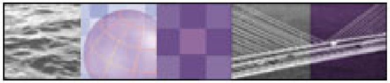
Introduction to WebSphere Platform Messaging (WPM)

WebSphere Training and
Technical Enablement



Unit Objectives

After completing this unit, you should be able to discuss:

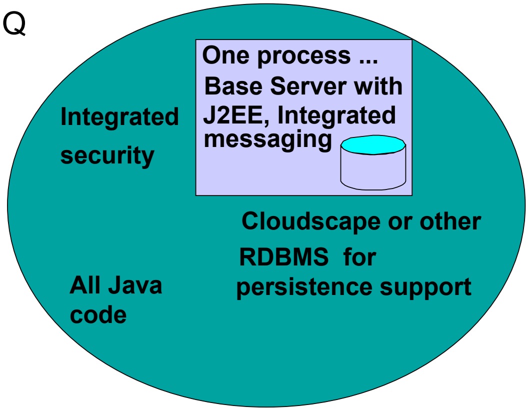
- Overview of WebSphere Messaging system
- Service Integration Bus Architecture and components
- Sample topologies
- Support for other JMS providers
- Summary

WebSphere Messaging – Big Picture

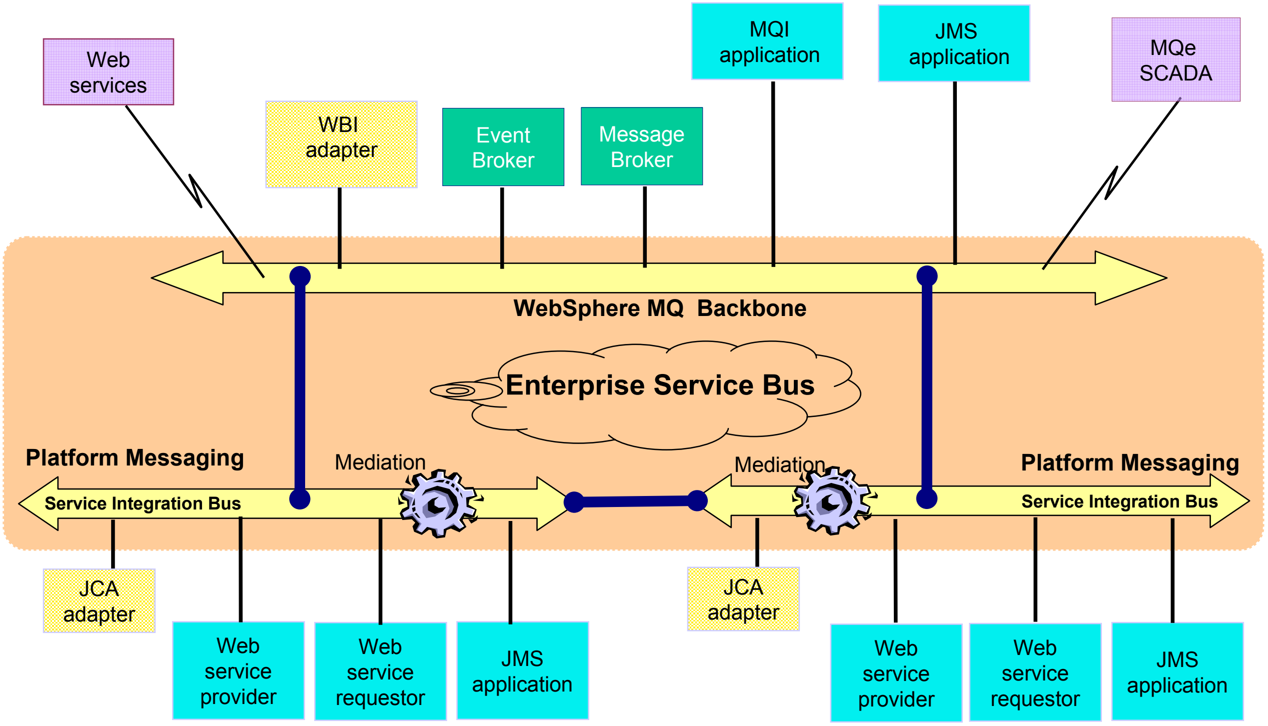
- Integrated asynchronous capabilities for the WebSphere platform
  - Integral JMS messaging service for WebSphere Application Server
  - Fully compliant JMS 1.1 provider
- Service Integration Bus
  - Intelligent infrastructure for service-oriented integration
  - Unifies Service-Oriented Architecture (SOA), messaging, message brokering and publish/subscribe
- Complement and extend WebSphere MQ and WebSphere Application Server
  - Share and extend messaging family capabilities

