IBM WebSphere Application Server

Complex Service Integration Bus Topologies

David Ware
WebSphere Messaging Development

05 March 2009

Legal Disclaimer

- **All content from this presentation is the property of IBM and should not be reproduced without IBM’s consent.**

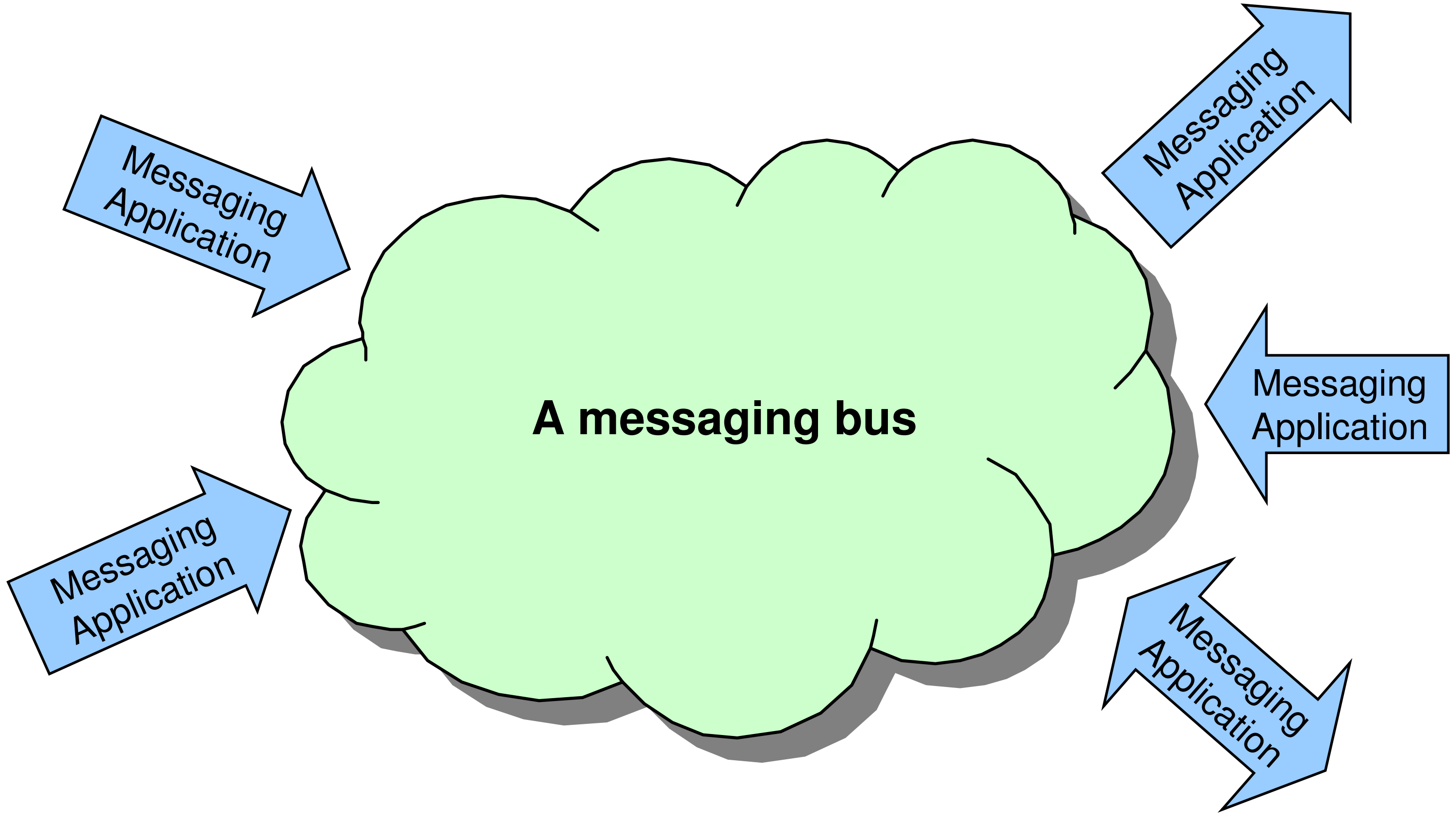
Objectives

- **Explain the factors that lead to more complex SIBus messaging topologies and explore some of the subtler aspects of SIBus topologies that ensure a correctly functioning, manageable and scalable system.**
  - This presentation will examine a number of SIBus topologies and work through the pros and cons of each, enabling you to know how to get the best out of them.

- **Provide an overview of *some* of the improvements made to the default messaging in WAS V7.**
  - And see how these improvements relate to the topologies.

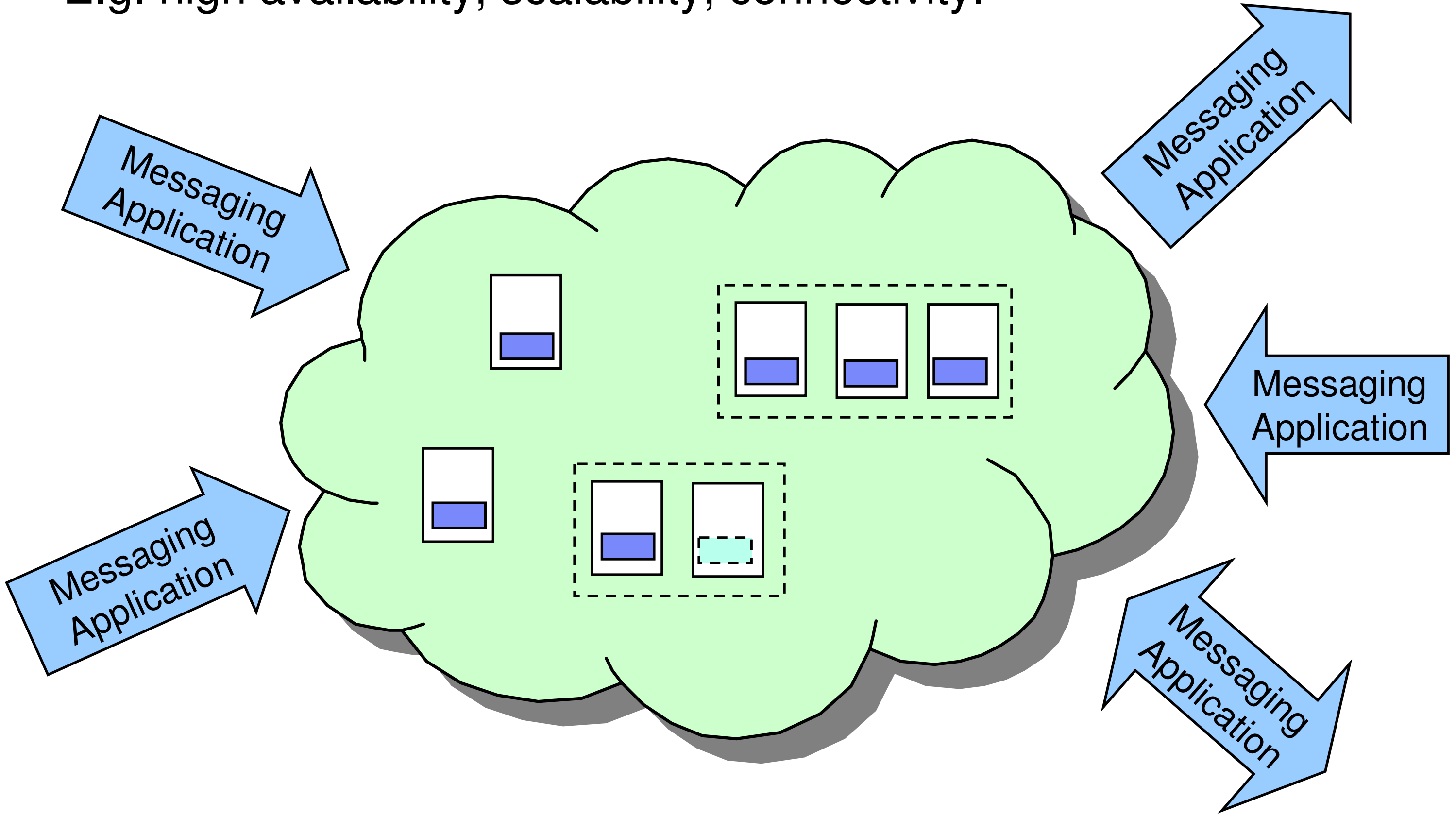
The bus

- **The initial design point of an SIBus was to be able to use it as a *location transparent messaging system*.**



The bus

- **Within that bus would be a messaging system capable of dealing with all the messaging requirements.**
  - E.g. high availability, scalability, connectivity.



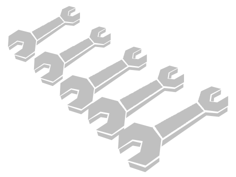
The bus

- **A bus will functionally work* irrespective of its topology, but unless carefully considered, can present many problems:**
  - Performance
  - Manageability
  - Availability
  - Serviceability

* There are cases where the bus configuration can affect the messaging applications' function, e.g. when using scalable clusters, as we'll see.

Keep it simple

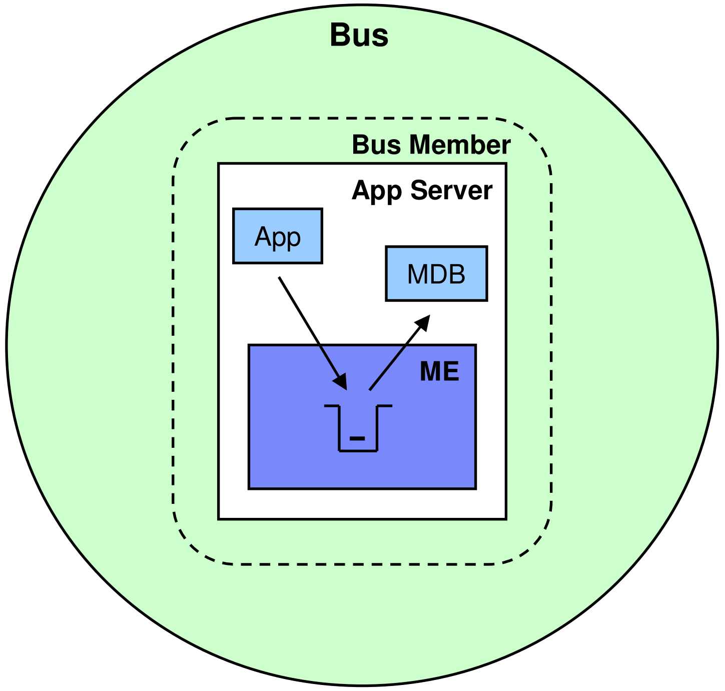
- **The best approach is to start with the simplest possible configuration and build from there when necessary…**



Terminology reminder

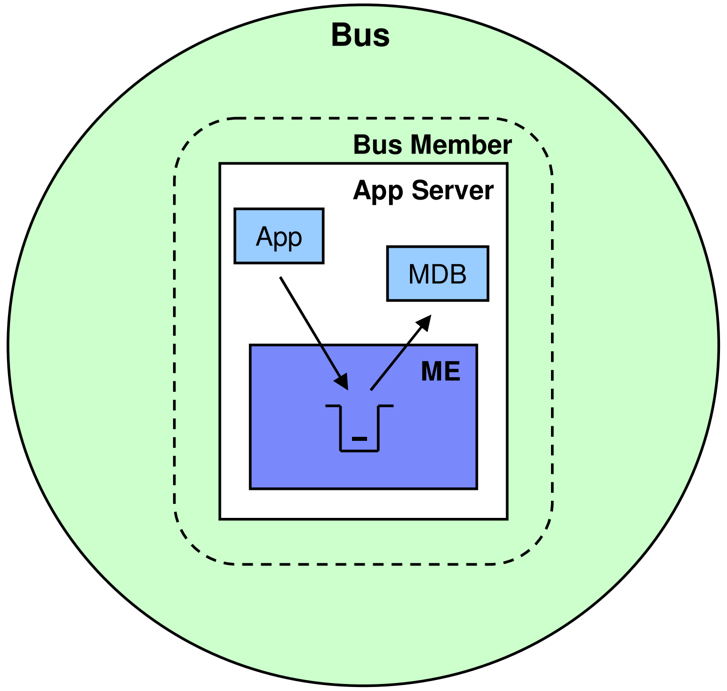
- **Bus Member**
  - An application server or cluster that has been made a member of a bus.
- **Messaging Engine (ME)**
  - The runtime instance(s) of a bus member. Each ME can only run in a single server at any one time. Each ME has its own in-memory and persistent state.
- **Queue**
  - A messaging destination for point-to-point messaging. A queue is assigned to a Bus Member.
- **Queue Point**
  - The runtime representation of a Queue, used to store messages.
  - Each ME in the assigned Bus Member will have its own Queue Point. Any message sent to the Queue will be stored on a single Queue Point.

Single server, single ME bus



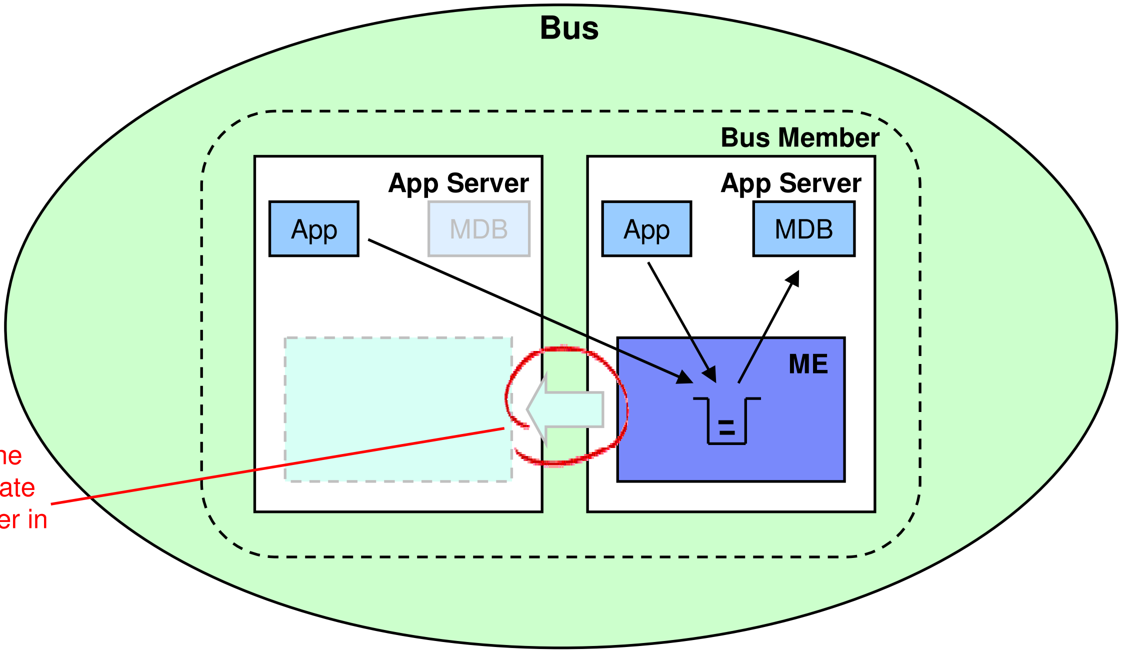
- Pros
  - Everything in one place
    - Aids performance as all messages and application connections are on the same ME
      - This minimises path length
    - Aids manageability as all messages and application connections are on the same ME
      - This minimises configuration and runtime objects to monitor

