Shared Q using Shared Message Data Sets

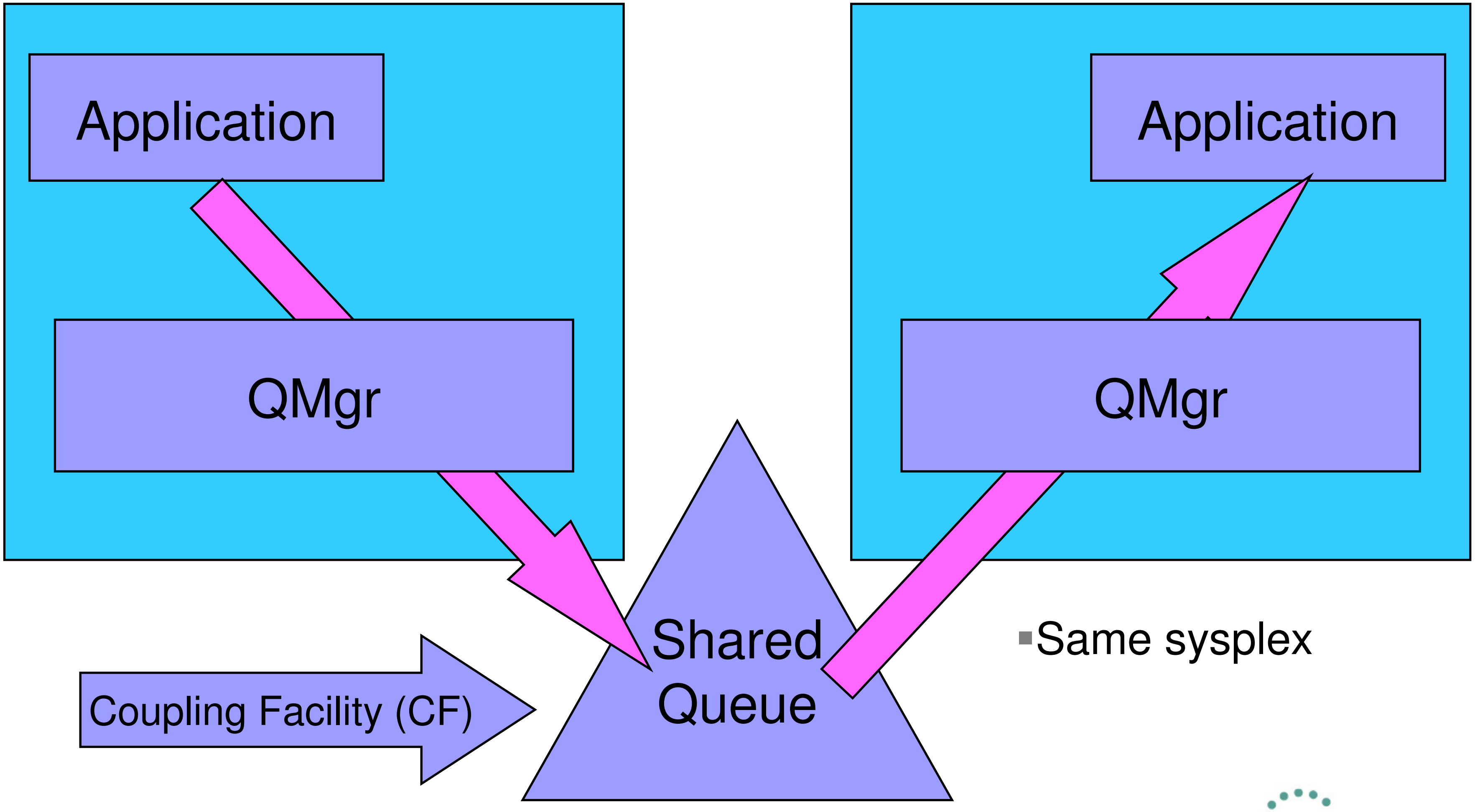
Session 11514

Paul S Dennis
dennisps@uk.ibm.com

Agenda

- What are Shared Queues
- Large messages with DB2
- SMDS
- Structures – Persistence and recovery
- Client Channels

Shared Queues



- Same sysplex

Shared Queues

N O T E S

Chart shows an application put to a shared target queue -- that is, the target queue is local to more than one queue manager. This put does not use the mover:

1. Application puts to shared target queue
2. Remote application can now get the message.

The remote application can put a message to the reply-to queue using the same method.

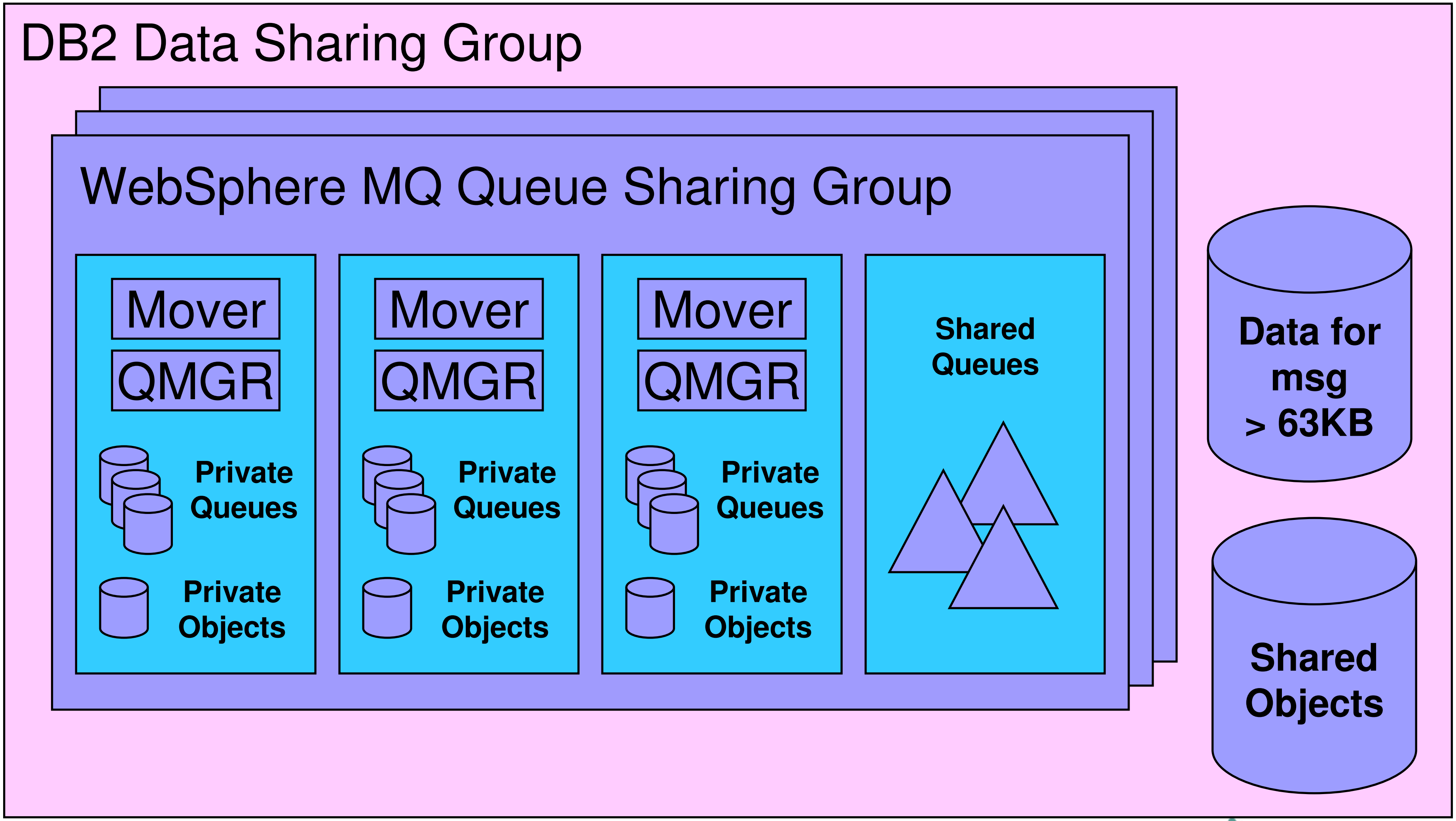
Note that applications connected to any queue manager with access to the shared queue can get the message. To access the same shared queues, queue managers must be:

- In the same z/OS sysplex
- In the same queue-sharing group (QSG) -- we will explain QSGs later.

There are restrictions on shared queues, for example:

- CF capacity is limited (compared to DASD).

Queue Sharing Groups (QSGs)



Queue-Sharing Groups (QSGs)

NOTES

Chart shows how queue managers are organized into queue-sharing groups (QSGs) and the relationship to DB2 data-sharing groups.

A queue-sharing group can contain one or more queue managers:

- Each queue manager has its own private (not shared) queues and object definitions.
- All queue managers in a QSG share the same set of shared queues and shared object definitions
- A queue manager cannot belong to more than one QSG.

