Migrating an existing IBM MQ configuration to an IBM Multi-site Workload Lifeline supported IBM MQ configuration

The purpose of this white paper is to aid customers that use traditional IBM MQ configurations, using direct queue manager to queue manager communication, to migrate to an MQ cluster configuration. IBM Multi-site Workload Lifeline relies on IBM MQ clustering technology to direct messages to the correct queue managers within the MQ cluster.

IBM MQ

IBM MQ provides a flexible, robust messaging backbone, with assured delivery of messages across a wide range of operating systems and platforms. An important feature of messaging is to remove complexity from application code – the reliable delivery and recovery of data is the job of the messaging provider, IBM MQ. A key concept of IBM MQ is its asynchronous processing - applications don't rely on each other's availability, or the availability of the network, to send data. If the applications and network are available, messages are immediately delivered. However, if the target application or network is not available, IBM MQ temporarily stores the data (i.e. messages) until it can be delivered.

The fundamental components of IBM MQ are its messages, queues, queue managers, and channels.

- **Messages** are data blocks that have some meaning to an application. IBM MQ adds information about how to store and deliver the message before IBM MQ forwards the message, and strips the information before delivering the message to the target application.
- **Queues** are data objects used to store messages, either until the message is forwarded to the appropriate queue manager or until the message is requested by the target application.
- **Queue managers** provide the messaging services and manage the queues and channels. Queue managers handle the delivery and recovery of messages. They are responsible for ensuring that messages are transferred to other queue managers by using channels.
- **Channels** represent the logical communication links, typically TCP socket connections, between queue managers and between a queue manager and a connecting application.

Figure 1 shows an example of an IBM MQ topology.

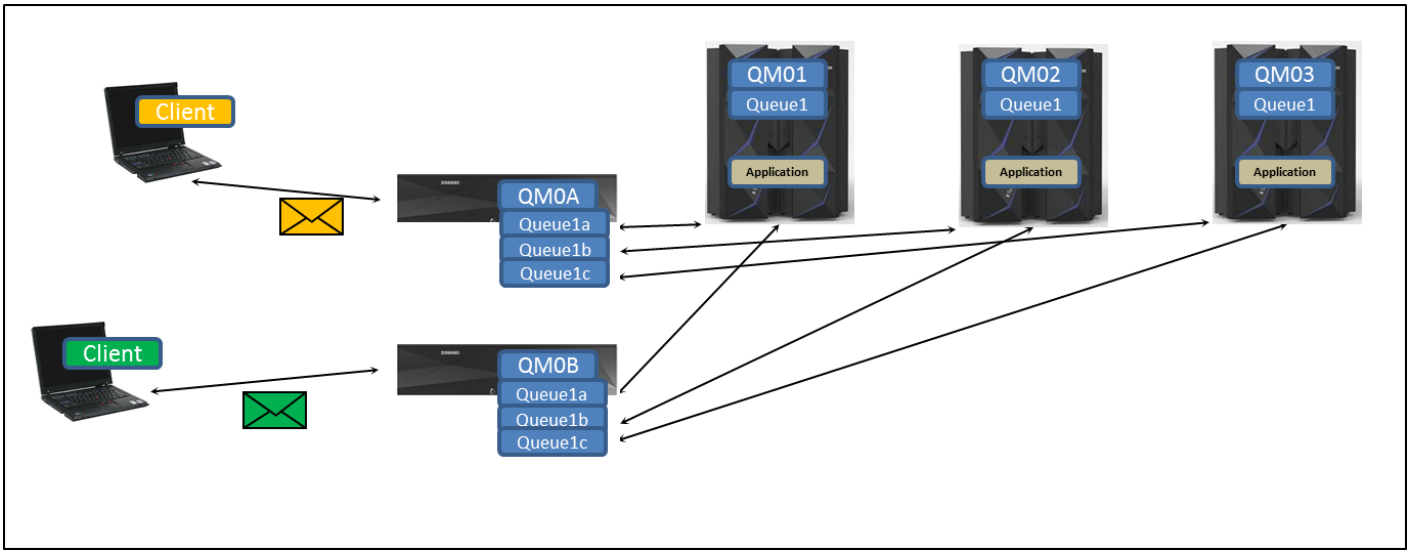

Figure 1 – IBM MQ topology

In Figure 1, clients connect to local queue managers to send and retrieve messages. These messages typically are destined to queues on remote queue managers, where the application processing the messages is located. After processing a message, the application usually sends a response message back to the local queue manager to be delivered to the client. In this example, clients use their local queue managers, QM0A or QM0B, to deliver messages to queue Queue1 that is configured on three remote queue managers, QM01, QM02, and QM03. When messages are delivered to a remote queue manager, they are queued there until the application can process them. Each application connects to its colocated queue manager to extract and process the messages that are queued on that queue manager.

To forward messages between the local and remote queue managers, channels are configured between each local queue manager and all possible remote queue managers. The local queue managers must define different local queue names that map to each remote queue manager and queue name pair, to target a specific queue manager. In this example, Queue1a maps to queue manager QM01 and Queue1, Queue1b maps to queue manager QM02 and Queue1, and so on. When delivering a message to a queue on a remote queue manager, the client needs to specify the local queue name that maps to the queue on the requested remote queue manager. If that remote queue manager is unavailable at the time the messages are sent by the client, the client's messages are queued on its local queue manager until that remote queue manager has recovered. The application retrieving messages from its colocated queue manager is unable to extract queued messages until that queue manager has recovered.

IBM MQ shared queues and shared channels

When queue managers are configured on z/OS systems, two additional features can be used to provide higher resiliency and availability – shared queues and shared channels. To use either of these features, each of the z/OS queue managers must be part of the same queue sharing group. A DB2 data sharing group must be configured and the queue sharing group defined within the DB2 data sharing environment. See the *IBM MQ: Installing IBM MQ* manual for information about configuring and using shared queues and shared channels.

- In a shared queue environment, a client can deliver messages to any of the z/OS queue managers within the queue sharing group. Because all the z/OS queue managers in the queue sharing group can access the same shared queue, the client does not depend on the availability of any one z/OS queue manager to be able to deliver messages. Applications connecting to their colocated z/OS queue managers retrieve messages from the same shared queue. This gives better availability if a z/OS queue manager becomes unavailable because the remaining z/OS queue managers in the queue sharing group can continue delivering messages from the shared queue to their colocated application.
- In a shared channel configuration, the z/OS queue managers all listen on the same z/OS Sysplex Distributor dynamically routable IP address (DRVIPA). Channel connections created by local queue managers to z/OS queue managers use this DRVIPA to create a channel to any one of the z/OS queue managers. All messages for a queue forwarded by a local queue manager are sent to the same z/OS queue manager while that z/OS queue manager is available. However, shared channels provide better resiliency if a z/OS queue manager is no longer available; the local queue manager forwarding messages can restart the channel using the same DRVIPA to establish a channel connection to another, available z/OS queue manager.

Figure 2 shows an example of an IBM MQ topology that uses shared queues and shared channels.