Single server, single ME bus



- Cons
  - Scalability and performance
    - of applications
    - of messaging
  - High availability
    - of applications
    - of messaging

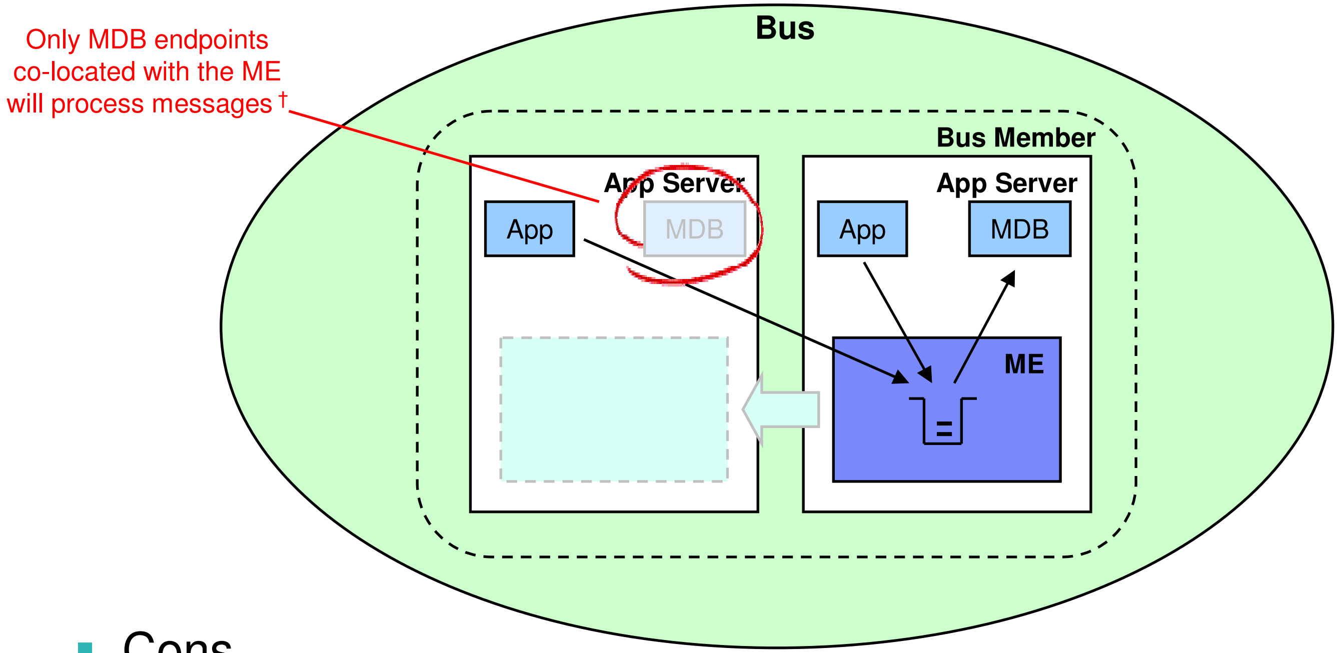
Application server cluster, single ME bus



In the event of a failure, the ME and all its persistent state can failover to another server in the cluster.

- Pros
  - All messaging is still in one place
  - Scalability and performance of applications
  - High availability of applications
  - High availability of messaging

Application server cluster, single ME bus

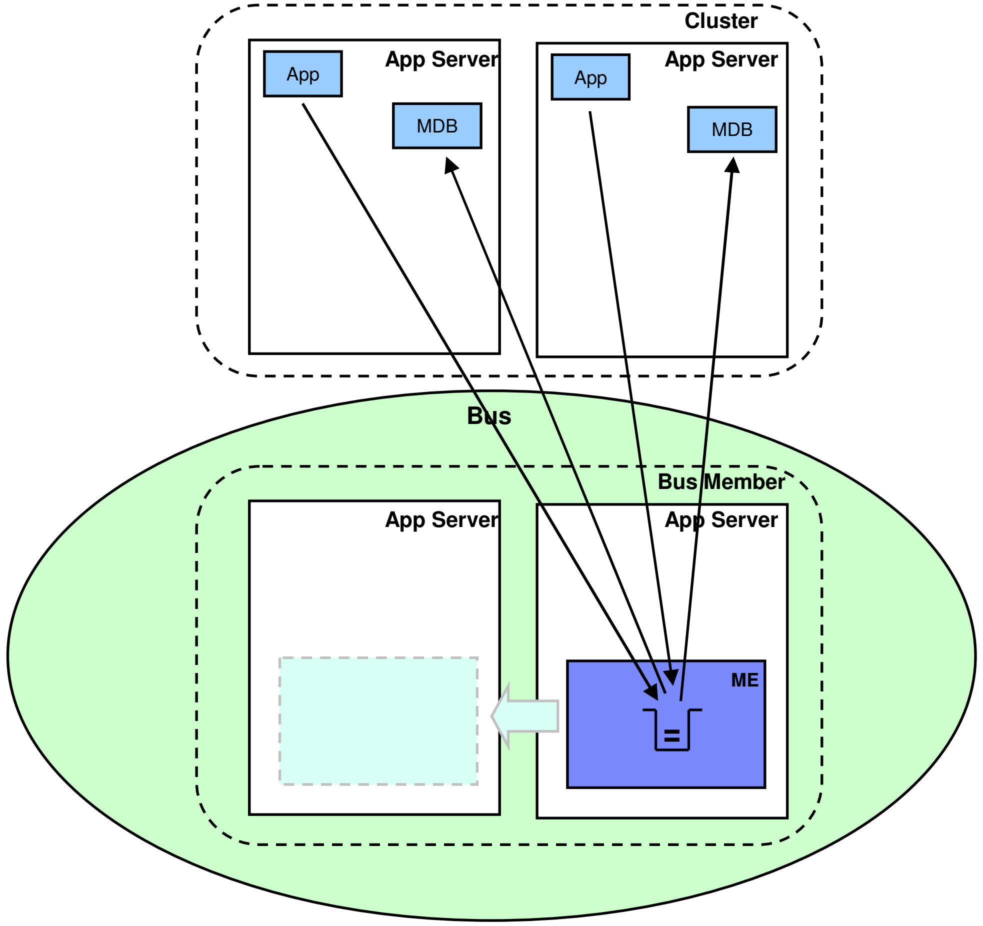

Only MDB endpoints co-located with the ME will process messages †
Bus
Bus Member
App Server
App
MDB
App Server
App
MDB
ME

- Cons
  - Scalability and performance
    - of messaging
    - of MDBs
  - Configuring a cluster bus member can be non-trivial
    - (although in the above topology it still is trivial)

† V7 introduces the option to modify this behaviour.

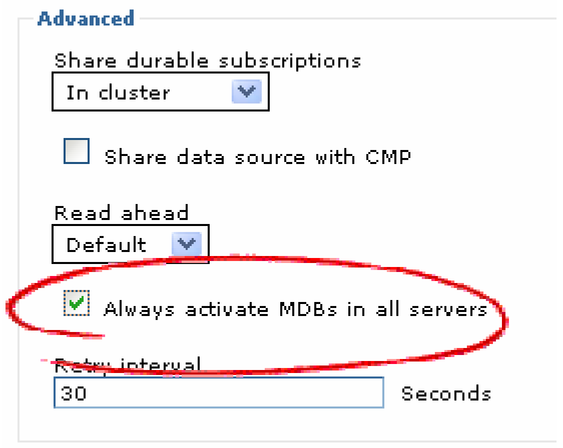
Application server cluster, single ME bus

- **To overcome the scalability of MDBs issue, one common configuration in V6 and V6.1 is the following:**



Application server cluster, single ME bus

- **In V7 it is possible to enable all MDB endpoints, irrespective of where the ME is running, so the extra server/cluster is no longer essential.**
  - Some may prefer to keep the separation to maintain a clear division of application runtime and messaging runtime.
  - This is configured using the a new option on the MDB's activation specification:
    - ***'Always activate MDBs in all servers'***



Advanced

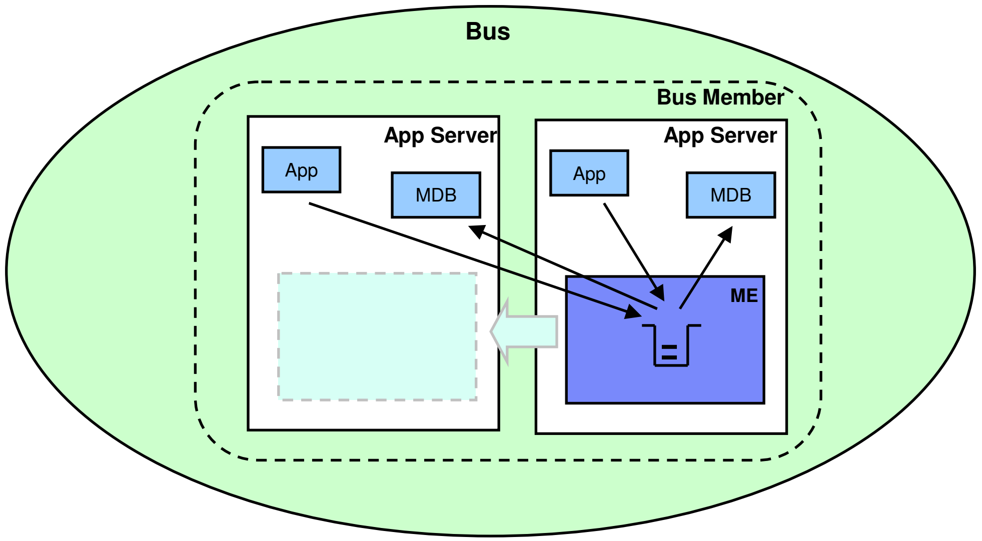
Share durable subscriptions: In cluster

☐ Share data source with CMP

Read ahead: Default

☑ Always activate MDBs in all servers

Retry interval: 30 Seconds



Application server cluster, single ME bus

- **Correctly configuring a cluster bus member in V6 and 6.1 involved configuring the following discrete information:**
  - The SIBus bus member
  - A corresponding Core Group policy and its Match Criteria may also be required for certain cluster configurations.

- **In V7 a** *configuration wizard* **provides a single point of configuration for cluster bus members and any associated Core Group policies.**

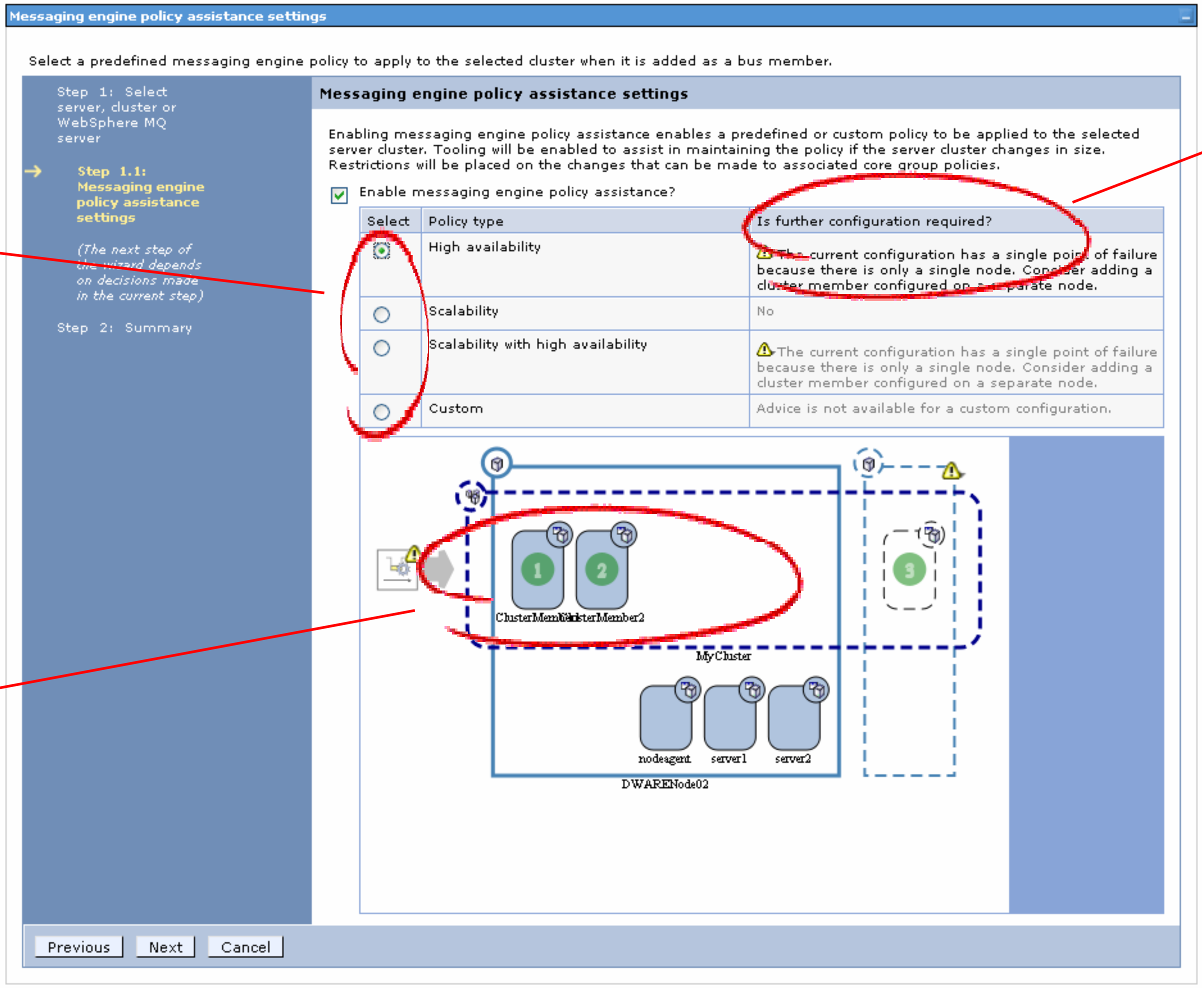
- **V7 also provides** *policy assist* **that helps in the configuration of typical cluster bus member policies**

Application server cluster, single ME bus

Policy assist provides automatic configuration of the Core Group policies to provide the most typical SIBus cluster configurations

The wizard generates a graphical representation of the selected cluster configuration

If the configuration is non-optimal or fails to satisfy the HA requirement *suggestions* are made



Messaging engine policy assistance settings

Select a predefined messaging engine policy to apply to the selected cluster when it is added as a bus member.

Step 1: Select server, cluster or WebSphere MQ server

→ **Step 1.1: Messaging engine policy assistance settings**

(The next step of the wizard depends on decisions made in the current step)

Step 2: Summary

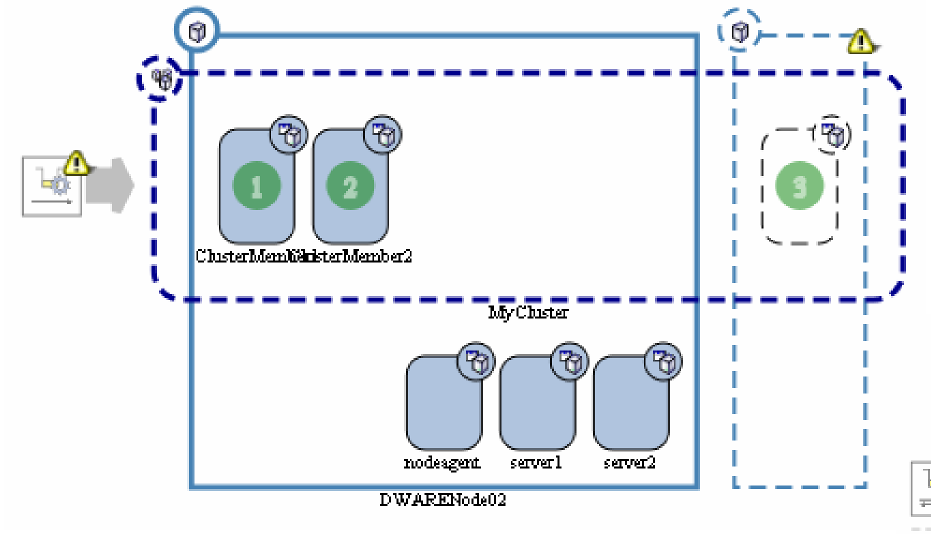
Messaging engine policy assistance settings

Enabling messaging engine policy assistance enables a predefined or custom policy to be applied to the selected server cluster. Tooling will be enabled to assist in maintaining the policy if the server cluster changes in size. Restrictions will be placed on the changes that can be made to associated core group policies.

