issues a syncpoint, IBM MQ for z/OS and any other resource managers are notified by the phase 1-PREPARE command that is issued by the coordinator.
3. When IBM MQ or the connection fails at any time before this point, IBM MQ backs out this UR without the need for further interaction with the coordinating Transaction Manager.
4. IBM MQ for z/OS completes its phase 1 by writing a log record, and then the coordinating transaction manager is notified about IBM MQ being "ready to commit" this UR.

The UR enters the INDOUBT state in IBM MQ. Now if the systems or the connections fail, IBM MQ neither commits or backs out this UR, waiting until directed by the coordinating transaction manager when communication is reestablished.

5. When all systems complete stage 1 processing, the coordinator (CICS, IMS, RRS, or WebSphere Application Server) records this point of the commit process on its log. After the recording of the commit process, all changes must be committed.
6. The coordinating transaction manager notifies all participating resource managers, including IBM MQ, with the phase 2-COMMIT command, and they perform their phase 2 processing by committing their changes.

Arrival of the COMMIT or BACKOUT notification ends the INDOUBT state of the UR; IBM MQ commits or backs out this UR even if one of the systems or the connection fails.

7. IBM MQ for z/OS logs the completion of its phase 2 processing and notifies the coordinator that a new point of consistency is reached.
8. When all notifications are received, the coordinator system logs its own point of consistency. The data that is controlled now is CONSISTENT among all participating systems.

From an administrative view, the critical consideration is the possibility of any units of recovery that might remain in the INDOUBT state within IBM MQ, and across an IBM MQ for z/OS restart. The restart topic in this unit addresses inquiry and handling of INDOUBT situations.

Currently, active units of recovery and their states can be displayed by use of the DISPLAY CONN command.

This command returns detailed unit of recovery information, such as:

- The UR type (QMGR, CICS, IMS, RRS, XA)
- The UR state, as discussed earlier, where the values shown are:
 - ACTIVE for in-flight tasks
 - PREPARED for tasks that are in the process of being committed
 - UNRESOLVED for in-doubt tasks
- For CICS applications, the CICS transaction code and task number
- For IMS applications, the IMS PSB name
- The user ID associated with the application
- The time when the unit of work started and the first record was written to the log

Log shunt

- Detect long-running units of work at checkpoint time
- Rewrite UOW log data nearer the log tail to make the UOW a "shunted UOW"
- Issue console messages when a shunt of a UOW is successful
- Perform UOW backout based on shunt records

- Reduced quantity of log data to be retained for UOW coordination
- Queue manager restart time reduced, as less log data must be traversed
- Shunt records not usable for media recovery

© Copyright IBM Corporation 2015

Figure 13-8. Log shunt

WM3021.1

Notes:

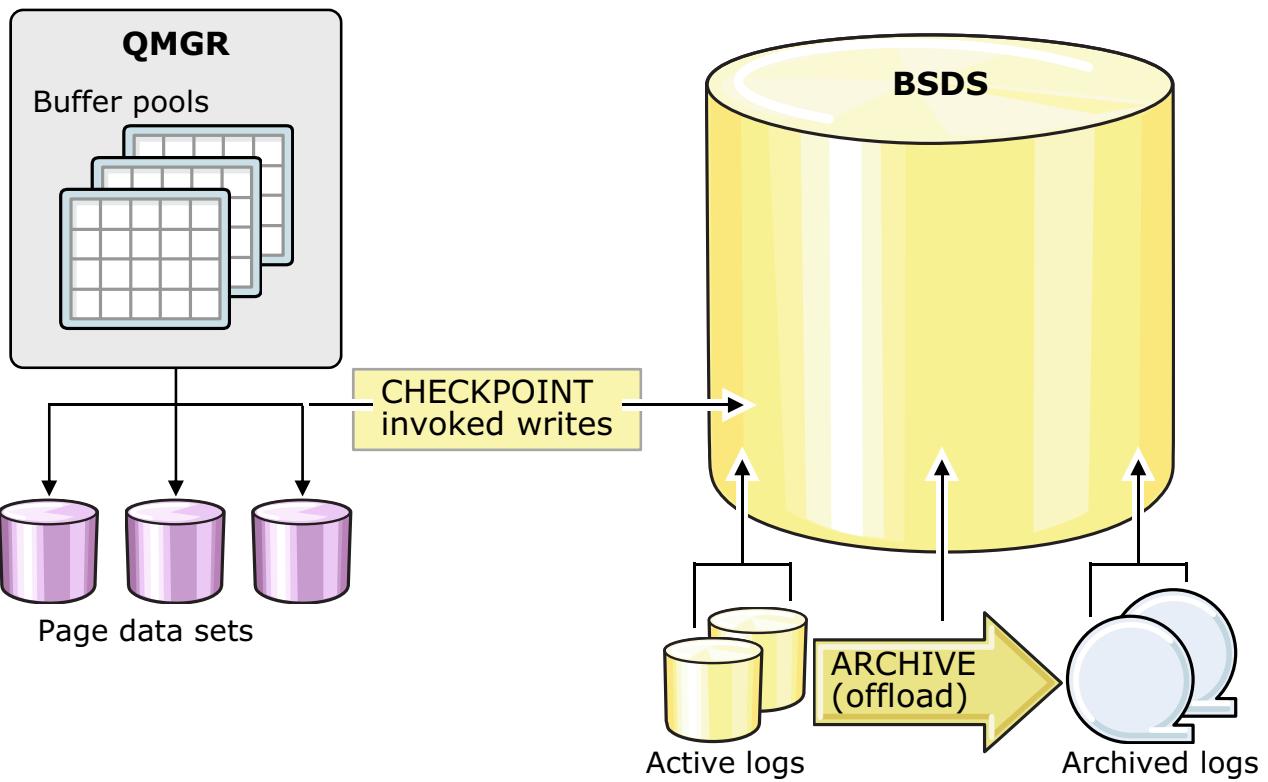
The Log shunt process is also performed at checkpoint time to mitigate problems with long-running tasks. The concept of shunting is to place the minimum information that is required to back out a task close to the end of the log to reduce the back-out effort in case of an abend.

- A task is considered “long-running” if it remains active for longer than three checkpoint intervals.
- Every three checkpoints, a long-running unit of recovery is shunted, that is, all the control data relevant to the back out is rewritten to the log.

A restarting queue manager never needs to return more than to the third checkpoint (current checkpoint minus 2).

Shunted records cannot be used if a page set failure occurs because they contain only the part of the original log record that is necessary for back out.

IBM MQ use of the bootstrap data set



© Copyright IBM Corporation 2015

Figure 13-9. IBM MQ use of the bootstrap data set

WM3021.1

Notes:

The bootstrap data set (BSDS) is a VSAM that stores information pertaining to:

- The active and archived log data sets
- The log records to enable log reads during normal processing
- Find key log records for restart

IBM MQ uses the inventory kept in the BSDS each time an archive log data set is defined or an active log data set is reused. If active, you can determine which log is full and which log is available. A key piece of information that is kept in the BSDS is the RBA of each portion of the log.

Log data sets are defined to the BSDS by using the change log inventory CSQJU003 utility.

Contents of the BSDS (1 of 2)

```

CSQJ440I CSQJU004 IBM WebSphere MQ for z/OS V8.0.0
CSQJ444I CSQJU004 PRINT LOG MAP UTILITY - 2014-10-23 10:26:22
=====
LOG MAP OF BSDS DATA SET COPY 1, DSN=WMQ.MQ00.BSDS01
BSDS VERSION      - 1
SYSTEM TIMESTAMP   - 2014-10-17 18:55:41.34
UTILITY TIMESTAMP  - 2014-08-21 08:13:13.63
HIGHEST RBA WRITTEN 000000000023A170 2014-10-17 18:55:41.3
HIGHEST RBA OFFLOADED 0000000000000000
=====
ACTIVE LOG COPY 1 DATA SETS
START RBA/TIME/LRSN END RBA/TIME/LRSN CREATED DATA SET INFORMA
----- -----
EMPTY DATA SET          2014-08-21 DSN=WMQ.MQ00.LOGCOPY.DS
0000-00-00 00:00:00.0 0000-00-00 00:00:00.0 08:13 PASSWORD=*****
/ 000000000000 / 000000000000 STATUS=
                           NEW,
                           REUSABLE
...

```

© Copyright IBM Corporation 2015

Figure 13-10. Contents of the BSDS (1 of 2)

WM3021.1

Notes:

The visual shows the first part of output that is created by the Print Log Map Utility (CSQJU004) that is used to list the information that is held in the BSDS.

The utility can be run at any time, whether IBM MQ is running or stopped. The listing shows:

- The BSDS version (Version 1 before IBM MQ V8 feature activation, or Version 2 if the BSDS is converted to the new IBM MQ V8 and later format to handle 8-byte RBA).
- Utility and system time stamps.
- The highest log RBA written to active logs and to archives (“offloaded”), if archiving is enabled.
- Inventories of active and archived logs with their data set names and associated RBA ranges and time stamps.
- Status of active log data sets, which can be:
 - NEW: The data set is defined but never used.
 - REUSABLE: The log data is available, maybe it is offloaded or archiving is off.
 - NOT REUSABLE: The data set contains records that are not offloaded.
 - STOPPED: An error was encountered during offload. The data set is not to be reused, but IBM MQ for z/OS attempts to read from it if it is required for recovery.
 - TRUNCATED: Writing to this data set has stopped.

Contents of the BSDS (2 of 2)

```

... ... ... ...
ARCHIVE LOG COPY 2 DATA SETS
NO ARCHIVE DATA SETS DEFINED FOR THIS COPY
=====
CSQJ401E RECORD NOT FOUND -      RESTART CONTROL
=====

        CHECKPOINT QUEUE
          AT 2014-10-23 10:26:22
DATE      TIME      START RBA      END RBA
2014-10-17 18:55:41 00000000023A27C 000000000023ACC2
2014-09-26 11:45:33 000000000182256 0000000000182C9C
2014-09-26 11:44:49 0000000001800C4 0000000000180A82
        SHUTDOWN CHECKPOINT
2014-09-15 09:25:08 000000000149242 0000000000149C88
2014-09-15 09:24:04 000000000147000 00000000001479BE
... ... ...
=====
CSQJ401E RECORD NOT FOUND -      ARCHIVE LOG COMMAND HISTORY
=====
CSQJ200I CSQJU004 UTILITY PROCESSING COMPLETED SUCCESSFULLY

```

© Copyright IBM Corporation 2015

Figure 13-11. Contents of the BSDS (2 of 2)

WM3021.1

Notes:

The display shows a partial view of the second part of the output that is created by the Print Log Map Utility (CSQJU004).

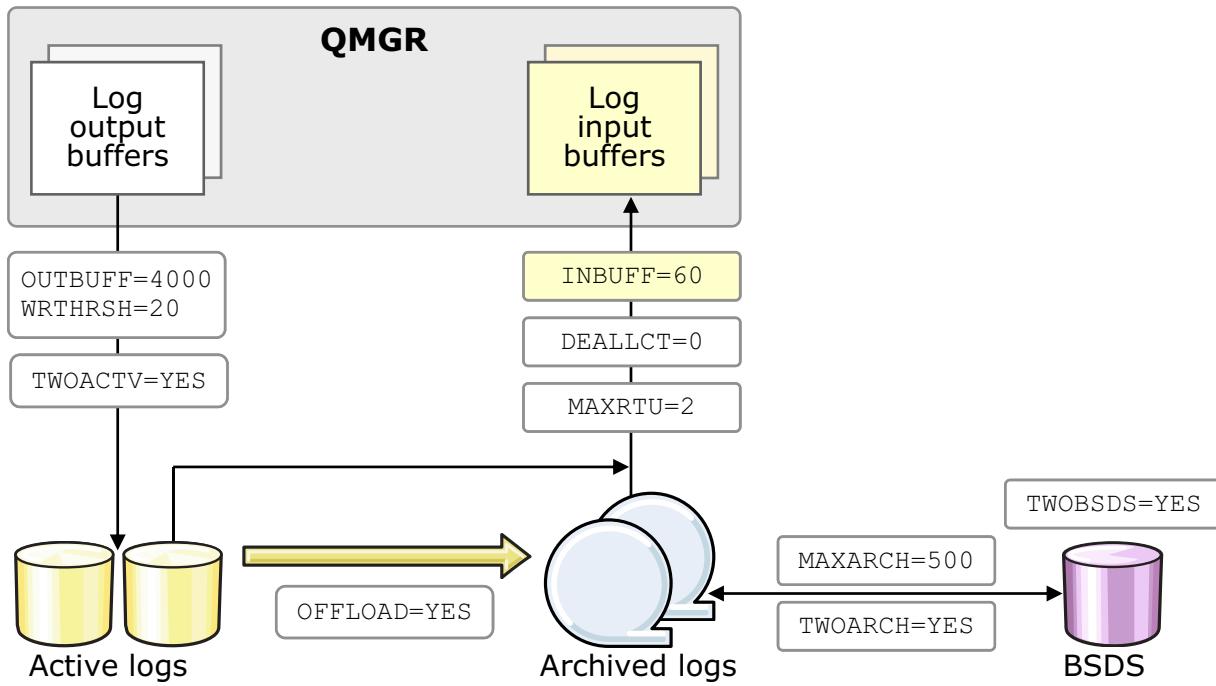
No archiving is set up for this queue manager. No restart control exists in this BSDS.

The checkpoint directory shows the times when checkpoint information was written to the log and tells the log RBA range it occupies, starting from most recent to oldest entries.

If applicable, another table would document the times when a log switch is initiated by using the ARCHIVE LOG command, but there is no activity in the display.

Log control ZPARM macro CSQ6LOGP

- DISPLAY / SET LOG



© Copyright IBM Corporation 2015

Figure 13-12. Log control ZPARM macro CSQ6LOGP

WM3021.1

Notes:

The options that are shown on this page are specified to IBM MQ through the CSQ6LOGP option of the CSQZPARM module. The name of this module must be used when starting IBM MQ.

The CSQ6LOGP macro parameters control:

- The size of the log output buffer and its write threshold, which might be affecting overall performance.
- The number of input buffers for both active and archive log reads in case of backout and recovery, and the number of archive log data sets that can be allocated at a time.
- Whether archiving is used.
- Whether active logs, archive logs, and bootstrap data sets are run in single or dual mode.
- The number of log data sets that are registered in the BSDS.

The DISPLAY LOG command returns the settings and some logging status information. It is automatically issued as part of IBM MQ start processing. Output might look as follows (the example was taken from an education lab system):

```
CSQJ322I MQ00 DISPLAY LOG report ... 884
Parameter Initial value      SET value
-----
INBUFF      60
OUTBUFF    4000
MAXRTU      2
MAXARCH    500
TWOACTV     YES
TWOARCH     YES
TWOBSDS    YES
OFFLOAD     YES
WRTHRSH    20
DEALLCT     0
COMPLOG    NONE
End of LOG report
CSQJ370I MQ00 LOG status report ... 885
Copy %Full DSName
 1    45 WMQ.MQ00.LOGCOPY1.DS02
 2    45 WMQ.MQ00.LOGCOPY2.DS02
Restarted at 2014-10-17 18:55:40 using RBA=0000000000239000
Latest RBA=00000000041CF84
Offload task is BUSY, allocating archive data set
Full logs to offload - 2 of 8
CSQ9022I MQ00 CSQJC001 ' DIS LOG' NORMAL COMPLETION
```

The SET LOG command can be used to change the values of the **DEALLCT**, **MAXARCH**, **MAXRTU**, and **WRTHRSH** parameters.

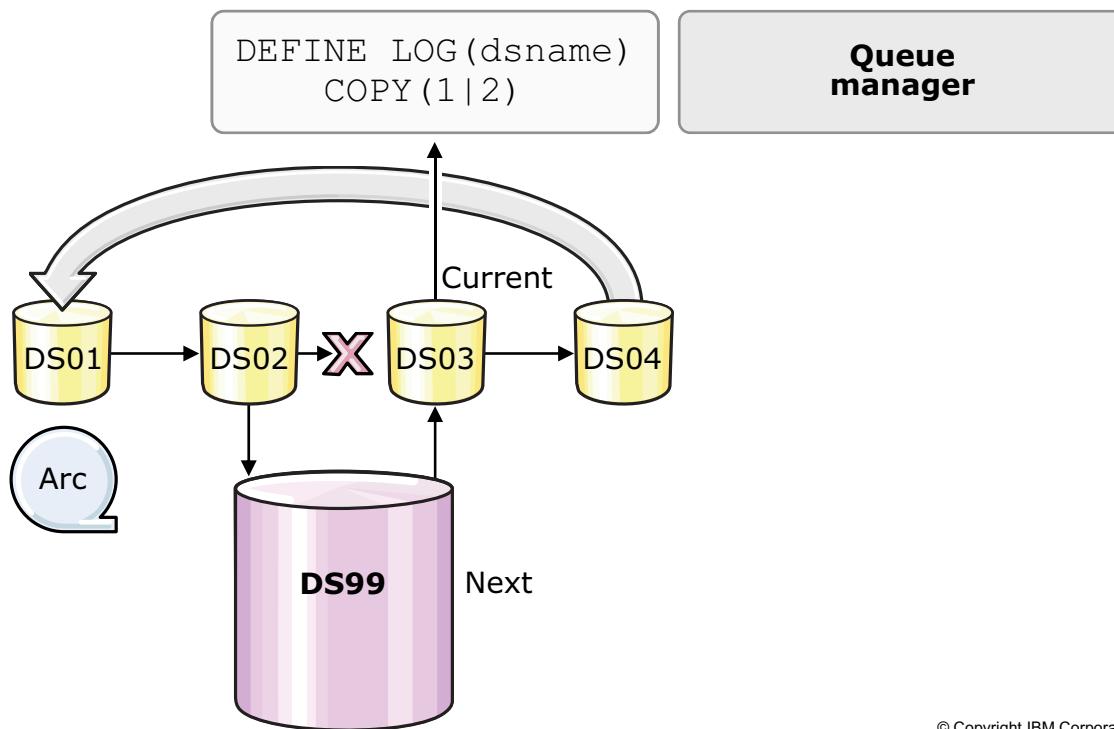
Updated values would appear in the right column of the DISPLAY LOG output.

Values that are changed with the **SET** option do not survive a queue manager restart, as the parameter module is interpreted unconditionally at start time.

SET LOG(DEFAULT) resets the parameters to the values specified in the archive system parameters set by CSQ6LOGP.

Dynamic addition of active log data sets

- Data set dynamically allocated
- Used as next in indicated “ring”



© Copyright IBM Corporation 2015

Figure 13-13. Dynamic addition of active log data sets

WM3021.1

Notes:

Logs can be added dynamically by using the `DEFINE LOG` MQSC command.

- The named VSAM linear data set (which must be already defined with IDCAMS and optionally, formatted with utility CSQJUFMT) is dynamically allocated to the queue manager.
- If the data set has been successfully opened for input and output, and the first record is either blank or contains the eye catcher indicating it has been preformatted with log preformat utility, the data set is added to the active log data set ring indicated by the `COPY` parameter. The data set is added in such a position so that *it will be the next one used* after the current data set is full. This position allows the command to succeed even if the queue manager is already frozen.
- The BSDS is updated so that this data set will still be used by the queue manager after restart.