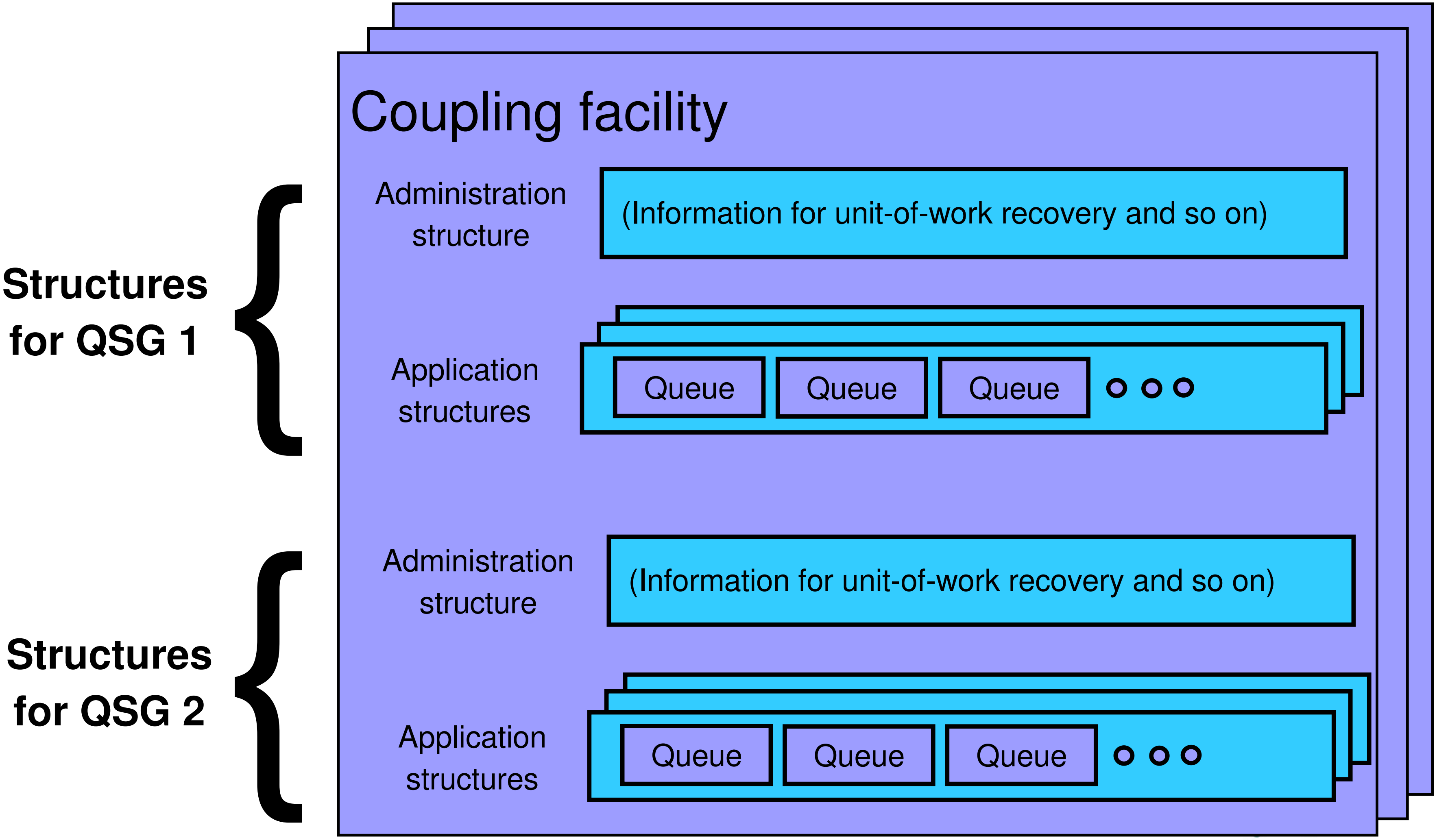
Shared object definitions for a QSG are maintained for WebSphere MQ by DB2. Shared access to these definitions is by DB2 data sharing:

- You must have DB2
- You can have more than one data-sharing group, but all members of one QSG must be members of the same data-sharing group
- Shared object definitions are cached in the queue managers.
- A DB2 outage does not bring down the QSG (but you cannot add or change shared objects if DB2 is down).

You do not have to define any queue-sharing groups if you do not run a sysplex (or if you just don't want to).

If using shared messages > 63KB then a small portion for the message is stored in the CF, and the rest is stored in DB2 or with V7.1 or higher there is the option of using Shared Message Data Sets (SMDS) for storing the rest of the message data.

CF Structures for shared-queues



CF Structures for shared-queues

N O T E S

Chart shows organization of WebSphere MQ data in coupling facility (CF) structures (actually *list* structures).

For clarity the chart shows:

- All structures in one CF -- actually they can be spread arbitrarily over many CFs
- Only WebSphere MQ structures -- actually other subsystems and applications can have structures in the same CF as Websphere MQ.

Each queue-sharing group needs:

- One administration structure -- this is used for information that WebSphere MQ itself needs, for example to manage unit-of-work recovery
- One or more (up to a maximum of 63) application structures -- these are used to hold the shared queues.

Each application structure can hold up to 512 shared queues.

Creating CF structures and shared queues

- Define a structure to z/OS (not to WebSphere MQ) by updating the CFRM policy (see *System Setup Guide*):
  - Structure is known to WebSphere MQ by its 12-character *str-name*.
  - Structure is known to z/OS by the 16-character name formed by:
    - *qsg-name* || *str-name*   (Application structures)
    - *qsg-name* || CSQ_ADMIN   (Administration structure)
- Define a shared queue using the DEFINE QLOCAL command on any queue manager in the QSG**:**
  - DEFINE QLOCAL(*queue-name*) QSGDISP(SHARED) CFSTRUCT(*str-name*)
- z/OS creates the structure when required (first use).
- WebSphere MQ creates the queue when required (first use).

Creating CF structures and shared queues

N O T E S

Chart shows the processes for creating CF list structures for use by WebSphere MQ QSGs and for creating shared queues in these structures.

The z/OS CFRM policy for the sysplex specifies how z/OS should allocate resources for each structure:

- What type of CF (for example, CF must have battery back-up)
- How big to make the structure.

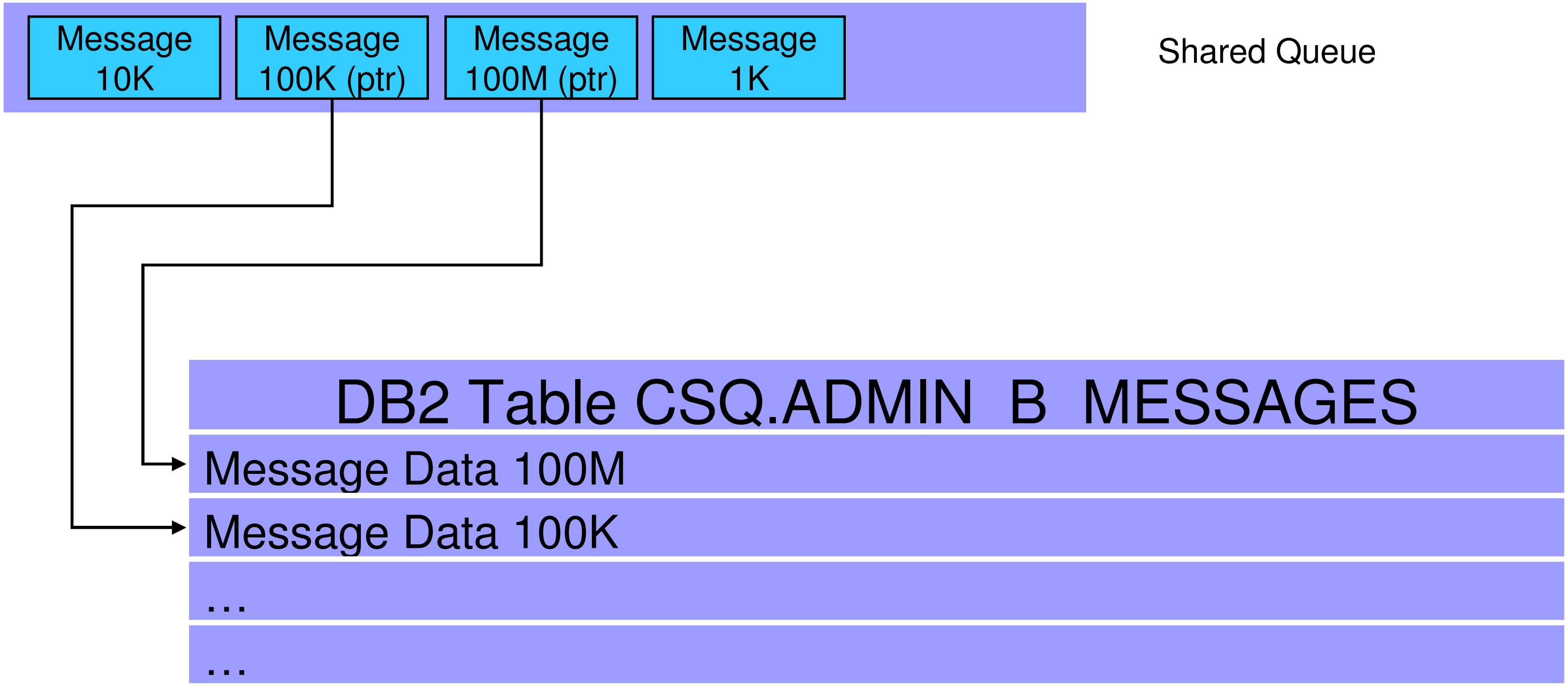
z/OS does not actually allocate any resource for the structure until first use -- in our case, the first time a queue manager connects to the structure:

- At startup for the administration structure
- At first queue open for application structures.