JMS Support

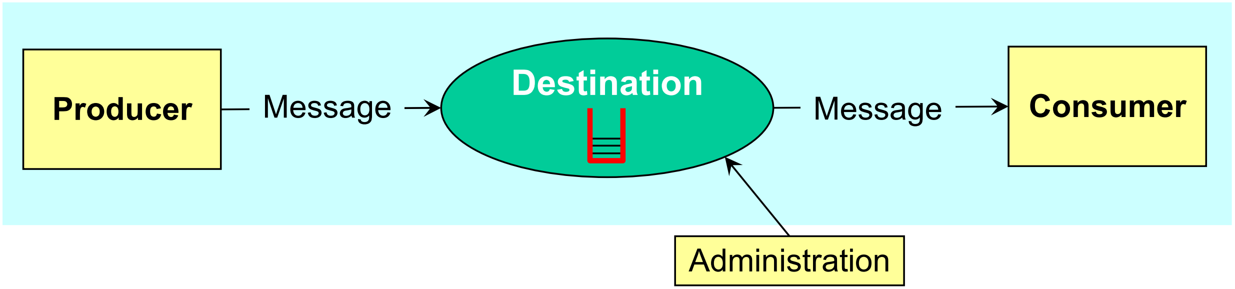
- WebSphere V6 provides a pure Java JMS 1.1 provider that is installed as part of the base server installation
  - Runs completely inside the application server JVM
- Persistent messages are stored either in an embedded Cloudscape database or an external database of customer choice (DB2, Oracle, and so forth) via JDBC driver
- Each application server, or cluster, can host a messaging engine. Messaging engines can be interconnected to form a messaging bus
- Fully integrated with application server management including high availability. Messaging engines will failover along with application servers
- Interoperable with WebSphere MQ


One process ...
Base Server with J2EE, Integrated messaging
Integrated security
Cloudscape or other RDBMS for persistence support
All Java code

Service Integration as Part of ESB



Messaging – Basic Flow

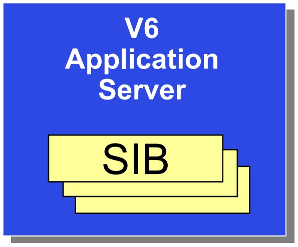


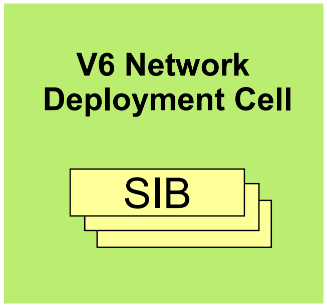
•Producers send/put messages to destinations
•Consumers receive/get messages from destinations
•Destinations are platform messaging managed points of communication rendezvous

- JMS queues
- JMS topics
- Web service endpoints

Service Integration Bus (SIBus)

- It's a communication infrastructure that provides service integration through synchronous and asynchronous messaging
- Can have multiple interconnected buses in a cell or stand-alone node (single server)
  - A common pattern is to have one SIBus in a stand-alone single server
- For WebSphere, SIBus consists of:
  - Bus member (application server or cluster)
  - Messaging engines in the server, or cluster, that manage the bus resources
  - Destinations that are linked to messaging engines
- When SIBus is used for JMS applications, it is referred to as a messaging bus