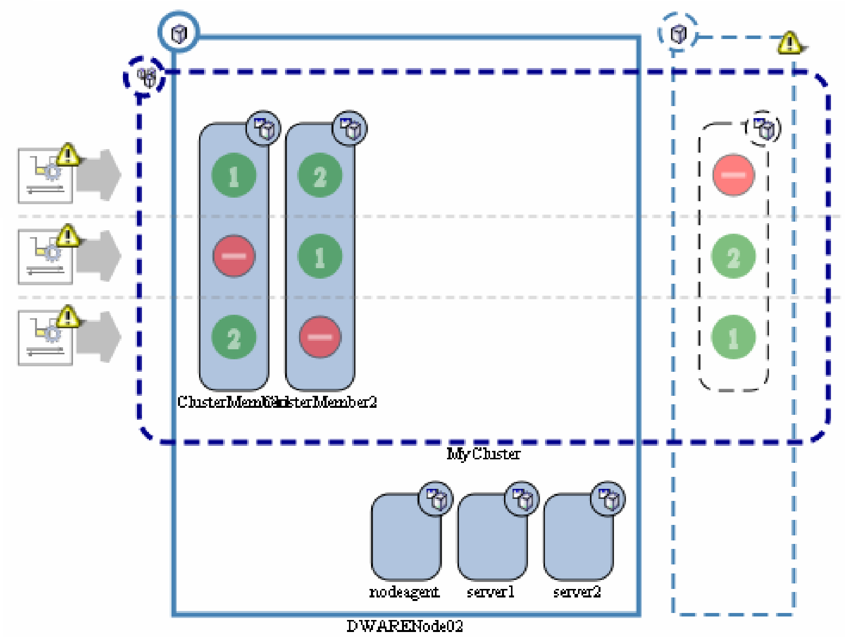
✓ Enable messaging engine policy assistance?

Select	Policy type	Is further configuration required?
●	High availability	⚠ The current configuration has a single point of failure because there is only a single node. Consider adding a cluster member configured on a separate node.
○	Scalability	No
○	Scalability with high availability	⚠ The current configuration has a single point of failure because there is only a single node. Consider adding a cluster member configured on a separate node.
○	Custom	Advice is not available for a custom configuration.

Application server cluster, single ME bus



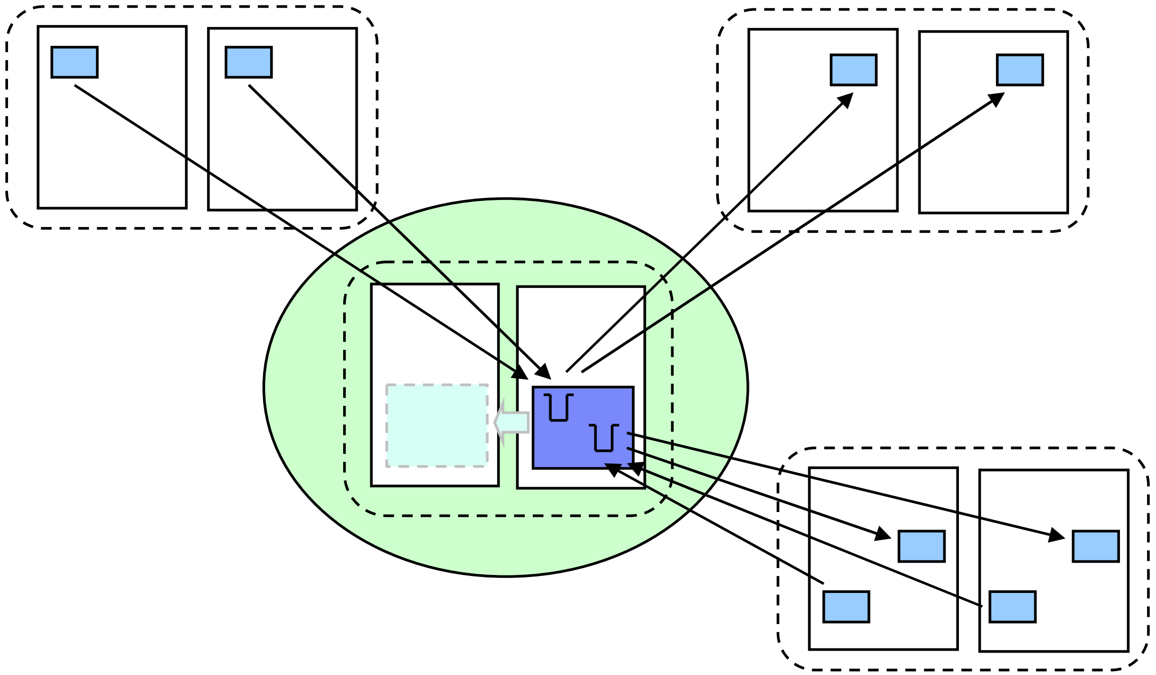
A highly available, single ME, cluster bus member



A highly available, three ME, cluster bus member

Multiple application server clusters, single ME bus

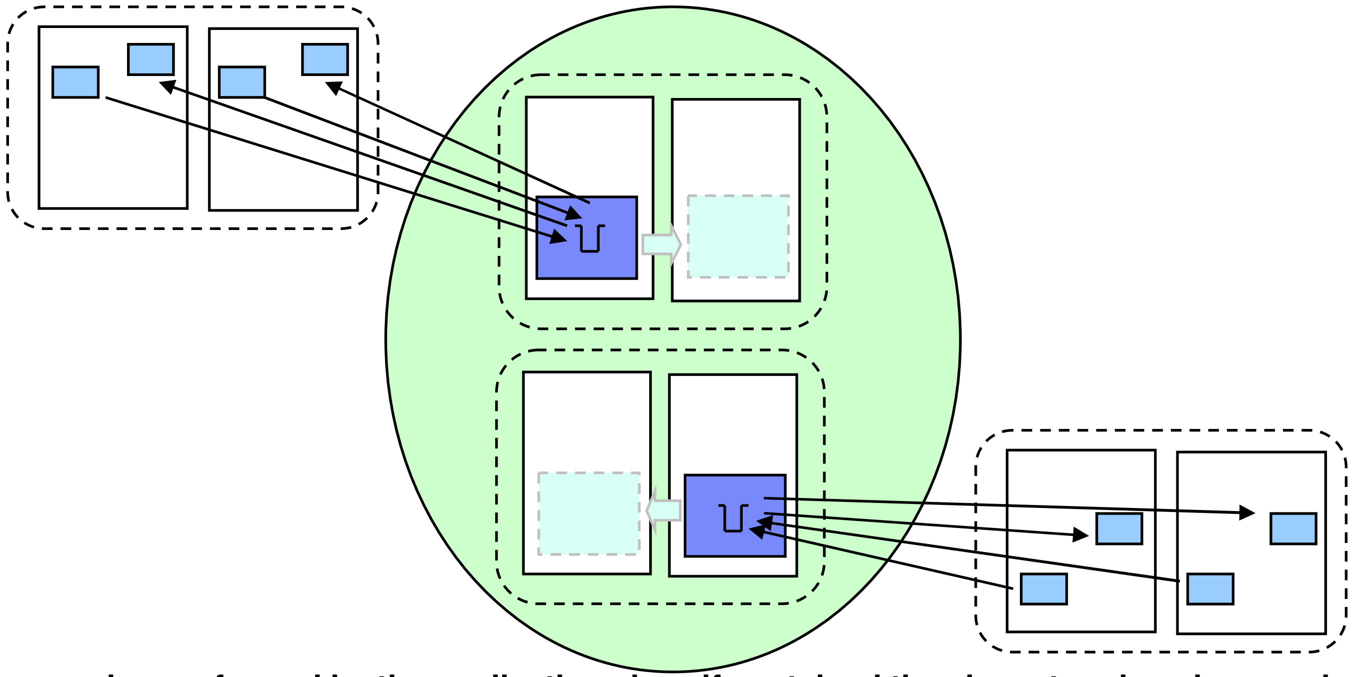
- **The previous cluster configuration can be scaled up to multiple application server clusters whilst keeping a single ME bus member.**
  - This allows further scaling of the applications where necessary, but maintains a simplistic messaging topology.



- **If the messaging traffic is too high the single ME may become a bottleneck**

Multiple bus member bus

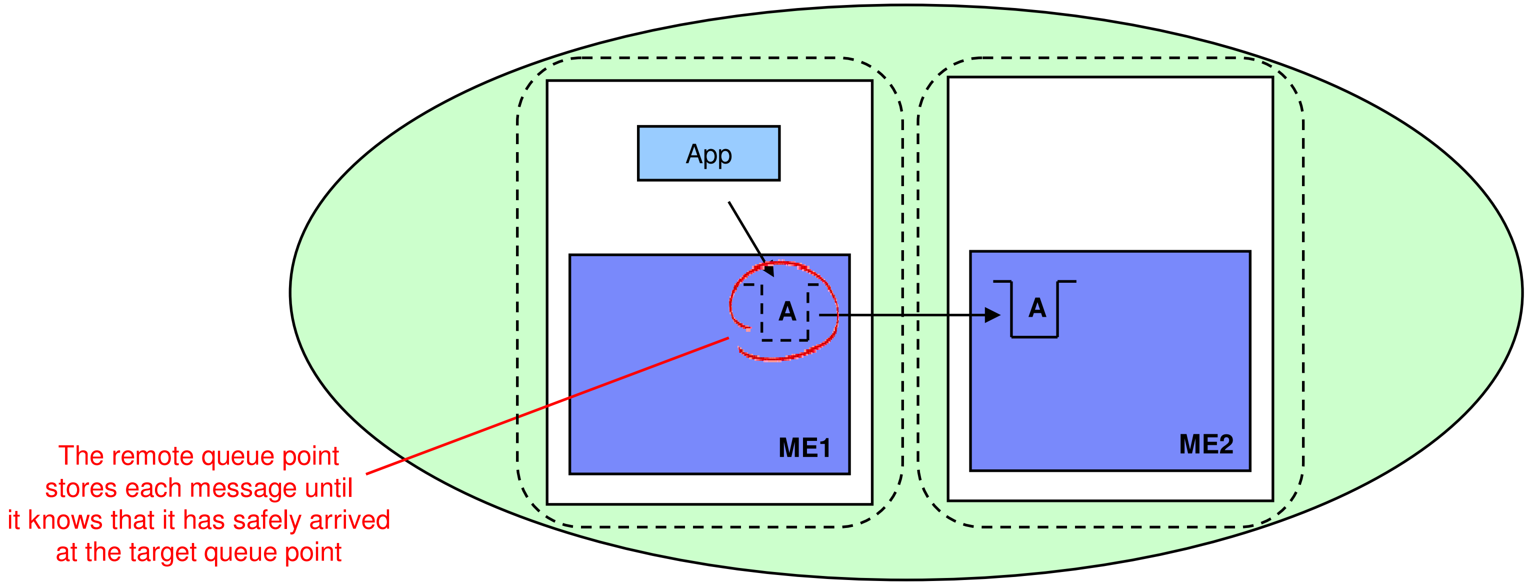
- **If the messaging traffic is expected to be too great for a single ME (hence JVM), multiple bus members should be considered where messaging traffic can be suitably divided (for example per application or queue).**



- **When the messaging performed by the applications is self contained the above topology is very similar at runtime to any single ME topology.**
- **However, once applications on different servers start to communicate with each other using messages, the runtime starts to change…**

Multiple bus member bus - sending

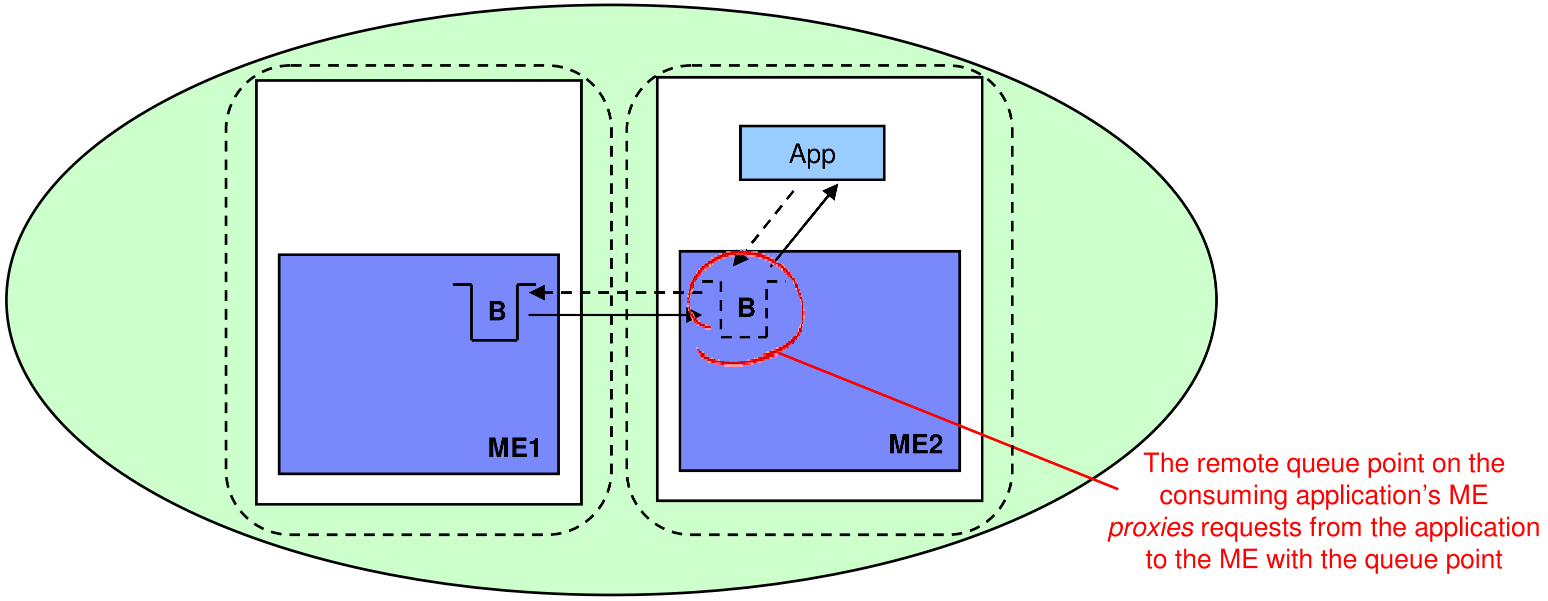
- **When messages are sent they are initially** *stored* **on the ME that the sending application is connected to *(an application connects to a* single *ME in the bus)*.**
  - Messages are initially stored on a ***Remote Queue Point*** ( )
- **The messages are then** *forwarded* **to an ME where a suitable Queue Point exists.**
- **Once received, the original copy of the message is deleted**
- **This is referred to as** *store and forward***.**
- **Store and forward can be beneficial as it allows message production to continue while a Queue Point's ME is unavailable.**