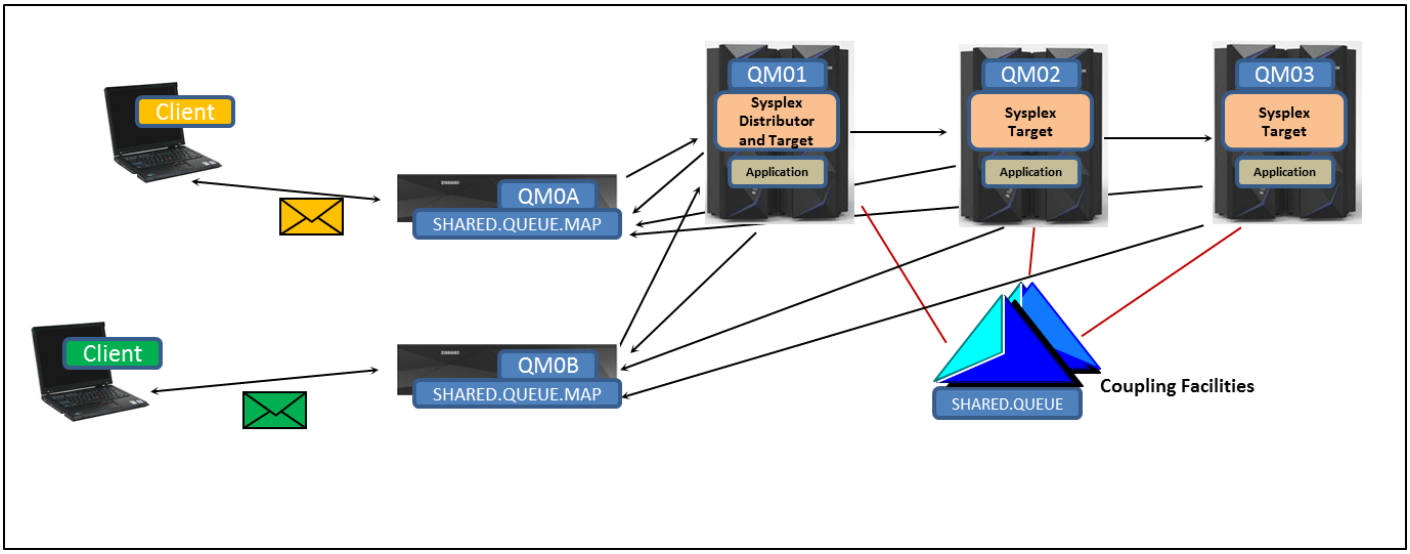

Figure 2 – IBM MQ shared queues and shared channels

In Figure 2, clients connect to local queue managers to send and retrieve messages. In this example, clients use their local queue managers, QM0A or QM0B, to deliver messages to queue SHARED.QUEUE that is configured as a shared queue. The shared queue is defined in a z/OS coupling facility, and queued messages are stored in coupling facility structures. The three z/OS queue managers, QM01, QM02, and QM03 all have access to this shared queue. When messages are delivered to a z/OS queue manager, they are queued in the shared queue until the application running on any of these remote systems can process them. Each application connects to its colocated z/OS queue manager, but can extract and process the messages that are queued on the same shared queue.

To forward messages between the local queue manager and remote z/OS queue managers, a single shared channel can be configured between a local queue manager and any of the z/OS queue managers. One of the z/OS queue managers defines the shared receiver channel, and its definition is propagated to all other z/OS queue managers in the queue sharing group. All local queue managers define a sender channel to this shared receiver channel. As each local queue manager starts the shared channel, the z/OS Sysplex Distributor selects one of the z/OS queue managers as the endpoint for this channel connection. Each of the local queue managers defines the same local queue name that maps to the shared queue configured on each z/OS queue manager. In this example, the local queue SHARED.QUEUE.MAP maps to any z/OS queue manager and the shared queue SHARED.QUEUE. The client needs to specify the local queue name that maps to the specific z/OS queue manager where the shared queue resides. If the z/OS queue manager becomes unavailable, the local queue manager automatically restarts its sender channel and the z/OS Sysplex Distributor selects a new z/OS queue manager as the endpoint for this channel connection. The application processing messages from the unavailable z/OS queue manager is unable to extract queued messages until the z/OS queue manager is recovered. However, because the queue is shared, applications colocated with the remaining z/OS queue managers continue to be able to extract queued messages.

IBM Multi-site Workload Lifeline

Although a z/OS sysplex environment provides high availability for workloads, some scenarios require more availability needs. For example, during an application upgrade or subsystem migration to a newer release, all systems in a z/OS sysplex might need to be taken offline to perform the upgrade. During this planned outage, any workloads that depend on the application or subsystem are unavailable until the upgrade is complete and service is restored. In another example, in the unlikely event of a disaster, such as an environmental issue, that results in the shutdown of the entire data center where the z/OS sysplex resides, all workloads running in this z/OS sysplex are unavailable.

IBM Multi-site Workload Lifeline (Lifeline) is a product that provides near continuous availability for business-critical workloads by providing workload monitoring and routing across two z/OS sysplex environments. So, in the event of a planned outage such as a workload upgrade, or an unplanned outage such as a data center power disruption, Lifeline detects and reacts to this event, and reroutes requests for the workload(s) to an alternate z/OS sysplex residing in another data center. This greatly reduces the window where business-critical workloads are not available.

Lifeline supports many different methods for how workload requests are presented to the z/OS sysplex, such as HTTP(S), JDBC connections, and 3270 access to legacy SNA applications. Lifeline also supports workloads where IBM MQ messaging is used to deliver workload requests to a z/OS sysplex. For IBM MQ, Lifeline depends on the environment being configured as an MQ cluster that spans both z/OS sysplexes. The next section discusses an MQ cluster configuration and how it differs from the IBM MQ configurations described earlier.

See the *IBM Multi-site Workload Lifeline V2.5* manual for more detailed information about Lifeline.

IBM MQ clusters

An IBM MQ cluster consists of a set of queue managers that communicate with each other by using cluster channels. Cluster channels are essentially the same channels as described above in Figure 1, except that most of the channel connections are dynamically created by the MQ cluster infrastructure, helping to reduce the number of manual definitions required. Each queue manager that is configured in an MQ cluster defines a cluster receiver channel that indicates to other queue managers how it can be reached. Each queue manager also defines a cluster sender channel that points to a queue manager managing a full repository. A channel is dynamically defined and started whenever there is a need to forward messages between a pair of queue managers.

In the IBM MQ configurations described previously, a single point-to-point channel is used to forward messages from a local queue manager to one remote queue manager. In an MQ cluster configuration, the local queue manager can select a different remote queue manager for each message being forwarded, providing a better distribution of messages from the local queue manager.

A local database, called a repository, holds the information that is used by a queue manager in an MQ cluster to know the location and availability of other queue managers and their cluster queues within the MQ cluster. Within an MQ cluster, two queue managers are designated as **full** repositories, meaning they understand the entire topology of the MQ cluster. All other queue managers in the MQ cluster hold **partial** repositories. When one of these partial repository queue managers requires information about the MQ cluster, the queue manager retrieves the information from a queue manager that manages a full repository, and stores this information locally in its partial repository.

As channels are defined and started as needed within an MQ cluster, shared channels are not necessary nor supported in an MQ cluster. However, shared queues can still be used in an MQ cluster.