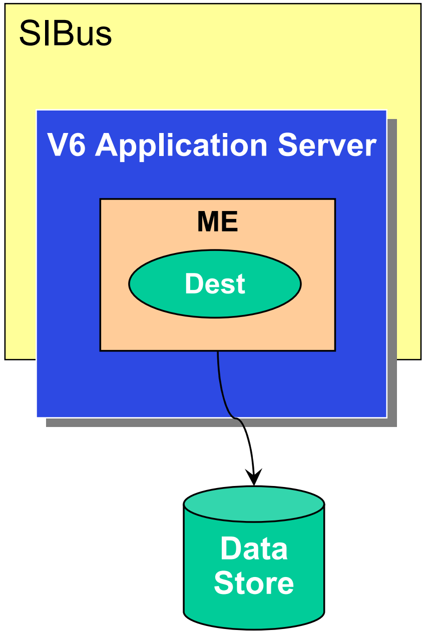


Bus Member

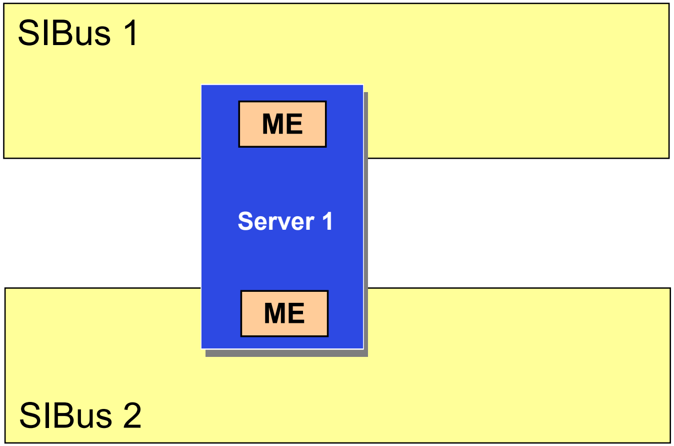
- Bus members of SIBus are application servers and/or clusters on which the messaging engines are defined
- When a new bus member is defined, one messaging engine (ME) is automatically created on the corresponding application server or cluster
- For an ND cell, you can add additional MEs to a cluster to provide scalability
- Can add or remove bus members – this effectively adds/removes the messaging engines

Messaging Engine (ME)

- MEs run inside the application server, or cluster, and manage messaging resources
  - A common pattern is one ME per server
- Each ME has its own set of tables in a data store (JDBC database)
- Queue-like destinations are associated with one or more MEs
  - Allows administrator to control which database is used for persistence
- MEs provide a connection point for clients to put or get messages

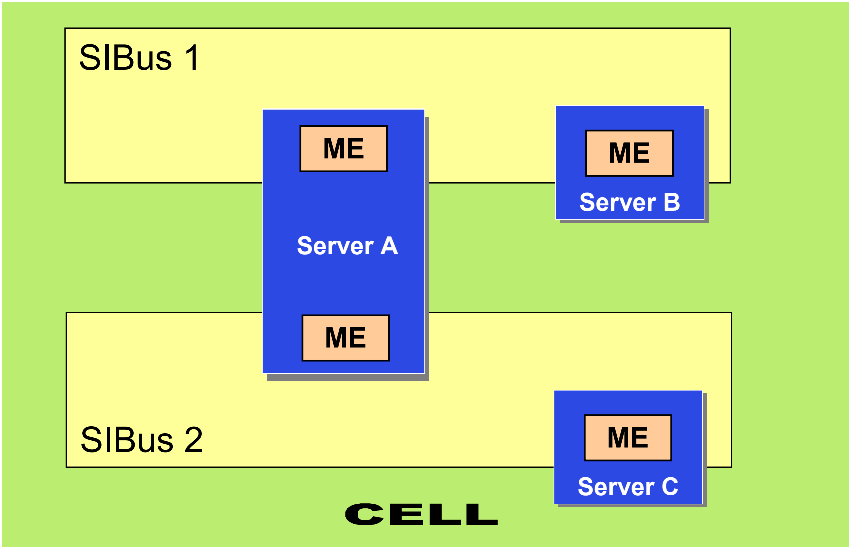


SIB and MEs in a Stand-alone Node



- A stand-alone node can have multiple **buses**
- Each bus can have servers as **bus members**
- When a server is made a bus member, a **Messaging Engine** is created

Example: SIB and MEs in a ND Cell



- A WebSphere Application Server cell can have multiple **buses**
- Each bus can have servers and clusters as **bus members**
- When a server, or cluster, is made a bus member, **Messaging Engine** is created

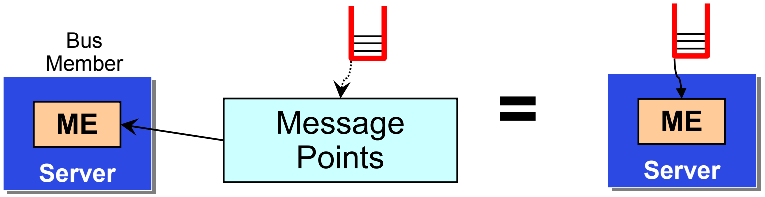
Bus Destinations

- Bus destination is a virtual place within an SIBus, to which applications (producers, consumers, or both) attach to exchange messages
- Bus destinations can be permanent or temporary
  - Temporary - Created and deleted automatically for API specific destinations
    - Created programmatically, usually to specify a JMSReplyTo destination within a message
  - Permanent – Created by administrator
    - Deleted only when administrator deletes it