As with private queues, defining the queue to WebSphere MQ does not create the queue. The queue is created when it is first used.

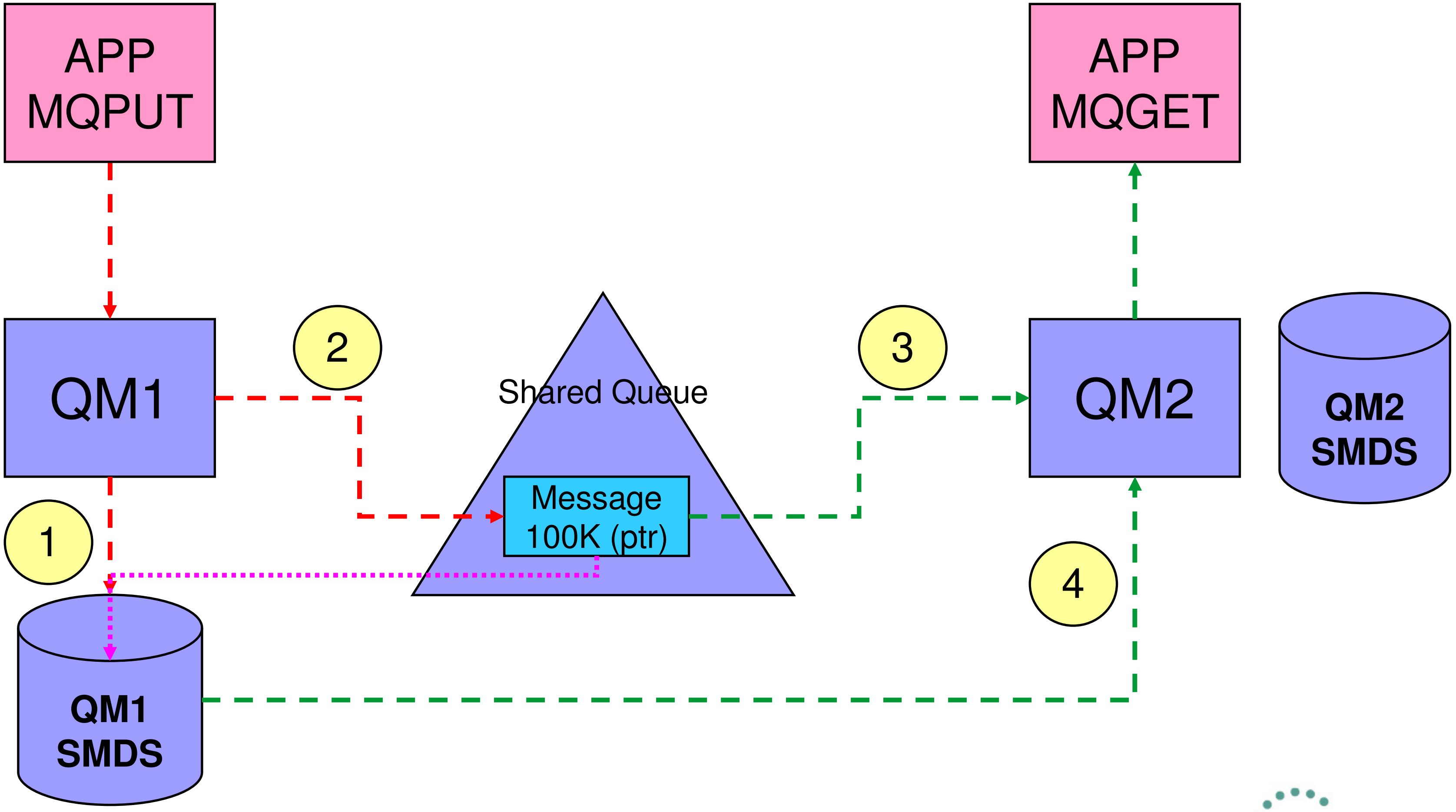
It is best to allocate queues so that (as far as possible) all the queues accessed by any one unit-of-work are in the same structure.

Large Shared Queue Messages (using DB2)



Large Shared Queue Messages (using SMDS)

V7.1



Shared message data set concepts

N O T E S

Offloaded message data for shared messages is stored in data sets.

Each application structure has an associated group of shared message data sets, with one data set per queue manager.

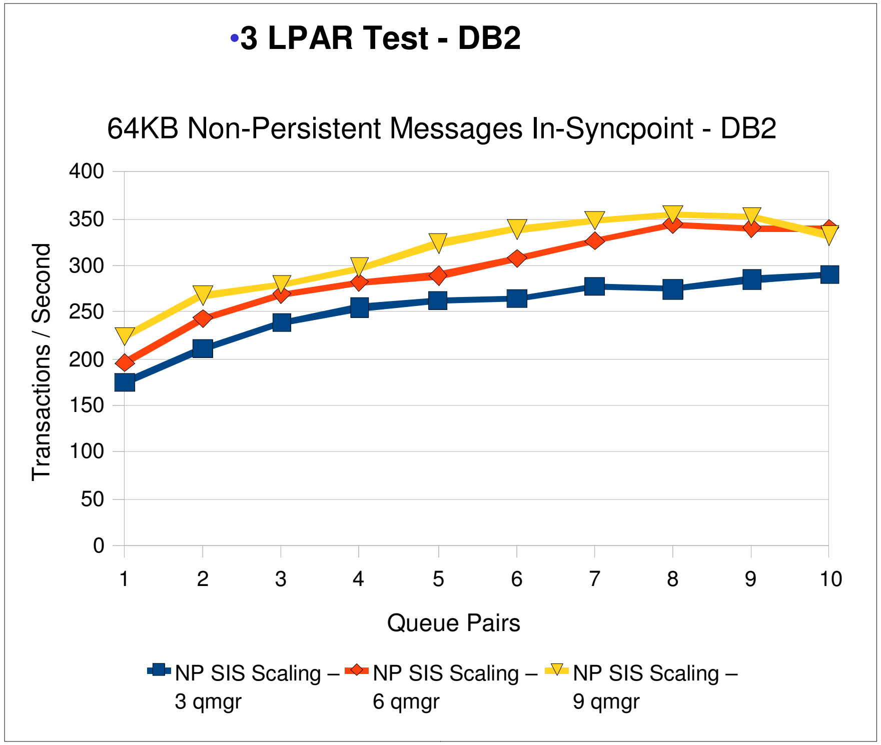
Named using DSGROUP parameter on CFSTRUCT definition.

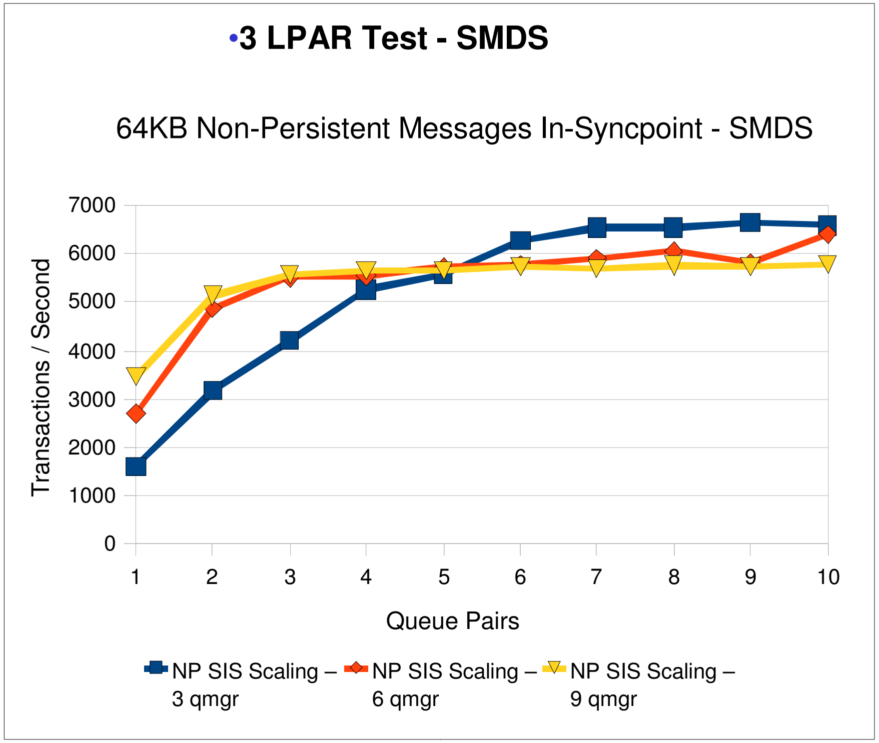
Each queue manager owns a data set for each structure, opened for read/write access, which it uses to write new large messages.

Each queue manager opens the data sets for the other queue managers for read-only access, so it can read their message data.