Figure 3 shows an example of an IBM MQ topology using an MQ cluster.

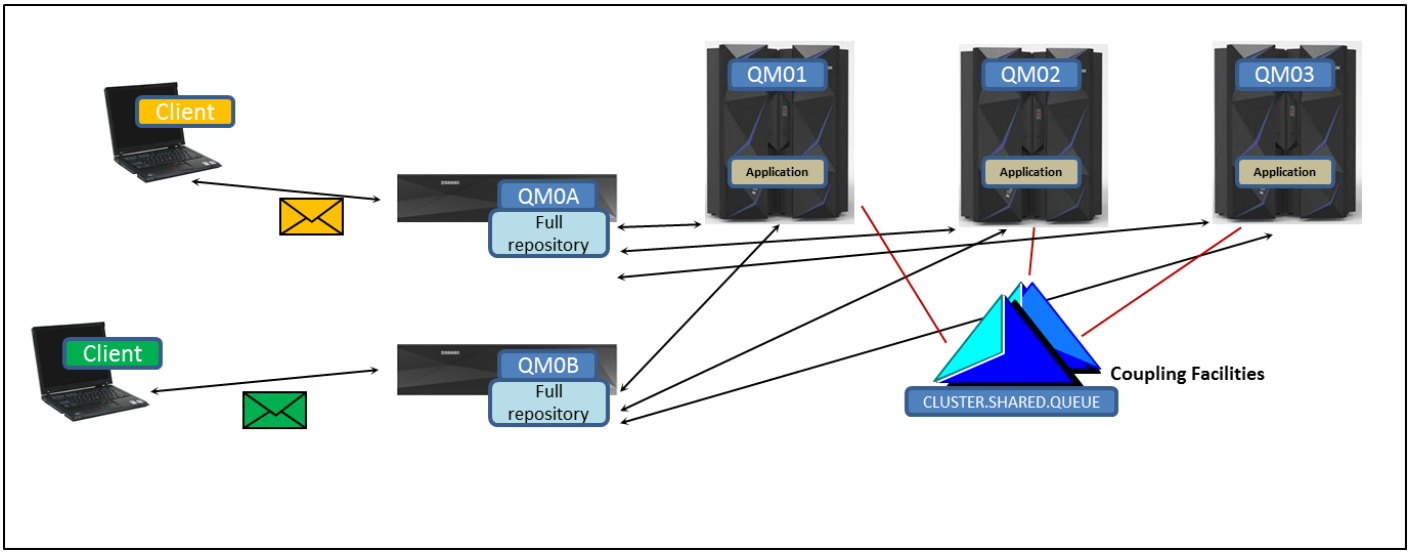

Figure 3 – IBM MQ clusters

In Figure 3, clients connect to local queue managers to send and retrieve messages. In this example, clients use their local queue managers, QM0A or QM0B, to deliver messages to queue CLUSTER.SHARED.QUEUE that is configured as a shared queue. The three z/OS queue managers, QM01, QM02, and QM03 all have access to this shared queue. When messages are delivered to a z/OS queue manager, they are queued in the shared queue until the application running on any of these remote systems can process them. Each application connects to its colocated z/OS queue manager, but extracts and processes the messages that are queued on the same shared queue.

To forward messages between the local and z/OS queue managers, the local queue manager, acting as a full repository, determines the z/OS queue managers that are available and host the cluster queue, and dynamically starts a channel connection, if not yet started, to one of the z/OS queue managers. The local queue managers do not need to configure any local queue name to map to the shared queue defined on the z/OS queue managers. Instead, the client specifies the cluster queue name of the shared queue residing within the MQ cluster. If the channel connection to the z/OS queue manager becomes unavailable, the local queue manager automatically starts a new channel connection to a different, available z/OS queue manager as the endpoint for this channel.

Note that in this example, only a single queue manager is defined on each system. With IBM MQ, it is possible to have multiple queue managers configured and active on a system.

Shared channel and shared queue configuration

For shared queues, the z/OS queue managers all share a single queue definition. One of the z/OS queue managers is responsible for defining the shared queue, and the definition is sent to all other z/OS queue managers in the queue sharing group. A typical definition for QM01, QM02, or QM03 from Figure 2 looks like this:

```
DEFINE QLOCAL( SHARED.QUEUE ) +
       QSGDISP( SHARED )  CFSTRUCT( MQSTRUCT ) +
       DESCR( 'Shared queue' )
```

For shared channels, the z/OS queue managers all share a single receiver channel definition. One of the z/OS queue managers is responsible for defining the receiver channel, and the definition is sent to all other z/OS queue managers in the queue sharing group. A typical definition for QM01, QM02, or QM03 from Figure 2 looks like this:

```
DEFINE CHANNEL( 'SHARED.RCVR.CHANNEL' ) +
       DESCR( 'Shared Receiver Channel' ) +
       CHLTYPE( RCVR ) +
       QSGDISP( GROUP )
```

When the channel initiator is started for each z/OS queue manager, the listener is configured to use the same DRVIPA, for example:

```
START LISTENER TRPTYPE( TCP ) PORT( 1414 ) +
        IPADDR( 10.10.10.1 ) INDISP( GROUP )

START CHANNEL( SHARED.RCVR.CHANNEL )
```

All local queue managers, QM0A and QM0B from Figure 2, that needed to create channel connections to the z/OS queue managers use the following definition to connect to this DRVIPA:

```
DEFINE QLOCAL( LOCAL.XMIT.QUEUE )
        DESCR( 'Local transmission queue' ) USAGE( XMITQ )

DEFINE CHANNEL( SDR.CHANNEL )
        CHLTYPE( SDR ) XMITQ( LOCAL.XMIT.QUEUE )
        CONNAME( '10.10.10.1(1414)' )
        DESCR( 'Sender channel to shared receiver channel' )

START CHANNEL( SDR.CHANNEL )
```

When the sender channel is started on the local queue manager, the z/OS Sysplex Distributor that is managing the DRVIPA routes the channel connection to one of the eligible z/OS queue managers that has this receiver channel defined.

The local queue managers need to define a receiver channel so that all z/OS queue managers create a channel connection back to it. Use the following statements to define the receiver channel for QM0A in Figure 2:

```
DEFINE CHANNEL( RCVR.CHANNEL ) CHLTYPE( RCVR )
        CONNAME( '10.10.20.1(1414)' )
        DESCR( 'Receiver channel' )

START CHANNEL( RCVR.CHANNEL )
```

Use a similar definition for the local queue manager QM0B in Figure 2, with the IP address in the CONNAME keyword mapping to its local IP address.

For all z/OS queue managers to be able to send a response message back to the local queue manager that sent the request message, sender channels are defined on each z/OS queue

manager, one to each local queue manager that it needs to respond to. A typical definition for QM01, QM02, and QM03 from Figure 2 looks like this:

```
DEFINE QLOCAL( LOCAL.XMIT.QUEUE ) +
       DESCR( 'Local transmission queue' ) USAGE( XMITQ )

DEFINE CHANNEL( QM0A.SNDR.CHANNEL )  +
       CHLTYPE( SDR ) XMITQ( LOCAL.XMIT.QUEUE ) +
       CONNAME( '10.10.20.1(1414)' ) +
       QSGDISP( QMGR ) +
       DESCR( 'Sender channel to QM0A' )

DEFINE CHANNEL( QM0B.SNDR.CHANNEL ) +
       CHLTYPE( SDR ) XMITQ( LOCAL.XMIT.QUEUE ) +
       CONNAME( '10.10.20.2(1414)' ) +
       QSGDISP( QMGR ) +
       DESCR( 'Sender channel to QM0B' )
```

Each of these sender channels needs to be started when the channel initiator is started for the z/OS queue manager.