Archive control ZPARM macro CSQ6ARVP

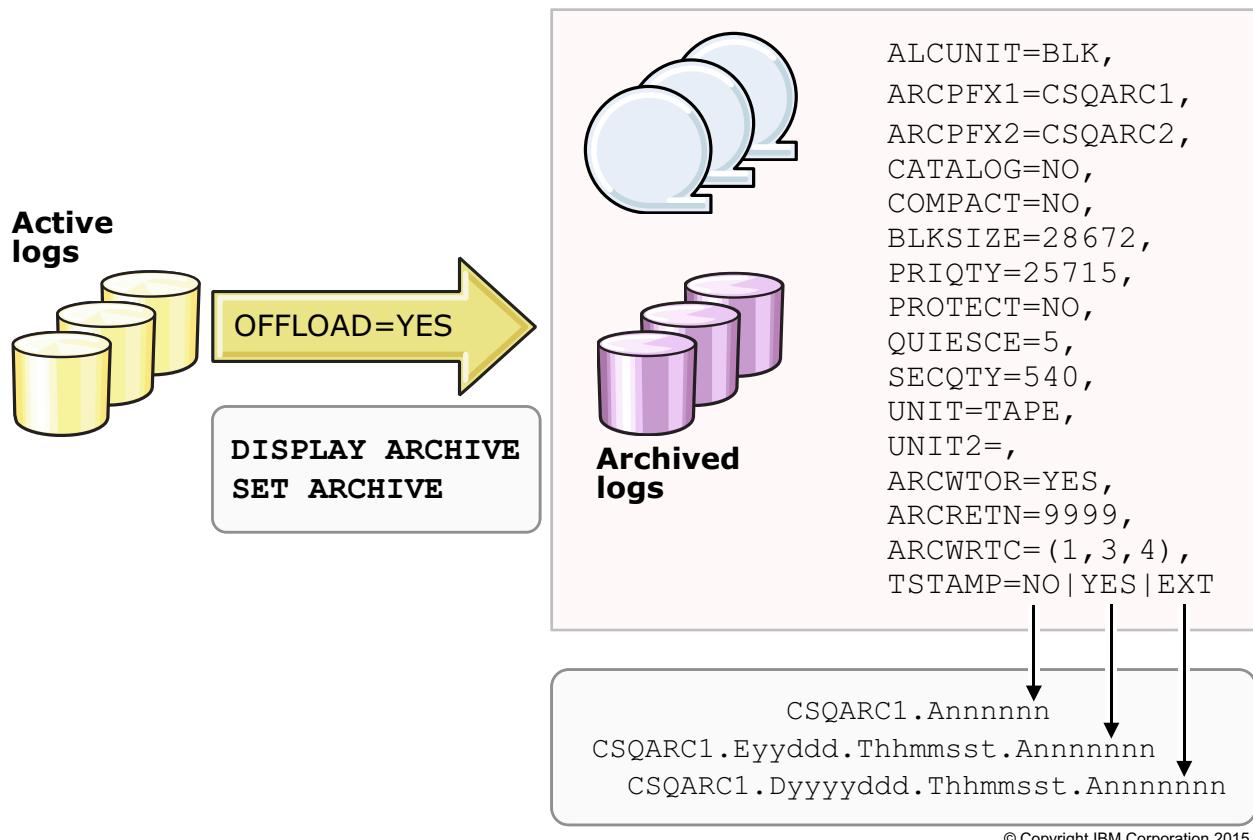


Figure 13-14. Archive control ZPARM macro CSQ6ARVP

WM3021.1

Notes:

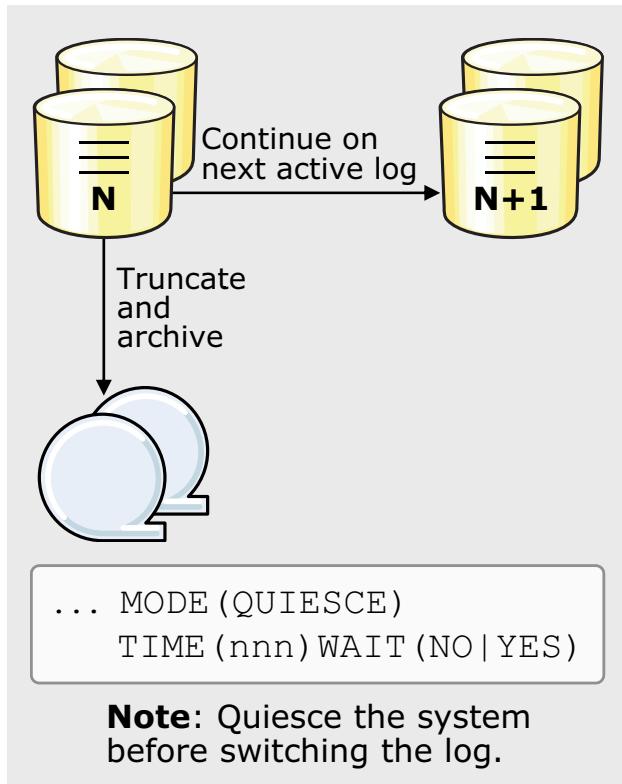
Macro CSQ6ARVP, which is within the initialization parameter file, contains options that control archive data set-related functions and names. They include:

- The data set names of the archives: This value is the high-level qualifier, and whether a time stamp becomes part of the name, as shown by the visual.
The maximum length of the prefix that is specified depends on the TSTAMP option used.
- Whether archive data sets are on DASD or on TAPE.
- Options how to handle the archives regarding cataloging and retaining.

Some of these archiving options can be changed online, as for the log values:

- DISPLAY ARCHIVE returns a report that shows the initial values for the archiving parameters and the current values, as changed by the SET ARCHIVE command.
- Again, changes persist for the current execution of the queue manager.
- On IBM MQ for z/OS start, a DISPLAY ARCHIVE command is started automatically, so its results are shown as part of the queue manager message log.

ARCHIVE LOG command



- **ARCHIVE LOG:**

- Truncate the current active log data sets
- Continue logging, and then switch to the next active log data set
- Start a task to offload the data sets

© Copyright IBM Corporation 2015

Figure 13-15. ARCHIVE LOG command

WM3021.1

Notes:

The ARCHIVE LOG command takes a copy of the current active log (or both logs if dual logging is used) and switches to the next active log or logs in the cycle, as shown in the figure.

Using the command with the **MODE(QUIESCE)** option might be of particular interest:

- **MODE(QUIESCE)** prevents new URs from starting, and delays the log switch until all active users or URs reach a commit point, thus ensuring that the archived data represents a system-wide point of consistency.

This process works only if all applications quiesce within the time that is specified in the **QUIESCE** system parameter (CSQ6ARVP macro), which can be overwritten by the **TIME(nn)** parameter that is issued with the ARCHIVE LOG command.

- If the system does not quiesce within the time that is specified, the ARCHIVE LOG command is canceled, and the logs are not switched.
- **WAIT(NO|YES)** specifies whether the issuer of the ARCHIVE command is able to issue further requests while waiting for the quiesce time to complete.

SUSPEND and RESUME QMGR LOG commands

SUSPEND QMGR LOG :

- Immediately stop logging
- Write (flush) log buffers
- Initiate system checkpoint



Take a fast copy (snapshot) of logs, BSDSs, and page sets

RESUME QMGR LOG :

- Resume logging



© Copyright IBM Corporation 2015

Figure 13-16. SUSPEND and RESUME QMGR LOG commands

WM3021.1

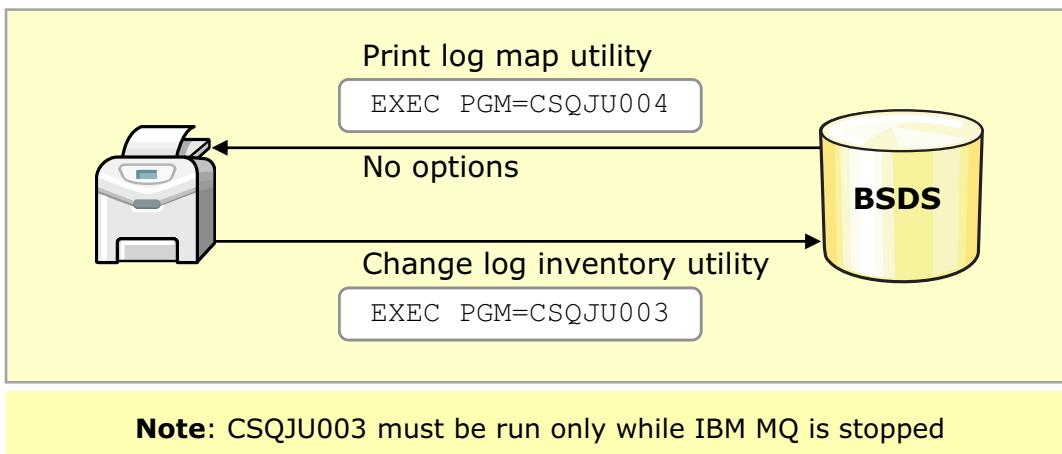
Notes:

This set of commands provide you with an option to suspend and later resume queue manager logging.

- While logging is suspended, you can take fast (snapshot) copies of your page sets and logs. This option gives you a point-in-time copy of your queue manager state.
- After you take the fast copies, you enter the command to resume queue manager logging.
- A non-deletable highlighted message reminds you that queue manager logging is suspended.

As suspending logging operations immediately stops all update activity on behalf of applications, this option should not be used during busy times.

List and change the BSDS contents



Note: CSQJU003 must be run only while IBM MQ is stopped

Command	Description
NEWLOG	Defines new active and archive logs
DELETE	Removes log entries
CRESTART	Controls the next restart

© Copyright IBM Corporation 2015

Figure 13-17. List and change the BSDS contents

WM3021.1

Notes:

The print log map utility (CSQJU004) output has been shown earlier.

The Change Log Inventory Utility (CSQJU003) performs the following BSDS update functions:

- **NEWLOG** informs the BSDS about an active or archive log data set to be added. This file might be:
 - A newly defined VSAM file that is used as an active log data set
 - An active log data set that replaces one, which encountered I/O errors
 - An archive log data set volume
- **DELETE** deletes active or archive log data set information. This file might be an active log data set that is replaced or an outdated archive data set.
- **CRESTART** controls the log RBA range to be accessed during the next restart.

CSQJU003 must be run only when the queue manager is stopped.

MQ for z/OS termination

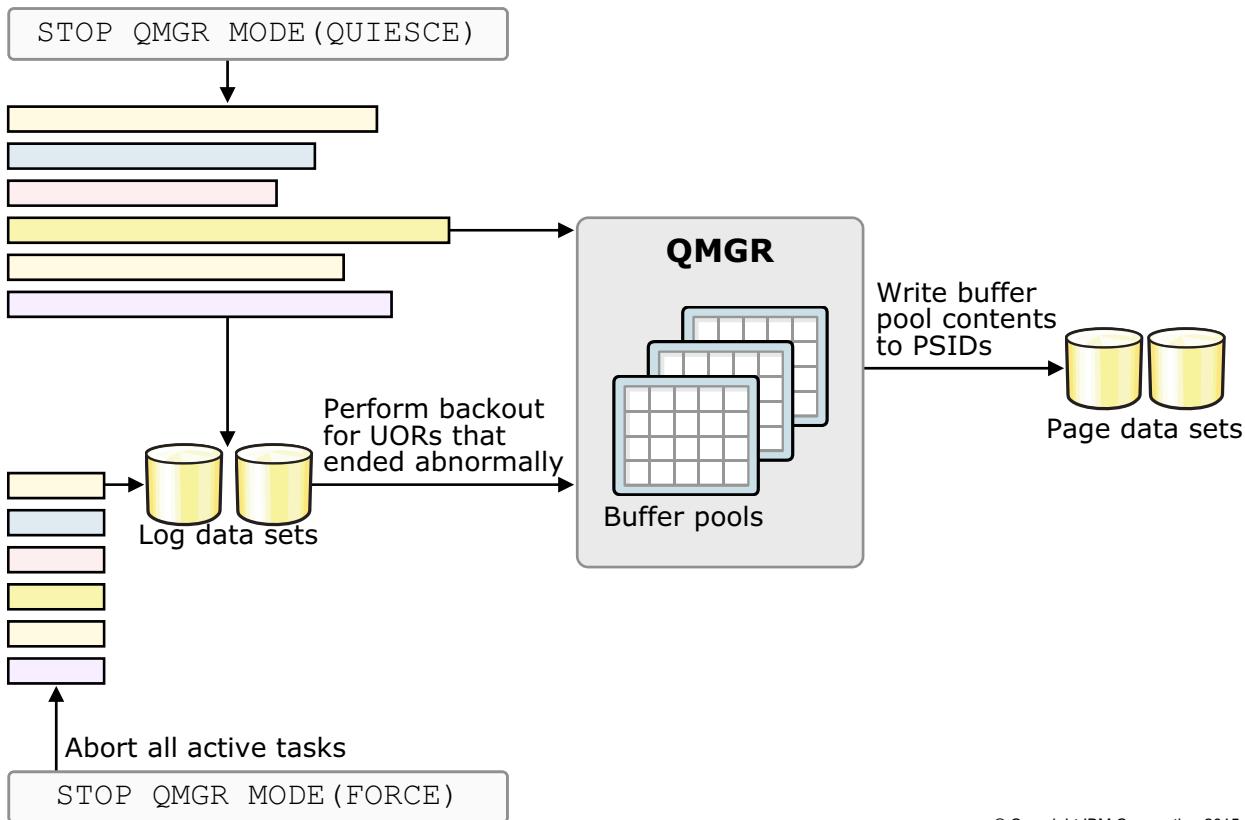


Figure 13-18. MQ for z/OS termination

© Copyright IBM Corporation 2015

WM3021.1

Notes:

What occurs on IBM MQ for z/OS restart, and how to manage and control these restart activities?

A restart requires IBM MQ for z/OS to run for some time and then be brought down in some way. The way this step happens determines what activities must be performed on restart. There are two ways to shut down IBM WebSphere MQ for z/OS in a controlled way, by use of the `STOP QMGR` command:

- STOP QMGR MODE (QUIESCE)
 - Allows all active units of recovery to complete; even a new UR might start, but no new **MQCONNECTs** are permitted.

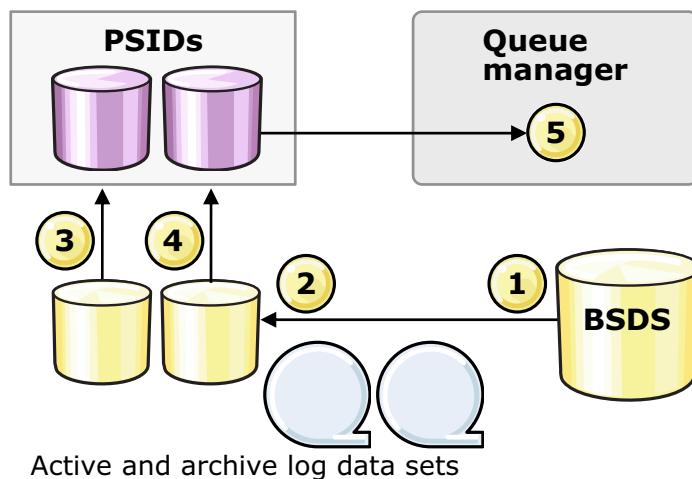
This method is the suggested way to stop the queue manager and is the default option. Applications should be encouraged to code the `FAIL_IF_QUIESCING` options with the appropriate MQI verbs. This method gives the applications an indication that the queue manager is attempting to shut down.
 - STOP QMGR MODE (FORCE)
 - Stops all active units of work and backs out changes; no new URs are permitted to start.

- After all threads are closed, both modes of controlled shutdown proceed with the same operational steps:
 - All outstanding updates from buffer pages to page sets are completed.
 - The shutdown checkpoint is taken and the BSDS is updated.
- In both cases, units of recovery that are in an INDOUBT state are not affected by a queue manager shutdown; they remain in-doubt.
- Only in the event of an *abnormal termination*, caused, for example, by an abend or a CANCEL command, the page set data is left in an inconsistent state.

IBM WebSphere MQ for z/OS automatically issues a DISPLAY THREAD(*) TYPE(INDOUBT) command when shutdown processing has started:

```
10.26.28 STC00194 CSQY009I +QM09 'STOP QMGR' COMMAND ACCEPTED FROM USER(INGMUSR),  
STOP MODE(QUIESCE)  
10.26.28 STC00194 CSQY002I +QM09 QUEUE MANAGER STOPPING  
10.26.28 STC00194 +QM09 DISPLAY THREAD(*) TYPE(INDOUBT)  
10.26.28 STC00194 CSQV401I +QM09 DISPLAY THREAD REPORT FOLLOWS -  
10.26.28 STC00194 CSQV420I +QM09 NO INDOUBT THREADS FOUND  
10.26.28 STC00194 CSQ9022I +QM09 CSQVDT 'DISPLAY THREAD' NORMAL COMPLETION
```

Restart activities



Step	Description
1	Log initialization
2	Current status rebuild
3	Forward
4	Backward log recovery
5	Queue index rebuild, which is controlled by the system parameter: QINDXBLD (WAIT NOWAIT)

© Copyright IBM Corporation 2015

Figure 13-19. Restart activities

WM3021.1

Notes:

When a queue manager starts, it always performs the following actions:

- During the log initialization phase, IBM WebSphere MQ for z/OS locates the last log record that is written before termination. Log data set names are obtained from the BSDS.

If the previous termination was a normal one, it is indicated by a shutdown checkpoint record on the BSDS. This checkpoint allows IBM WebSphere MQ to skip all the following steps except the last one.

Only steps 2 – 4 must be performed in the case of a restart after an abnormal termination.

- The status rebuild phase processes all log records starting with the last complete checkpoint to:
 - Identify all incomplete units of recovery
 - Identify all objects open at the time of failure
 - Determine the log RBA to start recovery of page sets to the current position, if necessary

3. The forward log recovery phase performs a forward pass of the log, from the lowest log RBA determined previously:
 - a. It uses the REDO log records to repeat all changes and restore the page sets to the point of failure.
 - b. It ensures that all committed changes, which might have been in virtual storage at termination, are reflected in the data sets.
4. The backward log recovery phase reads backwards through the log to back out all in-flight transactions identified earlier:
 - a. It uses the UNDO log records, as in the normal backout sequence.
 - b. It writes compensating REDO records to facilitate future media recovery.
5. The queue index rebuilding phase rebuilds new indexes in virtual storage for indexed queues containing persistent messages.

The completion of the restart can be delayed until this activity is completed (WAIT) or the restart might complete without all indexes built (NOWAIT). This value is based on the value of the system parameter QINDBLD as an option within the CSQ6SYSP macro.

- Previous releases enforced the WAIT behavior.
- Specifying QINDBLS(NOWAIT) speeds up queue manager restart but MQGET and MQPUT calls to indexed queues are suspended until the index is fully built. This option might cause applications that were running previously to time out.

Finally, a new checkpoint is taken.

System restart messages

```

CSQR001I MQ00 RESTART INITIATED
CSQR003I MQ00 RESTART - PRIOR CHECKPOINT RBA=000000000005C946
CSQR004I MQ00 RESTART - UR COUNTS - 946
IN COMMIT=0, INDOUBT=0, INFLIGHT=0, IN BACKOUT=0
CSQI049I MQ00 Page set 0 has media recovery 947
RBA=0000000000052A51, checkpoint RBA=0000000000052A51
CSQI049I MQ00 Page set 1 has media recovery 948
RBA=000000000005C946, checkpoint RBA=000000000005C946
... ... ...
CSQR030I MQ00 Forward recovery log range 952
from RBA=0000000000043CFA to RBA=000000000005D326
CSQR005I MQ00 RESTART - FORWARD RECOVERY COMPLETE - 953
IN COMMIT=0, INDOUBT=0
CSQR032I MQ00 Backward recovery log range 954
from RBA=000000000005D326 to RBA=000000000005D326
CSQR006I MQ00 RESTART - BACKWARD RECOVERY COMPLETE - 955
INFLIGHT=0, IN BACKOUT=0
CSQM559I MQ00 CSQMCTL Loading of cluster transmission 956
queue state started

```

© Copyright IBM Corporation 2015

Figure 13-20. System restart messages

WM3021.1

Notes:

The restart steps that are named on this slide are formally processed, even if it proves that no real restart processing is required, because the previous shutdown was a controlled one. This figure shows a message log extract from a controlled restart.