When a message with offloaded data needs to be deleted, it is passed back to the queue manager which originally wrote it, so that the queue manager can free the data set space when it deletes the message.

SMDS Performance Improvement





- Tests show comparable CPU savings making SMDS a more usable feature for managing your CF storage
- SMDS per CF structure provides better scaling than DB2 BLOB storage

Selecting which messages to offload

V7.1

- Messages too large for CF entry (> 63K bytes) are always offloaded.
- Other messages may be selectively offloaded using offload rules.
  - Each structure has three offload rules, specified on the CFSTRUCT definition.
  - Each rule specifies message size in Kbytes and structure usage threshold, using two parameters:
    - OFFLDnSZ(size) and OFFLDnTH(percentage), where n = 1, 2, 3.
  - Data for new messages exceeding the specified size is offloaded (as for a large message) when structure usage exceeds the specified threshold.
  - Default rules are provided which should be useful in most cases.
  - Rules can be set to dummy values if not required.
- Without offloading data, it is possible to store 1.25M messages of 63KB on a 100GB structure
- When offloading all messages, possible to store approx 140M messages on the same structure, irrespective of message size

Selecting which messages to offload

NOTES

As with previous releases of MQ, the amount of data that can be stored in the CF for a single message is limited to 63KB. This means that if the message is over 63KB in size then it must be “offloaded”. With V7.1 there are two offload methods, already seen, for offloading the data, using either DB2 or SMDS.

In addition to offloading all messages over 63KB, it is possible to specify that messages smaller than this size should also be offloaded. There are three sets of rules that are used to control this, and each set is made up of two parameters, the size of the message and how full the CF structure is when the message is put. These offload rules enable a balancing to be performed between performance and CF capacity. For example, you might use a rule that says that when the CF structure is 70% full then all messages over 32KB will be offloaded, and then another rule that says that when the CF structure is over 80% full, all messages over 4KB will be offloaded. When migrating a structure from CFLEVEL 4 to CFLEVEL 5 (required to use these rules), the defaults will be set to mimic the CFLEVEL 4 behavior. When defining a new structure at CFLEVEL 5, the default rules will be set as follows:

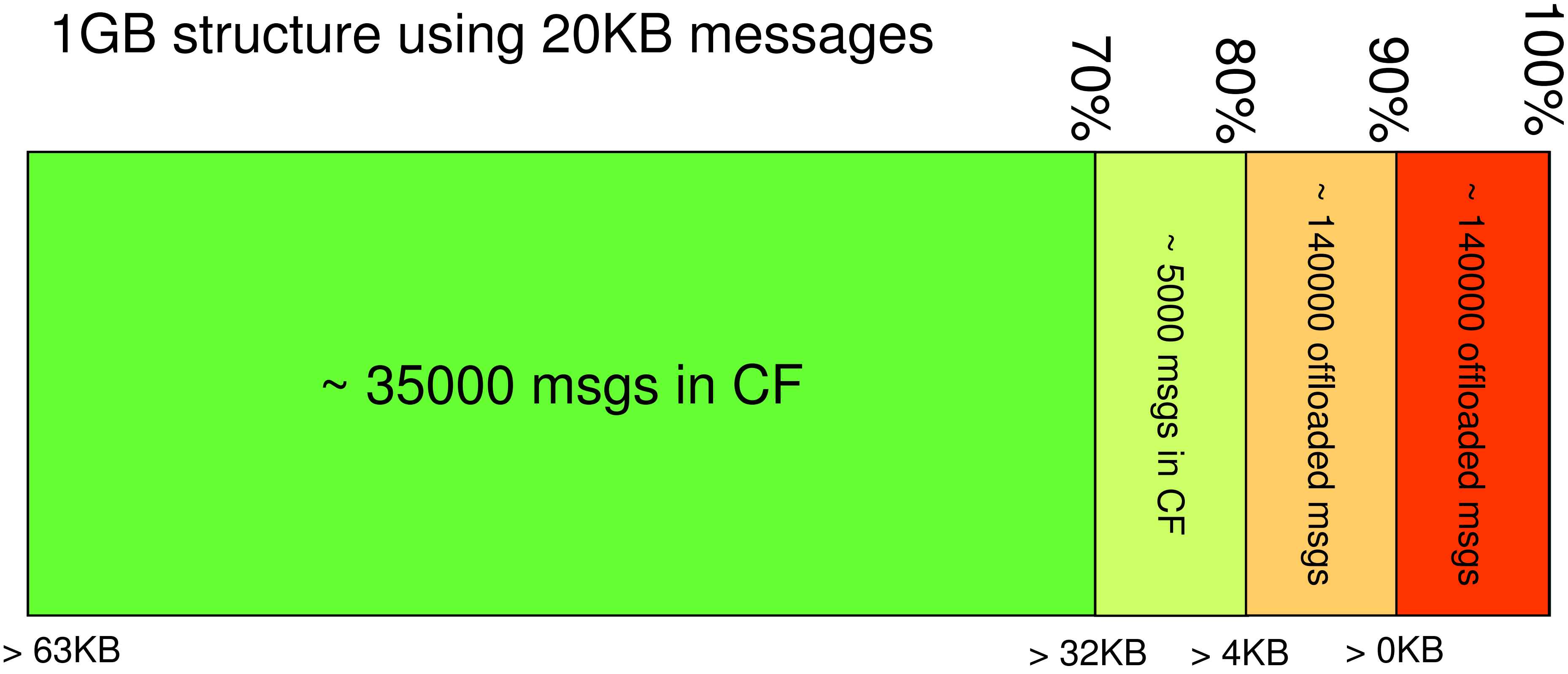
OFFLD1SZ = 32K	OFFLD1TH = 70
OFFLD2SZ = 4K	OFFLD2TH = 80
OFFLD3SZ = 0K	OFFLD3TH = 90

Typical use of offload rules

- The three offload rules have no fixed order but are typically intended to be used as follows:
  - Rule 1 is used to save space for fairly large messages by offloading them, with little performance impact, even when plenty of space left.
    - SMDS defaults: **OFFLD1SZ(32K)**, **OFFLD1TH(70)**
  - Rule 2 is used as an intermediate step between rules 1 and 3, to start saving more space as the structure usage increases, in exchange for a minor performance impact.
    - SMDS defaults: **OFFLD2SZ(4K)**, **OFFLD2TH(80)**
  - Rule 3 is used to maximize the remaining space when the structure is nearly full, by offloading everything possible.
    - SMDS defaults: **OFFLD3SZ(0K)**, **OFFLD3TH(90)**

Storage benefits of offloading

1GB structure using 20KB messages



~ 320000 msgs using offloading vs ~ 50000 without offloading

Creating a shared message data set

- SMDS is defined as a VSAM linear data set using **DEFINE CLUSTER**.
  - Requires **LINEAR** option.
  - Control interval size must be 4096, which is the default for linear.
  - Requires **SHAREOPTIONS(2 3)**, allowing one queue manager to write and other queue managers to read at the same time.
  - If maximum size may need to exceed 4GB, requires SMS data class which has VSAM extended addressability attribute.
  - If automatic expansion is to be supported, requires an appropriate secondary space allocation (although a default of 20% will be used if an expansion attempt fails because of no secondary allocation).
- Can optionally be pre-formatted, for example using CSQJUFMT.
  - Otherwise formatted automatically when first opened.

Creating a shared message data set

- The **DSGROUP** parameter on the **CFSTRUCT** definition specifies the group of data sets associated with the application structure.
  - It is specified as a generic data set name with a single asterisk as the point where the owning queue manager name is to be inserted.
  - It is required when the option **OFFLOAD**(**SMDS**) is specified.
- CSQ4SMDS in SCSQPROC provides JCL to define and format a single dataset

```
DEFINE CLUSTER                                  -
         (NAME(++HLQ++.++QMGR++.++CFSTRUCT++.SMDS)   -
          MEGABYTES(++PRI++ ++SEC++)             -
          LINEAR                                 -
          DATACLAS(EXTENDED)                     -
          SHAREOPTIONS(2 3) )                    -
     DATA                                        -
         (NAME(++HLQ++.++QMGR++.++CFSTRUCT++.SMDS.DATA) )
```

Access to shared message data sets

- Shared message data sets must be on shared direct access storage accessible to all queue managers within the QSG.
- Normal running:
  - Queue manager opens own data set read/write.
    - Requires **UPDATE** access to own data set.
  - Queue manager opens other data sets read-only.
    - Requires **READ** access to all other data sets.
- Media recovery processing:
  - Queue manager performing recovery opens own data set and all other data sets for read/write access.
    - Requires **UPDATE** access to all data sets.