For the local queue managers to identify the shared queue on the remote z/OS queue managers, clients must use a local mapping definition to connect and deliver messages. A typical definition for QM0A and QM0B from Figure 2 looks like this:

```
DEFINE QREMOTE( SHARED.QUEUE.MAP )
       RNAME( SHARED.QUEUE ) RQMNAME( MQSG )
       XMITQ( LOCAL.XMIT.QUEUE )
       DESCR( 'Mapping to shared queue' )
```

The keywords on this definition indicate the following:

- QREMOTE specifies the queue name that clients use when connecting to this local queue manager
- RNAME identifies the actual name of the shared queue as defined on the z/OS queue managers
- RQMNAME specifies the name of the queue sharing group defined for IBM MQ

When a client connects to its local queue manager and sends a message to this locally defined queue name, the local queue manager maps it to the correct shared queue and forwards the message to one of the z/OS queue managers that has defined the shared queue. For example, in Figure 2, the client connecting to its local queue manager QM0A sends a message to queue SHARED.QUEUE.MAP. The QM0A queue manager forwards the message to one of the z/OS queue managers, QM01, QM02, or QM03, for queue SHARED.QUEUE.

Converting to an IBM MQ cluster

To convert to an MQ cluster, the first step is to identify the queue managers that are to serve as full repository queue managers. It is recommended that two full repositories be configured in an MQ cluster. In Figure 3, local queue managers QM0A and QM0B were selected as full repositories. To configure a queue manager as a full repository, the following IBM MQ definition is needed:

```
DEFINE NAMELIST( CLUSTERLIST ) DESCR( 'List of clusters' ) NAMES( CLUSTER1 )
ALTER QMGR REPOSNL( CLUSTERLIST )
```

The first statement defines a list of the one or more MQ clusters that this queue manager is configured - in this example, the single MQ cluster named CLUSTER1. The next statement identifies this queue manager as a full repository queue manager for the single MQ cluster in the list.

For MQ clusters, the shared queues are defined as cluster queues. The z/OS queue managers continue to share a single queue definition. One of the z/OS queue managers is responsible for defining the shared cluster queue CLUSTER.SHARED.QUEUE, and the definition is sent to all other z/OS queue managers in the queue sharing group. A typical definition for QM01, QM02, or QM03 from Figure 3 looks like this:

```
DEFINE QLOCAL( CLUSTER.SHARED.QUEUE ) +
       QSGDISP( SHARED ) CFSTRUCT( MQSTRUCT ) +
       CLUSTER ( CLUSTER1 ) +
       DESCR( 'Shared cluster queue for CLUSTER1' )
```

The key difference in this definition is:

- the CLUSTER keyword is added to indicate the MQ cluster that is configured for the shared cluster queue

For MQ clusters, the sender and receiver channels are defined as cluster channels, so the definitions for sender and shared receiver channels shown previously are no longer used. Cluster receiver channels are defined on the full repository queue managers to allow all other queue managers in the MQ cluster to create channel connections to one of them. Use the following statements to define the cluster receiver channel for QM0A in Figure 3:

```
DEFINE CHANNEL( CLUSTER.RCVR.CHANNEL ) CHLTYPE( CLUSRCVR )
        CLUSTER( CLUSTER1 ) CONNAME( '10.10.20.1(1414)' )
        DESCR( 'Cluster CLUSTER1 receiver channel for QM0A' )

START CHANNEL( CLUSTER.RCVR.CHANNEL )
```

The key differences in this definition are:

- the channel type is now specified as CLUSRCVR
- the CLUSTER keyword is added to indicate the MQ cluster that is configured for the cluster receiver channel

Use a similar definition for the full repository queue manager QM0B in Figure 3, with the IP address in the CONNAME keyword mapping to its local IP address.

The only cluster sender channel definition that is needed on each of the full repository queue managers is to each other. For example, use the following statements for the full repository queue manager QM0A to create a channel to QM0B in Figure 3:

```
DEFINE CHANNEL( CLUSTER.SDR.CHANNEL ) CHLTYPE( CLUSSDR )
        CLUSTER( CLUSTER1 ) CONNAME( '10.10.20.2(1414)' )
        DESCR( 'Cluster CLUSTER1 sender channel to QM0B' )

START CHANNEL( CLUSTER.SDR.CHANNEL )
```

The key differences in this definition are:

- the channel type is now specified as CLUSSDR
- the CLUSTER keyword is added to indicate the MQ cluster that is configured for the cluster sender channel

Use a similar definition for the full repository queue manager QM0B in Figure 3, with the IP address in the CONNAME keyword mapping to QM0A's local IP address.

The z/OS queue managers also create cluster sender and receiver channels. The cluster sender channel creates a channel connection to one of the full repository queue manager and the z/OS queue manager dynamically discovers how to create channel connections to the other full repository queue manager. The cluster receiver channel is advertised to the full repository queue managers so that channel connections are dynamically created to the z/OS queue managers, when needed.

For example, use the following statements for the z/OS queue manager QM01 to create a channel connection to full repository queue manager QM0A in Figure 3 and to define the endpoint for other queue managers to dynamically create channel connections to QM01:

```
DEFINE CHANNEL( CLUSTER.SDR.CHANNEL ) CHLTYPE( CLUSSDR ) +
       CLUSTER( CLUSTER1 ) CONNAME( '10.10.20.1(1414)' ) +
       QSGDISP( QMGR )  +
       DESCR( 'Cluster CLUSTER1 sender channel to QM0A' )

DEFINE CHANNEL( CLUSTER.RCVR.CHANNEL ) CHLTYPE( CLUSRCVR ) +
       CLUSTER( CLUSTER1 ) CONNAME( '10.10.30.1(1414)' ) +
       QSGDISP( QMGR )  +
       DESCR( 'Cluster CLUSTER1 receiver channel for QM01' )
```

The key differences in this definition are:

- the channel type is now specified as CLUSSDR or CLUSRCVR
- the CLUSTER keyword is added to indicate the MQ cluster that is configured for the cluster sender and receiver channels
- the channel disposition, QSGDISP, is no longer specified as GROUP for the cluster receiver channel, as it was for a shared receiver channel definition
- the IP address specified on the CONNAME keyword for the cluster receiver channel is a local IP address, not a DRVIPA

Use a similar cluster sender and receiver channel definition for the other z/OS queue managers, QM02 and QM03 in Figure 3, with the IP addresses specified on the CONNAME keyword in the cluster receiver channel definition mapping to their own local IP address.

Each of these cluster channels needs to be started when the channel initiator is started for the z/OS queue manager.