Remember how incomplete units of recovery must be processed:

- *In-flight* units of recovery, for which the first phase of syncpoint is not done yet: These units are backed out on restart.
- *In-doubt* units of recovery, which completed phase 1 but awaited notification from the co-ordinator to begin phase 2: These units remain in-doubt until the syncpoint coordinator (CICS, IMS, or RRS) gives the appropriate notification.
- *In-commit* units of recovery for which IBM MQ already started phase 2 of the commit process: These units are committed on restart.
- *In-backout* units of recovery for which IBM MQ started backout but could not complete it. These units are backed out on restart.

Caring for in-doubt threads (CICS sample)

- Determine in-doubt threads

```
DIS THREAD(*) TYPE(INDOUBT)
```

- Returns the following output for in-doubt CICS threads:

CSQV401I + DISPLAY THREAD REPORT FOLLOWS -

CSQV406I + INDOUBT THREADS -

NAME	THREAD-XREF	NID
VICIC1	00012C20	VICIC1.A6E5A6F0E2178D25
VICIC1	00012420	VICIC1.A6E5A6F055B2AC25
VICIC1	0000F820	VICIC1.A6E5A6EFFD60D425
VICIC1	00012620	VICIC1.A6E5A6F07AB56D22

DISPLAY THREAD REPORT COMPLETE

CSQ9022I + CSQVDT 'DIS THREAD' NORMAL COMPLETION

© Copyright IBM Corporation 2015

Figure 13-21. Caring for in-doubt threads (CICS sample)

WM3021.1

Notes:

When a connection between IBM WebSphere MQ and CICS (or IMS) breaks, one or more Units of Recovery can be left in-doubt on the IBM MQ side.

- When the systems reconnect, they determine automatically whether resync activity is required. IBM MQ issues a system message for every in-doubt UR, identifying it by its CICS UOWID.
- Resolution is performed automatically, based on the CICS log information, and the result is displayed in a system message for each UR.

Any in-doubt units of recovery that cannot be resolved automatically must be investigated:

- Issue the `DISPLAY THREAD(connection-name) TYPE(INDOUBT)` command in IBM WebSphere MQ for z/OS to list the in-doubt units of recovery for that connection (see output in the preceding figure).
- List the CICS system log to locate the log entries for the particular unit of recovery. How to do this process depends on the CICS version you are running.
- Use the IBM WebSphere MQ for z/OS Log Print Utility (CSQ1LOGP) to locate the corresponding IBM MQ for z/OS log entries (see last page of this topic).

- Use the CICS system log entries to decide whether the IBM WebSphere MQ for z/OS in-doubt UOWs should be committed or backed out. If no other reason can be found why these units of recovery were not resolved automatically, use the RESOLVE INDOUBT command:

```
RESOLVE INDOUBT (connection-name) ACTION(action) NID(nid)
```

Where:

- connection-name is the CICS applid
- action is COMMIT or BACKOUT
- nid is one or more of the network IDs displayed in the NID column of the DIS THREAD report.

Manually resolving IMS units of recovery

- For in-doubt IMS threads, the DIS THREAD TYPE (INDOUBT) output is provided here:

```
CSQV401I + DISPLAY THREAD REPORT FOLLOWS -
CSQV406I + INDOUBT THREADS -
NAME      THREAD-XREF   NID
IMSA      0002MQMPRG1  IMSJ.5600000000
IMSA      0001MQMINQ   IMSJ.5700000000

DISPLAY THREAD REPORT COMPLETE
CSQ9022I + CSQVDT 'DIS THREAD' NORMAL COMPLETION
```

- The command to specify COMMIT or BACKOUT for each in-doubt UR is:

```
RESOLVE INDOUBT(connection name)
ACTION(COMMIT|BACKOUT) NID(network-id)
```

© Copyright IBM Corporation 2015

Figure 13-22. Manually resolving IMS units of recovery

WM3021.1

Notes:

When the *IMS* adapter is restarted after a connection has been broken, the following resynchronization is performed:

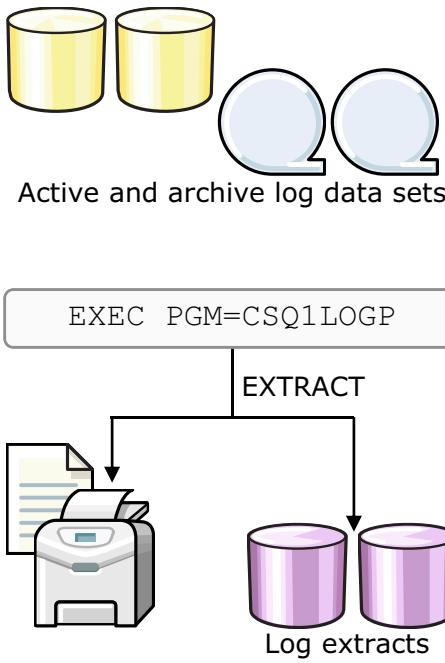
- IMS presents a list of unit of work (UOW) IDs it believes to be in-doubt to IBM MQ, by using the IMS adapter with a resolution parameter of **COMMIT** or **BACKOUT**.
- If there are any mismatches with its own data, IBM MQ issue message CSQQ010E for the particular unit of work ID.
- The IMS adapter then receives from IBM MQ a list of any unit of work (UOW) IDs that it still holds in-doubt for this connection, and reports these values to the IMS master terminal in message CSQQ008I.

Any in-doubt units of recovery that cannot be resolved automatically must be investigated:

- Issue the DISPLAY THREAD(connection-name) TYPE(INDOUBT) command in IBM MQ to list the in-doubt units of recovery for that connection (see output in the preceding figure).
- Force the IMS log closed, by using /DBR FEOF, and archive the IMS log. Then, use DFSERA10 to print the required log records for the last transaction in each dependent region.
- Back out each PSB involved that has not reached a commit point by using the DL/1 batch utility.

If necessary, use the RESOLVE INDOUBT command, as for CICS URs, where the connection name is the IMS system ID.

Log print utility CSQ1LOGP



- DD statements for input:
 - BSDS
 - ACTIVE
 - ARCHIVE (multiple)
- Input control (select) parameters:
 - RBASTART
 - RBAEND
 - PAGESET
 - URID
 - RM (RECOVERY | DATA | BUFFER)
- EXTRACT option writes selected records to data sets:
 - CSQBACK: From backed out UOWs
 - CSQCMIT: From committed UOWs
 - CSQBOTH: From both committed and backed out UOWs
 - CSQINFLT: From not-completed UOWs
 - CSQOJBS: Definition changes

© Copyright IBM Corporation 2015

Figure 13-23. Log print utility CSQ1LOGP

WM3021.1

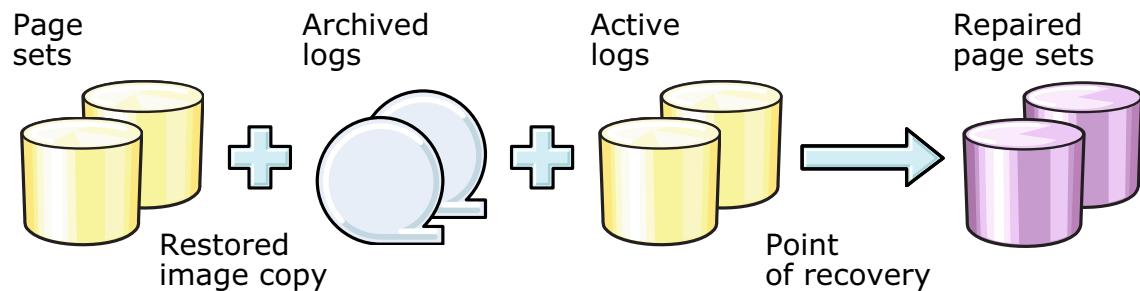
Notes:

- The utility can be run against active or archived logs. If you are processing active log data sets, the utility runs even if IBM WebSphere MQ is running, if the BSDS and active log data sets are defined by using at least SHAREOPTIONS(2 3).
- Logs can be selected directly by DD statements that identify the active or archive log data sets or through the BSDS.
 - If input selection is based on BSDS, the log RBA range to be searched must be specified.
 - Further input control statements identify the unit of recovery ID (URID), the page set number, or both as selection criteria for the log records you are interested in.
 - Only records of the IBM MQ resource manager specified with the RM parameter are processed.
- The utility provides a **SUMMARY** option to request a shortened summary report before listing all details.
- CSQ1LOGP allows the extraction of log records to data sets based on the unit of work state of a task at the time that is associated to the end RBA of the specified range.

Log extraction might be useful for:

- Message auditing: Prove that particular data was put and committed, and who received it.
- Object changes: When? Who?
- Persistent data recovery, if you cannot wait for media recovery.
- Problem determination: Determine information such as putter/getter response times, and time that is spent on queue.

Principle of media recovery



© Copyright IBM Corporation 2015

Figure 13-24. Principle of media recovery

WM3021.1

Notes:

What can be done if IBM MQ page set, log, or bootstrap data sets are damaged, and what actions should be taken to minimize risk?

If your current data is lost, for whatever reason, you need to do the following steps to recover it to the latest valid state:

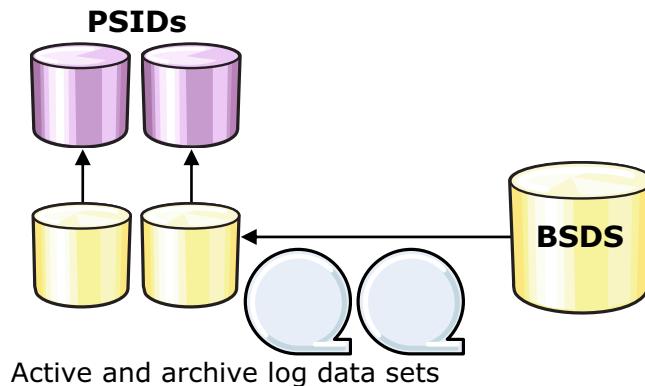
- Restore saved backups ("image copies") of your page sets.
This option requires that you made these backups previously.
- Get a complete set of logging information that contains all of the updates applied to the data from the time the latest image copy was written.
This option requires that logging is active always and that the log data is available.
- Run some recovery process that merges the logs with the old image copies, thus re-creating the latest state of committed data.

In the case of IBM MQ:

- The "data" that can be subject to recovery are the contents of the page set data sets.
- Logging is performed by the system automatically.

- You must make sure that page data sets are backed up regularly, so that a useful image copy is available when a data set or a volume crashes.
- You must make sure that all the logs required for recovery are available and recorded in the BSDS.

PSID recovery is part of MQ restart



- Restart IBM MQ for z/OS with the contents of the latest backup (image copy) of the corrupted page set
- IBM MQ will:
 - Read the recovery log sequence number (LSN) of each page set
 - Determine the last complete checkpoint for each page set
 - Read the logs forward from this checkpoint and REDO log records that are applied to PSIDs, if the record number is less than the page-LSN

© Copyright IBM Corporation 2015

Figure 13-25. PSID recovery is part of MQ restart

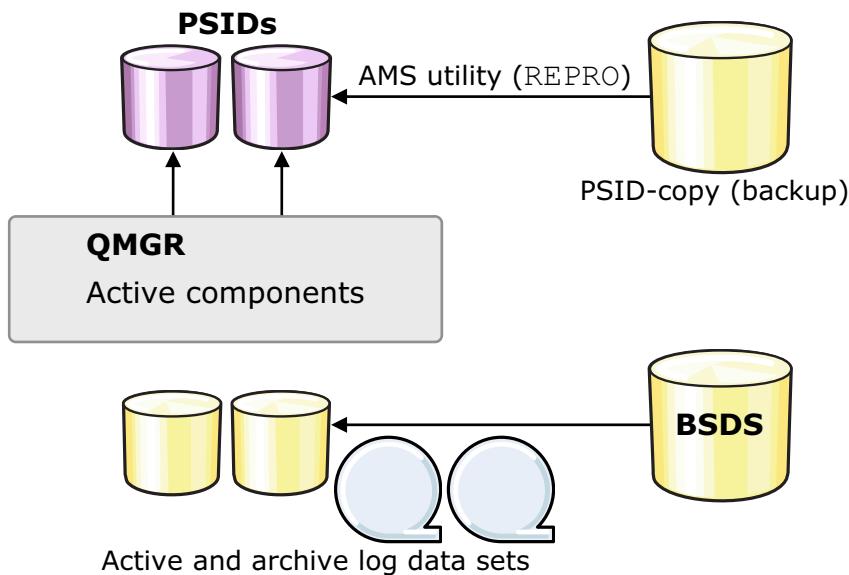
WM3021.1

Notes:

Recovery of damaged page sets is always done through restarting IBM MQ with page set data at the latest available level. Access to all required logs, both active and archive, must be ensured.

- In page 0 of each page set, IBM MQ maintains a log RBA number, called the *log sequence number* (LSN), which is the starting log RBA from which the page set can be recovered. Up to this log RBA, all updates to persistent messages are already written to the page set.
- At shutdown checkpoint time during normal termination, the LSN is set to the highest used RBA, indicating that the page set contains the latest level of committed data.
- If a page set LSN that is lower than the shutdown checkpoint RBA is detected or a shutdown checkpoint does not exist, IBM MQ determines the checkpoint that is the most recent to the lowest page set LSN. Starting from that point, if any updates, it recovers one or more of the PSIDs affected by the special *undo* and *redo* logging technique, as written to the log. The recovery process finishes only if all the necessary logs are registered and available to IBM MQ.

Taking page set backups



- Full backup
 - When the queue manager is stopped
- "Fuzzy" backup
 - While the queue manager is running

© Copyright IBM Corporation 2015

Figure 13-26. Taking page set backups

WM3021.1

Notes:

- Page sets can be backed up at any time by use of the REPRO AMS utility (or an equivalent).
- Backups that are taken while IBM MQ is running are called *fuzzy backups* because PSID contents might not reflect a consistent state of message data because more or fewer updates might not yet have been written to disk and are still in virtual storage buffers.
- A *full backup* requires that the queue manager is taken down using the `STOP QMGR` command, so that all updates are forced on to the page sets.

When a backup is taken afterward, and while the queue manager is not active, you can free all logs written before this point if there were no in-doubt units of recovery when the queue manager shutdown. You can recover from this backup requiring only log records that are written after this time.

Useful commands for PSID backup handling

- **DISPLAY THREAD TYPE (INDOUBT)**
 - Make sure that there are no in-doubt URs
 - If there are any, get rid of them by using RESOLVE INDOUBT
- **ARCHIVE LOG MODE (QUIESCE)**
 - Switch logs and establish a *system-wide point of consistency*, which might be the target for a controlled (manipulated) system recovery
- **SUSPEND/RESUME QMGR LOG**
 - Create a target for a time stamp recovery
- **DISPLAY USAGE**
 - Informs about usage of active PSIDs and the log records
 - Needed in case of recovery from fuzzy backups

© Copyright IBM Corporation 2015

Figure 13-27. Useful commands for PSID backup handling

WM3021.1

Notes:

You must implement a recovery strategy that includes rules for when to take what types of backups and how long to keep the archives.

- If you can afford to shut down the queue manager daily, you can take full backups and do not have to keep any archives beyond that time.
Another suitable approach might be to save GDG of backups and the logs back to the earliest GDG date.
- Most installations need to use fuzzy backups. Determine the log or archive data that is required for the DISPLAY USAGE command.
- If some time stamp recovery is of interest, then you must use the ARCHIVE LOG the SUSPEND/RESUME LOG commands, or both:

Offering just the archives that are produced until and by a successful ARCHIVE LOG QUIESCE command allows recovery to the system-wide point of consistency that was reached when the command was processed, when you start with the latest PSID backups created before the ARCHIVE LOG

- Restarting IBM WebSphere MQ with the logs, BSDS, and page sets “snapshot” copy that were created while logging was suspended has the same final effect as if the whole system had crashed now: incomplete work is backed out, and new processing starts with the committed contents of all queues.

Use of ARCHIVE LOG QUIESCE versus SUSPEND LOG or RESUME QMGR LOG

- Theoretically, SUSPEND LOG or RESUME LOG processing is possible at any time. The impact on active applications must be taken into account; while the ARCHIVE LOG QUIESCE command will not complete if an application cannot finish within the active quiesce time.
- The recovery point that is created by the use of ARCHIVE LOG QUIESCE is a system-wide point of consistency, which can be used with recovering another resource manager such as database data. Performing this recovery needs the PSID backups from an earlier time (either Full or Fuzzy), and all the log (archive) data that is produced since then (and some time before) until the time when the ARCHIVE LOG command was issued and processed.

The snapshot copies of logs, BSDS, and PSIDs taken during a SUSPEND LOG phase can be kept together and form a set of files so that a queue manager can be restarted directly. This phase usually occurs without the need for any other log or archive data, unless a long-running task was active at the time of the snapshot. In this case, backing out would require access to data that is not contained in the log copies.

Display usage for PAGESET command output

MQ00 DIS USAGE TYPE(PAGESET)

CSQI010I MQ00 Page set usage ... 125

Page set	Buffer pool	Total pages	Unused pages	Persistent data pages	NonPersist data pages	Expansion count
-	0	322	287	35	0	USER 0
-	1	322	301	21	0	USER 0
-	2	322	318	4	0	USER 0
-	3	322	322	0	0	USER 0
-	4	322	303	17	2	USER 0

End of page set report

CSQI065I MQ00 Buffer pool attributes ... 126

Buffer pool	Available buffers	Stealable buffers	Stealable percentage	Page class	Location
-	0	50000	49966	99	4KB BELOW
-	1	20000	19999	99	4KB BELOW
-	2	50000	50000	100	4KB ABOVE
-	3	20000	19993	99	4KB BELOW

End of buffer pool attributes

CSQI024I MQ00 CSQIDUSE Restart RBA for system as 127

configured=00000000000601DD

© Copyright IBM Corporation 2015

Figure 13-28. Display usage for PAGESET command output

WM3021.1

Notes:

DISPLAY USAGE offers two options:

- **TYPE(PAGESET)** returns information about the size and use of all page sets known to the queue manager. This information might be used as a basis for administering page sets, as covered by the next topic.

If page sets are backed up currently to recovery administration, it gives the lowest *recovery log sequence number* of all page sets, which is a possible point of recovery.

- Take a (fuzzy) backup of the page data sets afterward.
- To ensure restart with this set of backups, you must keep all logs, including any logs that are newer than the one containing the system restart RBA.