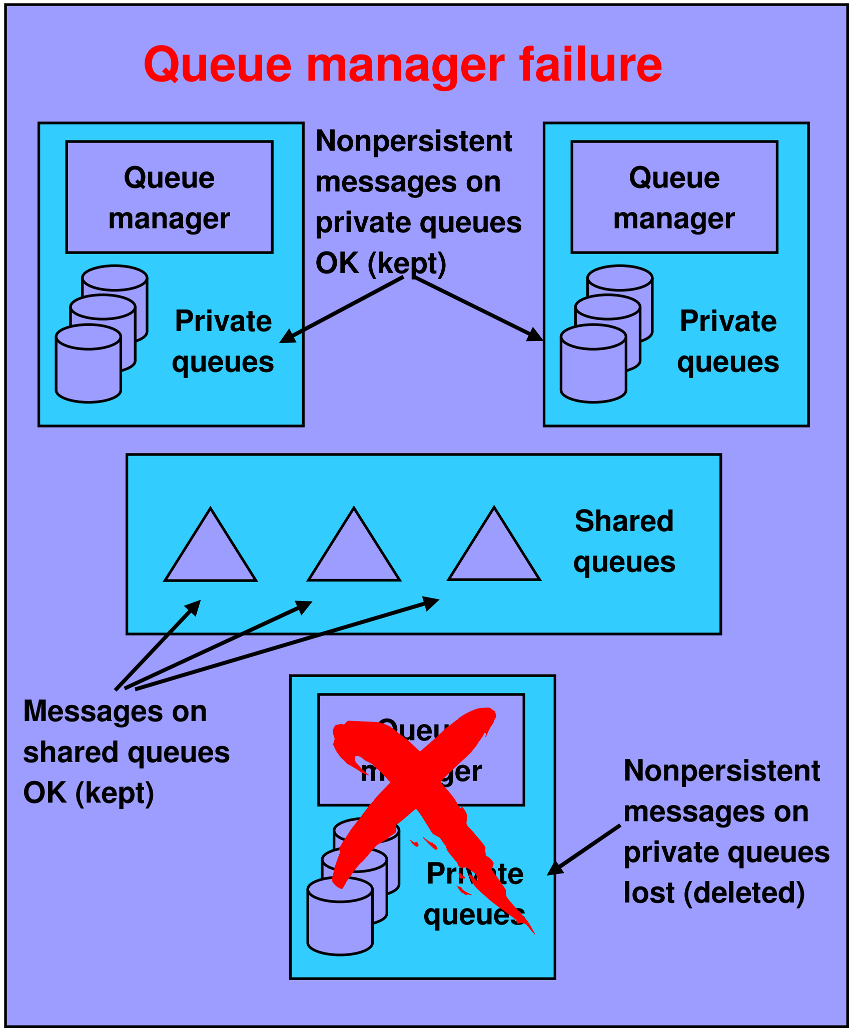
Shared message data set capacity considerations

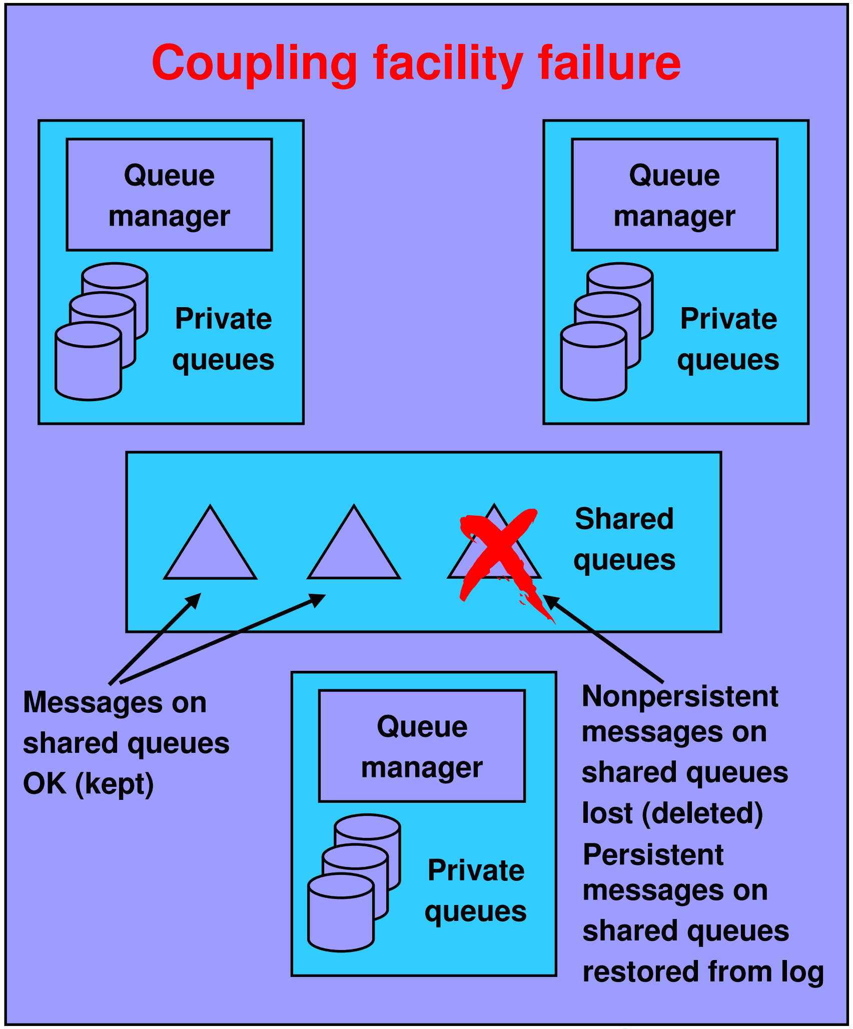
- Each shared message data set only contains data for large messages written via its owning queue manager.
- Message size calculation:
  - Each stored message includes standard headers (usually 352 bytes).
  - Each message is stored as one or more message blocks.
  - Each message block is stored in a range of consecutive 4K pages on the data set, with a very small header (32 bytes).
  - Approximate data set space required per large message, in bytes, is given by size of message plus header rounded up to next 4K.
- Multiply by maximum anticipated backlog of messages written via that queue manager (plus some safety margin) to estimate size needed for data set.

SMDS capacity considerations – expansion

- Data set can be automatically expanded when necessary.
  - Normally set by **DSEXPAND(YES|NO)** option on **CFSTRUCT**, which specifies default option for data set group.
  - Can also be overridden for individual data sets using **DSEXPAND** option on **ALTER SMDS**.
- Expansion attempt is automatically triggered when 90% full.
  - If no secondary allocation was specified, VSAM error message will appear, but queue manager will retry using a default secondary allocation of 20% of the existing size.
  - If expansion fails (not enough space available), queue manager sets **DSEXPAND(NO)** to prevent further attempts. Operator can use **ALTER SMDS** to set **DSEXPAND(YES)** again after problem is fixed.
  - If maximum extents are reached, data set cannot be expanded any further. (It could however be marked unavailable then copied to a larger data set which is then renamed back to the original name).

Failure and persistence


Queue manager failure
Queue manager
Private queues
Nonpersistent messages on private queues OK (kept)
Queue manager
Private queues
Shared queues
Messages on shared queues OK (kept)
Queue manager
Private queues
Nonpersistent messages on private queues lost (deleted)


Coupling facility failure
Queue manager
Private queues
Queue manager
Private queues
Shared queues
Messages on shared queues OK (kept)
Queue manager
Private queues
Nonpersistent messages on shared queues lost (deleted)
Persistent messages on shared queues restored from log

Failure and persistence

N O T E S

Chart shows implications of failures in a queue-sharing group.

Left side of chart shows queue manager failure. If one or more queue managers in a queue-sharing group fail, or are stopped normally:

- Nonpersistent messages on queues private to the failing queue manager or managers are lost -- in fact they are deleted when a queue manager restarts
- Messages on shared queues are not lost, they are kept -- even if *all* queue managers in the queue-sharing group fail.

Right side of chart shows coupling facility structure failure (for simplicity the chart shows an entire CF failing). If one or more CF structures fail:

- Messages on queues in other CF structures are not lost
- Nonpersistent messages on queues in failing CF structures are lost
- Persistent messages on queues in failing CF structures must be restored from backup and log information on the logs
- Restoring queue manager accesses logs of all queue managers in the QSG.

If the administration structure fails, all the queue managers in the QSG fail.

Admin Structure Recovery

V7.0.1

- Prior to V7.0.1 each queue manager would rebuild own admin structure entries
  - Particularly an issue in a DR situation.
    - Need to start all queue managers to rebuild admin structure
    - Once recovered, application structures could be recovered
- At V7.0.1 active queue managers notice if other queue managers don’t have entries, and initiate rebuild on their behalf

Admin Structure Recovery

V7.0.1

NOTES

If the Admin Structure was lost for some reason (DR situation, loss of power to the CF etc), then prior to V7.0.1 each queue manager had to rebuild its own Admin Structure entries. As the admin structure needs to be complete for application structure recovery to take place, it was necessary in a DR situation to start up all the queue managers in a QSG before application structure recover could take place.