App
A
A
ME1
ME2
The remote queue point stores each message until it knows that it has safely arrived at the target queue point

- **However, this results in additional potential places where messages may become blocked for various reasons, complicating general problem determination.**

Multiple bus member bus - receiving

- **When consuming from a Queue Point that is not located on the same ME that the application is connected to, a process similar to *store and forward* is occurring, this is generally referred to as** *remote get.*
  - **The connected ME (ME2) requests a message from the ME with the Queue Point (ME1)**
  - **The ME1 selects a message, persistently locks it for the ME2 and sends a copy of the message to ME2**
  - **The consuming application receives the message, processes it and commits the receive.**
  - **The ME2 persists the act of committing the message and asynchronously informs ME1 of the commit.**
  - **Asynchronously, ME1 deletes the message and ME2 removes the commit state.**


App
B
B
ME1
ME2
The remote queue point on the consuming application's ME *proxies* requests from the application to the ME with the queue point

- **Unlike store and forward, remote get requires the Queue Point's ME to be available for messages to be consumed**

Multiple bus member bus

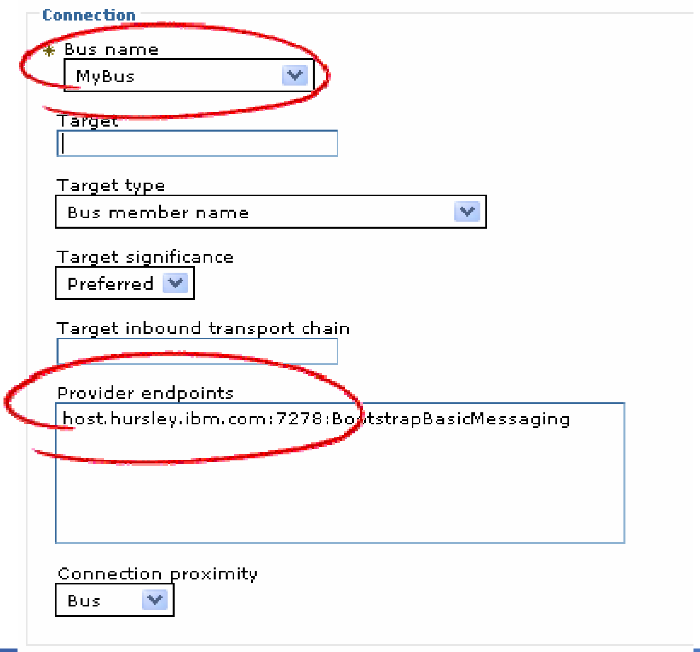
- **Ideally, cases of store and forward and remote get should be minimised to only where necessary.**
- **To achieve this, careful planning is obviously even more important**
  - Typically, messaging applications will ***randomly*** connect to any available messaging engine in the bus.
    - The only default behaviour that overrides this is when an application connects to a bus that has an ME running in the same server.
  - This can result in non-optimal paths for messages to take to and from the applications to the queues or subscriptions.
    - This does not just cost in performance, but manageability and serviceability of the system due to the unpredictable nature of connections and the resulting variable message routing.
  - The general rule is to connect directly to the bus member that owns the Queue. And if you really have to choose between a producing application or a consuming application, connect the consumers directly to the Queue Point’s ME and allow the producer to store and forward.

- The answer is to always *target* connections and activation specifications.

Multiple bus member bus - targeting

- **When JMS applications connect to a bus the minimum information required is the name of the *bus*.**
  - If the application is not running in a server, e.g. the client container, at least one ***provider endpoint*** is also needed to bootstrap to a bootstrap bus member server (a server that is a bus member or one that has the SIB service enabled).
  - The provider endpoint chosen does **not** automatically influence the choice of ME chosen for the connection, it is simply used as a bootstrap.

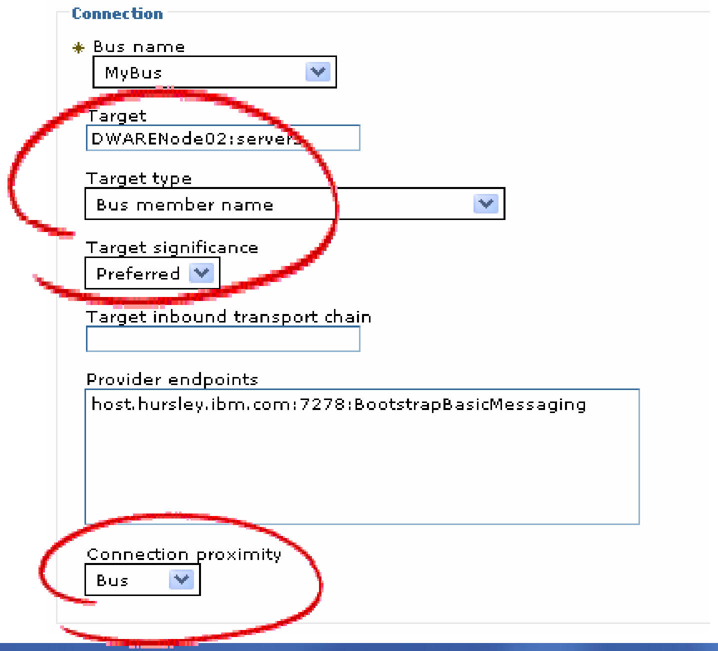
- **With only the above configured, a JMS connection will be made to *any* ME in the bus, based on WAS WLM logic.**


Connection

* Bus name	MyBus
Target	
Target type	Bus member name
Target significance	Preferred
Target inbound transport chain	
Provider endpoints	host.hursley.ibm.com:7278:BootstrapBasicMessaging
Connection proximity	Bus

Multiple bus member bus - targeting

- **By *targeting* the connection, varying levels of control over the actual connection made into the bus can be defined.**
  - **Target type**
    - The type of entity that is to be targeted
    - ***Bus member*** - any available ME in that bus member.
    - ***Messaging engine*** - a specific ME.
    - ***Custom messaging engine group*** - a manually configured set of MEs.
  - **Target significance**
    - ***Required*** - The connection will ***only*** be made to the specified target.
    - ***Preferred*** - If the specified target is available it will be used, otherwise any other available ME in the bus will be used.
  - **Target**
    - The name of the above target
  - **NB:** When using connection pooling, connections in the pool to non-preferred targets may still exist even once the preferred target becomes available.
    - This can be minimised by configuring the connection pool settings (e.g. by reducing the ***Aged timeout).***
- ***Connection proximity***
  - This provides an additional level of control, based on the physical location relative to the application or bootstrap server
    - E.g. within the same server or cluster, or on the same host.


Connection

Bus name: MyBus

Target: DWARENode02:server

Target type: Bus member name

Target significance: Preferred

Target inbound transport chain:

Provider endpoints: host.hursley.ibm.com:7278:BootstrapBasicMessaging

Connection proximity: Bus

Multiple bus member bus - targeting

- **For example, when using a connection to** send **messages to a queue the connection could be configured to** *prefer* **the** *bus member* **that owns that queue. This would ensure a direct send of messages while the bus member is available but also allow store and forward of the messages from another ME if it should become unavailable.**
  - This maintains HA support whilst optimising flow of messages under normal circumstances.

- **If using a connection to** receive **messages,** *requiring* **the** *bus member* **where the queue resides makes sense as messages can only be received while an ME in that bus member is available, so nothing is gained by being able to connect elsewhere.**
  - This optimises and simplifies the flow of messages to the application.

- **As for connections for consuming applications, when configuring an** Activation Specification **for an MDB, unless the application is co-located with the bus member with the queue, that** *bus member* **should be a** *required* **target.**

Multi-bus topologies

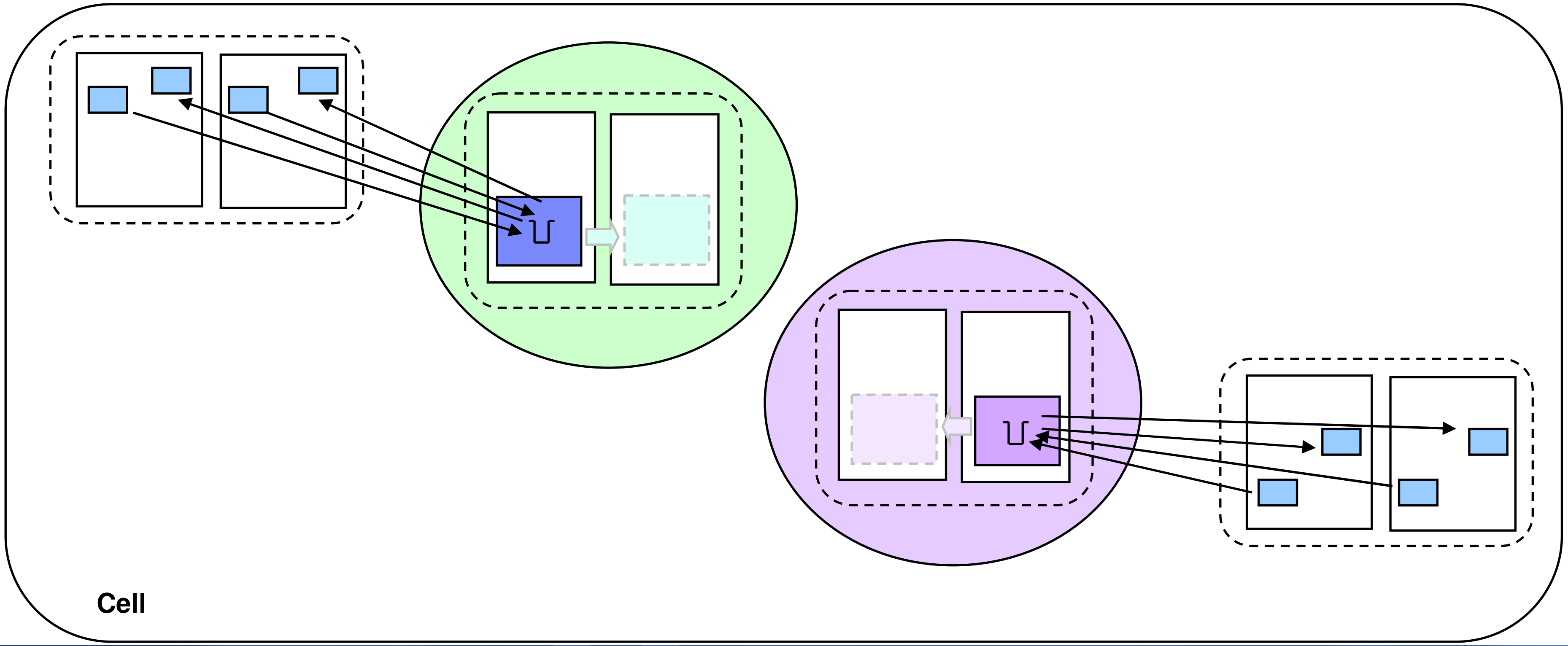
- **There are two main reasons for having a multi-bus SIB topology:**

1. Because, for simplicity or isolation, you want to scope unrelated messaging applications within a cell.

2. Because related messaging applications span multiple WAS cells and a bus is restricted to a single cell.

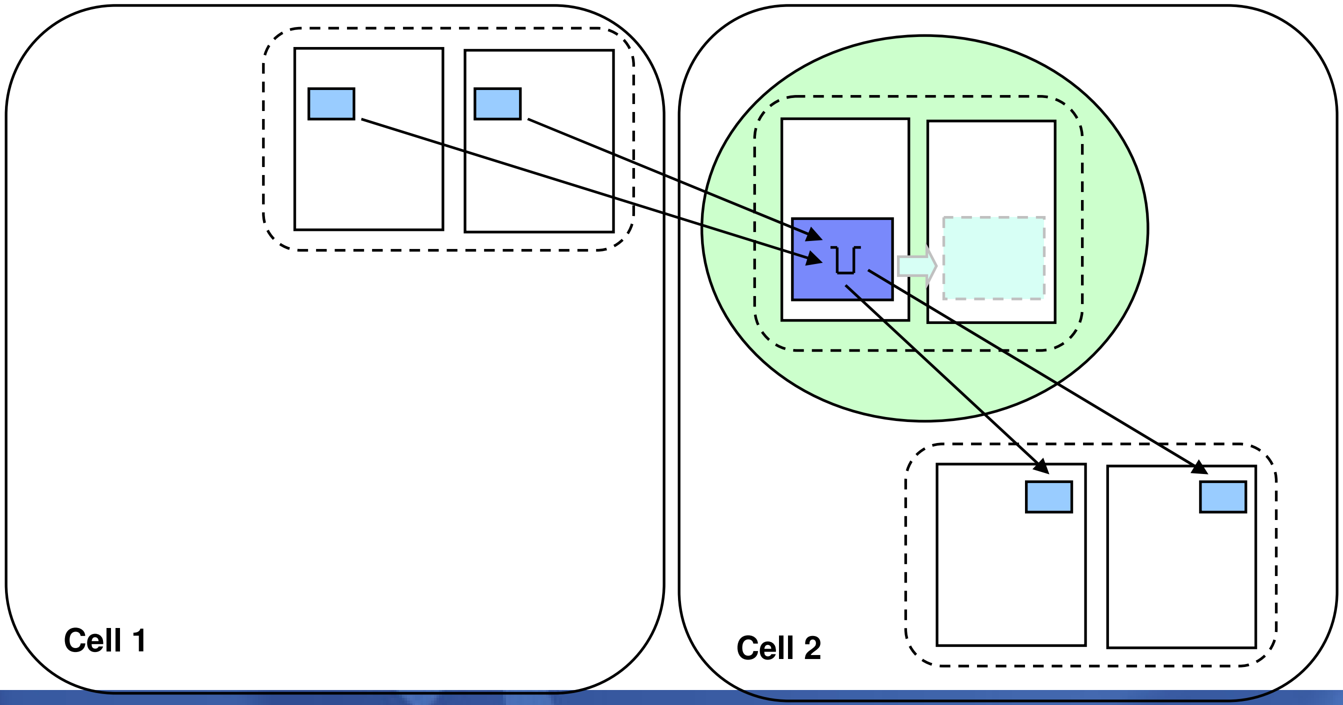
Multi-bus topologies (1)

- **If you’re using multiple buses to scope unrelated messaging applications, no inter-bus linkage will be required.**
- **This doesn’t change any configuration or runtime aspects of the multiple buses, they exist in simply ignorance of each other, despite potentially running in the same cell.**