Display usage for DATASET command output

```

MQ00 DIS USAGE TYPE(DATASET)
CSQI070I MQ00 Data set usage ... 130
Data set   RBA/LRSN           DSName
Log, oldest with active unit of work:
      - 000000000006DC20  MQ00.LOGCOPY1.DS01
Log, oldest for page set recovery:
      - 00000000000601DD  MQ00.LOGCOPY1.DS01
End of data set report
CSQ9022I MQ00 CSQIDUSE 'DIS USAGE' NORMAL COMPLETION

```

© Copyright IBM Corporation 2015

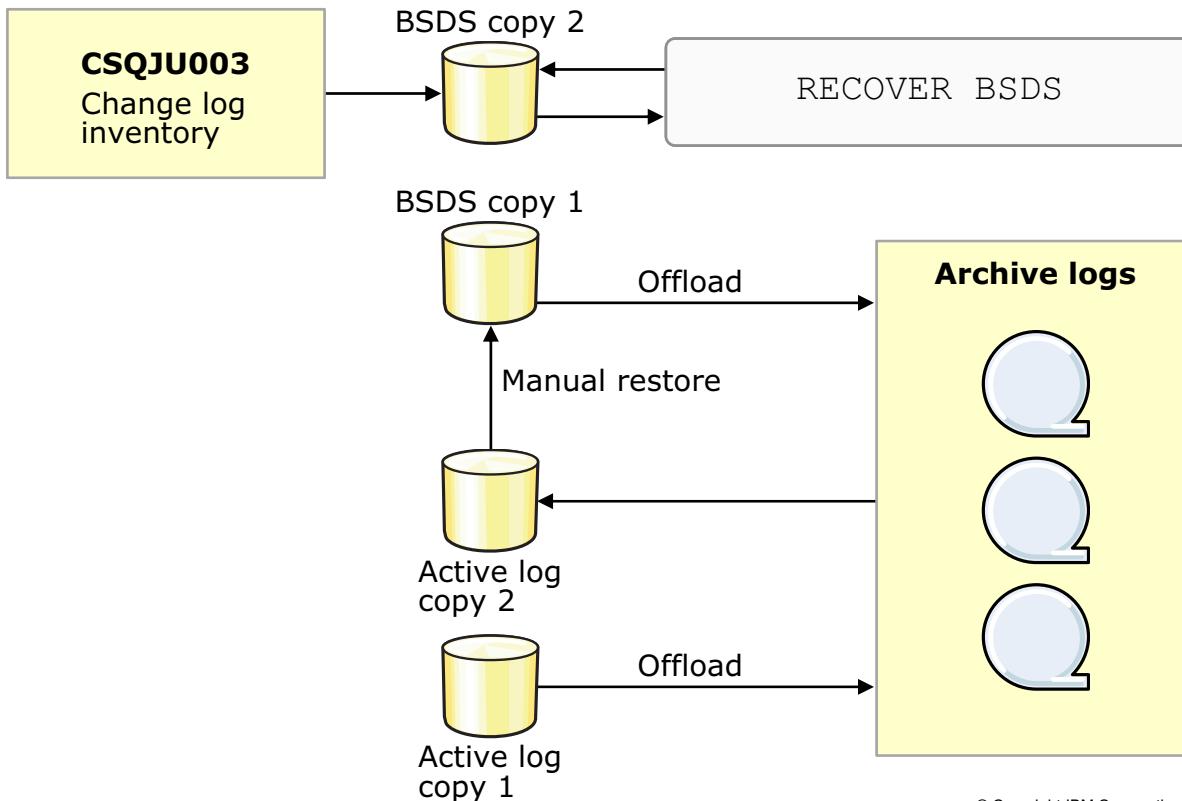
Figure 13-29. Display usage for DATASET command output

WM3021.1

Notes:

- **TYPE(DATSET)** returns log data set-oriented results. It names the log data set where the corresponding RBA is located. Additionally, it indicates the start RBA of the oldest active unit of work and the associated log data set. Comparing this RBA to the current one might indicate the existence (or absence) of long-running tasks.
- **TYPE(ALL)** combines both PAGESET and DATASET types.

BSDS recovery considerations



© Copyright IBM Corporation 2015

Figure 13-30. BSDS recovery considerations

WM3021.1

Notes:

The easiest and suggested way to minimize the impact of a damaged BSDS is by using DUAL bootstrapping.

If dual BSDS data sets are used, the queue manager ensures that these sets remain synchronized.

At restart, the queue manager checks that both BSDS data sets exist and that they are identical in content. If the time stamps in the BSDSs are different, IBM MQ detects the mismatching time stamps on start (message CSQJ120E) and resynchronizes the BSDS data sets by using the more recent copy. The queue manager starts normally, issuing the following message:

```
14.45.45 STC23412 CSQJ120E +Q10D DUAL BSDS DATA SETS HAVE UNEQUAL TIME 331
      STAMPS, SYSTEM BSDS1=2008-01-30 13:50:22.49, BSDS2=2008-01-30
      331          14:42:41.61, UTILITY BSDS1=2006-10-06 14:36:03.28,
      BSDS2=2006-10-06
      331          14:36:03.28
14.45.45 STC23412 CSQJ130I +Q10D DUAL BSDS MODE RESTORED FROM BSDS
14.45.45 STC23412 CSQJ127I +Q10D SYSTEM TIME STAMP FOR BSDS=2008-01-30 14:42:41.61
```

If an I/O error occurs on one of the BSDS data sets while the queue manager is active, a message is issued that identifies the damaged copy. IBM MQ reverts to SINGLE BSDS mode.

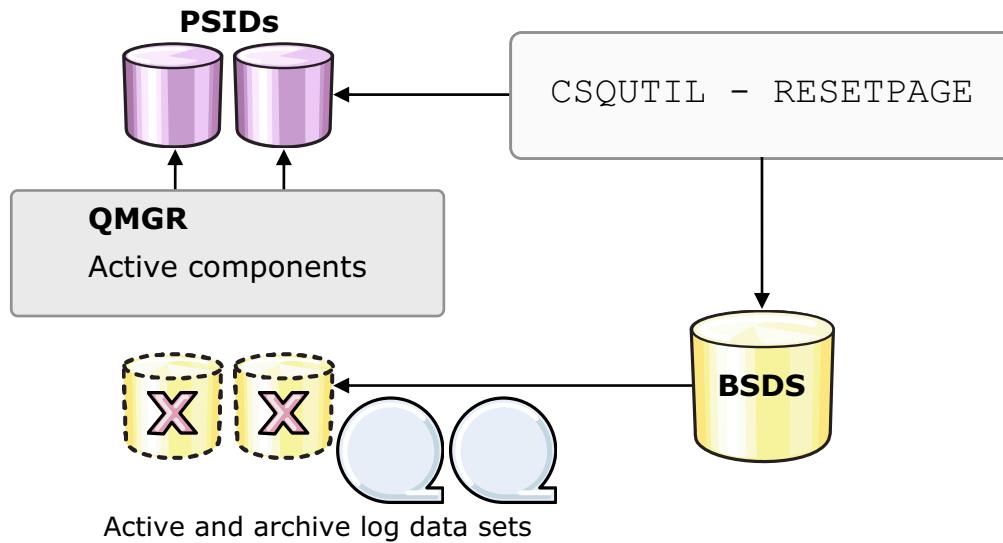
- While the queue manager is still running, you can delete and redefine a new copy for the damaged data set, and then use the RECOVER BSDS command to rebuild the second copy and reinstate dual BSDS mode.

If you get BSDS errors when running in SINGLE BSDS mode (or errors on BOTH copies running in DUAL BSDS mode), you must recover BSDS from the latest available archive:

- Whenever an active log is archived, the BSDS is also archived. When archiving to DASD, the BSDS archive data set uses the same naming convention as the associated log archive, except that the last qualifier starts with a “B” instead of an “A.”
- As you cannot determine offloads from a BSDS, you might have to search your MVS log for this information; look for the message CSQJ003I.
- Delete or rename the damaged BSDSs and define a new one by using the CSQ4BSDS JCL as a template.
- Use AMS REPRO to copy the archive BSDS to the replacement BSDS.
- Use the change log inventory utility (CSQJU003) to update the archive and (if required) the active log data set inventories and the active log RBA ranges on the replacement BSDS.
- Copy the updated BSDS to the second replacement BSDS, if you use dual BSDS.

Restart the queue manager with the newly constructed BSDS.

When logs are lost or damaged



- No way to recover logs: Always use dual logging
- To restart IBM MQ at the risk of lost or inconsistent data:
 - Manipulate PSIDs:
CSQUTIL - RESETPAGE
 - Manipulate BSDS: Perform a “cold start”

© Copyright IBM Corporation 2015

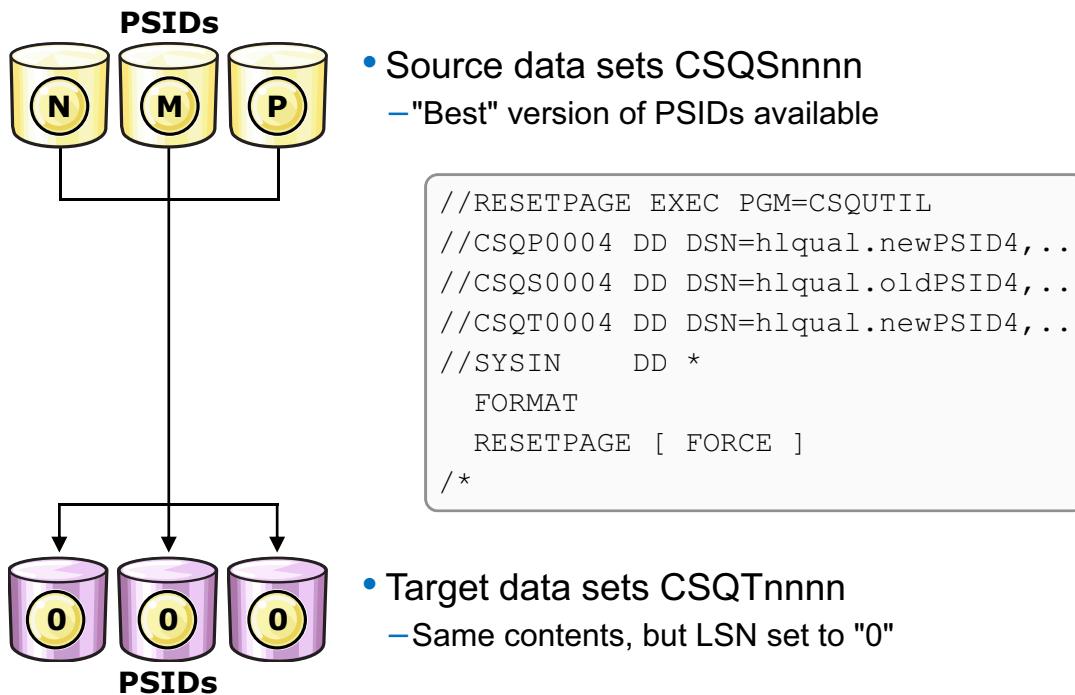
Figure 13-31. When logs are lost or damaged

WM3021.1

Notes:

- If you lose your logs, and the queue manager stops or is aborted, there is no way to recover. Some application data is lost and will probably remain in an inconsistent state.
 - Log data is the critical information that ensures recoverability of the user data. It is not protected except through dual logging.
 - If you are not using RAID or mirrored DASD, always use dual logging in a production environment.
- To restart the queue manager in this situation, there are two utility functions:
 - CSQUTIL option **RESETPAGE** is used to proceed with any available page set contents.
 - CSQJU003 option **CRESTART** is used to specify a reduced log range to be accessed through restart.

CSQUTIL RESETPAGE command



© Copyright IBM Corporation 2015

Figure 13-32. CSQUTIL RESETPAGE command

WM3021.1

Notes:

- The **RESETPAGE** option resets the Log Sequence Number (LSN) that is kept in the first page of each page set to zero.
- It is meant to be used to update a set of *consistent page sets* so that they can be used with a set of new (clean) BSDS and log data sets to start the queue manager.
- This option causes the queue manager to start from RBA 0, without performing any recovery actions. Thus, any contents of page sets are kept and take effect.
- CSQUTIL **RESETPAGE** might be used in one of two ways:
 - Page sets are copied (from CSQS to CSQT), and the RESET is applied to the new data sets.
 - With the **FORCE** option that is specified, RESET is done in place against the page sets referenced by DDnames CSQP.
- The **RESETPAGE** function must never be used for a subset of page data sets.
- Only "clean" page sets (that is, full backups) can be RESET this way.

CSQJU003 CRESTART command

```
//JU003 EXEC PGM=CSQJU003
//SYSUT1 DD DSN=hlqual.BSDS1,..
//SYSUT2 DD DSN=hlqual.BSDS2,..
//SYSIN     DD *
      CRESTART
      CREATE,ENDRBA=endrba
/*

```

- Creates a conditional restart control record (CRCR) in the BSDS
 - CRCR specifies a reduced log range to be used for restart
 - Log records beyond **endrba** are ignored and discarded
 - No page set must have LSN > **endrba**

© Copyright IBM Corporation 2015

Figure 13-33. CSQJU003 CRESTART command

WM3021.1

Notes:

- The conditional restart control record (CRCR) is recorded in the BSDS. It is used to restrict the scope of log records that are used by the queue manager restart to a specified range.
- It can be created and canceled at any time, whether the queue manager is running.
- Only the last CRCR specified is active, but previous ones remain recorded in the BSDS for documentation.
- No page set LSN must be higher than the **endrba** specified for restart.
- This option supports point-of-time recovery by using the data that is captured in response to ARCHIVE LOG QUIESCE processing.



Warning

Be sure that you know what you are doing.

Usually, application data is lost, the user data might be in an inconsistent state, or both.

Queue manager cold start

- Offline preparation:
 - Run CSQUTIL MAKEDEF with "DISPLAY ALL," including DISPLAY QMGR
 - Save the definition file that is created
- In case of disaster:
 - Re-create all page set, log, and BSDS data sets
 - Put the definition file into CSQINP2
 - Start the queue manager
- Result:
 - All objects are preserved or re-created
 - All messages are lost



Ensure all alternatives are exhausted before continuing with this procedure

© Copyright IBM Corporation 2015

Figure 13-34. Queue manager cold start

WM3021.1

Notes:

This is the suggested method to prepare for a queue manager restart after a disaster situation, if either one of the options that are shown earlier cannot be used or if continuing work with an old or inconsistent set of messages seems less desirable than dropping all messages.

- Prepare a CSQUTIL COMMAND-MAKEDEF job that explicitly displays commands naming each object type, as there is no "DISPLAY ALL" command.
This job should be executed at regular intervals, possibly as part of the page set backup procedure.

The **CSQUTIL MAKEDEF** option produces an **ALTER QMGR** command with the complete set of queue manager attributes when processing **DISPLAY QMGR**.

Thus, queue manager attributes can be saved and restored as part of the cold start processing.

Dynamic page set handling

```
DEFINE PSID(psid-number)
    BUFFPOOL(n) DSN(dsname)
    EXPAND(USER|SYSTEM|NONE)
```

- PSID data set must exist and must be preformatted
- Page set is dynamically allocated
- If the buffer pool specified does not exist, it is created

```
ALTER PSID(psid-number)
    EXPAND(USER|SYSTEM|NONE)
```

- Is used only to change the PSID expand option

```
DELETE PSID(psid-number)
```

- The page set identified must have no storage class (**STGCLASS**) referencing it

© Copyright IBM Corporation 2015

Figure 13-35. Dynamic page set handling

WM3021.1

Notes:

The page set management options of CSQUTIL provide tools to copy, save, and restore either the contents of a complete page set or the contents of single queues. These utility functions might be the basis for page set and queue management strategies that you develop and implement as an administrator. Page sets can be dynamically allocated, and brought into use by using the `DEFINE PSID` command.

- The page set must be created first and must be prepared by the CSQUTIL FORMAT function with the appropriate number named by the DD statement.
- The new page set can be associated with an existing buffer pool or a new one.

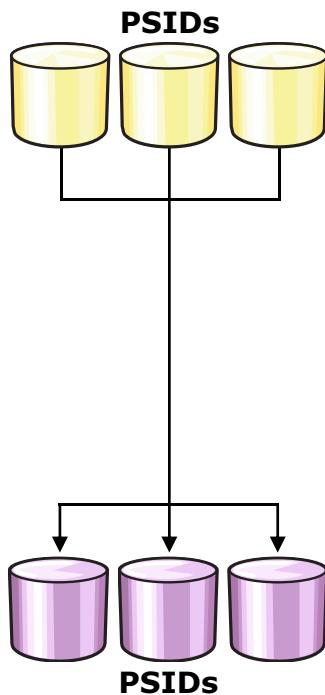
A new buffer pool is allocated with 1000 buffers, if available. Its size can be changed by use of the `ALTER BUFFPOOL` command afterward.

- If the new page set is to be used permanently, it should be added to the *MSTR JCL. However, dynamic addition of a page set is “remembered” over a queue manager restart and the queue manager attempts dynamic allocation at restart time if the page set does not appear in the JCL.

The `ALTER PSID` command is used for changing the expand setting dynamically. This setting is logged, and thus kept over a queue manager restart.

An old page set can be deallocated by use of the `DELETE PSID` command. No storage class must use the named page set, regardless of whether a queue uses the **STGCLASS**.

CSQUTIL COPYPAGE command



- Source data sets CSQSnnnn

```
//COPYPAGE EXEC PGM=CSQUTIL
//CSQP0004 DD DSN=hlq.newPSID4,...
//CSQS0004 DD DSN=hlq.oldPSID4,...
//CSQT0004 DD DSN=hlq.newPSID4,...
//SYSIN      DD *
      FORMAT
      COPYPAGE
/*
```

- Target data sets CSQTnnnn
 - Same contents, but more space
 - Used to EXPAND page sets

© Copyright IBM Corporation 2015

Figure 13-36. CSQUTIL COPYPAGE command

WM3021.1

Notes:

The **COPYPAGE** function is meant to be used for *expanding* page sets, not for taking backups. It can be used only when the queue manager is not running.

- The target page sets identified by DDnames CSQT must be formatted before the **COPYPAGE** takes place; this step can be done in a single run of CSQUTIL.
- The target data sets must be larger than the appropriate source data sets.
- Multiple page sets can be copied through a single invocation of CSQUTIL; the DDnames tell the utility what to do.
- All queues and messages on the page set are copied, and so are the queue manager object definitions on page set zero.

Details are documented in the CSQUTIL section of the IBM MQ Knowledge Center.

CSQUTIL queue and page set management

```
//COPYPAGE EXEC PGM=CSQUTIL,PARM='MQ01'
//CSQUOUT   DD DSN=hlqual.PSID03.SEQCOPY,...
//OUTPUTX   DD DSN=hlqual.QUEUE.UNLOAD,...
//SYSIN     DD *
```

Command	Description
COPY	Copy the whole contents of a page set or named queues to sequential files while the Queue Manager is running
LOAD	Restore copied PSIDs and queues
SCOPY	Same as COPY, but to be used when the Queue Manager is stopped (PSID data sets must be named in JCL)
EMPTY	Delete all messages from a named queue or all queues on a page set

© Copyright IBM Corporation 2015

Figure 13-37. CSQUTIL queue and page set management

WM3021.1

Notes:

All of the functions that are shown in the figure can either be applied to a complete page set or to single queues. The online functions of CSQUTIL use regular application MQI calls, so processing is treated like regular application work.

- Within the **COPY** and **SCOPY** functions, options can be added to include all queues or predefined queues only, thus excluding dynamic queues.
- **COPY**, **EMPTY**, and **LOAD** can be used to move queues between page sets or even between queue managers.
 - In this case, applications must be prevented from accessing the queue in work, and the queue must be redefined before it is loaded.
- Details are documented in the CSQUTIL section of the IBM MQ Knowledge Center.

Implementing above the bar buffer pools: BUFFPOOL

- **LOCATION**, or **LOC** attribute used to set buffer pool location to **LOC (ABOVE)**
 - Include the buffer pool allocations in CSQINP1, or
 - Change dynamically with the **ALTER BUFFPOOL (nn) LOC (ABOVE)**
- If LOC is ABOVE, BUFFERS attribute valid range accepts value 999,999,999
- New option to back buffer pool by real storage for the life of the queue manager
 - **PAGECLAS** set to **FIXED4KB**

Requires
OPMODE=NEWFUNC



To mitigate impact to other address spaces, ensure that there is enough real storage available before using **PAGECLAS** at **FIXED4KB**

MQ00 ALTER BUFFPOOL(2) LOC(ABOVE)

```
CSQP024I MQ00 Request initiated for buffer pool 2
CSQ9022I MQ00 CSQPALTB ' ALTER BUFFPOOL' NORMAL COMPLETION
CSQP054I MQ00 Buffer pool 2 is now located above the bar
CSQP023I MQ00 Request completed for buffer pool 2, now 067
has 50000 buffers
CSQY220I MQ00 CSQSCTL Queue manager storage usage: 068
local storage: used 388MB, free 1307MB: above bar: used 448MB, free 1GB
```

© Copyright IBM Corporation 2015

Figure 13-38. Implementing above the bar buffer pools: BUFFPOOL

WM3021.1

Notes:

Above the bar buffer pools is a new IBM MQ for z/OS V8 capability. OPMODE must be set to (NEWFUNC, 800) to use the above the bar buffer pool enhancement.