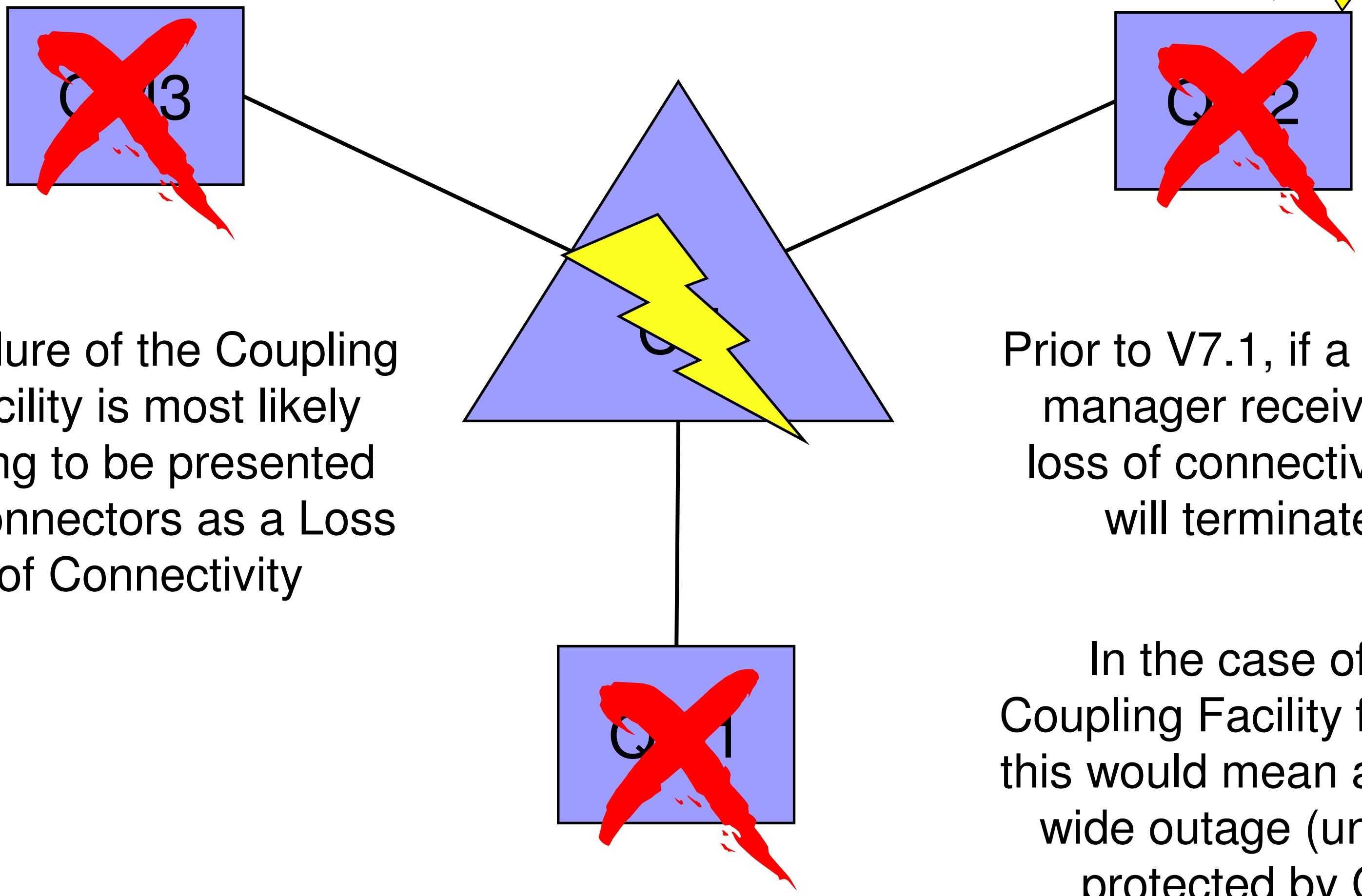
In V7.0.1 an enhancement has been made to admin structure recovery so that a single queue manager is able to recover the admin structure entries for all the other queue managers in the QSG. If a V7.0.1 (or higher) queue manager notices that the admin structure entries are missing for another queue manager then it will attempt to recover them on behalf of the other queue manager. It can only do this if the other queue manager is not running at the time. In a DR situation this means that it is only necessary to start a single queue manager at V7.0.1 (or higher) before being able to recover the application structures.

A V7.0.1 queue manager can recover the entries on behalf of any version of queue manager, so you don't need to have all queue managers in the QSG to be running at V7.0.1 before this functionality will take place.

CF Loss of Connectivity Tolerance

V7.1

Pre V7.1 Queue Managers



A failure of the Coupling Facility is most likely going to be presented to connectors as a Loss of Connectivity

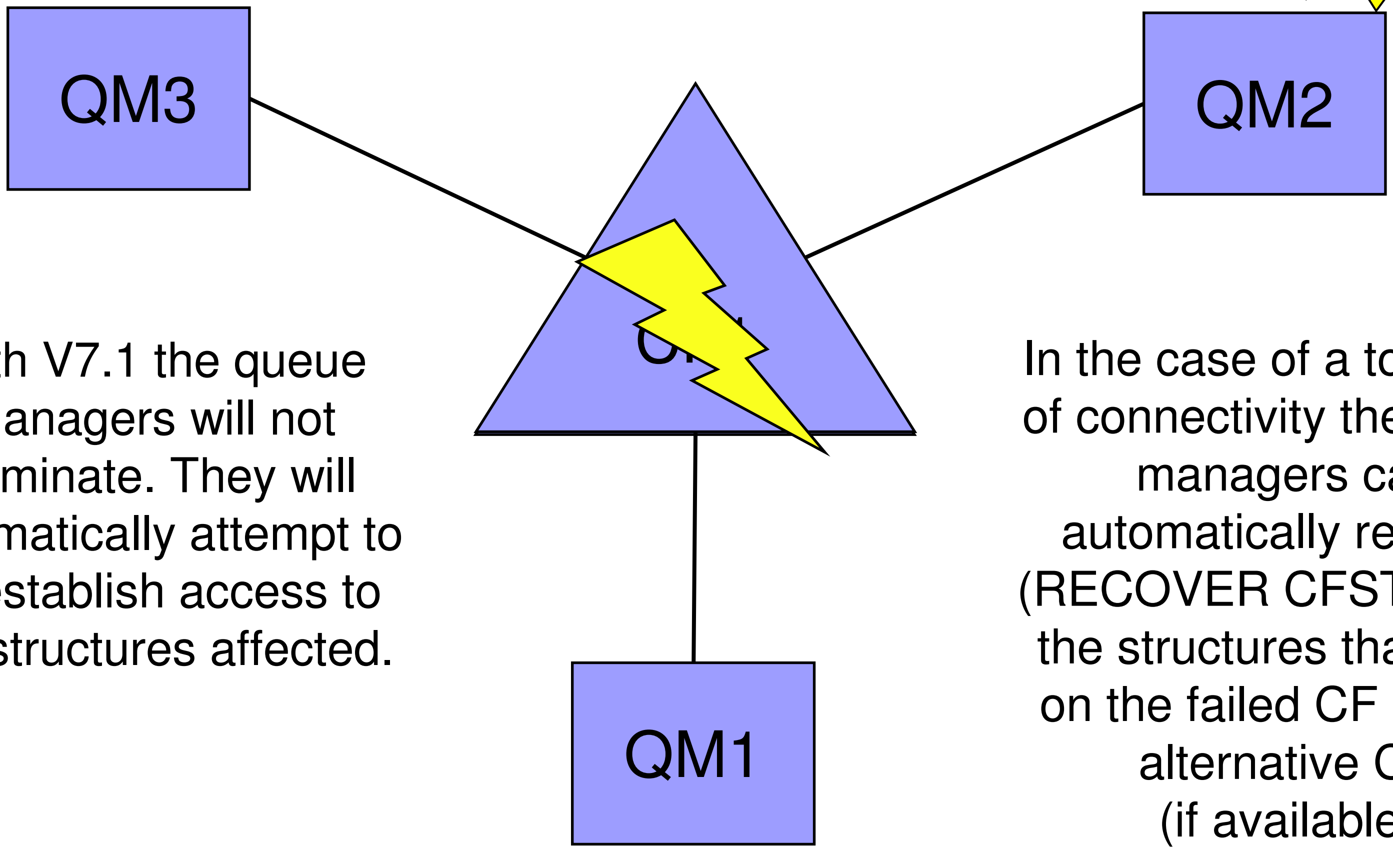
Prior to V7.1, if a queue manager receives a loss of connectivity, it will terminate.

In the case of a Coupling Facility failure, this would mean a QSG wide outage (unless protected by CF Duplexing)

CF Loss of Connectivity Tolerance

V7.1

V7.1+ Queue Managers



With V7.1 the queue managers will not terminate. They will automatically attempt to re-establish access to the structures affected.

In the case of a total loss of connectivity the queue managers can automatically recover (RECOVER CFSTRUCT) the structures that were on the failed CF into an alternative CF (if available)

Admin structure loss of connectivity

V7.1

NOTES

Queue managers will tolerate loss of connectivity to the admin structure without terminating if:

the QMGR CFCONLOS attribute is set to TOLERATE
all the queue managers in the QSG are at V7.1

All queue managers in the QSG will disconnect from the admin structure, then attempt to reconnect and rebuild their own admin structure data.

If a queue manager cannot reconnect to the admin structure, for example because there is no CF available with better connectivity, some shared queue operations will remain unavailable until the queue manager can successfully reconnect to the admin structure and rebuild its admin structure data.

The queue manager will automatically reconnect to the admin structure when a suitable CF becomes available on the system.

Failure to connect to the admin structure during queue manager startup is not tolerated, regardless of the value of CFCONLOS.