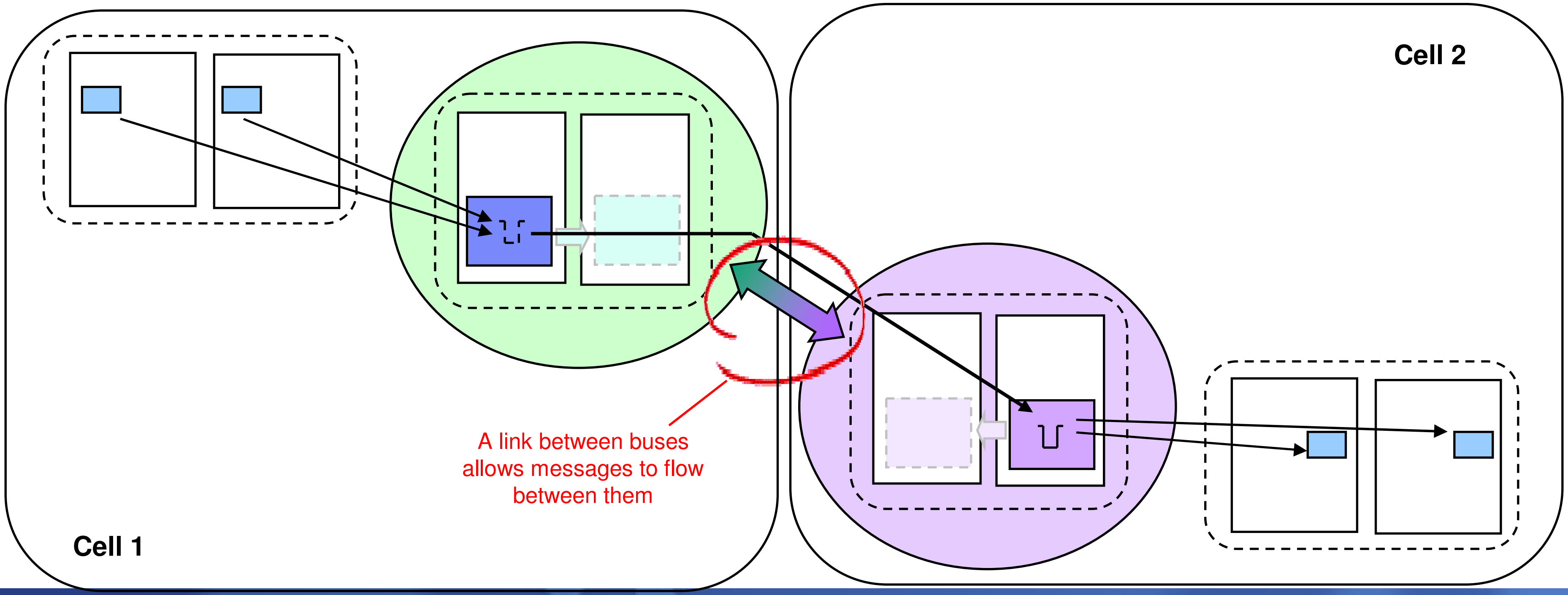
Multi-bus topologies (2)

- **If messaging spans cells, the first option to consider is to have a single bus in one of the cells and for the messaging application to connect across-cells to access the messages.**
  - In V6 and V6.1 MDB's require use of a Core Group Bridge between cells to allow this.
  - V7 makes it possible for an MDB's Activation Specification to specify a provider endpoint in a different cell from the MDB application.



Multi-bus topologies (2)

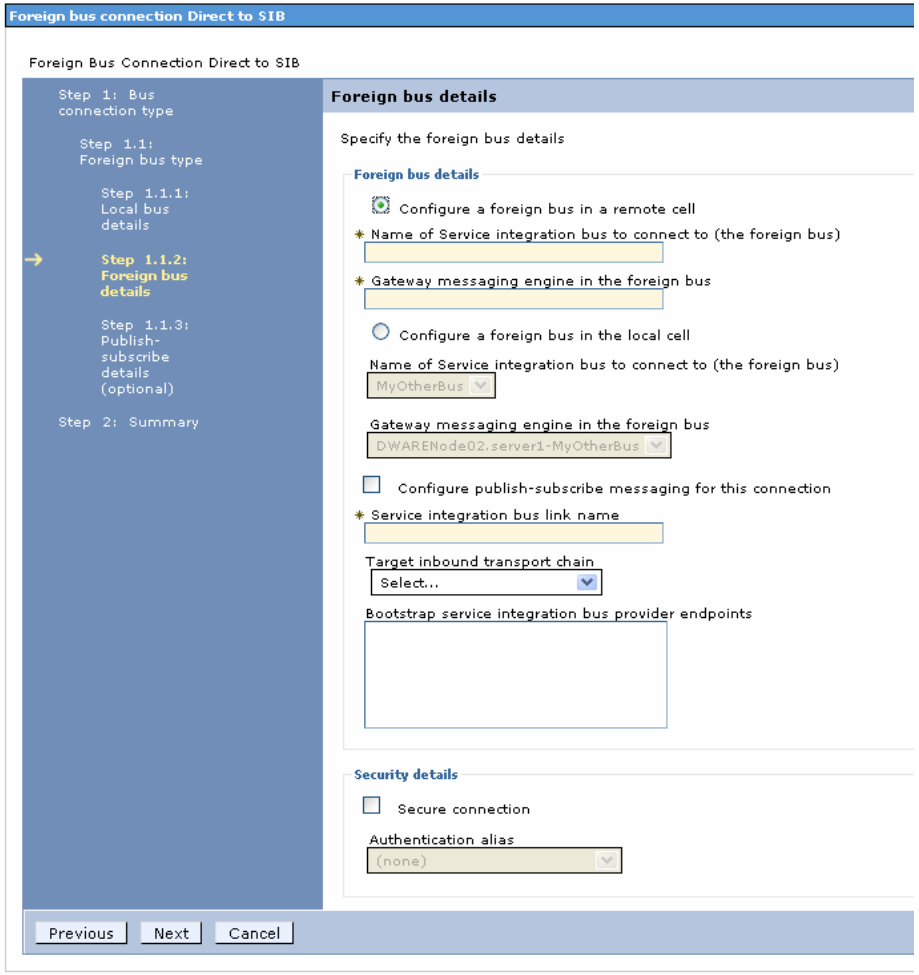
- **If a single bus is not appropriate, multiple linked buses will be required.**
  - This introduces more complexity, both for the initial configuration and the manageability of the resultant system.
  - Messages are *stored and forwarded* between buses
  - Messages cannot be consumed remotely from destinations in a different linked bus



Multi-bus topologies (2)

- **Prior to V7, the separate configuration and management of** *foreign buses* **and** *SIB links* **is required to link two buses together.**

- **The V7 admin console has introduced the concept of** *foreign bus connections* **and a single configuration wizard to create the underlying foreign bus and link that are required.**


Foreign bus connection Direct to SIB

Foreign Bus Connection Direct to SIB

Step 1: Bus connection type
Step 1.1: Foreign bus type
Step 1.1.1: Local bus details
→ Step 1.1.2: Foreign bus details
Step 1.1.3: Publish-subscribe details (optional)
Step 2: Summary

Foreign bus details

Specify the foreign bus details

Foreign bus details

[✓ Configure a foreign bus in a remote cell]

* Name of Service integration bus to connect to (the foreign bus)

* Gateway messaging engine in the foreign bus

Configure a foreign bus in the local cell

Name of Service integration bus to connect to (the foreign bus): MyOtherBus

Gateway messaging engine in the foreign bus: DWARENode02.server1-MyOtherBus

☐ Configure publish-subscribe messaging for this connection

* Service integration bus link name

Target inbound transport chain: Select...

Bootstrap service integration bus provider endpoints

Security details

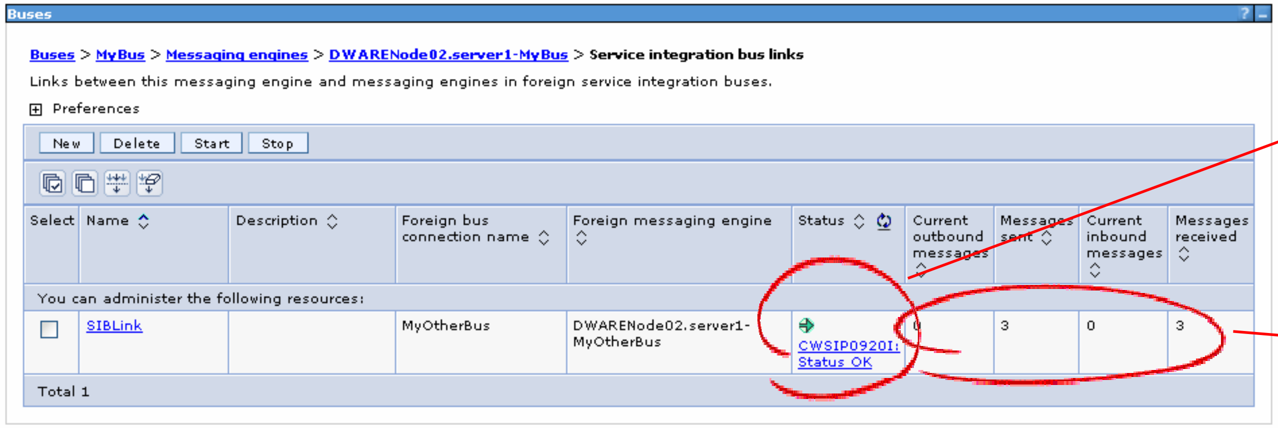
☐ Secure connection

Authentication alias: (none)

Multi-bus topologies (2)

- In V6 and V6.1 messages in transit between MEs within the same bus are visible and controllable using *Remote Queue Points*. However, messages in transit between buses are not visible. This is a problem when monitoring a multi-bus system.
- V7 has made messages in transit between buses (both SIB and WMQ) visible and controllable.
  - The runtime information provides the overall state of a foreign bus connection, readily identifying potential stoppages and problems and allowing drilling down to the reasons for problems.

Multi-bus topologies (2)



Buses

Buses > MyBus > Messaging engines > DWARENode02.server1-MyBus > Service integration bus links

Links between this messaging engine and messaging engines in foreign service integration buses.

⊞ Preferences

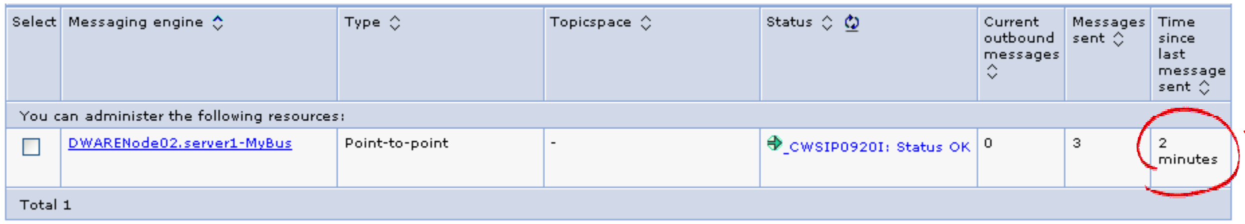
New Delete Start Stop

Select	Name	Description	Foreign bus connection name	Foreign messaging engine	Status	Current outbound messages	Messages sent	Current inbound messages	Messages received
You can administer the following resources:									
☐	SIBLink		MyOtherBus	DWARENode02.server1-MyOtherBus	CWSIP0920I: Status OK	0	3	0	3
Total 1									

An overall status of the link is shown at each level

A few messages have successfully been sent and received on this connection.
None are currently queued for transmission.

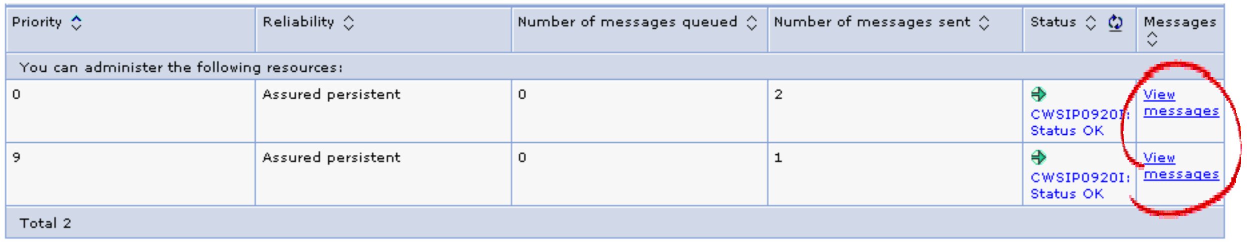
The whole link



Select	Messaging engine	Type	Topicspace	Status	Current outbound messages	Messages sent	Time since last message sent
You can administer the following resources:							
☐	DWARENode02.server1-MyBus	Point-to-point	-	CWSIP0920I: Status OK	0	3	2 minutes
Total 1							

No messages have been transmitted for two minutes.

All link transmitters for the link



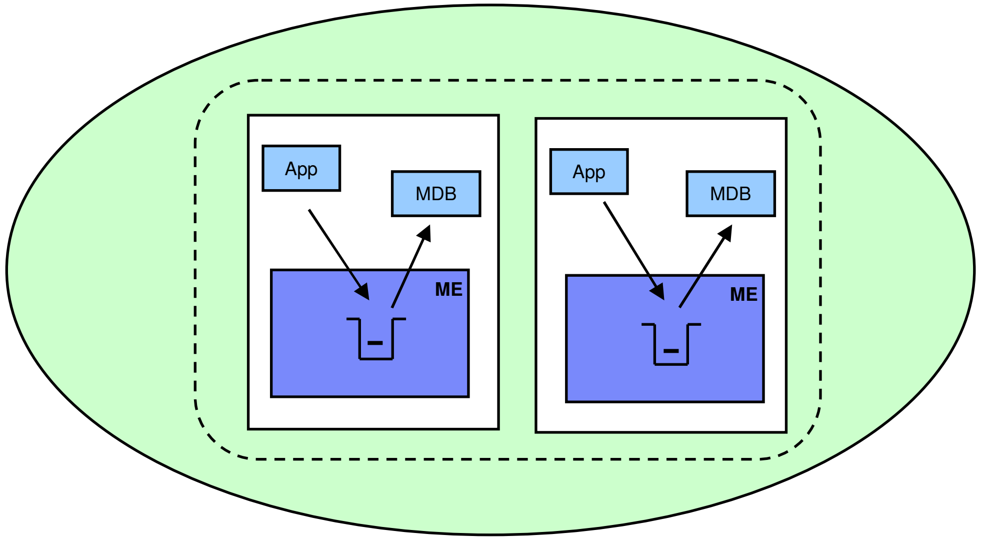
Priority	Reliability	Number of messages queued	Number of messages sent	Status	Messages
You can administer the following resources:					
0	Assured persistent	0	2	CWSIP0920I: Status OK	View messages
9	Assured persistent	0	1	CWSIP0920I: Status OK	View messages
Total 2					

If messages were queued up for transmission we'd be able to view them here.

An individual transmitter

Scalable cluster bus members

- **The one factor not yet addressed is scalability of messaging for a single application, or a particular flow of messages.**
- **To achieve this we need to parallelise the messaging work, not just at the application layer but also in the messaging layer.**
- **This is through use of** *multiple MEs*, **using a** *scalable* **cluster bus member**



Scalable cluster bus members

- **This adds an extra dimension to the complexity of the messaging.**
  - Up until this point, a queue could be thought of as a single *pot* of messages, located in a single place, i.e. the *Queue Point*.
  - Once a bus member contains multiple MEs ***any*** queue defined to that bus member will be split (or partitioned) across all MEs in the bus member, resulting in multiple Queue Points.
  - Each message sent to the queue will have a Queue Point chosen for it by the system and that is where the message will be stored.
  - Prior to V7, to be able to consume that message, a consuming application must be attached to the queue point containing the message.