Unit summary

- Describe the contents and use of the IBM MQ for z/OS log and bootstrap data sets
- Describe and use the options to dynamically adjust the number and sizes of logs, page data sets, and buffers
- Determine the appropriate log and archive parameter module options according to requirements
- Describe how IBM MQ for z/OS maintains consistency with IMS, CICS, and Resource Recovery Services (RRS)
- Use the appropriate commands to display and control affected units of recovery
- Describe how IBM MQ for z/OS cleans up inconsistent data on restart
- Perform queue and page set management functions, such as expanding page data sets and restoring unloaded data

© Copyright IBM Corporation 2015

Figure 13-39. Unit summary

WM3021.1

Notes:

Checkpoint questions (1 of 2)

1. Checkpoint frequency is controlled by:
 - a. LOGLOAD parameter
 - b. WRTHRSH parameter
 - c. CHCKPNT parameter
 - d. OFFLOAD parameter

2. What command or utility helps identify which logs or archive data sets are needed to do recovery from fuzzy backups?
 - a. DIS ARCHIVE
 - b. DIS USAGE
 - c. CSQJU004 utility
 - d. CSQUTIL MADEDEF

© Copyright IBM Corporation 2015

Figure 13-40. Checkpoint questions (1 of 2)

WM3021.1

Notes:

Write your answers here:

1.

2.

Checkpoint questions (2 of 2)

3. True or False: New pages and logs can be added only when the queue manager is created.

4. What features or enhancements require the OPMODE parameter to be set to NEWFUNC?
 - a. Use of a client ID greater than 12 clients
 - b. Implementation of 8-byte RBA
 - c. Buffers above the 2-GB line
 - d. Channel authentication

© Copyright IBM Corporation 2015

Figure 13-41. Checkpoint questions (2 of 2)

WM3021.1

Notes:

Write your answers here:

3.

4.

Checkpoint answers (1 of 2)

1. Checkpoint frequency is controlled by:

- a. LOGLOAD parameter
- b. WRTHRSH parameter
- c. CHCKPNT parameter
- d. OFFLOAD parameter

Answer: a, LOGLOAD parameter in the CSQ6SYSP macro

2. What command or utility helps identify which logs or archive data sets are needed to do recovery from fuzzy backups?

- a. DIS ARCHIVE
- b. DIS USAGE
- c. CSQJU004 utility
- d. CSQUTIL MADEDEF

Answer: b, DIS USAGE

© Copyright IBM Corporation 2015

Figure 13-42. Checkpoint answers (1 of 2)

WM3021.1

Notes:

Checkpoint answers (2 of 2)

3. True or False: New pages and logs can be added only when the queue manager is created.

Answer: False. New pages and logs can be added to an existing queue manager.

4. What features or enhancements require the OPMODE parameter to be set to NEWFUNC?
- Use of a client ID greater than 12 clients
 - Implementation of 8-byte RBA
 - Buffers above the 2-GB line
 - Channel authentication

Answer: a, b, c

© Copyright IBM Corporation 2015

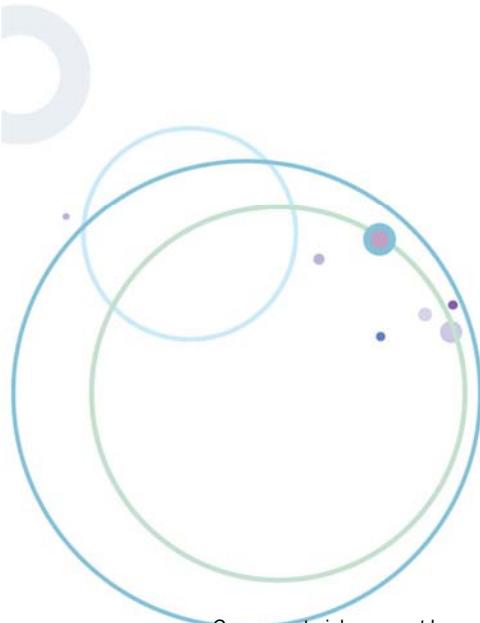
Figure 13-43. Checkpoint answers (2 of 2)

WM3021.1

Notes:

Exercise 10

Working with file handling utilities



© Copyright IBM Corporation 2015

Course materials may not be reproduced in whole or in part without the prior written permission of IBM.

10.0

Figure 13-44. Exercise 10

WM3021.1

Notes:

Exercise objectives

After completing this exercise, you should be able to:

- Dynamically create a page set
- Move buffers for a page set to reside above the 2GB line
- Move queues to a new page set
- Dynamically add and switch to a new log
- Back up object definitions
- List the contents of the BSDS data set

© Copyright IBM Corporation 2015

Figure 13-45. Exercise objectives

WM3021.1

Notes:

Unit 14. Support for CICS, IMS, and HTTP applications

What this unit is about

This unit covers the ways in which WebSphere MQ for z/OS interacts with CICS and IMS. It explains what is necessary during setup and during ongoing production, what steps need to be considered, and alternative approaches.

What you should be able to do

After completing this unit, you should be able to:

- Describe how to set up an adapter and bridge for CICS and IBM MQ
- Describe how to configure and use the IBM MQ IMS OTMA Bridge
- Describe how to configure the IBM MQ IMS Adapter
- Compare the IBM MQ IMS OTMA Bridge and the IBM MQ IMS adapter
- Summarize the IBM MQ Bridge for HTTP

Unit objectives

- Describe how to set up an adapter and bridge for CICS and IBM MQ
- Describe how to configure and use the IBM MQ IMS OTMA Bridge
- Describe how to configure the IBM MQ IMS Adapter
- Compare the IBM MQ IMS OTMA Bridge and the IBM MQ IMS adapter
- Summarize the IBM MQ Bridge for HTTP

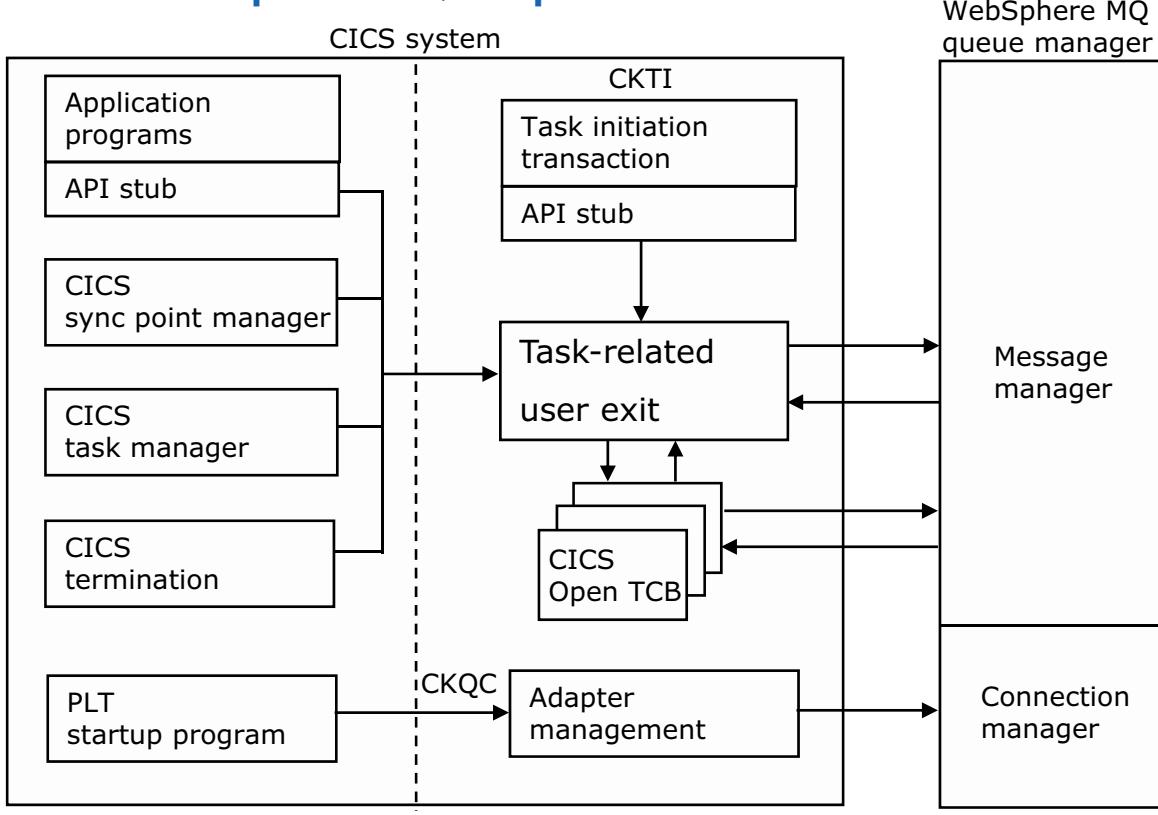
© Copyright IBM Corporation 2015

Figure 14-1. Unit objectives

WM3021.1

Notes:

CICS WebSphere MQ adapter



© Copyright IBM Corporation 2015

Figure 14-2. CICS WebSphere MQ adapter

WM3021.1

Notes:

CICS contains a general-purpose interface for communicating with other resource managers, such as database management systems (DBMS), for granting CICS applications access to the external resources. This interface is called the resource manager interface (RMI). For each external resource manager (ERM) connected, there must be an attachment component specific to the ERM.

The ERM attachment for CICS interfacing WebSphere MQ for z/OS is called an adapter. The CICS WebSphere MQ adapter:

- Is made up of transactions, programs, and map sets that are delivered with WebSphere MQ for z/OS
- Allows CICS programs to access WebSphere MQ for z/OS objects by using the Message Queue Interface (MQI)
- Provides a ready-to-use trigger mechanism that allows CICS transactions to be started when a WebSphere MQ message queue meets a trigger criterion
- Contains a BMS-based command interface to display and control the connection between the CICS TS region and an instance of WebSphere MQ

A CICS TS region can be connected to only one WebSphere MQ system (queue manager) at a time.

CICS TS and the WebSphere MQ Queue Manager must operate in the same MVS.

Communication is by threads by using MVS cross-memory services.

Since CICS TS V3.2, WebSphere MQ components are supplied in CICS TS. The components that are transferred include the CICS IBM MQ adapter, the CICS IBM MQ trigger monitor, and the CICS IBM MQ bridge. The components were made threadsafe, and are enabled to use CICS open TCBs. Exploitation of OTE benefits threadsafe applications by using WebSphere MQ. For multiple WebSphere MQ requests, TCB switching can be avoided, resulting in a saving of processor and an increase in overall throughput, because applications can now run on multiple open TCBs and avoid bottlenecking.

The CICS region uses open TCBs in L8 mode to connect to WebSphere MQ queue managers. When a CICS task makes a request to connect to WebSphere MQ, it obtains an L8 TCB from the pool in the CICS region. It keeps the L8 TCB from the time it is allocated to the end of the task. Even if the CICS task switches back to run on the QR TCB or makes no further requests to connect to WebSphere MQ, the L8 TCB is not released until the CICS task ends. Each concurrent CICS task that connects to WebSphere MQ therefore requires one L8 TCB during the task.

Starting and stopping the adapter

Starting

- SIT based auto-connect
- Define the connection between CICS and WebSphere MQ
- PLTPI based auto-connect
- CKQC transaction command interface (START option)

MQCONN=YES

MQCONN connection definition

**DFHPLT TYPE=ENTRY,
PROGRAM=DFHMQC0D**

Stopping

- Automatically at CICS shutdown
- Transaction **CKQC Stop option QUIESCE|FORCE**

© Copyright IBM Corporation 2015

Figure 14-3. Starting and stopping the adapter

WM3021.1

Notes:

MQCONN is a CICS resource that defines the attributes of the connection between CICS and WebSphere MQ. You must install an MQCONN resource before you start the connection between CICS and WebSphere MQ.

The MQCONN resource specifies the following settings for the connection between CICS and WebSphere MQ:

- The name of either a single WebSphere MQ queue manager or a queue-sharing group of WebSphere MQ queue managers, to which CICS connects (MQNAME attribute)
- The resynchronization strategy that CICS adopts if the connection ends with units of work outstanding (RESYNC MEMBER attribute)
- The name of the default initiation queue for the connection (INITQNAME attribute)

Before any MQI activity can occur, the connection must be started in one of two ways:

- Either automatically at initialization by using the SIT parameter MQCONN=YES or the program list table entry for program CSQCC0DF
- Or by using the CSQC adapter control transaction

All methods refer to options specified in the MQCONN CICS resource definition to identify the WebSphere MQ subsystem and other subordinate specifications.

The connection will be created, and the threads activated, after the CONTROL IS GIVEN TO CICS start message.

- The WebSphere MQ system ID must be determined either from the MQCONN definition or through the CKQC input panel.
- On non-immediate shutdowns of CICS TS, the WebSphere MQ adapter automatically quiesces active threads.
- Be careful when using the FORCE option to stop the adapter or do an IMMEDIATE shutdown of CICS TS; this action is likely to abend active threads.

CICS IBM MQ adapter CICS resources

- DFHMQ group in DFHLIST supplies CSD definitions for the CICS-WebSphere MQ adapter
- WebSphere MQ supplies input to DFHCSDUP
- CICS-WebSphere MQ adapter transactions
 - **CKQC** Adapter control interface transaction
 - **CKTI** Trigger monitor transaction
 - **CKAM** Alert monitor transaction and others

© Copyright IBM Corporation 2015

Figure 14-4. CICS IBM MQ adapter CICS resources

WM3021.1

Notes:

CICS supplies CSD definitions for the CICS WebSphere MQ adapter in group DFHMQ as part of DFHLIST. WebSphere MQ provides the definitions for sample application programs in the CSQ4SAMP group.

DFHMQ contains these definitions:

- The supplied adapter programs
- The supplied adapter management transactions
- The supplied sets of BMS maps, which are required for the adapter panels

Program names have the format `DFHMQxxxx` (and some `CSQCxxxx` aliases that are provided for compatibility), and transaction names have the format `CKxx`.

CKQC is the CICS transaction that starts the WebSphere MQ adapter display and control function.

One or more instances of **CKTI** can be started as trigger monitors, each against a separate initiation queue.

CICS IBM MQ bridges

Two approaches for different kinds of CICS non-MQ literate applications that are to serve any IBM MQ queues:

- CICS DPL bridge
 - Interfaces with nonterminal-related CICS server programs
- CICS 3270 bridge
 - Interfaces with earlier CICS 3270 applications

The IBM MQ CICS adapter is required as a prerequisite for both bridge modes.

© Copyright IBM Corporation 2015

Figure 14-5. CICS IBM MQ bridges

WM3021.1

Notes:

Bridges are used for interfacing IBM MQ applications that can run under any IBM MQ platform with programs that do not use the MQI.

For CICS, these programs might be:

- Server programs that are started by using EXEC CICS LINK and that also communicate with the caller by using a COMMAREA
- Transactions that are normally started from 3270 terminals and use CICS terminal control functions (EXEC CICS RECEIVE and EXEC CICS SEND) for data exchange

The bridge mechanism must map and convert the contents of an IBM MQ message queue to a structure that is acceptable to the CICS program.

The most interesting use of bridges is to interface with existing applications that are not IBM MQ aware.

As provided by IBM MQ, the bridge mechanisms run within CICS themselves, and need the CICS IBM MQ adapter to be active to access IBM MQ message queues.

IBM MQ CICS bridge

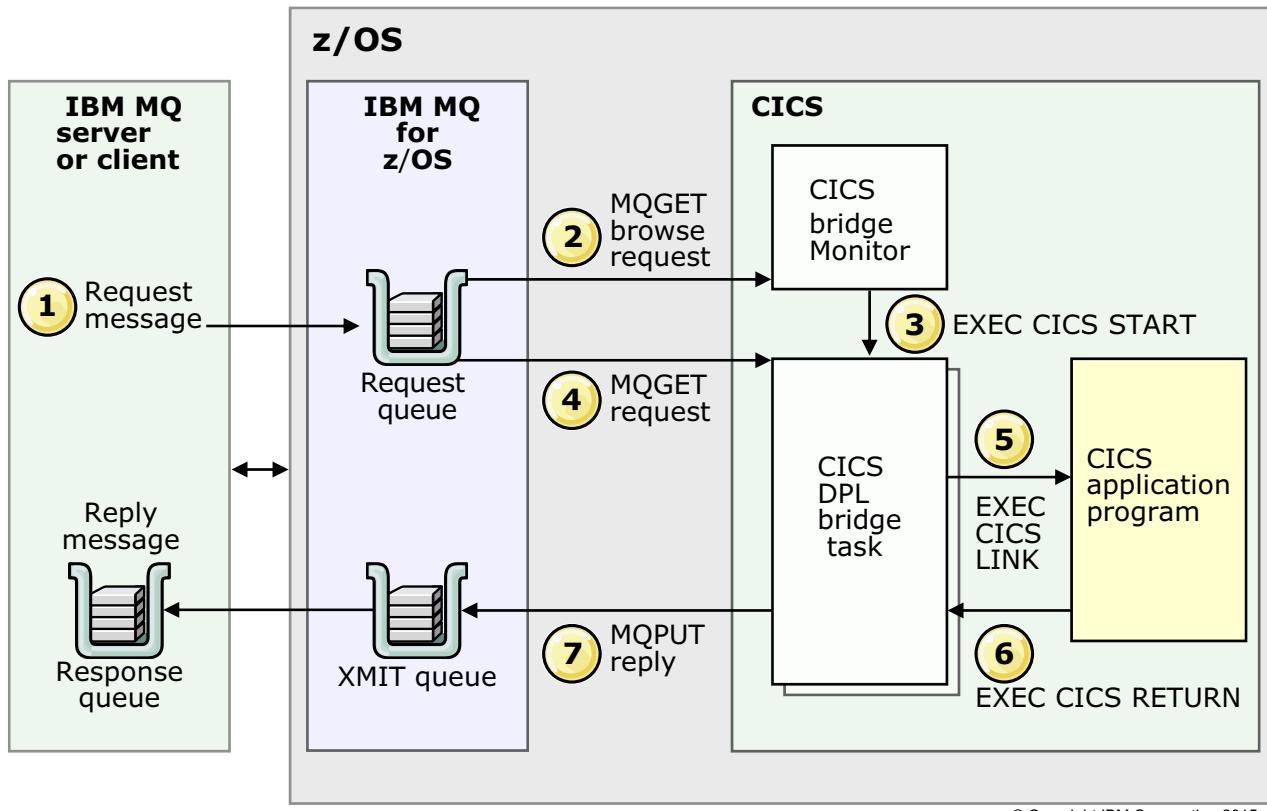


Figure 14-6. IBM MQ CICS bridge

WM3021.1

Notes:

The bridge makes it possible for an application running outside CICS to use IBM MQ messages to run a program or transaction on CICS and get a reply.

The bridge provides for MQI support, so that the application program running outside CICS does not need to issue IBM MQ calls. This capability allows reuse of older applications.

There are two CICS applications compatible with the bridge:

- DPL programs, which use EXEC CICS LINK
- CICS 3270 transaction programs that use either basic mapping support (BMS) or control commands

Looking at the figure, the process followed by the DPL variation of the bridge is:

1. The application message requesting a CICS program to run is put on the request queue; the originating application might run on any IBM WebSphere MQ server or client platform.
2. The monitor task CKBR (program CSQCBR00) constantly browses the queue and recognizes that a new request is waiting.