For an MQ cluster, the local queue managers no longer need to define a local mapping to the remote shared queue. Clients connect to the local queue manager and deliver messages directly to the name of the shared cluster queue that is defined on the z/OS queue managers. For example, in Figure 3, the client connecting to its local queue manager QM0A sends a message to queue CLUSTER.SHARED.QUEUE. The QM0A queue manager determines from its full repository which of the z/OS queue managers, QM01, QM02, or QM03, to forward the message.

IBM Multi-site Workload Lifeline support for IBM MQ clusters

Lifeline provides nearly continuous availability for workloads that use IBM MQ to deliver workload messages to a z/OS sysplex. Lifeline requires that an MQ cluster be configured for the workload, and that the MQ cluster include queue managers configured on two z/OS sysplexes.

Lifeline monitors the availability of the MQ cluster, the z/OS queue managers comprising the MQ cluster, and the z/OS queue manager cluster queues. Lifeline influences the MQ cluster infrastructure so that messages get delivered to the correct z/OS queue managers, based on the state of the IBM MQ workload. Lifeline provides additional support so that any messages that remain queued in cluster queues on z/OS queue managers are transferred to cluster

queues on z/OS queue managers in the alternate z/OS sysplex, before switching the IBM MQ workload to the alternate site. See the *IBM Multi-site Workload Lifeline V2.5* manual for more information about how Lifeline manages IBM MQ workloads.

Lifeline recommends that z/OS queue managers that are configured in an MQ cluster be using v8.0 or higher, while non-z/OS queue managers should be using v7.5 or higher.

Figure 4 shows an example of an IBM MQ topology using an MQ cluster spanning two z/OS sysplexes.

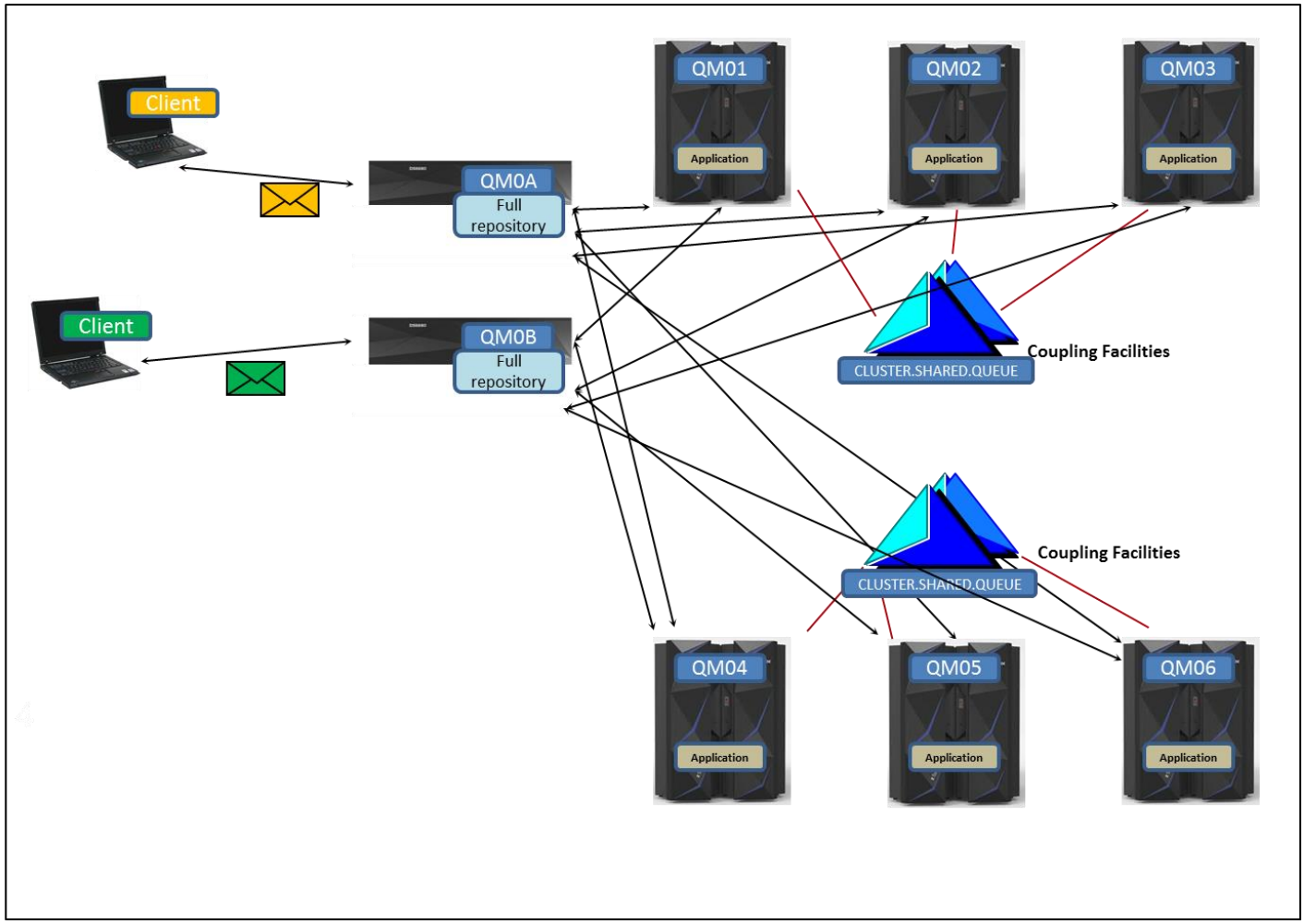

Figure 4 – Multi-sysplex IBM MQ cluster

In Figure 4, clients connect to local queue managers to send and retrieve messages. In this example, the remote z/OS queue managers reside in two different z/OS sysplexes. In this example, clients use their local queue managers, QM0A or QM0B, to deliver messages to queue CLUSTER.SHARED.QUEUE that is configured as a shared queue. Because the MQ cluster spans two z/OS sysplexes, the shared queue is defined in a z/OS coupling facility on both z/OS sysplexes, and queued messages are stored in their respective coupling facility structures. The three z/OS queue managers in one z/OS sysplex, QM01, QM02, and QM03 all

have access to the shared queue in their z/OS coupling facility, while the three z/OS queue managers in the other z/OS sysplex, QM04, QM05, and QM06 all have access to the shared queue in their z/OS coupling facility. When messages are delivered to a z/OS queue manager, they are queued in the shared queue on the z/OS sysplex where the z/OS queue manager resides, until the application running on any of these remote systems in that z/OS sysplex can process them. Each application connects to its colocated queue manager, but can only extract and process the messages that are queued on the same shared queue within its z/OS sysplex.

To forward messages between the local and remote z/OS queue managers, the local queue manager, acting as a full repository, determines the z/OS queue managers that are available across both z/OS sysplexes and host the cluster queue, and dynamically starts a channel connection to one of the available z/OS queue managers. A remote z/OS queue manager is considered available when all the following conditions are true:

- the queue manager is active
- the cluster queue is defined on the queue manager and can accept messages (i.e. the cluster queue is enabled for “puts”)
- the “rank” of the cluster queue on the queue manager is at least as high as any other queue manager in the MQ cluster
- the cluster receiver channel for the queue manager is started

If the channel connection to the z/OS queue manager becomes unavailable, the local queue manager automatically starts a new channel connection to a different, available z/OS queue manager as the endpoint for this channel.