- **For these reasons, application design has to be very carefully factored into these topologies.**

Scalable cluster bus members

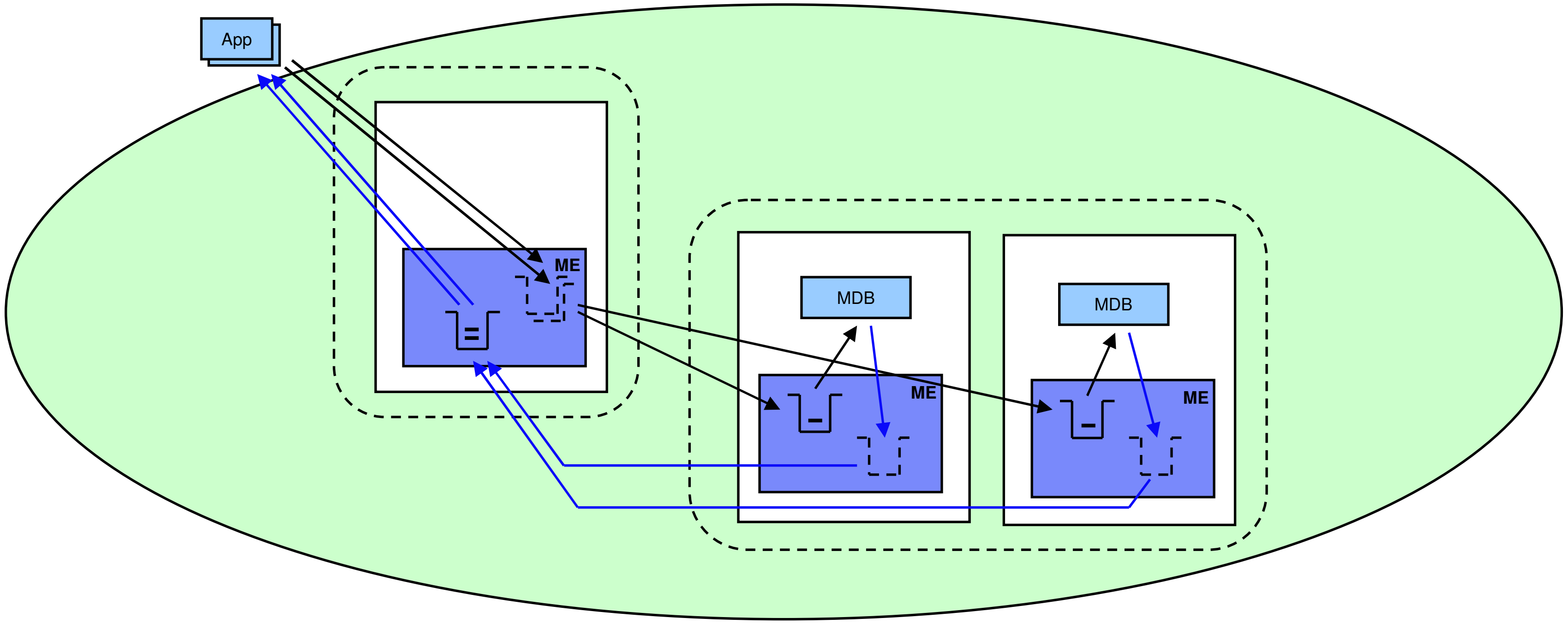
- **All queues defined to a scalable bus member will have multiple queue points*, so to be able to correctly use scalable bus members the following factors need to be carefully considered.**
  - Message **producing** applications connected to an ME in the queue's bus member will ***prefer*** to send all messages to a queue point on that ME. When connected to an ME in a different bus member, all messages will be ***workload balanced*** across all available queue points. †
  - Typically, message **consuming** applications (including MDBs) only consume from a ***single queue point*** for the lifetime of the consumer (e.g. the JMS MessageConsumer), so any application running a single instance will not necessarily see all the messages (e.g. an application waiting for a specific reply message).†
  - Typically, message **consuming** applications connected to an ME in the queue's bus member will only consume messages from the queue point on that ME. When connected to an ME in a different bus member the system will ***randomly*** pick one of the queue points at the time the consumer is created and **only** consume messages from that single queue point. †
  - Message order can no longer be guaranteed due to the workload balancing of messages across queue points. †

* ***Dynamically created temporary queues will only have a single queue point on the ME that the creating application is connected to (and also durable subscriptions when using publish/subscribe).***

† **V7 introduces the option to modify this behaviour**

Scalable cluster bus members

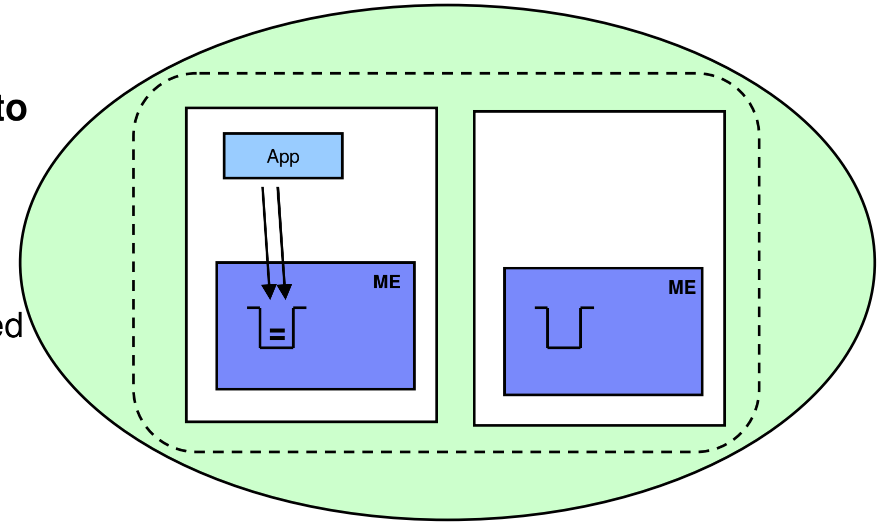
- **The previous reasons obviously limit the scenarios where scalable bus members can be used, and is very dependent on the relative locations of the application connections to achieve the correct behaviour.**
- **A *classic* example of a scalable bus member prior to V7 would be where a co-located MDB services a partitioned queue and any replies are sent to non-partitioned queue(s) on different bus members.**
  - The non-partitioned queue is required to ensure each reply message goes to where the consumer is waiting.



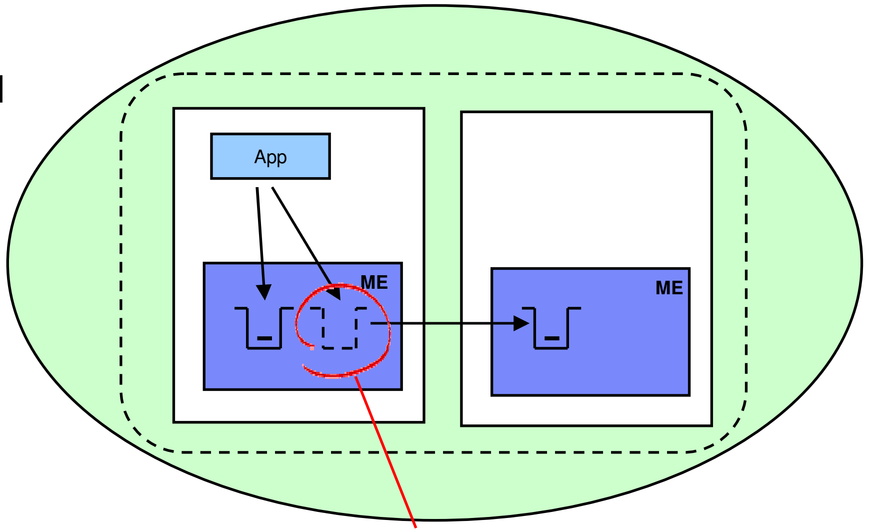
- **V7 provides a host of new options to allow finer control of how messaging applications interact with multi-partition queues…**

Local queue point preference in V7

- **As described previously, whether messages are workload balanced across all queue points or sent to the local queue point (if possible) is dependent on the location of the message producing JMS application's connection.**
  - If an available queue point exists on the locally connected ME, messages will be sent there (as the top illustration shows), otherwise each individual message is workload balanced across all available queue points.

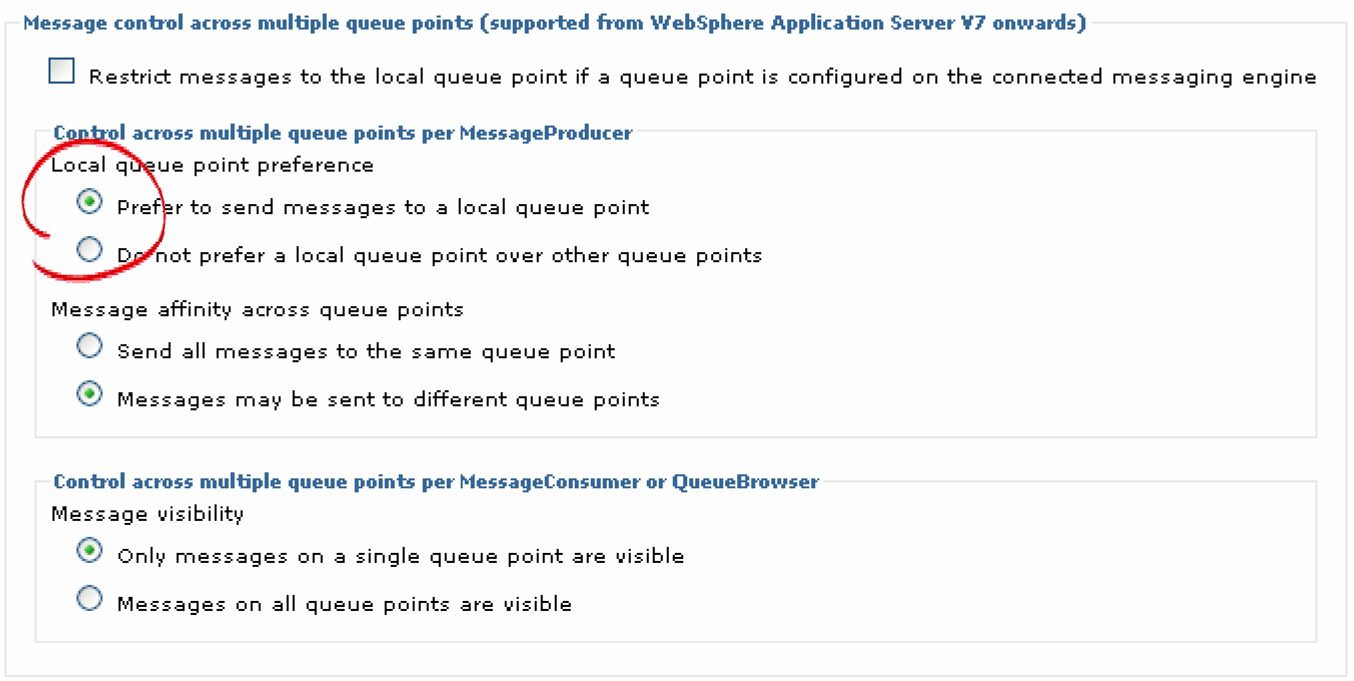


- **V7 allows this default behaviour to be overridden by a JMS application, allowing applications connected directly to an ME with a queue point to still workload balance their messages across all available queue points (as the bottom illustration shows).**
  - Inter-bus links can also now operate in this way.



Store and forward is used to enable workload balancing

Local queue point preference in V7



Message control across multiple queue points (supported from WebSphere Application Server V7 onwards)

☐ Restrict messages to the local queue point if a queue point is configured on the connected messaging engine

Control across multiple queue points per MessageProducer

Local queue point preference

- ● Prefer to send messages to a local queue point
- ○ Do not prefer a local queue point over other queue points

Message affinity across queue points

- ○ Send all messages to the same queue point
- ● Messages may be sent to different queue points

Control across multiple queue points per MessageConsumer or QueueBrowser

Message visibility

- ● Only messages on a single queue point are visible
- ○ Messages on all queue points are visible

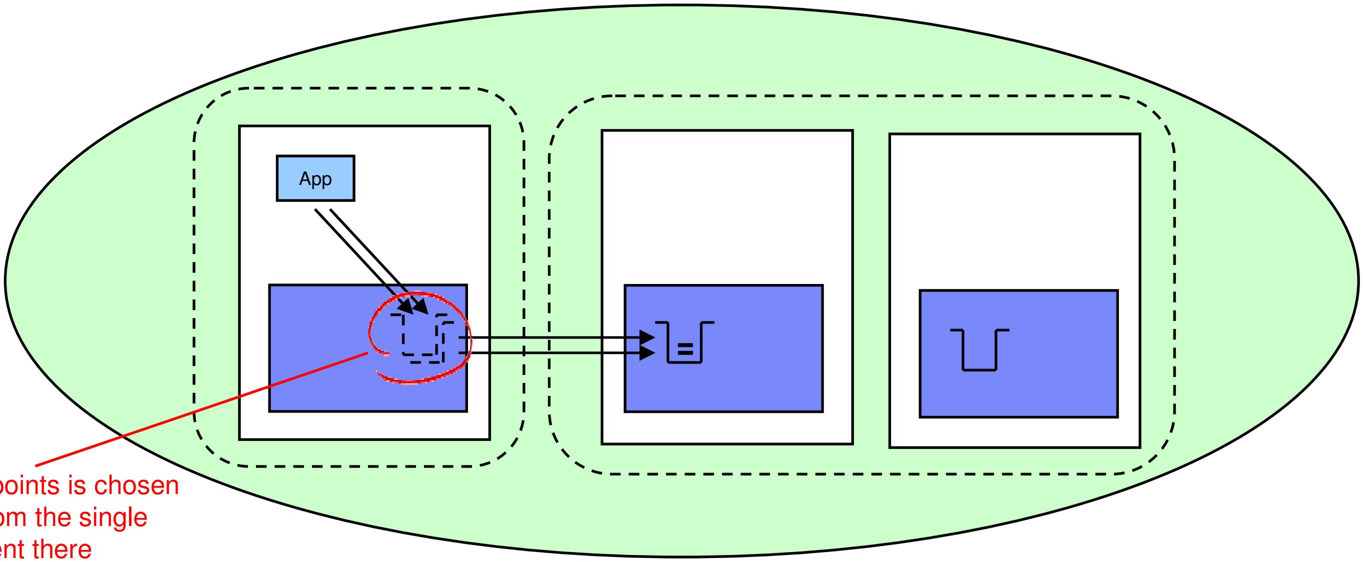
JMS Queue configuration

Message affinity in V7

- **Prior to V7, when a JMS producer sends multiple messages to a multi-queue point queue it is not possible to ensure all messages will go to the same queue point.**
  - Even when connected locally to an ME with a queue point, messages may be re-directed to another queue point if the local one becomes unavailable (for example, full).
- **This prevents guaranteeing any** *ordering* **of messages produced by a single producer to a multi-queue point queue.**

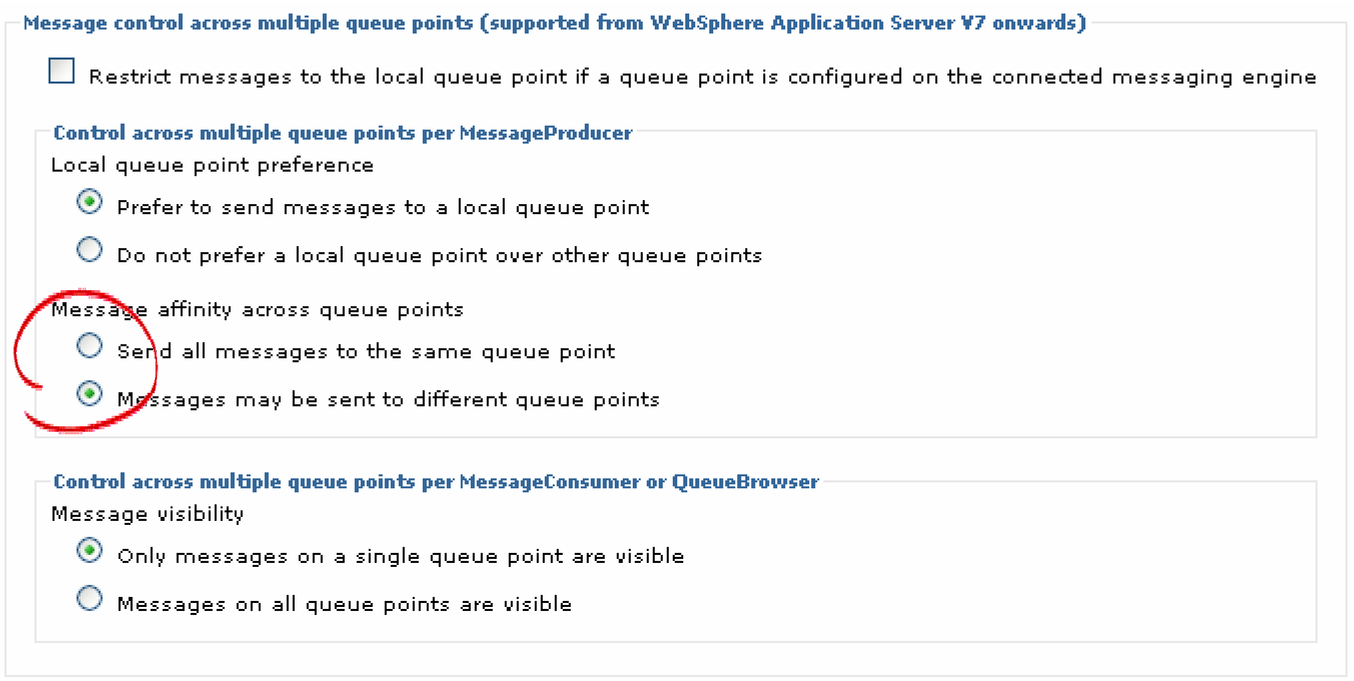
Message affinity in V7

- **V7 allows a producer to specify that all messages sent under the same JMS MessageProducer must go to the same queue point.**
  - If local queue point preference disabled or not applicable then that single queue point is *randomly* chosen by the system (using WAS WLM).
  - If some messages can not be delivered to the chosen queue point (for example it becomes full), the messages will be stored in transit or a failure indicated to the application, just as if a single-queue point queue was being used.