Application structure loss of connectivity

V7.1

NOTES

Queue managers will tolerate loss of connectivity to application structures if:

they are at CFLEVEL(5)
the CFCONLOS attribute is set to TOLERATE

All queue managers that lose connectivity to an application structure will disconnect from the structure.

The next action depends on whether it is a partial or total loss of connectivity (according to MQ’s definition).

- loss of connectivity is partial if there is at least one system in the sysplex that still has connectivity to the CF that the structure is allocated in.
- loss of connectivity is total if all systems in the sysplex have lost connectivity to the CF that the structure is allocated in.

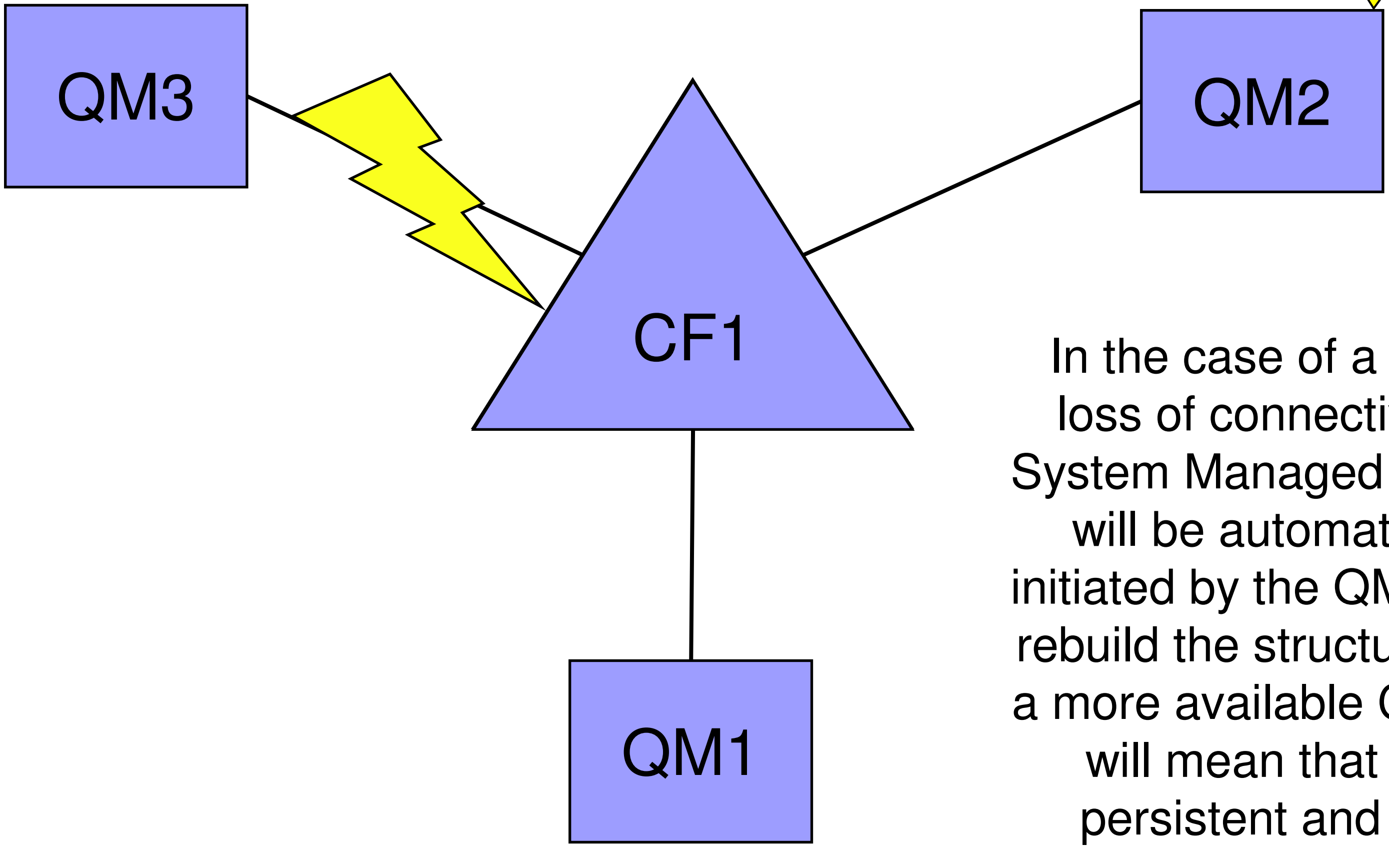
In the case of total loss of connectivity

- the structure will (probably) need to be recovered using the RECOVER CFSTRUCT command.
- non-persistent messages will be lost.

CF Loss of Connectivity Tolerance

V7.1

V7.1+ Queue Managers



In the case of a partial loss of connectivity, a System Managed Rebuild will be automatically initiated by the QMGRs to rebuild the structures into a more available CF. This will mean that both persistent and non-persistent messages will be retained.

Application structure loss of connectivity

V7.1

NOTES

In the case of partial loss of connectivity

- queue managers that lost connectivity to the structure will attempt to initiate a system-managed rebuild in order to move the structure to another CF with better connectivity.
- if the rebuild is successful, both persistent and non-persistent messages will be copied to the other CF.
- queue managers that didn’t lose connectivity to the structure may experience a slight delay during system-managed rebuild processing, but shared queues will remain available.

If an application structure cannot be reallocated in another CF with better connectivity, queues on the structure will remain unavailable until connectivity is restored to the CF that the structure is currently allocated in.

Queue managers will automatically reconnect to the structure when it becomes available.

CF Loss of Connectivity Tolerance

V7.1

- QMGR CFCONLOS(TERMINATE|TOLERATE)
  - Specifies whether loss of connectivity to the admin structure should be tolerated
  - Default is TERMINATE
  - Can only be altered to TOLERATE when all QSG members are at 7.1

- CFSTRUCT CFCONLOS(TERMINATE|TOLERATE|ASQMGR)
  - Specifies whether loss of connectivity to application structures should be tolerated
  - Only available at CFLEVEL(5)
  - Default is ASQMGR for new CFLEVEL(5) structures, and TERMINATE for structures altered to CFLEVEL(5)

- CFSTRUCT RECAUTO(YES|NO)
  - Specifies whether application structures should be automatically recovered
  - Only available at CFLEVEL(5)
  - Default is YES for new CFLEVEL(5) structure, and NO for structures altered to CFLEVEL(5)

CFRM Policy Considerations

```
CFSTRUCT(TEST1)
    CFLEVEL(5)
    CFCONLOS(TOLERATE)
    RECAUTO(YES)
    OFFLOAD(SMDS)
```

```
STRUCTURE NAME(SQ27TEST1)
     SIZE(50000)
     INITSIZE(20000)
     DUPLEX(DISABLED)
     ALLOWAUTOALT(YES)
     PREFLIST(P5CF01,P5CF02)
```

- If using CFCONLOS(TOLERATE) also need to consider multiple CFs in PREFLIST
- ALLOWAUTOALT(YES) enables CF to adjust entry/element ratio, and also automatically resize structure up to SIZE value (can also adjust down to MINSIZE!!)
- MQ structures can be duplexed… this will make most types of failures transparent to MQ

Client Channels

- Client channels are stateless, so don’t use synchronization queues
  - Only benefit of using a shared channel is the shared status
  - Can cause performance issues if using shared channel
    - Needs to update DB2 status for each connect/disconnect
- Can configure a generic port to point at INDISP(QMGR) listener on each queue manager
  - Can still benefit from failover and balancing of client connections without using a shared channel, and can still use QSG name on the MQCONN
- Will not work for Extended Transactional Client (including WAS 2-Phase Commit over client conn) until at V7.0.1

Client Channels

N

O

T

E

S

As client channels are stateless, they don't use a synchronization queue. The only benefit of using a shared channel for client channels is the shared status information. However, the use of a shared server-connection channel has drawbacks as it means each connection/disconnect will cause the queue manager to update the shared channel status, which is held in DB2. This could lead to performance issues if there are lots of clients connecting.

It is still possible to use a generic port to provide workload distribution and failover in the QSG, but rather than targeting an INDISP(SHARED) listener on each queue manager, the INDISP(QMGR) listener should targeted.