Lifeline influences which z/OS queue managers are available in the MQ cluster by changing the state of cluster receiver channels, modifying cluster queue ranks (CLWLRANK), and changing the cluster receiver channel weights (CLWLWGHT).

On the active site for the IBM MQ workload, Lifeline starts the cluster receiver channels for the queue managers in that site, and sets the cluster queue rank for the workload’s cluster queues to a non-zero value. Lifeline also alters the cluster receiver channel weights to have more messages forwarded to the z/OS queue managers on the active site that are capable of handling additional work – as perceived from z/OS Workload Manager (WLM) and observed queue manager health.

On the alternate site, Lifeline stops the cluster receiver channels for the queue managers in that site, and sets the cluster queue rank for the workload’s cluster queues to zero

Figure 5 shows an example of an IBM MQ topology using an MQ cluster with Lifeline Advisors and Agents when the workload is active to a site.

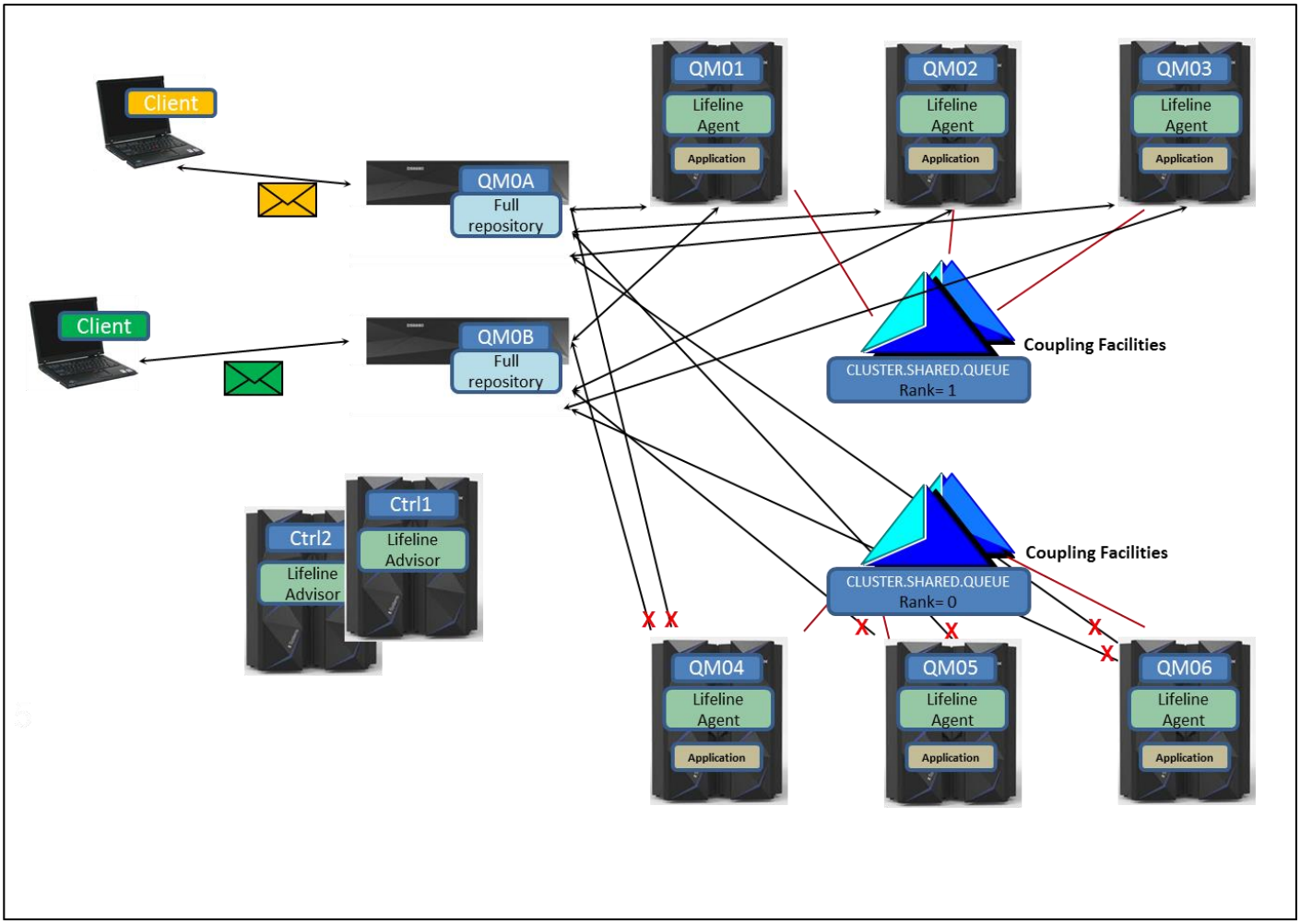

Figure 5 – Multi-sysplex IBM MQ cluster with Lifeline and active workload

In Figure 5, Lifeline Advisors reside on z/OS systems (known as controllers) outside of either z/OS sysplex. Lifeline Agents reside on the systems within both z/OS sysplexes, wherever z/OS queue managers are configured to host the cluster queue(s) for the workload. There is a primary Lifeline Advisor that communicates with each of the Lifeline Agents as well as the secondary Lifeline Advisor residing on a backup controller. If the primary Lifeline Advisor fails, the secondary Lifeline Advisor assumes the primary role and takes over communication with the Lifeline Agents.

Note that although the Lifeline Advisors are residing on separate z/OS systems, it is also acceptable to configure them to reside within the z/OS sysplexes where the workload applications are active.

The Lifeline Agents connect with the queue managers that are active and colocated on the same system where the Lifeline Agent is running. If a security product such as RACF has defined resource profiles and enabled any of the MQADMIN, MQCONN, or MQUEUE class, ensure the Lifeline Agents also have access to these defined resources. See the *IBM Multi-site Workload Lifeline V2.5* manual for more information about allowing Lifeline Agent access to IBM MQ resources.

When a workload that uses an MQ cluster is activated by Lifeline, the Lifeline Advisor notifies all the Lifeline Agents of the change in the workload state. In Figure 5, the active site is where queue managers QM01, QM02, and QM03 reside. The Lifeline Agents on the active site communicate with their colocated queue managers to ensure that the cluster queue rank is set to a non-zero value and the cluster receiver channels for each of these queue managers is started. The cluster receiver channel weights are also set based on WLM and health recommendations, relative to the other queue managers in the active site. The Lifeline Agents on the alternate site communicate with their colocated queue managers to ensure that the cluster queue rank is set to zero and the cluster receiver channels for each of these queue managers is stopped. Any messages for the workload that arrive on local queue managers QM0A and QM0B and for cluster queue CLUSTER.SHARED.QUEUE are forwarded to only the queue managers in the active site, QM01, QM02, and QM03.

Note that in this example, only a single z/OS queue manager is defined on each system. With IBM MQ, it is possible to have multiple z/OS queue managers configured and active on a system. The Lifeline Agent running on that system connects to all the queue managers active on that system and controls how each queue manager operates in the MQ cluster.