One of the two queue points is chosen and all messages from the single producer are sent there

Message affinity in V7



Message control across multiple queue points (supported from WebSphere Application Server V7 onwards)

☐ Restrict messages to the local queue point if a queue point is configured on the connected messaging engine

Control across multiple queue points per MessageProducer

Local queue point preference

- [✓ Prefer to send messages to a local queue point]
- Do not prefer a local queue point over other queue points

Message affinity across queue points

- Send all messages to the same queue point
- [✓ Messages may be sent to different queue points]

Control across multiple queue points per MessageConsumer or QueueBrowser

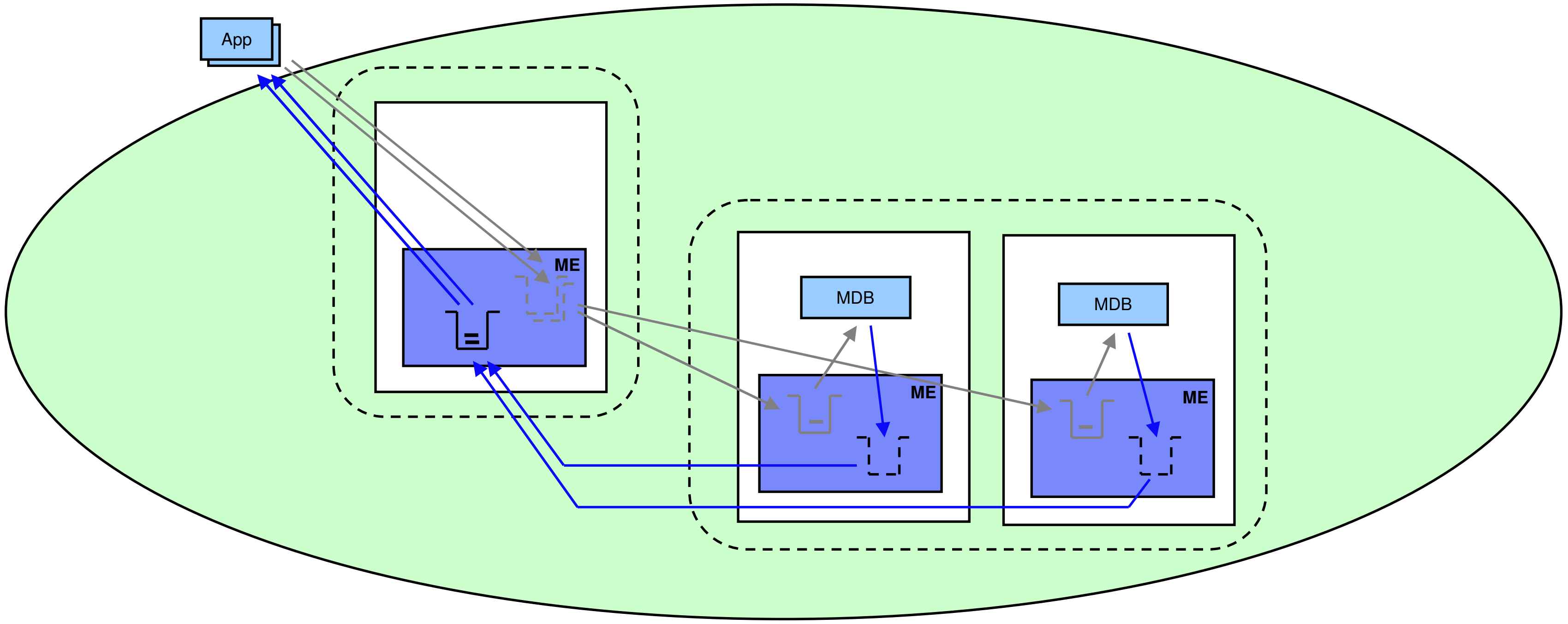
Message visibility

- [✓ Only messages on a single queue point are visible]
- Messages on all queue points are visible

JMS Queue configuration

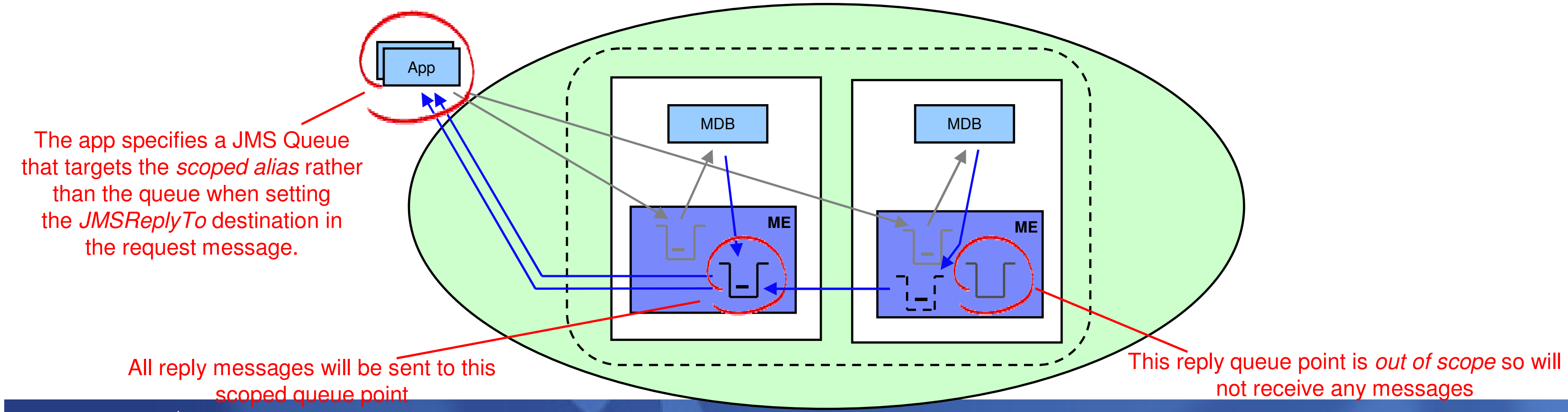
Scoped alias queues in V7

- A common problem with scalable bus members is that *all* queues defined to the bus member are partitioned across all MEs in the bus member, not just the ones that need to be partitioned for scalability reasons.
- If messages on those queues need to be seen by a specific consumer (e.g. reply messages), prior to V7, an additional, single-ME, bus member would be required.



Scoped alias queues in V7

- **V7 introduces the concept of** ***scoped alias destinations*****.**
  - These allow a multi-queue point queue to be scoped down to a subset of its queue points, e.g. one.
  - This allows a mixture of multi-queue point queues and single-queue point alias queues being defined on a single scalable bus member.
    - In keeping with the targeting advice, the requesting application should try to connect to the ME with the scoped queue point
  - It also allows each individual queue point of a queue to be directly addressable as individual JMS Queues, allowing work to be statically shared across those queue points.
  - This is achieved by creating a different scoped alias for each queue point.



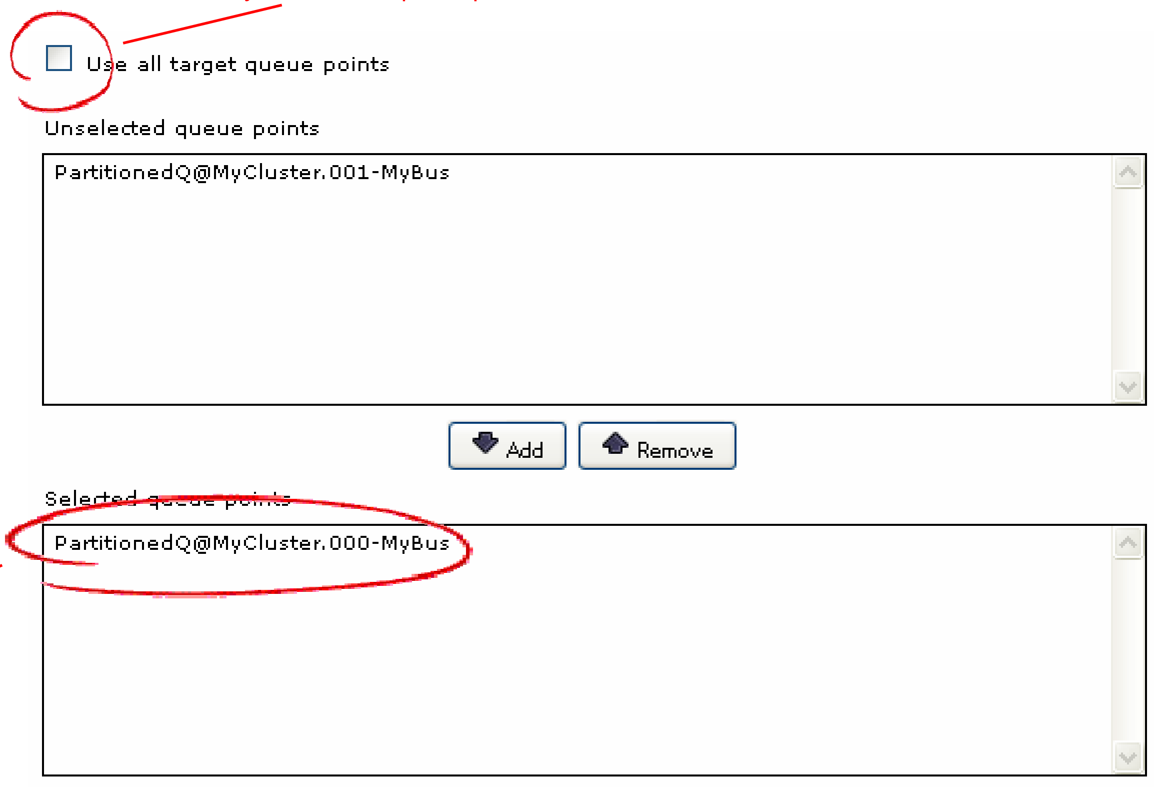
The app specifies a JMS Queue that targets the *scoped alias* rather than the queue when setting the *JMSReplyTo* destination in the request message.

All reply messages will be sent to this scoped queue point

This reply queue point is *out of scope* so will not receive any messages

Scoped alias queues in V7

By default, all queue points are included in the alias



☐ Use all target queue points

Unselected queue points

PartitionedQ@MyCluster.001-MyBus

Selected queue points

PartitionedQ@MyCluster.000-MyBus

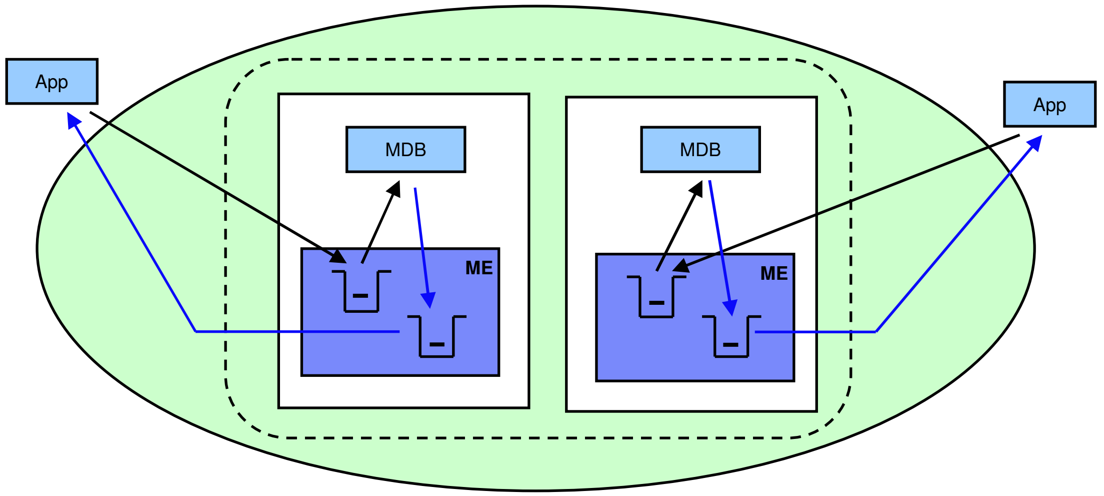
Individual queue points can be included in the scope of the alias