3. Relevant authentication tasks are done and a bridge task with the default transaction code of CKBM (program CSQCBP00) is started with the appropriate authority.
The bridge task already provides the CICS task environment for the user or application program to be started, which is comparable to a CICS mirror task.
4. The bridge task removes the message from the request queue.
5. The bridge task builds a COMMAREA with the data from the message and issues the EXEC CICS LINK for the program requested.
6. The program returns the response in the COMMAREA used by the request.
7. The bridge task reads the COMMAREA, creates a message, and puts it on the reply-to queue specified in the request message.
8. The CICS bridge task ends.
 - There might be one or more CKBR monitor task transactions that are active, but typically every CKBR task serves its request queue exclusively.
 - While a CKBM bridge task is serving a request, another message can arrive on the request queue and cause another bridge task to start, perhaps for same or a different application program.

So, as with CICS mirror tasks, multiple bridge tasks might be active at the same time, and are independent from each other. The only thing that they have in common is the request queue through which the invocation messages arrive.

The 3270 bridge follows a similar process:

- The 3270 bridge is for interfacing with existing 3270 applications, that is, traditional (*existing*) CICS transactions where the presentation logic is not separated from the business logic.
- The bridge is able to intercept terminal commands that are issued by the CICS transaction and present them to a “bridge exit” for both inbound and outbound (“receiving” and “sending”) communication by using an architected interface that is called the bridge exit area (BRXA).
- The bridge monitor task that is supplied by WebSphere MQ (CKBR) is the same that serves the DPL bridge, although it determines its behavior differently from the request message:
 - This program does an MQGET of the request message from the request queue, performs the interactions with the user transaction, and does an MQPUT of the application response to the reply-to queue.
 - The requester application might process the reply as soon as it is visible on the reply-to queue.
- The user transaction is not aware of the different way it is started and processed, including any IBM WebSphere MQ-related issues. All the interfacing is performed by the bridge exit program that is supplied by IBM WebSphere MQ and the CICS TS 3270 bridge system function.

CICS bridge request message (MQCIH) contents

Term	Description
Link Type	DPL or 3270
UoW Control	<ul style="list-style-type: none"> • Single • First, Middle, or Last • Commit or Backout
Transaction ID	3270 User Transaction or Bridge Task Tx
Conversational	Yes or No (DPL)
NextTransid	As specified by user program (3270)
Authenticator	Password or Passticket
Facility	Termid returned for Pseudoconv. (3270)
Attention ID	Initial value of AID key (3270)
Cursor Position	Initial value (3270)

© Copyright IBM Corporation 2015

Figure 14-7. CICS bridge request message (MQCIH) contents

WM3021.1

Notes:

- All specifications that are required for the particular processing within CICS must be put into the request message by the client application.
- The CICS components that are supplied by IBM WebSphere MQ, that is, the monitor and the bridge task program (for DPL) or the bridge exit program (for 3270), recognizes these options and act appropriately.
- The requester (client) application must be *CICS aware* as to the contents and structure of the request message.

This component is the only *adoption* related to CICS that must be coded in the requesting application.

- As can be seen from the sample fields that are shown, most parts of the MQCIH structure address the 3270 bridge.

A DPL program might be run without using the MQCIH message format, if default options are suitable.

Preparing CICS IBM MQ applications

Use regular CICS procedures (DFHYIT*) with other specifications:

- SYSLIB concatenation of the compile step:
 - Add IBM MQ language support libraries for expanding IBM MQ structures and constants:
SCSQASM, SCSQCOB, SCSQC370, SCSQPLI
- Linkedit step:
 - Add the following JCL to access the IBM MQ program stub:

```
//LKED.CSQSTUB DD DSN=MQS.SCSQLOAD,DISP=SHR
//LKED.SYSIN      DD *
      INCLUDE CSQSTUB(CSQCSTUB)
      NAMEpgmname (R)
```

© Copyright IBM Corporation 2015

Figure 14-8. Preparing CICS IBM MQ applications

WM3021.1

Notes:

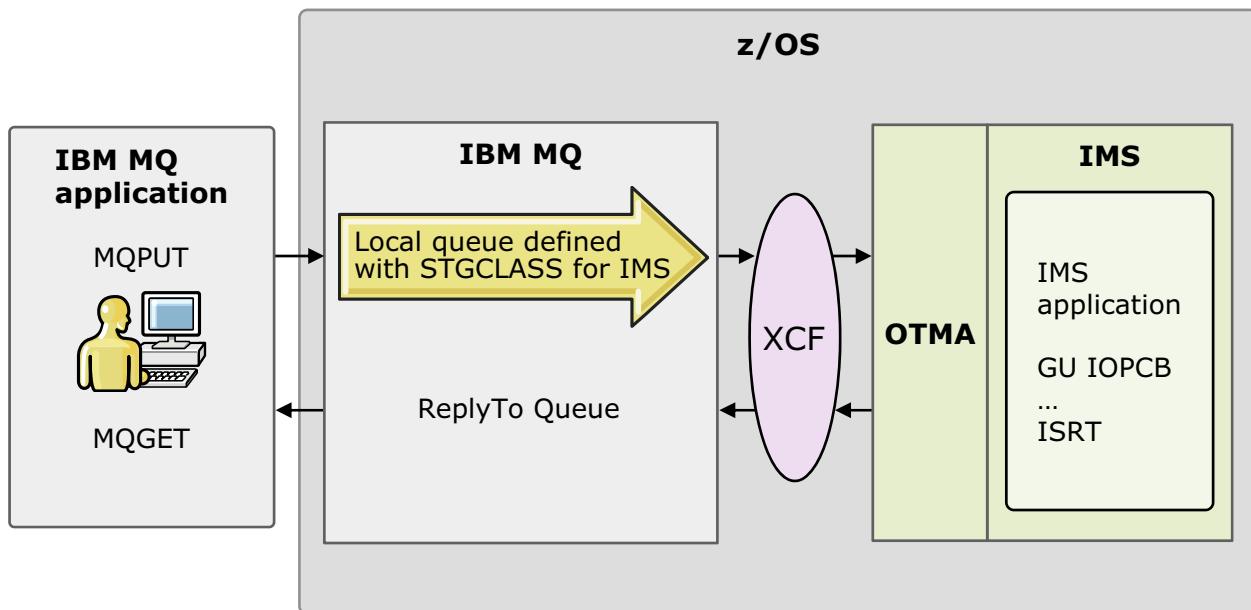
As the MQI is a call interface, there is no need for a special precompile.

For use of the MQI, application programs refer to WebSphere MQ structures that must be made available to the compile step.

To interface with the attachment, CICS programs that use the MQI must be link-edited with the MQ-supplied program stub CSQCSTUB.

Existing CICS WebSphere MQ applications can run unchanged with the CICS WebSphere MQ adapter that is shipped with CICS. It is not necessary for you to compile or link-edit them again. For new or changed applications, you can use existing link-edit procedures. You can use stubs that are shipped with CICS or WebSphere MQ, unless the application uses the new API calls that were added in version 7 of WebSphere MQ. The WebSphere MQ Version 7 API calls are supported in CICS only when you use the stubs that are shipped with CICS, not the stubs that are shipped with WebSphere MQ.

IBM MQ IMS OTMA bridge



© Copyright IBM Corporation 2015

Figure 14-9. IBM MQ IMS OTMA bridge

WM3021.1

Notes:

The IBM WebSphere MQ-IMS bridge provides the following overall functionality enables invocation of existing IMS transactions by placing a message in an IBM MQ queue. The IMS transaction does not require any changes. The bridge enables implicit IBM WebSphere MQ API support.

The IMS transaction does its work as before:

- When the IMS transaction does a GET UNIQUE, OR GU to the IOPCB, it gets the incoming message.
- Any output that the IMS transaction returns by using an INSERT (ISRT) to the IOPCB is put on the Reply_To_Queue specified in the client.

Unlike the CICS bridge facilities, the IMS bridge is not dependent on an adapter. The bridge uses the z/OS cross-system coupling facility (XCF) and the IMS Open Transaction Manager Access protocol.

Applications that use OTMA issue XCF calls across the two subsystems, much in the same way as any client and server application. OTMA is used to code IMS clients such as the TCP client. The IBM MQ IMS bridge is an OTMA client. The queue inbound to IMS is defined with a special IBM MQ storage class that ties it to the OTMA group.

IBM MQ IMS OTMA bridge implementation checklist

- Ensure IBM MQ and IMS are at required maintenance levels
- Define bridge to IMS
- Define bridge to IBM MQ:
OTMACON parameter of CSQ6SYSP
- Define “bridge” STGCLASS
- Define “bridge” queues
- Create RACF facility definitions
- Code or modify IBM MQ client application to include correct format and *ReplyToQ*

© Copyright IBM Corporation 2015

Figure 14-10. IBM MQ IMS OTMA bridge implementation checklist

WM3021.1

Notes:

This slide summarizes the steps that are required to configure the bridge on the IMS and IBM MQ subsystems.

Although the bridge configuration can be tested with security disabled, one of the challenging parts to the configuration is getting the RACF definitions that are done in time. The rest of the configuration is simple.

Define bridge to IMS

Definition in DFSPBxx:

- OTMA=Y ,
- GRNAME=HARRY ,
- OTMANM= APPL8

XCF Group Name: **HARRY**
IMS Member Name: **APPL8**
IBM MQ Member Name: **VC5**

- Indication of successful definition: Find message:

DFS2360I 12:50:39 XCF GROUP JOINED SUCCESSFULLY

© Copyright IBM Corporation 2015

Figure 14-11. Define bridge to IMS

WM3021.1

Notes:

To define the bridge to IMS, update the DFSPBxx macro. After it is completed, a message that the definition is successful is displayed.



Define bridge to IBM MQ

- Definition in CSQ6SYSP macro of zparms:
- OTMACON= (HARRY , VC5 , DFSYDRU0 , 2147483647 , CSQ)
- Indication of successful definition: Find message:
**CSQ2010I (VC5 CSQ2MEM0 CONNECTED TO PARTNER,
XCFGNAME=HARRY XCFMNAME=APPL8**

XCF Group Name: **HARRY**
IMS Member Name: **APPL8**
IBM MQ Member Name: **VC5**

© Copyright IBM Corporation 2015

Figure 14-12. Define bridge to IBM MQ

WM3021.1

Notes:

In the IBM MQ side, the bridge is configured in the OTMACON parameter of the CSQ6SYSP macro. As with the IMS side after the queue manager restarts with the updated OTMACON, look for a success message as indicated in the figure.

IMS /DIS OTMA command

- A successful configuration displays “ACCEPT TRAFFIC”

GROUP/MEMBER	XCF-STATUS	USER-STATUS	SECURITY
HARRY			
-APPL8	ACTIVE	SERVER	FULL
-VC5	ACTIVE	ACCEPT TRAFFIC	
00041/111632	SYS3		

© Copyright IBM Corporation 2015

Figure 14-13. IMS /DIS OTMA command

WM3021.1

Notes:

After IMS, IBM MQ, and the RACF definitions are completed, the outcome of the configuration can be checked with the display OTMA command.

In the display that is shown in the figure, security is activated. You need to look for the “ACCEPT TRAFFIC” status.

If you have a safe developing environment, that is if you trust the inbound clients, it is possible to temporarily disable security to recheck the /DIS OTMA status with the command /SECURE OTMA NONE. Setting security off might help determine whether any problems preventing receipt of an “ACCEPT TRAFFIC” status are caused by security.

However, the optimal case is to have full security implemented.

Coding messages short format: No IIH header

- Messages that use default bridge attributes must be prefixed by the following fields before the message data:
 - The length of the message + 12 (**LL**)
 - Binary zeroes required by IMS (**ZZ**)
 - Eight-character name of the IMS transaction (**TRANID**)

<LL><ZZ><TRANID><DATA>
- MQMD Format for messages that use default attributes:
MQFMT_IMS_VAR_STRING
- Default attributes are documented in the IBM MQ Application Programming Guide

```
length - 112 bytes
00: 0070 0000 C2D9 C9C4 C9D4 C2E3 D9D5 F0F1 '..H IMBTRN01BRID'
10: C7C5 40E3 C5E2 E340 D4C5 E2E2 C1C7 C540 'GE TEST MESSAGE '
20: E6C9 E3C8 D6E4 E340 C1D5 40C9 C9C8 40C8 'WITHOUT AN IIH H'
30: C5C1 C4C5 D900 0000 0000 0000 0000 0000 'EADER.....'
40: 0000 0000 0000 0000 0000 0000 0000 0000 '.....'
50: 0000 0000 0000 0000 0000 0000 0000 0000 '.....'
60: 0000 0000 0000 0000 0000 0000 0000 0000 '.....'
```

© Copyright IBM Corporation 2015

Figure 14-14. Coding messages short format: No IIH header

WM3021.1

Notes:

There are two ways to format IBM MQ message for the IMS bridge. If you can accept all the IIH header default values, the message might be formatted without an IIH.

For the length, add 12 to the length of the message.

In this example, the IMS transaction name is IMBTRN01, and is followed by the message data starting with "BRIDGE TEST".

For this option, use **MQFMT_IMS_VAR_STRING** in the MQMD Format field.



Note

If no IIH header is used, commit mode defaults to 0, COMMIT_THEN_SEND.

Messages that are sent with commit mode attribute set to COMMIT_THEN_SEND or zero, are required to be persistent.

Coding bridge messages using IIH header (1 of 2)

- Messages requiring specific attributes must be additionally prefixed by the IIH structure:
<IIH><LL><ZZ><TRANID><DATA>
 - **LL** is the message length + 12 (the IIH length is **not** added)
 - **MQMD Format** for messages using the IIH header:
MQFMT_IMS
- Fields in the IIH structure are documented in the IBM MQ Application Programming Reference

© Copyright IBM Corporation 2015

Figure 14-15. Coding bridge messages using IIH header (1 of 2)

WM3021.1

Notes:

If your application requires the IIH header, for instance, to set the COMMIT MODE, then it is necessary to include the IIH header, and set the MQMD Format to MQFMT_IMS.

The <LL><ZZ><DATA> follow the IIH header.

The length of the IIH header is not added to the LL or length field.

Coding bridge messages using IIH header (2 of 2)

```

length - 196 bytes
00: C9C9 C840 0000 0001 0000 0054 0000 0311 'IIH .....'
10: 0000 0000 4040 4040 4040 4040 0000 0000 '.... . . .'
20: 4040 4040 4040 4040 4040 4040 4040 4040 ' . . . '
30: 4040 4040 4040 4040 4040 4040 4040 4040 ' . . . '
40: 0000 0000 0000 0000 0000 0000 0000 0000 '..... . . .'
50: 40F1 D540 [0070 0000 C9D4 C2E3 D9D5 F0F3 ' 1N ...IMBTRN03'
60: D4C5 E2E2 C1C7 C540 E6C9 E3C8 40C9 C9C8 'MESSAGE WITH IIH'
70: 40C8 C5C1 C4C5 D940 E4E2 C9D5 C740 C3D6 ' HEADER USING CO'
80: D4D4 C9E3 40D4 D6C4 C540 F100 0000 0000 'MMIT MODE 1....'
90: 0000 0000 0000 0000 0000 0000 0000 0000 '..... . . .'
A0: 0000 0000 0000 0000 0000 0000 0000 0000 '..... . . .'
B0: 0000 0000 0000 0000 0000 0000 0000 0000 '..... . . .'
C0: 0000 0000 '.... . '

```

© Copyright IBM Corporation 2015

Figure 14-16. Coding bridge messages using IIH header (2 of 2)

WM3021.1

Notes:

This slide shows the contents of the message including the IIH. The dotted line delineates the start of the LLZZTRANID section.

Other bridge considerations

- Translation, either with or without IIH formats: Defer to z/OS
- IMS multi-segment messages should be in format:
`<LL><ZZ><TRANID><DATA><LL><ZZ><DATA> . . .`



ReplyToQueue might sometimes contain IMS DFSxxx messages

© Copyright IBM Corporation 2015

Figure 14-17. Other bridge considerations

WM3021.1

Notes:

While not a requirement, it is better to defer translation of any character sets to the z/OS, or receiving side.

If the IMS transaction expects multi-segment messages, then the message should follow the format that is shown in the preceding slide. Adjust the message length after the first occurrence.



Important

When receiving replies from the IBM MQ IMS bridge, make provision to handle IMS messages that might be returned instead of the expected reply. These messages start with the “DFS” prefix. The reply can be checked that it does not start with “DFS” as a validation.

Transaction pipes: Tpipes

- Tpipes are like LTERMS in that they point back to client request
- Tpipes are control blocks
- Up to two tpipes per queue depending on commit mode
- Tpipe names depend on the commit mode and the PCB the transaction replies to:
 - Replying to IOPCB (no CHNG)
 - CM0: CSQ0xxxx – SYN
 - CM1: CSQ8xxxx
- Replying to ALTPCB (CHNG):
 - User named tpipe

© Copyright IBM Corporation 2015

Figure 14-18. Transaction pipes: Tpipes

WM3021.1

Notes:

Transaction pipes represent IMS control blocks. The IBM MQ IMS bridge creates up to two TPIPES per queue, one tpipe for commit mode 0, or COMMIT_THEN_SEND, one tpipe for commit mode 1, or SEND_THEN_COMMIT.

When passing a tpipe name to an IMS transaction that does an ALTPCB CHNG call, care must be taken to adhere to valid tpipe naming conventions. Invalid named Tpipes can lead to IMS problems that are difficult to resolve. The tpipe substitution in a CHNG call would happen in the IMS transaction, however, if the transaction is coded in such a way that it takes the tpipe name from a bridge inbound IBM MQ message, then IBM MQ application needs to provide a valid tpipe name. The situation that is described is uncommon, but worth mentioning given the problem that the invalid tpipe might cause in IMS.

Attribute summary

COMMIT MODE	IMS transaction		Persistence	
	IOPCB	ALTPCB	YES	NO
COMMIT MODE 0 COMMIT_THEN_SEND	*CSQ00000 tpipe	USER NAMED tpipe	OK	
COMMIT MODE 1 SEND_THEN_COMMIT	*CSQ80000 tpipe	DFS2082 IMS ERROR	OK	OK

*Tpipe prefix as specified in the OTMACON macro



Commit mode 0 – non-persistent combination used to be an error.
Check documentation for the IMS version used to confirm recent, version-dependent

© Copyright IBM Corporation 2015

Figure 14-19. Attribute summary

WM3021.1

Notes:

This slide presents a summary of compatible combinations of IBM MQ persistence and the commit mode that is sent in the IIH header.



Important

The error situation resulting from a commit mode 0 in a non-persistent message is changed in later versions of IMS. Check the IBM MQ and IMS documentation to confirm behavior changes.

Tpipes

```
R 66,/DIS TMEMBER VC5 TPIPE ALL.
IEE600I REPLY TO 66 IS;/DIS TMEMBER VC5 TPIPE ALL.
DFS000I      MEMBER/TPIPE      ENQCT      DEQCT      QCT STATUS SYS3
DFS000I      VC5                      SYS3
1 DFS000I    -MICKEYMS          1          1          0      SYS3
2 DFS000I    -CSQ8000E          0          0          0      SYS3
3 DFS000I    -JUSTIN01          1          1          0      SYS3
4 DFS000I    -CSQ0000E          1          1          0  SYN  SYS3
5 DFS000I    -CSQ0000A          4          4          0  SYN  SYS3
6 DFS000I    -CSQ8000A          0          0          0      SYS3
DFS000I      *00041/135759*      SYS3
```

© Copyright IBM Corporation 2015

Figure 14-20. Tpipes

WM3021.1

Notes:

This slide shows how tpipes display according to commit mode and IMS PCB.