The default short retry interval (SHORTTMR) for cluster receiver channels is 60 seconds, while for the long retry interval (LONGTMR), the default is 1200 seconds. For the MQ cluster to quickly react to changes in the cluster receiver channel states, it is recommended that the z/OS queue managers be configured with the cluster receiver channel short and long retry intervals set to 5 seconds. Use the following definition to change the default retry intervals for the cluster receiver channel for QM01 in Figure 5:

```
DEFINE CHANNEL( CLUSTER.RCVR.CHANNEL ) CHLTYPE( CLUSRCVR ) +
       CLUSTER( CLUSTER1 ) CONNAME( '10.10.30.1(1414)' ) +
       QSGDISP( QMGR )  +
       SHORTTMR( 5 ) LONGTMR( 5 ) +
       DESCR( 'Cluster CLUSTER1 receiver channel for QM01' )
```

Use a similar cluster receiver channel definition for the other z/OS queue managers in Figure 5 to set their short and long retry intervals.

Note that when a cluster receiver channel is stopped for the MQ queue managers on the non-active site, these lower retry intervals increase the frequency that IBM MQ messages CSQX534E and CSQX599E are issued to the system console and IBM MQ joblog. To prevent these two messages from being issued, add them to the EXCLMSG system parameter for CSQ6SYSP in the IBM MQ system parameter module. See *IBM MQ: Installing IBM MQ* for information about the EXCLMSG system parameter.

When a z/OS queue manager is started, the IBM MQ workload that the queue manager is responsible for might be active on the alternate site. To prevent a queue manager from being started in an incorrect state and inadvertently receiving MQ messages, the cluster receiver channel for the queue manager should not be started. Use the following definition to ensure the

cluster receiver channel is stopped when the channel initiator is started for the queue manager QM01 in Figure 5:

```
STOP CHANNEL( CLUSTER.RCVR.CHANNEL )
```

Similar definitions are used for the other z/OS queue managers in Figure 5 to stop their cluster receiver channels when the channel initiator is started.

Once the Lifeline Agent running on the same system detects the started z/OS queue manager, the Lifeline Agent sets the appropriate cluster receiver state and weight, and cluster queue rank for the queue manager, based on the current state of the IBM MQ workload in that site.

Figure 6 shows an example of an IBM MQ topology using an MQ cluster with Lifeline Advisors and Agents when the workload is quiesced to both sites.

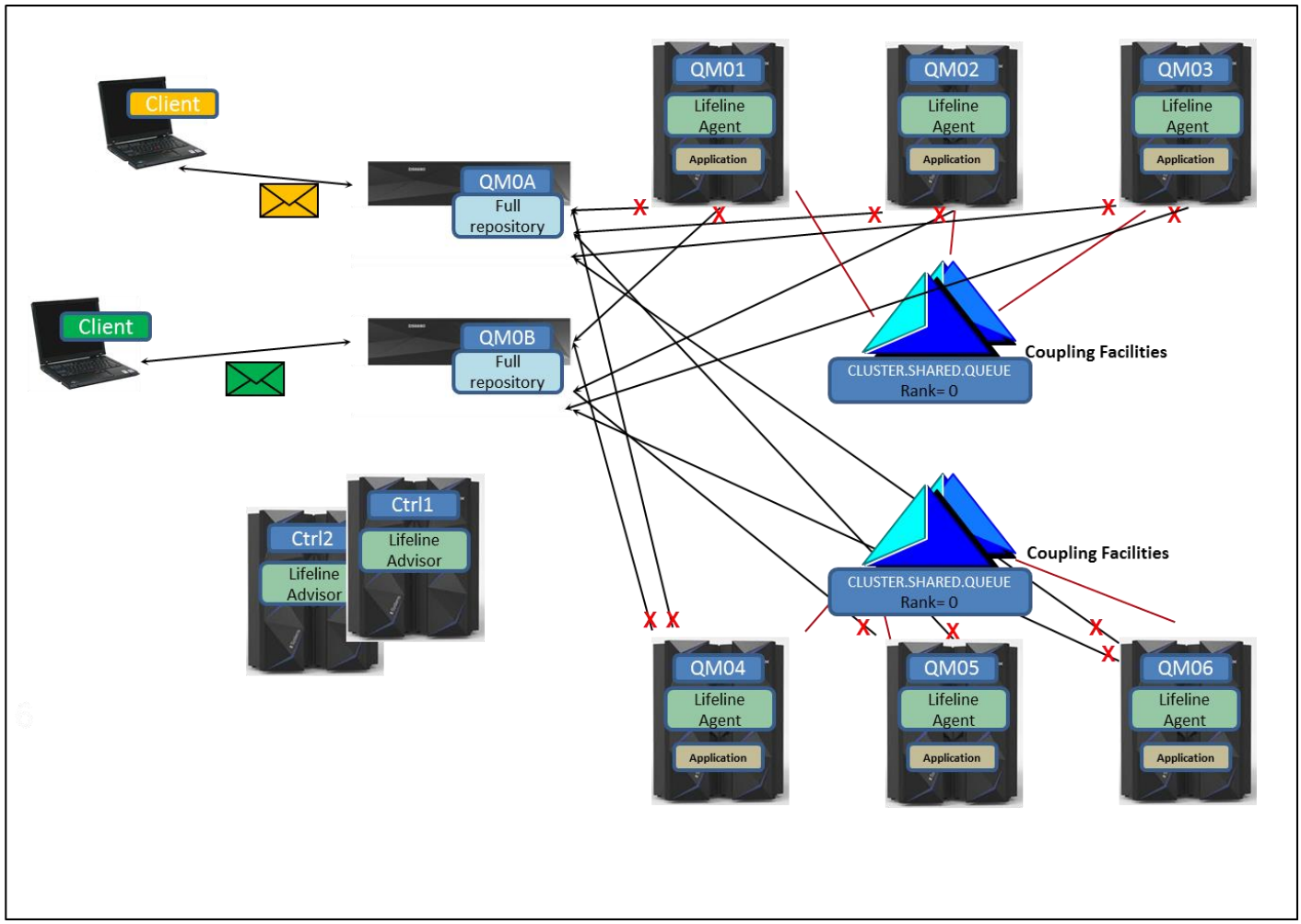

Figure 6 – Multi-sysplex IBM MQ cluster with Lifeline and quiesced workload

When a workload that uses an MQ cluster is quiesced by Lifeline, the Lifeline Advisor notifies all the Lifeline Agents of the change in the workload state. In Figure 6, the Lifeline Agents on both sites communicate with their colocated queue managers to ensure that the cluster queue rank is set to zero and the cluster receiver channels for each of these queue managers is stopped. Any messages for the workload that arrive on local queue managers QM0A and QM0B and for cluster queue CLUSTER.SHARED.QUEUE are queued on the local queue managers until the workload is activated to one of the sites.