Alias destination configuration

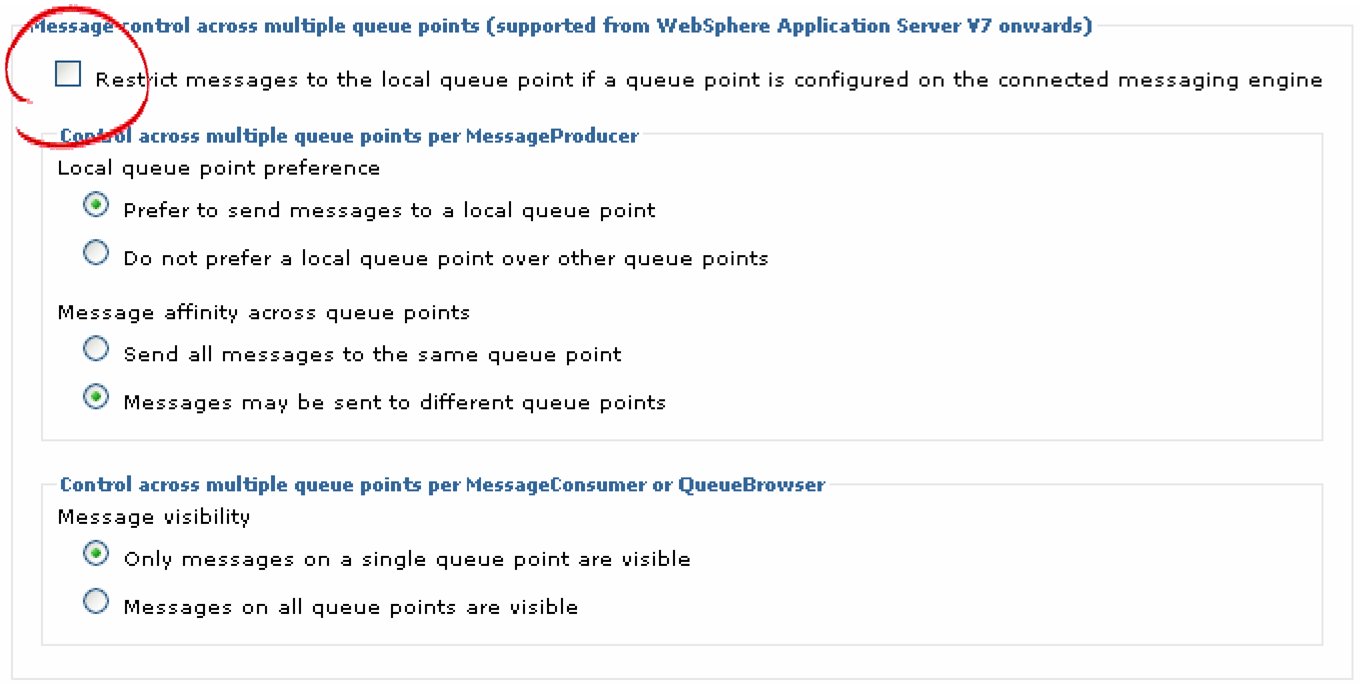
(this is only visible when viewing the configuration of an alias that directly targets a partitioned queue, not in the destination creation wizard)

Dynamic scope to local routing in V7

- **Configuring one or more scoped aliases is a *static* solution**
  - This can cause complications in an HA environment.
- **It is possible for a JMS application to *dynamically* scope all its messages to the queue points on the *connected* ME, preventing *any* cross-ME message traffic (even in the case of unavailable queue points).**
  - This applies when sending messages
  - Consuming messages
  - And the routing of any messages sent by another application, that is sending a reply message to this application's reply queue.
    - The 'local' is local to the application sending the request, not the application sending the reply.



Dynamic scope to local routing in V7



Message control across multiple queue points (supported from WebSphere Application Server V7 onwards)

☐ Restrict messages to the local queue point if a queue point is configured on the connected messaging engine

Control across multiple queue points per MessageProducer

Local queue point preference

- [✓ Prefer to send messages to a local queue point]
- Do not prefer a local queue point over other queue points

Message affinity across queue points

- Send all messages to the same queue point
- [✓ Messages may be sent to different queue points]

Control across multiple queue points per MessageConsumer or QueueBrowser

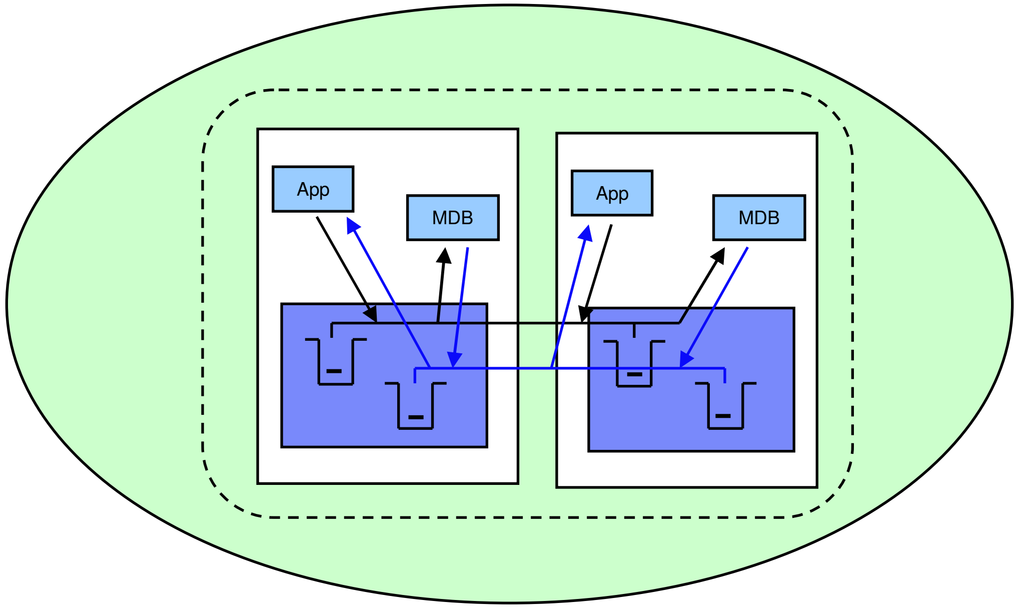
Message visibility

- [✓ Only messages on a single queue point are visible]
- Messages on all queue points are visible

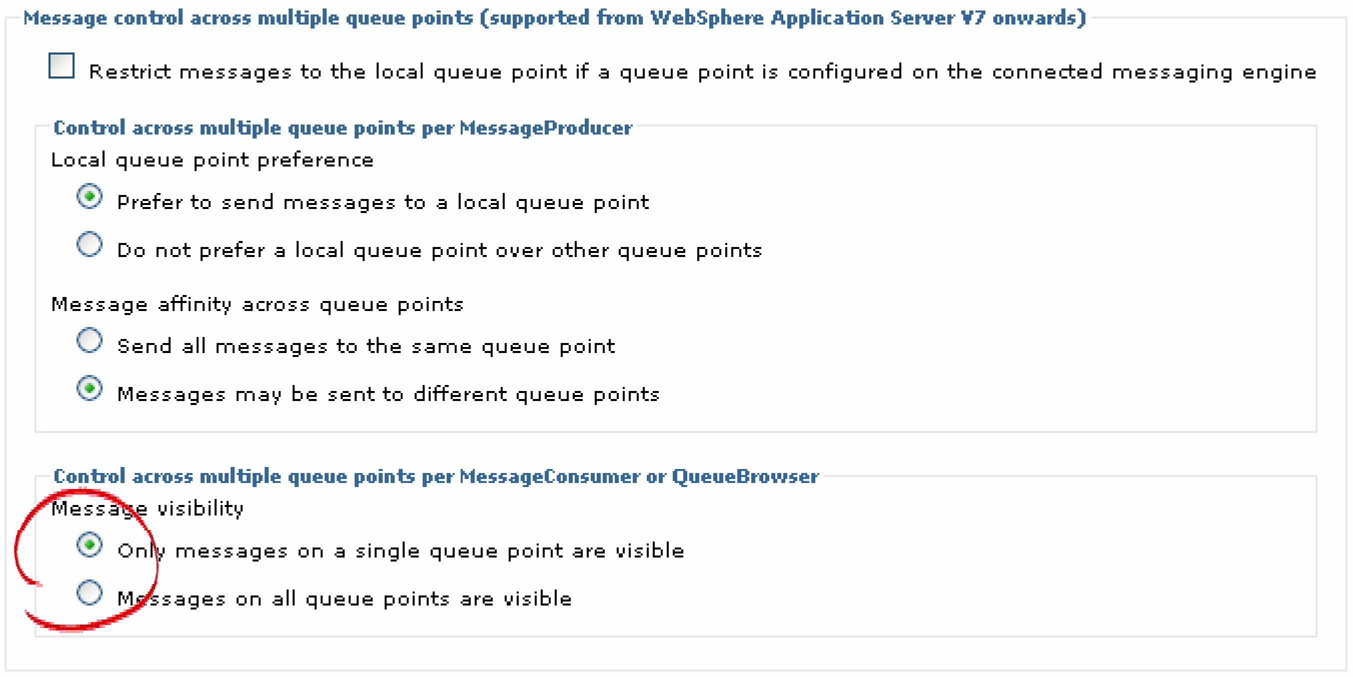
JMS Queue configuration

Message visibility in V7

- **By default a JMS consumer is only able to consume messages stored on the queue point that it is connected to.**
- **V7 adds the option to make all messages across all queue points of a queue** *visible* **to a consumer and therefore consumable from a single consumer.**
- **To the application, this makes multiple queue points appear as a single queue point again**
  - ***So you don't need to be so careful about your application and topology design now!...maybe?***



Message visibility in V7


Message control across multiple queue points (supported from WebSphere Application Server V7 onwards)

☐ Restrict messages to the local queue point if a queue point is configured on the connected messaging engine

Control across multiple queue points per MessageProducer

Local queue point preference

- [✓ Prefer to send messages to a local queue point]
- Do not prefer a local queue point over other queue points

Message affinity across queue points

- Send all messages to the same queue point
- [✓ Messages may be sent to different queue points]

Control across multiple queue points per MessageConsumer or QueueBrowser

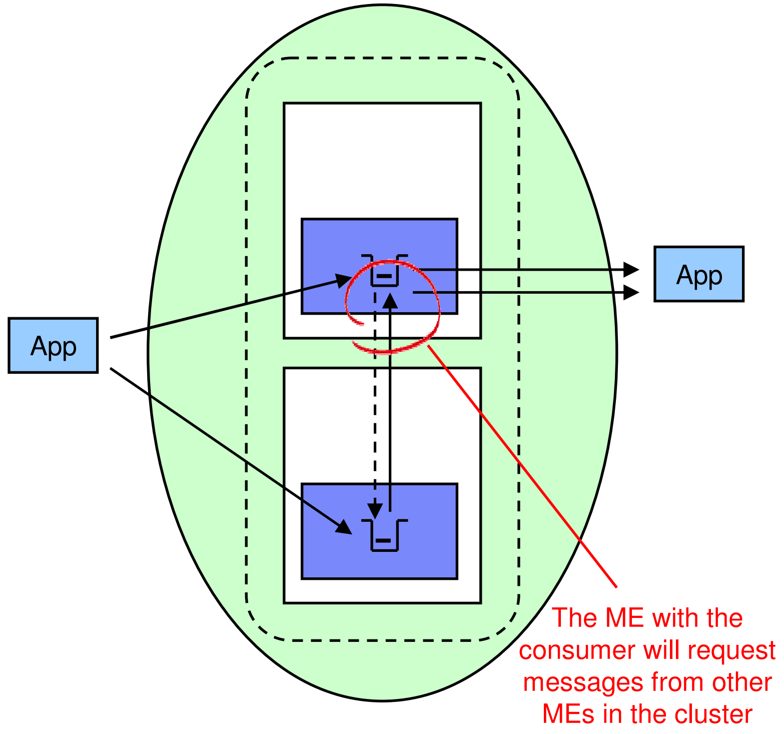
Message visibility

- [✓ Only messages on a single queue point are visible]
- Messages on all queue points are visible

JMS Queue configuration

Message visibility in V7

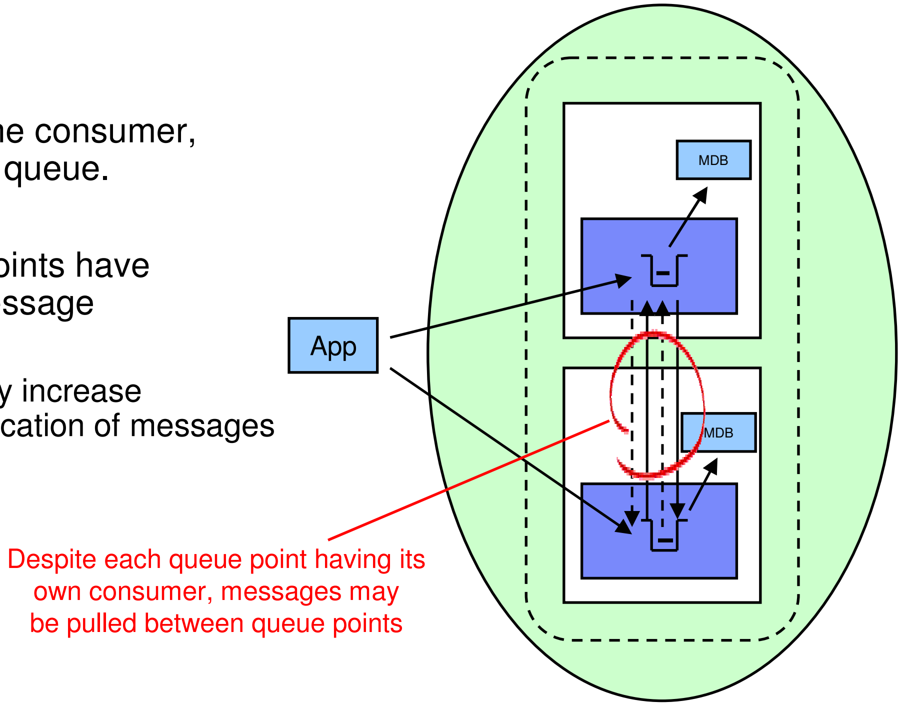
- **So that solves all the previous problems with scalable clusters…**
  - ***Doesn't it?***
- **Consider what the messages are now doing**
  - A consumer will still only connect to one ME and be attached to a single queue point.
  - The connected ME will do the job of *gathering* messages from other queue points when the consumer requests a message.
  - The gathering ME does not know which queue points have messages, which can result in it requesting messages from all queue points.
    - This uses the *remote get* protocol previously mentioned.
- **What is the benefit of this? Why was the queue partitioned in the first place?**
  - For scalability and performance?
- **What have you actually achieved?**
  - At its best, sideways shuffling of messages
  - At it's worst, workload balancing of messages followed by sucking them back into one place.


App
App
The ME with the consumer will request messages from other MEs in the cluster

Message visibility in V7

- **So when could you use this?**
  - Specific reply messages, when you can’t guarantee their location
    - For example, when a requesting application can disconnect between sending the request and receiving the reply (see later options for ways of minimising this).

- **So when shouldn’t you use this?**
  - When you have multiple instances of the consumer, all able to receive *any* message on the queue.
    - **E.g. An MDB.**
  - Why? Because even when all queue points have a share of the messages, sideways message gathering is still occurring.
    - This will impair performance and greatly increase the complexity when determining the location of messages as part of problem determination.


MDB
MDB
App
Despite each queue point having its own consumer, messages may be pulled between queue points

Other V7 improvements

- **Performance**
  - Focused on supporting larger messages.
- **Security administration**
  - SIB resource access control can now be administered through the admin console.
- **Auto-stop of MDBs**
  - Sequential message failures are detected, stopping an MDB before messages are moved to an exception destination.
- **WebSphere MQ interoperation**
  - Single bus interoperation with WMQ now on distributed platforms.
- **WebSphere MQ JMS provider**
  - Now runs as a JCA 1.5 resource adaptor.
- **…**

Summary

- **We've worked through the possible reasons for needing a more complex service integration bus messaging topology and some of the challenges that they can present.**

- **And provided an overview of *some* of the improvements made to the default messaging in WAS V7.**

References

- **Information Center topics of interest for V7**
  - –How a message-driven bean connects in a cluster
    - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/concepts/cjn_mdb_endpt_overview.html
  - –Service integration high availability and workload sharing configurations
    - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/concepts/cjt0007_.html
  - –Connecting buses
    - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/tasks/tjj0090_.html
  - –Workload sharing with queue destinations
    - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.pmc.nd.multiplatform.doc/concepts/cjt0014_.html

Questions

- **Thank you!!**