- Tpipes with names starting with “CSQ8” are commit mode 1. They show zeros as they do not get enqueued.
- Tpipes with names starting with “CSQ0” are commit mode 0. They show enqueue and dequeue counts.
- There are two other tpipes named MICKEYMS and JUSTIN01. These tpipes were created by an IMS transaction that issued an ISRT with alternative PCB (ALTPCB), and the value that is passed in IMS call resulted in the tpipe name.

IMS exits

- **DFSYPRX0** required only if:
 - Going from non-OTMA to OTMA: R15 = 4
 - Going from OTMA to non-OTMA: R15 = 8
- **DFSYDRU0:**
 - Used to build MQMD structure
 - Can be used to synchronize tpipe after CHNG



Before considering a DFSYDRU0 exit, review the asynchronous callout enhancements for IBM MQ in IMS V1.3.

© Copyright IBM Corporation 2015

Figure 14-21. IMS exits

WM3021.1

Notes:

IMS provides some exit points for IMS OTMA, some are shown on this slide.

The DFSYPRX0 exit switches the origin of an inbound transaction to respond to a different resource, for instance, an OTMA inbound transaction can send its reply to APPC by setting REG15 to 8 and providing the additional details.

DFSYDRU0 is of particular interest. The access point for this exit from an IMS transaction is the CHNG call. You look at sample DSFYDRU0 code in the next slide.

OTMA exits

- An IMS exit might alter the ReplyTo queue that is set by the application

```

CLC      8(8,R2),=CL8'MVSTP001'
        BE LABL01 ... ...
LABL01   DS 0H
        L R7,96(,R2) PARMS ADDR
        MVI 16(R7),X'80' SYN TP FLAG
        MVC MQMD REPLYTOQ,=CL48'APPLICATION.REPLYTOQ.01'
        MVC MQMD_REPLYTOQMGR,=CL48'MVS1'
        MVC MQMD_FORMAT,=CL8'MQIMSVS'
        MVC MQMD_USERIDENTIFIER,=CL12'ANYUSER'
        MVC 324(8,R6),=CL8'MQIMSVS'
        LA R0,MQPER_PERSISTENT
        ST R0,MQMD_PERSISTENCE ... ...

```



Code is for illustrative purposes only.

© Copyright IBM Corporation 2015

Figure 14-22. OTMA exits

WM3021.1

Notes:

An IBM MQ application sending a message to OTMA expects to receive a response, or perhaps a DFSxxx feedback message from IMS in the ReplyToQueue specified when sending the request message to IMS.

What happens if there are no errors, but the reply never arrives?

A DFSYDRU0 exit has full access to the MQMD, and is possible that after the message gets to IMS, the exit might change the ReplyTo destination to a different queue manager and queue. Any field in the MQMD might be changed, including the user ID.

When troubleshooting IBM MQ-OTMA applications for lost messages, ask the IMS team if they use DFSYDRU0 in the IMS region as a DFSYDRU0 exit can overwrite the ReplyToQueue set by the application.

IBM MQ IMS OTMA bridge security

- IBM MQ IMS bridge messages contain the following security information:
 - Either IBM MQ has CONTROL or ALTER access to the **IMSXCF.xcfgname.imsxcfname**, or a UTOKEN
 - If the MQIIH structure is present, the MQIIH designated SecurityScope (this setting is ignored if the **/SECURE OTMA PROFILE** is not used)
 - UserID is passed in the MQMD structure

/SECURE OTMA settings:

```
/SECURE OTMA NONE  
/SECURE OTMA CHECK  
/SECURE OTMA FULL  
/SECURE OTMA PROFILE
```

© Copyright IBM Corporation 2015

Figure 14-23. IBM MQ IMS OTMA bridge security

WM3021.1

Notes:

The IBM MQ-IMS OTMA bridge requires setup of the **IMSXCF.xcfgname.imsxcfname** RACF resource to be able to set full security.

Review the latest IBM MQ and IMS documentation for the possible settings of the SECURE OTMA command.

IBM MQ and IMS transaction expiration support

- For transactions with a specified expiry time, IMS expires at following intervals:
 - When IMS receives input message from XCF
 - Message enqueueing
 - Application GU time (none after GU)
- IBM expiration can be passed to IMS through the **IHH** header **MQIHH_PASS_EXPIRATION**
- Option to pass IMS the remaining IBM MQ expiration time
 - SERVICE attribute of the CSQ6SYSP macro for ZPRM module
CSQ6SYSP SERVICE=0000000001
 - SET SYSTEM command
SET SYSTEM SERVICE(0000000001)



When setting the SERVICE attribute, care must be taken of any other service parameters in the queue manager. If other parameters are in place, a logical “or” should be used to combine them.

© Copyright IBM Corporation 2015

Figure 14-24. IBM MQ and IMS transaction expiration support

WM3021.1

Notes:

IBM MQ IMS bridge uses the IMS Version 11 transaction expiration function to reduce CPU costs by preventing IMS from processing transactions that are no longer necessary to the remote application.

On the IMS side:

An expiration that is associated with a transaction can be specified:

- In the **IBM MQ OTMA** message header
- In IMS TRANSACT definition macro-in IMS DFSINSX0 exit for a new transaction, or
- By the IMS CREATE or UPDATE TRAN commands for the transaction

If the transaction has the expiration time specified, OTMA times out the input transactions from WebSphere MQ in three different points.

- When OTMA first receives the message from XCF
- Message enqueueing
- Application GU time (none after GU)

If the transaction is a Fast Path transaction, a conversational transaction, or a transaction that is routed to a remote IMS system through a Multiple Systems Coupling (MSC) link, OTMA monitors expiration only until the transaction enqueues to IMS.

If IBM MQ IMS Bridge does not specify an expiration time in the OTMA message header, OTMA uses the expiration value that is specified in the IMS transaction as the timeout value for transactions. If transaction expiration is specified in both on the IMS transaction and the IBM MQ OTMA message header, the timeout value that is specified by IBM MQ OTMA message header overrides the timeout value in the transaction. If either IBM MQ OTMA message header or the IMS transaction does not specify an expiration value, the transaction does not time out.

On the IBM MQ side:

- Any IBM MQ message can have an expiration time that is associated with it. The expiration interval is passed by the application to IBM MQ in the MQMD.Expiry field. The time is expressed in 10ths of a second and can be thought of as a 'time to live' for the message. An attempt to MQGET a message past the expiration time results in the message removed from the queue and expiry processing performed. As an IBM MQ message flows between queue managers in an IBM MQ network, the 'time to live' expiration field is decremented, while the message is waiting on a transmission queue for movement between queue managers.
- IBM MQ expiry processing is driven by the Report options that are specified by the application in the MQMD header of the message. Options include: generation of an expiry report, passing remaining expiry interval from a request message to a response message, or just discarding the expired message.

Common bridge problems

- Security related: Various
- Invalid length: Other than data +12
- Invalid attribute combination: CommitMode0 *must* be persistent or error CSQ2001 feedback 00230000 is displayed
- Invalid length: Overlooked bridge formatting
- Overlooked including **ReplyToQMgr**
- Queues **not intended** to be bridge queues defined with a bridge STGCLASS
- ReplyToQ name misspelled
- Faulty DLQ “handler” is discarding messages
- **Invalid Tpipe name** passed to IMS: This situation can be difficult to debug if not noticed
- ACCEPT TRAFFIC not showing on /DIS OTMA
- OTMA exit override of MQMD ReplyTo queue rerouting messages

© Copyright IBM Corporation 2015

Figure 14-25. Common bridge problems

WM3021.1

Notes:

This slide contains some common problems observed when starting to implement OTMA.

As noted earlier, the *Invalid attribute combination* problem is superseded by later versions of IMS, as this combination might now be acceptable. Check the IBM MQ and IMS documentation for the versions of the products in use.

Comparison: IBM MQ-IMS adapter to IBM MQ IMS OTMA bridge

IBM MQ-IMS adapter

- IMS GU obtains the name of the queue, but no data
- MQI calls need to be coded in IMS programs to obtain data
- Trigger monitor transaction required

IBM MQ IMS OTMA bridge

- IMS GU obtains the message data
- No changes to existing IMS transactions (in most cases)
- Coding of new IMS transactions simplified



Usually the best choice

© Copyright IBM Corporation 2015

Figure 14-26. Comparison: IBM MQ-IMS adapter to IBM MQ IMS OTMA bridge

WM3021.1

Notes:

Whenever possible, the IBM MQ IMS OTMA bridge should be used.

The IBM MQ-IMS adapter was the first IBM MQ IMS interface. There are many drawbacks, the biggest being the need to change existing IMS applications to include IBM MQ calls. IBM MQ IMS OTMA is a configuration effort, rather than the need to alter existing applications.

After the IBM MQ IMS OTMA bridge is configured, it is simple to continue to use IMS transactions unchanged with new IBM MQ applications.



Setting up the IMS adapter

- Why?
 - IMS programs that need to access IBM MQ facilities
 - Starting IMS transactions through triggers on local queues
- What must be done?
 - Define IBM MQ for z/OS as a subsystem to IMS
 - Define IMS Language interface tokens (LIT) to the adapter by using an IBM MQ macro
 - Link IMS programs with the IBM MQ
 - IMS program stub CSQQSTUB

© Copyright IBM Corporation 2015

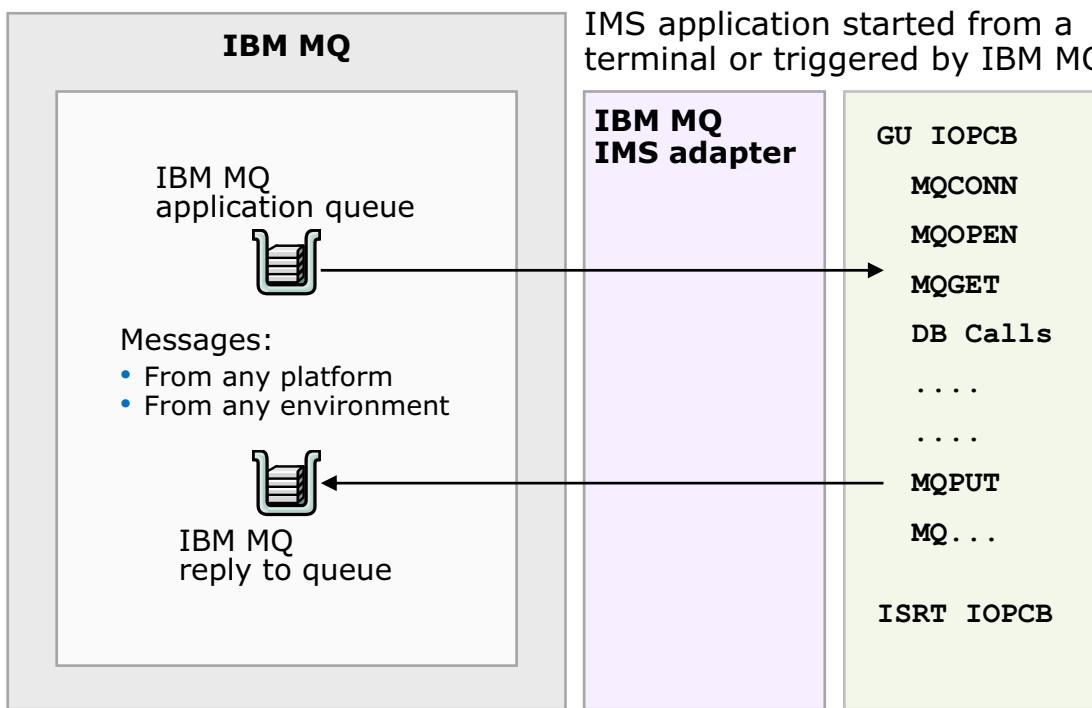
Figure 14-27. Setting up the IMS adapter

WM3021.1

Notes:

- All implementation activities are on the IMS side, both at the system and at the application level
- IBM MQ for z/OS provides the objects that are accessed by IMS instances

MQ-IMS adapter



© Copyright IBM Corporation 2015

Figure 14-28. MQ-IMS adapter

WM3021.1

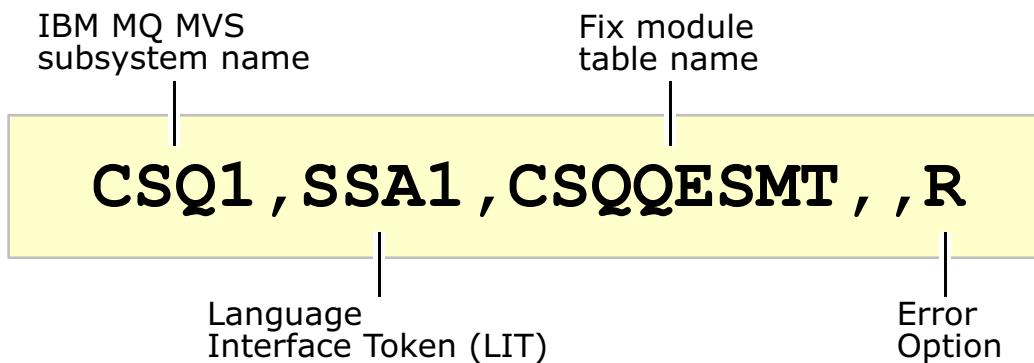
Notes:

The MQ-IMS adapter provides explicit MQI support to IMS programs, which means that IMS applications need to make changes to use **MQOPEN**, **MQPUT**, **MQGET** calls to access IBM MQ messages for in the IMS transaction.



Defining IBM MQ to IMS

- IMS Startup parameter **SSM = ssm**
- **IMS . PROCLIB (&imsid. &ssm)**
 - Defines external subsystems to IMS
 - One entry per queue manager



© Copyright IBM Corporation 2015

Figure 14-29. Defining IBM MQ to IMS

WM3021.1

Notes:

This token is the `Language_interface_token` as specified in the `CSQQDEFV` macro. For more information, see the next page.

The **Error_option** attributes are listed here:

- **R** - Pass return code to application
- **Q** - The application produces an abend with `RC=3051` (this output happens only on an attempt by IMS to access an unavailable IBM WebSphere MQ for z/OS subsystem), but tries to keep the message
- **A** - The application produces an abend with `RC=3047` and discards the message

If you run multiple IMS regions and multiple queue managers, you might determine by specific “SSM PROCLIB members” per region what (if any) queue managers might be accessed by a specific region.

Defining LITs to the adapter

- IBM MQ macro CSQQDEFV
- Tells adapter code which LIT to use for which IBM MQ system
- Defines a default queue manager

```
QM1 CSQQDEFX TYPE=DEFAULT,NAME=CSQ1,LIT=SSA1
QM2 CSQQDEFX TYPE=ENTRY,NAME=CSQT,LIT=SSAT
QM3 CSQQDEFX TYPE=ENTRY,NAME=MQ5,LIT=SSQ5
CSQQDEFX TYPE=END
```

© Copyright IBM Corporation 2015

Figure 14-30. Defining LITs to the adapter

WM3021.1

Notes:

- The IMS ESAF interface (External subsystem access facility) uses *language interface tokens* (LITs) for identifying the subsystems that are known and connected to IMS.
- These LITs must be defined and allocated to the queue manager names through macro structure CSQQDEFV.
- The macro is contained in **thlq.SCSQMAC**. A sample structure is in **thlq.SCSQPROC(CSQQDEFV)**.
- LIT names are arbitrary and not apparent to applications.
- As for batch applications, one queue manager might be defined as the default. This queue manager is chosen if an application does not specify a queue manager name on **MQCONN**.
- You might create multiple CSQQDEFX structures for different IMS regions to specify different default queue managers.

The load library containing the result CSQQDEFV load module must be placed ahead of **thlq.SCSQAUTH** in the IMS MPP start **STEPLIB**.

IBM MQ for z/OS IMS program stub

- STUB module CSQQSTUB for application programs
 - Contained in **thlq.SCSQLOAD**
 - INCLUDE on program link-edit
- IMS Language interface module DFSLI000 has been supporting IBM MQ and IBM MQ since IMS V4

© Copyright IBM Corporation 2015

Figure 14-31. IBM MQ for z/OS IMS program stub

WM3021.1

Notes:

The IMS program preparation link-edit step looks similar to the JCL display in these notes.

Example is for a C language sample:

```
//LKED      EXEC PPARM='....'
//CSQSTUB  DD DSN=MQS.SCSQLOAD,DISP=SHR
//IMSLIB    DD DSN=IMS.RESLIB,DISP=SHR
//SYSLIB    DD DSN=SYS1.SEDCBASE,DISP=SHR
//          DD DSN=SYS1.SIBMBASE,DISP=SHR
//          DD DSN=SYS1.SISPOLOAD,DISP=SHR  <-- for TSO pgms only
//SYSLMOD   DD DSN=your.load.library,DISP=SHR
//SYSLIN    DD DSN=PLKED.SYSMOD,DISP=(OLD,DELETE)  pre-link output
//          DD *
INCLUDE IMSLIB(DFSLI000)
INCLUDE CSQSTUB(CSQQSTUB)
NAME     pgmname
/*
```

Operating the IMS adapter

- /DISPLAY SUBSYS name|all
 - Displays connection status and thread activity
- /START SUBSYS name
 - Connects the IMS Control region to an IBM MQ for z/OS system
- /STOP SUBSYS name
 - Disconnects IMS from an IBM MQ for z/OS subsystem
- /DISPLAY OASN SUBSYS name
 - Displays outstanding recovery elements
- /CHANGE SUBSYS name
 - Deletes in-doubt unit of recovery from IMS

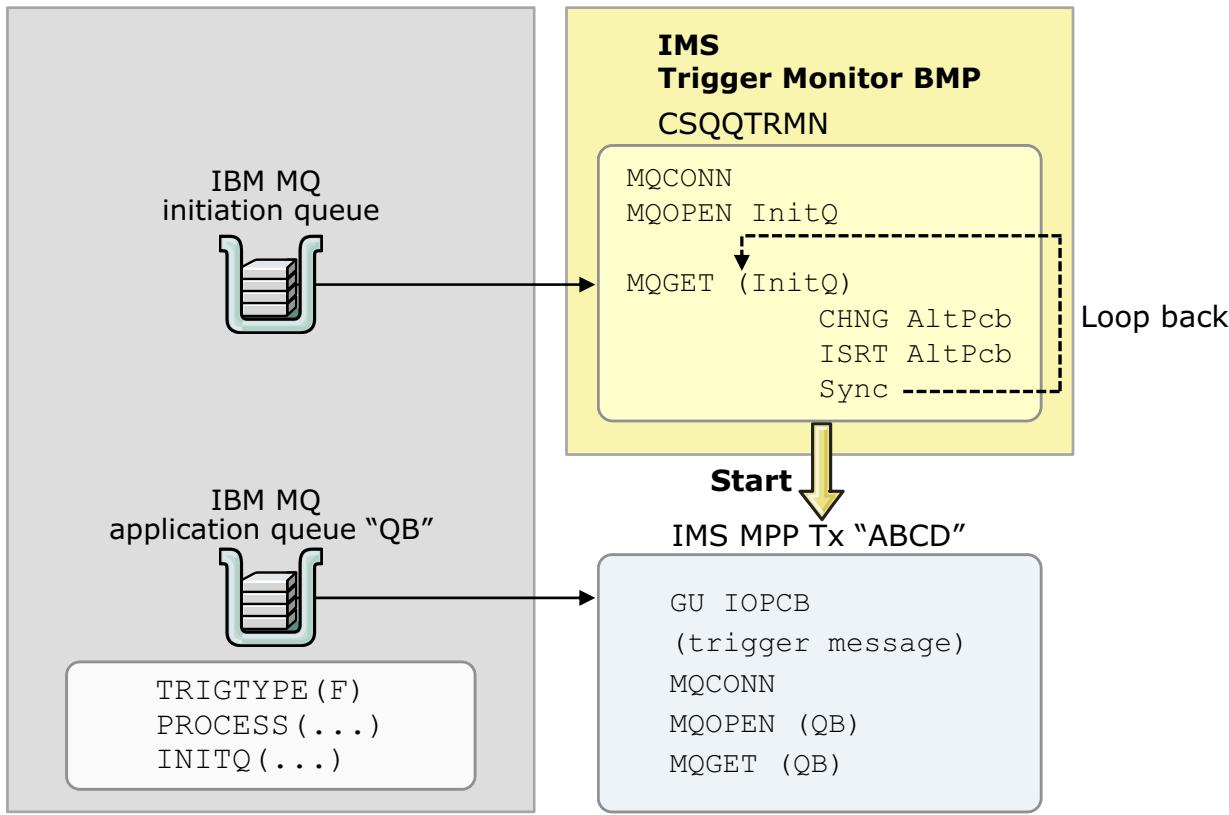
© Copyright IBM Corporation 2015

Figure 14-32. Operating the IMS adapter

WM3021.1

Notes:

IBM MQ supplied IMS trigger monitor



© Copyright IBM Corporation 2015

Figure 14-33. IBM MQ supplied IMS trigger monitor

WM3021.1

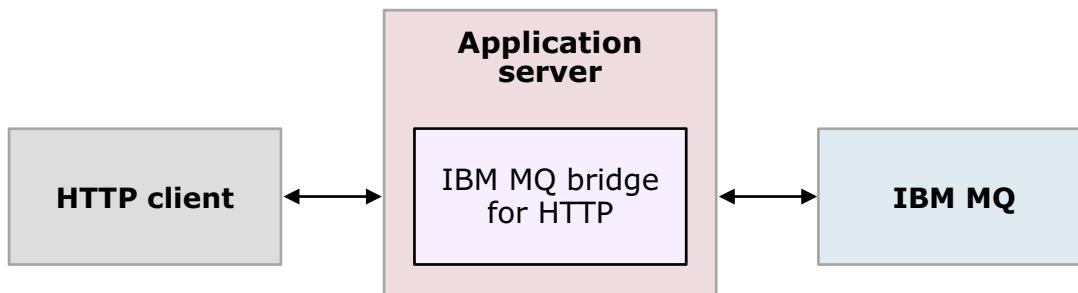
Notes:

- Program and program specification block (PSB) name is CSQTRMN
- Runs as a batch-oriented batch messaging processing (BMP) program
- It is capable of starting IMS online transactions that are specified in IBM WebSphere MQ process objects through the IMS CHANGE PCB / ISRT PCB mechanism
- Stops when IMS goes down or by MVS stop command
- Names of queue manager and the initiation queue to monitor are specified on a start DD statement:

```
//CSQQUT1 DD *
QMGRNAME=CSQ1
INITQUEUENAME=IMSTMN.INITQ
....
```

IBM MQ bridge for HTTP

- Web application that receives HTTP requests, interacts with IBM MQ, and returns HTTP responses
- Allows sending IBM MQ messages over HTTP for environments such as:
 - Environments without enough storage to install an IBM MQ client
 - Environments that support HTTP but not IBM MQ
- Handles three different types of HTTP requests
 - POST: Puts a message to a queue or topic
 - GET: Non-destructive MQGET to use with point-to-point only
 - DELETE: Destructive MQGET to use with point-to-point and publish/subscribe



© Copyright IBM Corporation 2015

Figure 14-34. IBM MQ bridge for HTTP

WM3021.1

Notes:

The IBM MQ bridge for HTTP is a Java Platform, Enterprise Edition servlet, which interfaces with IBM MQ on behalf of a client.

- Receives HTTP requests from one or more clients
- Returns HTTP responses to the clients

The servlet processes three types of HTTP requests:

- POST as shown on slide
- GET excludes publish/subscribe
- DELETE as shown on slide

The IBM MQ bridge for HTTP provides sample applications simulate the amqsput and amqsget samples for a point-to-point environment.

- HTTPPOST - Uses the IBM MQ bridge for HTTP to send HTTP POST requests in a Java application to put messages to a queue, and handles the response.
- Sends HTTPDELETE - Uses the IBM MQ bridge for HTTP to send HTTP DELETE requests in a Java application to get messages from queue, and handles the responses containing message.

For more information about running the samples, see the IBM MQ product documentation.



IBM MQ bridge for HTTP components

- On z/OS, `WMQHTTP.war`, is included as part of the IBM MQ z/OS UNIX System Services Components feature
- `WMQHTTP.war` is installed to
`PathPrefix/usr/lpp/mqm/V8R0M0/http/`
where `PathPrefix` is an optional customer defined prefix

© Copyright IBM Corporation 2015

Figure 14-35. IBM MQ bridge for HTTP components

WM3021.1

Notes:

For more information about the installation details, see the IBM MQ product documentation.

Unit summary

- Describe how to set up an adapter and bridge for CICS and IBM MQ
- Describe how to configure and use the IBM MQ IMS OTMA Bridge
- Describe how to configure the IBM MQ IMS Adapter
- Compare the IBM MQ IMS OTMA Bridge and the IBM MQ IMS adapter
- Summarize the IBM MQ Bridge for HTTP

© Copyright IBM Corporation 2015

Figure 14-36. Unit summary

WM3021.1

Notes:

Checkpoint questions

1. What component is required when setting up the CICS DPL or the CICS 3270 bridge?
 - a. IBM MQ CICS BMS interface
 - b. IBM MQ CICS adapter
 - c. CKTI enabled on CSQ6SYSP
 - d. All of the above
2. Multiple answers: What statements are correct when comparing the IBM MQ IMS adapter and the IBM MQ OTMA bridge?
 - a. The IBM MQ IMS OTMA bridge does not require changes to IMS transactions.
 - b. The IBM MQ adapter requires a trigger monitor transaction.
 - c. The IBM MQ adapter GU call obtains the queue name. IBM MQ calls need to be coded to get the messages.
 - d. All IBM MQ messages to OTMA must contain an IIH header.
3. True or False: The IBM MQ bridge for HTTP handles POST, GET, and DELETE requests.

© Copyright IBM Corporation 2015

Figure 14-37. Checkpoint questions

WM3021.1

Notes:

Write your answers here:

- 1.
- 2.
- 3.

Checkpoint answers

1. What component is required when setting up the CICS DPL or the CICS 3270 bridge? **Answer: b, the IBM MQ CICS adapter**
 - a. IBM MQ CICS BMS interface
 - b. IBM MQ CICS adapter
 - c. CKTI enabled on CSQ6SYSP
 - d. All of the above
2. Multiple answers: What statements are correct when comparing the IBM MQ IMS adapter and the IBM MQ OTMA bridge.
Answers: a, b, and c. The IBM MQ IMS OTMA bridge handles messages without an IIH header.
 - a. The IBM MQ IMS OTMA bridge does not require changes to IMS transactions.
 - b. The IBM MQ adapter requires a trigger monitor transaction.
 - c. The IBM MQ adapter GU call obtains the queue name. IBM MQ calls need to be coded to get the messages.
 - d. All IBM MQ messages to OTMA must contain an IIH header.
3. True or False: The IBM MQ bridge for HTTP handles POST, GET, and DELETE requests. **Answer: True**

© Copyright IBM Corporation 2015

Figure 14-38. Checkpoint answers

WM3021.1

Notes:

Unit 15. Course summary

What this unit is about

This unit summarizes the course and provides information for future study.

What you should be able to do

After completing this unit, you should be able to:

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

Unit objectives

- Explain how the course met its learning objectives
- Access the IBM Training website
- Identify other IBM Training courses that are related to this topic
- Locate appropriate resources for further study

© Copyright IBM Corporation 2015

Figure 15-1. Unit objectives

WM3021.1

Notes:

Course learning objectives

After completing this course, you should be able to:

- Describe message-oriented middleware and the capabilities it must provide
- Identify the key components of IBM MQ for z/OS
- Summarize the responsibilities of the IBM MQ administrator
- Configure IBM MQ V8 for z/OS
- Enable IBM MQ for z/OS eight-byte RBA and buffers above 2 GB
- Demonstrate how to create and change queues and place and retrieve messages from a queue
- Define and demonstrate how to set up and work with distributed queuing

© Copyright IBM Corporation 2015

Figure 15-2. Course learning objectives (1 of 2)

WM3021.1

Notes:

Course learning objectives

After completing this course, you should be able to:

- Differentiate between an IBM MQ queue manager and an IBM MQ client
- Describe and demonstrate how to set up an IBM MQ cluster
- Contrast point-to-point and publish/subscribe messaging styles
- Describe shared queues and queue sharing groups
- Summarize IBM MQ for z/OS recovery and restart activities
- Demonstrate how to use IBM MQ events for monitoring
- Summarize performance considerations

© Copyright IBM Corporation 2015

Figure 15-3. Course learning objectives (2 of 2)

WM3021.1

Notes:

Course learning objectives

After completing this course, you should be able to:

- Describe security considerations for IBM MQ for z/OS
- Describe and implement connection authentication and channel authentication
- Identify correct problem determination techniques for IBM MQ for z/OS
- Summarize basic use and configuration of IBM MQ Managed File Transfer
- Describe IBM MQ support for CICS and IMS interfaces

© Copyright IBM Corporation 2015

Figure 15-4. To learn more on the subject (1 of 2)

WM3021.1

Notes:

To learn more on the subject (1 of 2)

- IBM Training website:
www.ibm.com/training
- IBM support pack *MP16: Capacity Planning and Tuning for WebSphere MQ z/OS*:
www.ibm.com/support/docview.wss?uid=swg24007421
- IBM support pack *MP1J: WebSphere MQ for z/OS V8.0 Performance report*:
www.ibm.com/support/docview.wss?rs=171&uid=swg24038347
- IBM support pack *MP1H: WebSphere MQ for z/OS V7.1.0 Performance report*:
www.ibm.com/support/docview.wss?uid=swg24031663
- IBM support pack *MP1B: WebSphere MQ Interpreting accounting and statistics data, and other utilities*:
www.ibm.com/support/docview.wss?uid=swg24005907

© Copyright IBM Corporation 2015

Figure 15-5. To learn more on the subject (2 of 2)

WM3021.1

Notes:



To learn more on the subject (2 of 2)

- Redbooks publication **IBM MQ V8 Features and Enhancements**:
www.redbooks.ibm.com/abstracts/sg248218.html?Open
- Additional information by an IBM MQ product developer on connection authentication:
<https://www.ibm.com/developerworks/community/blogs/messaging/tags/connauth?sortby=0&maxresults=30&lang=en>

© Copyright IBM Corporation 2015

Figure 15-6. Unit summary

WM3021.1

Notes:

Appendix A. List of abbreviations

AMS	(IBM MQ) Advanced Message Security
APAR	authorized program analysis report
APF	Authorized Program Facility
API	application programming interface
APPCC	Advanced Program-to-Program Communication
BMP	batch message processing program
BMS	basic mapping support
BRXA	bridge exit area
BSDS	bootstrap data set
CA	certificate authority
CDDT	client channel definition table
CDT	class descriptor table
CF	coupling facility
CFRM	coupling facility resource management
CHIN	channel initiator
CICS	Customer Information Control System
COBOL	Common Business Oriented Language
CPF	command prefix
CPU	central processing unit
CRCR	conditional restart control record
CSD	CICS system definition data set
DASD	Direct Access Storage Device
DBMS	database management systems
DD	data definition (statement)
DLH	dead-letter header
DLL	dynamic link library
DLQ	dead-letter queue
DN	distinguished name
DNS	Domain Name Server
DPL	distributed program link
DQM	data quality management

ERM	external resource manager
ESAF	external subsystem access facility
ESM	external security manager
FFDC	first-failure data capture
FIFO	first-in first-out
FMID	function modification identifier
FTP	File Transfer Protocol
FTPS	File Transfer Protocol Secure
GDG	generation data group
GP	generic port
GTF	generalized trace facility
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IBM	International Business Machines Corporation
ICSF	Integrated Cryptographic Service Facility
IGQ	intra-group queuing
IIH	installation-initiating host
IMS	Information Management System
I/O	input/output
IP	Internet Protocol
IPCS	Interactive Problem Control System
IPL	Initial Program Load
ISPF	Interactive System Productivity Facility
IUCV	inter-user communication vehicle
IVP	installation verification procedure
JCL	job control language
JDBC	Java Database Connectivity
JES	job entry subsystem
JMS	Java Message Service
KB	kilobyte
LDAP	Lightweight Directory Access Protocol
LIT	language interface token
LP	local port

LPA	link pack area
LSN	log sequence number
LU	logical unit
LUWID	logical unit of work identifier
MCA	message channel agent
MFT	Managed File Transfer
MPP	message processing program
MQ	Message Queue
MQADV	(IBM) MQ Advanced
MQAMS	(IBM) MQ Advanced Message Security
MQI	Message Queue Interface
MQMD	MQ Message Descriptor
QMFT	(IBM) MQ Managed File Transfer
MQSC	MQ script command
MSC	Multiple Systems Coupling
MSN	message sequence number
MVS	Multiple Virtual Storage
NID	network identifier
OAM	object authority manager
OS	operating system
OTE	on-target earning
OTMA	Open Transaction Manager Access
PCB	program communication block
PCF	programmable command format
PDF	Portable Document Format
PDS	partitioned data set
PMO	put message options
PSB	program specification block
PSW	program status word
PSID	page set identification
QMID	queue manager ID
QSG	queue-sharing group
RACF	Resource Access Control Facility

RAID	Redundant Array of Independent Disks
RBA	relative byte address
RMF	Resource Monitoring Facility
RMI	resource manager interface
RRMS	Restart Recovery Management Services
RRS	Resource Recovery Services
SCM	storage class memory
SDK	software development kit
SDSF	System Display and Search Facility
SDWA	System Diagnostic Work Area
SIT	system initialization table
SMDS	shared message data set
SMF	System Management Facilities
SMP/E	System Modification Program / Extended
SMS	Storage Management Subsystem
SP	shared port
SQL	Structured Query Language
SSL	Secure Sockets Layer
STC	standard telegraphic code
SVC	supervisor call
TB	terabyte
TCB	task control block
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Level Security
TS	temporary storage
TSO	Time Sharing Option
UACC	universal access authority
UNIX	Uniplexed Information and Computing System
UOW	unit of work
UR	Unit of recovery
URID	unit of recovery ID
USS	UNIX System Services

VSAM	Virtual Storage Access Method
VTAM	Virtual Telecommunication Access Method
VUE	Value Unit Edition
WTO	write to operator
XA	extended architecture
XCF	cross-system coupling facility
zEDC	zEnterprise Data Compression
z/OS	zSeries operating system

Appendix B. Resource guide

Completing this WebSphere Education course is a great first step in building your WebSphere, CICS, and SOA skills. Beyond this course, IBM offers several resources to keep your WebSphere skills on the cutting edge. Resources available to you range from product documentation to support websites and social media websites.

Training

- **IBM Training website**
 - Bookmark the IBM Training website for easy access to the full listing of IBM training curricula. The website also features training paths to help you select your next course and available certifications.
 - For more information, see: <http://www.ibm.com/training>
- **IBM Training News**
 - Review or subscribe to updates from IBM and its training partners.
 - For more information, see: <http://bit.ly/IBMTrainEN>
- **IBM Certification**
 - You can demonstrate to your employer or clients your new WebSphere, CICS, or SOA mastery through achieving IBM Professional Certification. WebSphere certifications are available for developers, administrators, and business analysts.
 - For more information, see: <http://www.ibm.com/certify>
- **Training paths**
 - Find your next course easily with IBM training paths. Training paths provide a visual flow-chart style representation of training for many WebSphere products and roles, including developers and administrators.
 - For more information, see: <http://www.ibm.com/services/learning/sites.wss/us/en?pageType=page&c=a0003096>

Social media links

You can keep in sync with WebSphere Education, including new courses and certifications, course previews, and special offers, by visiting any of the following social media websites.

- **Twitter**

- Receive short and concise updates from WebSphere Education a few times each week.
- Follow WebSphere Education at: twitter.com/websphere_edu
- **Facebook:**
 - Become a fan of IBM Training on Facebook to keep in sync with the latest news and career trends, and to post questions or comments.
 - Find IBM Training at: facebook.com/ibmtraining
- **YouTube:**
 - Visit the IBM Training YouTube channel to learn about IBM training programs and courses.
 - Find IBM Training at: youtube.com/IBMTutorial

Support

- **WebSphere Support portal**
 - The WebSphere Support website provides access to a portfolio of support tools. From the WebSphere Support website, you can access several downloads, including troubleshooting utilities, product updates, drivers, and Authorized Program Analysis Reports (APARs). To collaboratively solve issues, the support website is a clearing house of links to online WebSphere communities and forums. The IBM support website is now customizable so you can add and delete portlets to the information most important to the WebSphere products you work with.
 - For more information, see:
<http://www.ibm.com/software/websphere/support>
- **IBM Support Assistant**
 - The IBM Support Assistant is a local serviceability workbench that makes it easier and faster for you to resolve software product issues. It includes a desktop search component that searches multiple IBM and non-IBM locations concurrently and returns the results in a single window, all within IBM Support Assistant.
 - IBM Support Assistant includes a built-in capability to submit service requests; it automatically collects key problem information and transmits it directly to your IBM support representative.
 - For more information, see: <http://www.ibm.com/software/support/isa>
- **WebSphere Education Assistant**
 - IBM Education Assistant is a collection of multimedia modules that are designed to help you gain a basic understanding of IBM software products

and use them more effectively. The presentations, demonstrations, and tutorials that are part of the IBM Education Assistant are an ideal refresher for what you learned in your WebSphere Education course.

- For more information, see:

<http://www.ibm.com/software/info/education/assistant/>

WebSphere documentation and tips

- **IBM Redbooks**

- The IBM International Technical Support Organization develops and publishes IBM Redbooks publications. IBM Redbooks are downloadable PDF files that describe installation and implementation experiences, typical solution scenarios, and step-by-step “how-to” guidelines for many WebSphere products. Often, Redbooks include sample code and other support materials available as downloads from the site.
- For more information, see: <http://www.ibm.com/redbooks>

- **IBM documentation and libraries**

- Information centers and product libraries provide an online interface for finding technical information on a particular product, offering, or product solution. The information centers and libraries include various types of documentation, including white papers, podcasts, webcasts, release notes, evaluation guides, and other resources to help you plan, install, configure, use, tune, monitor, troubleshoot, and maintain WebSphere products. The WebSphere information center and library are located conveniently in the left navigation on WebSphere product web pages.

- **developerWorks**

- IBM developerWorks is the web-based professional network and technical resource for millions of developers, IT professionals, and students worldwide. IBM developerWorks provides an extensive, easy-to-search technical library to help you get up to speed on the most critical technologies that affect your profession. Among its many resources, developerWorks includes how-to articles, tutorials, skill kits, trial code, demonstrations, and podcasts. In addition to the WebSphere zone, developerWorks also includes content areas for Java, SOA, web services, and XML.
- For more information, see: <http://www.ibm.com/developerworks>

WebSphere Services

- IBM Software Services for WebSphere are a team of highly skilled consultants with broad architectural knowledge, deep technical skills, expertise on suggested practices, and close ties with IBM research and development labs. The WebSphere Services team offers skills transfer, implementation, migration, architecture, and design services, plus customized workshops. Through a worldwide network of services specialists, IBM Software Service for WebSphere makes it easy for you to design, build, test, and deploy solutions, helping you to become an on-demand business.
- For more information, see:
<http://www.ibm.com/developerworks/websphere/services/>

IBM
®