Though messages are queued on the local queue managers when the workload is quiesced by Lifeline, a cluster queue and z/OS queue manager is selected by the local queue manager for each message. By default, cluster queues are defined with the DEFBIND(OPEN) setting, indicating that once the cluster queue is selected for a remote queue manager, the message remains queued locally until the cluster queue becomes available on the selected remote queue manager. In a Lifeline IBM MQ workload configuration, this default behavior is undesirable, if the cluster queue selected resides on a z/OS queue manager within the site that is not the active site. This implies that any messages that are to be forwarded to z/OS queue managers on the non-active site remain queued indefinitely, until the workload is activated to the site where these z/OS queue managers reside. To ensure that messages that are queued on the local queue managers get forwarded once the workload is activated, the default binding for the cluster queue must be set to NOTFIXED. This allows MQ cluster workload management to reallocate the messages to an available z/OS queue manager on the active site. Use the following definition to change the default binding for the cluster queue in Figure 5:

```
DEFINE QLOCAL( CLUSTER.SHARED.QUEUE ) +
       QSGDISP( SHARED )  CFSTRUCT( MQSTRUCT ) +
       CLUSTER ( CLUSTER1 ) +
       CLWLRANK( 0 ) +
       DEFBIND ( NOTFIXED ) +
       DESCR( 'Shared cluster queue for CLUSTER1' )
```

IBM Multi-site Workload Lifeline configuration for IBM MQ clusters

The Lifeline Advisor needs to identify the set of z/OS queue managers that are responsible for receiving messages for the workload. This information is passed to each of the Lifeline Agents, so the Lifeline Agents can locate the z/OS queue managers that are colocated on the same system where the Lifeline Agent is active.

For example, referring to the MQ cluster configuration in Figure 5, the following Lifeline Advisor mq_manager_list configuration statement is specified:

```
mq_manager_list
{
  QM01,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE1,MQWORKLOAD,INCLUDE,60
  QM04,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE2,MQWORKLOAD,INCLUDE,60
  QM02,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE1,MQWORKLOAD,INCLUDE,60
  QM05,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE2,MQWORKLOAD,INCLUDE,60
  QM03,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE1,MQWORKLOAD,INCLUDE,60
  QM06,CLUSTER1,CLUSTER.SHARED.QUEUE,SITE2,MQWORKLOAD,INCLUDE,60
}
```

The statement defines the following:

- z/OS queue managers used by the IBM MQ workload
- the name of the MQ cluster configured for the IBM MQ workload – in this example, the MQ cluster is CLUSTER1
- the cluster queue(s) on the z/OS queue manager that are used by the IBM MQ workload – in this example, the cluster queue for the workload is the shared queue CLUSTER.SHARED.QUEUE
- the site (i.e. z/OS sysplex name) where each z/OS queue manager resides
- the name of the IBM MQ workload – in this example, the workload name is MQWORKLOAD
- whether the cluster queue should be included or excluded when determining the cluster queues that should have their messages transferred to the alternate site – in this example, any messages on the shared queue, CLUSTER.SHARED.QUEUE, are transferred to z/OS queue managers on the alternate site when a Lifeline Advisor TRANSFER command is issued
- the failure detection interval, in seconds, before the Lifeline Advisor determines a workload failure has occurred

The order of queue managers within the mq_manager_list configuration statement is used to determine how to select a target cluster queue when transferring any remaining queued messages to the alternate site prior to a workload switch. For example, to transfer any remaining queued messages from the shared queue CLUSTER.SHARED.QUEUE on z/OS queue manager QM01, the Lifeline Advisor traverses the list of z/OS queue managers in the mq_manager_list to find the first z/OS queue manager on the alternate site where the shared queue CLUSTER.SHARED.QUEUE is available. In this example, z/OS queue manager QM04 is selected as the target z/OS queue manager to transfer these queued messages from z/OS queue manager QM01. See the *IBM Multi-site Workload Lifeline V2.5* manual for more information about the Lifeline Advisor mq_manager_list configuration statement.

To verify that Lifeline Agents are correctly monitoring their colocated z/OS queue managers, use the Lifeline Agent DISPLAY,MEMBERS command. This information includes the names of the z/OS queue managers configured for the IBM MQ workload that are colocated on this system, the name of the MQ cluster configured for the IBM MQ workload, and the group name associated with the cluster queue. For example, from Figure 5, on the system where z/OS queue manager QM01 is active, the Lifeline Agent display returns the following:

```
MODIFY AQSAGE,DISPLAY,MEMBERS
AQS0114I MEMBER SUMMARY
MQ CLUSTER      : CLUSTER1
  GROUP NAME    : CLUSTER1-CLUSTER.SHARED.QUEUE
    IPADDR..PORT: 10.10.30.1..1414
    MATCHES     : 001         PROTOCOL : TCP
    QMGR NAME   : QM01
1 OF 1 RECORDS DISPLAYED
```

See the *IBM Multi-site Workload Lifeline V2.5* manual for more information about the Lifeline Agent DISPLAY,MEMBERS command.

To verify that the Lifeline Advisor is receiving status updates for all z/OS queue managers, use the Lifeline Advisor DISPLAY,WORKLOAD,DETAIL command. This information includes the names of the z/OS queue managers configured for the IBM MQ workload and the availability of the cluster queues on each of the z/OS queue managers. The current maximum number of messages still queued on a cluster queue is also provided for each z/OS queue manager. For example, from Figure 5, the Lifeline Advisor display returns the following:

```
MODIFY AQSADV,DISPLAY,WORKLOAD=MQWORKLOAD,DETAIL
F AQSADV,DISPLAY,WORKLOAD,DETAIL
AQS0146I WORKLOAD DETAILS
TYPE : ACTIVE/STANDBY
  WORKLOAD NAME : MQWORKLOAD
    STATE : ACTIVE
      SITE : SITE1
    MQ QUEUE MANAGERS:
      QMGR : QM01          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.30.1..1414
      SYSTEM NAME : SYS1A            SITE : SITE1
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
 MQ QUEUE MANAGERS:
      QMGR : QM04          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.40.1..1414
      SYSTEM NAME : SYS2A            SITE : SITE2
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
 MQ QUEUE MANAGERS:
      QMGR : QM02          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.30.2..1414
      SYSTEM NAME : SYS1B            SITE : SITE1
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
 MQ QUEUE MANAGERS:
      QMGR : QM05          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.40.2..1414
      SYSTEM NAME : SYS2B            SITE : SITE2
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
 MQ QUEUE MANAGERS:
      QMGR : QM03          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.30.3..1414
      SYSTEM NAME : SYS1C            SITE : SITE1
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
 MQ QUEUE MANAGERS:
      QMGR : QM06          TRANSDEPTH : 0
        IPADDR..PORT : 10.10.40.3..1414
      SYSTEM NAME : SYS2C            SITE : SITE2
      QUEUE: CLUSTER.SHARED.QUEUE                          STATE : AVAIL
1 OF 1 RECORDS DISPLAYED
```

Conclusion

Customers have come to rely on IBM MQ to reliably deliver messages for their workloads. IBM MQ can be configured in several different ways – direct channels between queue managers, shared channels to a set of z/OS queue managers, or in a cluster of queue managers. Lifeline's support for workloads depending on IBM MQ messaging relies on MQ clusters to be configured. The previous sections described how non-clustering MQ environments worked, and the configuration changes required to convert to an MQ cluster that works with Lifeline. For further information on IBM MQ, see the *IBM MQ: Installing IBM MQ* manual. For further information on IBM Multi-site Workload Lifeline, see the *IBM Multi-site Workload Lifeline V2.5* manual.