When using client channels into a QSG it is not possible to use the Extended Transactional Client (or client connections from WAS) if you are using 2-phase commit, unless you are connecting into a V7.0.1 queue manager

2-Phase Commit Client Connections

V7.0.1

- When setting up the connection, specify the QSG name rather than QMGR name
  - In MQConnectionFactory if using JMS under WAS, you must ensure that you are only using shared resources
  - This causes a UR with GROUP disposition to be created, rather than QMGR
  - A GROUP UR can be inquired and resolved via any member of the QSG
    - If there is a failure, the transaction manager will reconnect to the QSG and request a list of in-doubt transactions. GROUP URs will be reported back no matter what QMGR they were started on

2-Phase Commit Client Connections

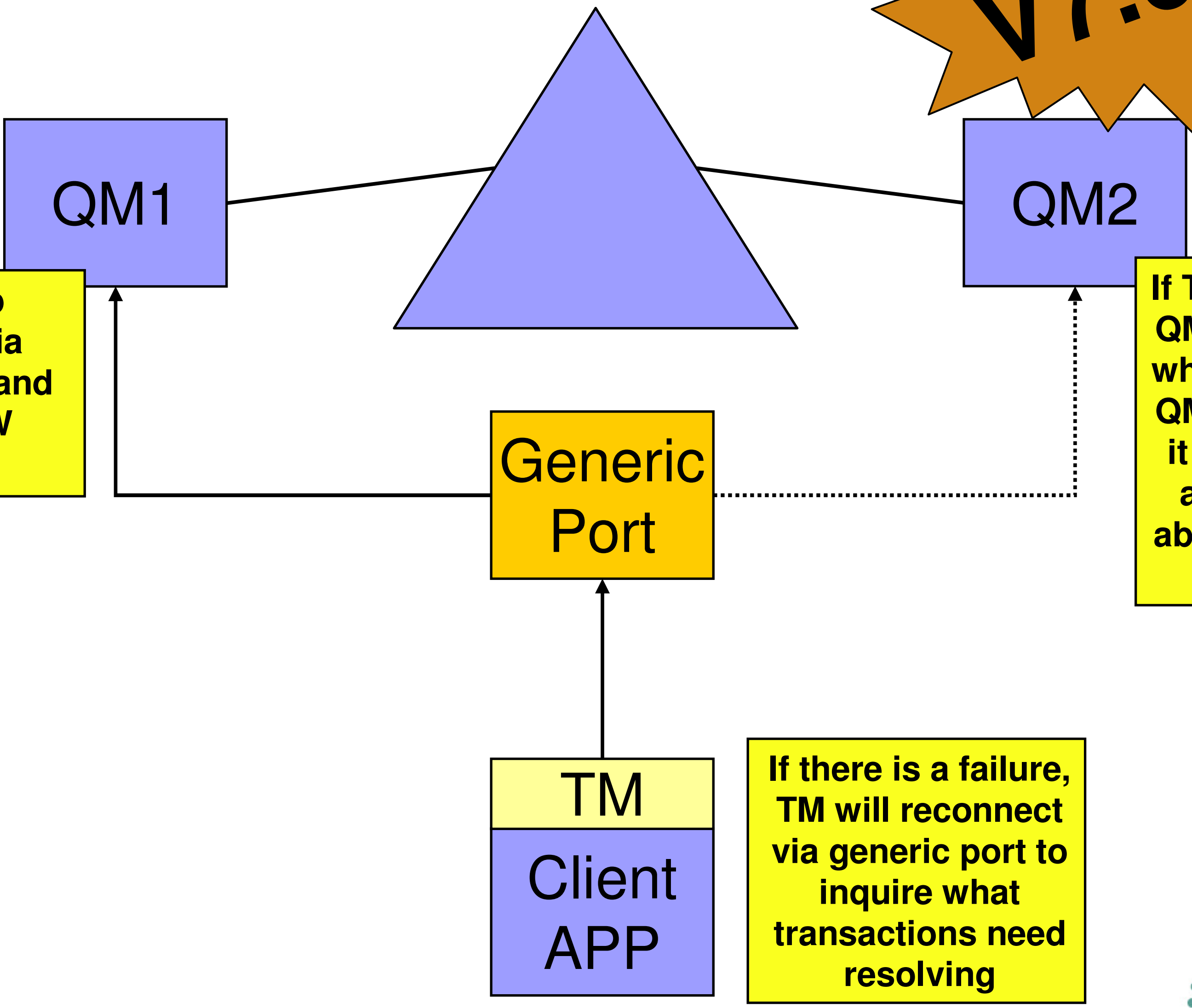
V7.0.1

N O T E S

When using the Extended Transactional Client, or the JMS transactional client (under WAS), it is possible to use 2-phase commit applications in a QSG. When specifying the connection options to the Transaction Manager it is necessary to provide the QSG name rather than the QMGR name, and also configure the client channel to be routed to a suitable (V7.0.1 or higher qmgr) in the QSG. When using this configuration, any Unit Of Recovery (UR) that is created will have a GROUP disposition. This means that it can be inquired and resolved on any qmgr in the QSG.

If a connection fails for some reason, and the TM reconnects to the QSG, it can inquire and resolve the transactions no matter which qmgr it is now connected to, and where the transactions were originally started.

GROUPUR – The Problem (Pre V7.0.1)

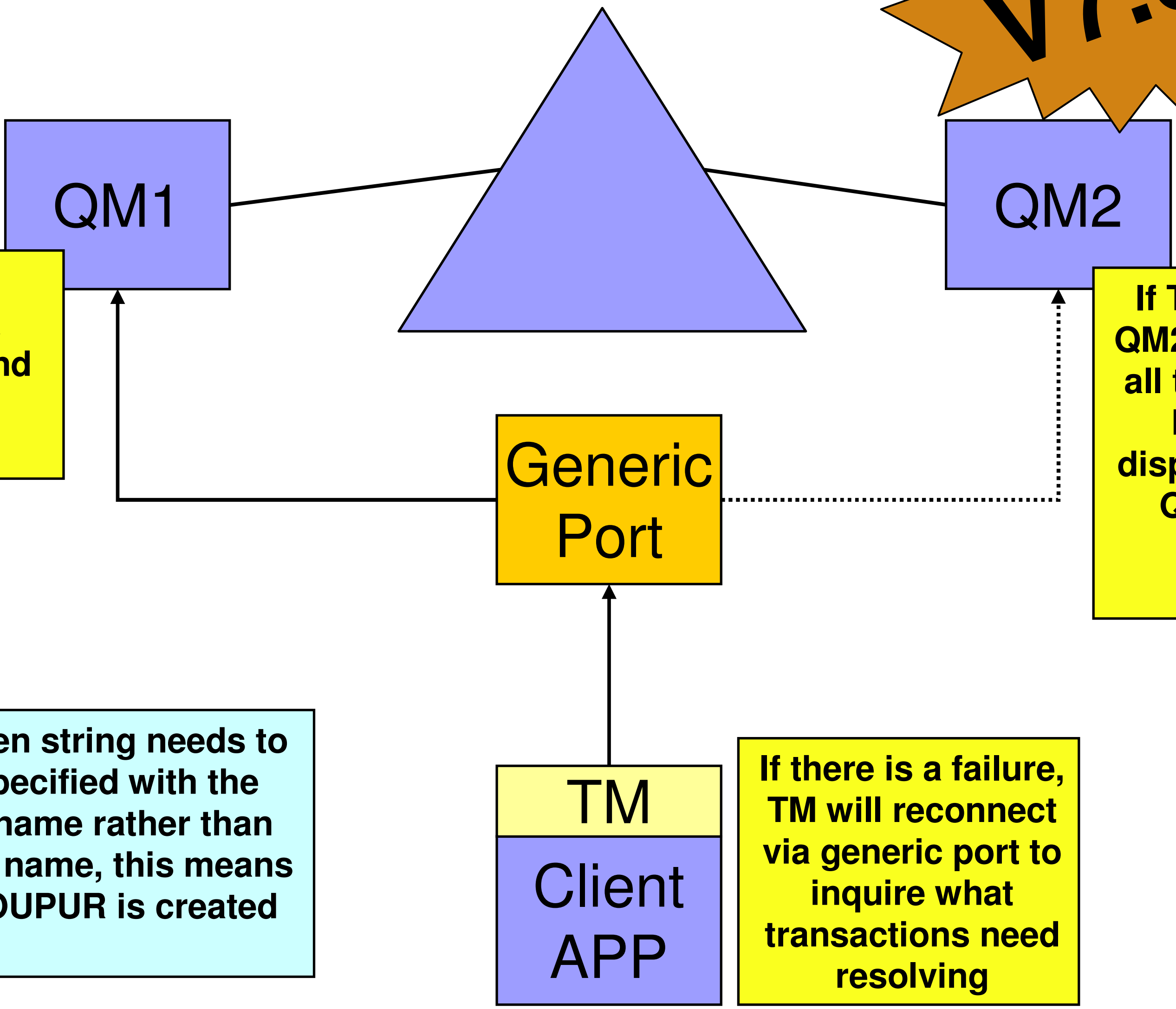


Client App connects via generic port and starts UOW

If TM reconnects to QM2 it only be told what is in-doubt on QM2, meaning that it will throw away any information about in-doubts on QM1

If there is a failure, TM will reconnect via generic port to inquire what transactions need resolving

GROUPUR – The Solution (V7.0.1)



Client App connects via generic port and starts UOW

If TM reconnects to QM2, QM2 will inquire all the in-doubts that have a GROUP disposition, whatever QMGR that were running on.

xa_open string needs to be specified with the QSG name rather than QMGR name, this means a GROUPUR is created

If there is a failure, TM will reconnect via generic port to inquire what transactions need resolving

More Information

- WebSphere MQ for z/OS Concepts and Planning Guide
- SupportPacs MP16, MP1E, MP1F, MQ1G
  - www.ibm.com/software/integration/support/supportpacs/perfreppacs.html
- RedPaper 3636 – WebSphere MQ Queue Sharing Group in a Parallel Sysplex environment
  - www.redbooks.ibm.com/redpieces/pdfs/redp3636.pdf

Any questions?

Please fill in evaluations at share.org/AnaheimEval #11514

Copyright Information