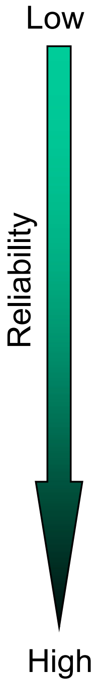
- Types of destinations
  - Queue - For point-to-point messaging
  - Topicspace - For publish-subscribe messaging
  - Alias - Destination that is an alias for another target destination
  - Foreign - Destination that identifies a destination on another bus
  - Exception – Destination that is used to handle messages that cannot be sent to intended bus destination

Linking Destinations to Bus Members

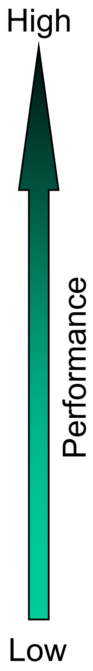
- Bus destinations are associated with one or more bus members, thereby associating it with the corresponding MEs
  - –Allows administrator to control which database is used for persistence
  - –In most cases, a destination is associated with one ME
  - –Multiple MEs provide scalability
- Queue for point-to-point messaging
  - Administrator defines a queue destination on one assigned bus member
  - Each ME in that assigned bus member has a queue point where messages are held
- Topicspace for publish-subscribe messaging
  - Every ME in the SIB is a publication point where messages are held



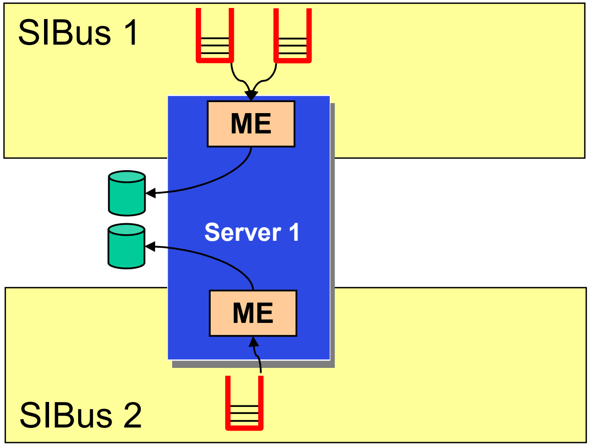
Destination Quality of Service for Reliability



- BEST_EFFORT_NONPERSISTENT
  - Messages are never written to disk
  - Throw away messages if memory cache over-runs
- EXPRESS_NONPERSISTENT
  - Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts
  - No acknowledgement that the ME has received the message
- RELIABLE_NONPERSISTENT
  - Same as Express_Nonpersistent, except, we have a low level acknowledgement message that the client code waits for, before returning to the application with an OK or not OK response
- RELIABLE_PERSISTENT
  - Messages are written asynchronously to persistent storage during normal processing, and stay persisted over server restarts
  - If the server fails, messages are lost if they are held in the cache at the time of failure
- ASSURED_PERSISTENT
  - Highest degree of reliability where assured delivery is supported



Message Store

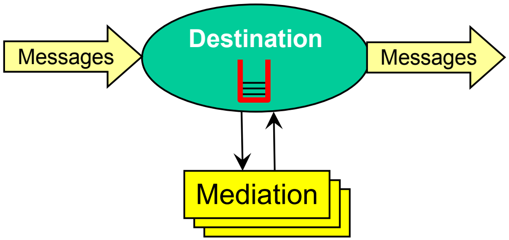


•Each ME has its own data store for storing messages, transaction states and delivery records

•ME requires persistent backing data store – JDBC database used in WebSphere implementation
•MEs may share the database, but each ME has its own schema within the database (which results in different tables)
•Cloudscape database is used as default in Base. In ND, a distributable database, such as DB2, is required

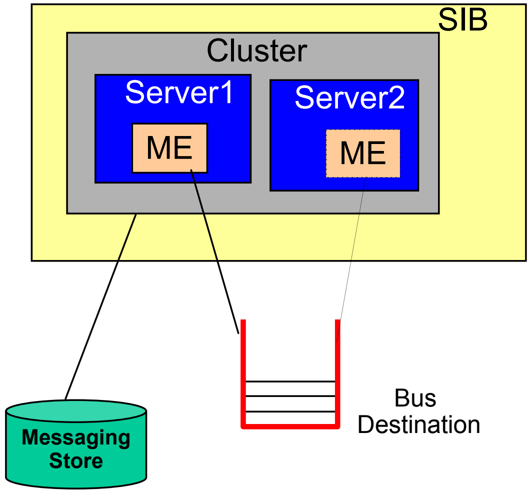
Mediation

- Mediation - The ability to manipulate a message as it traverses the messaging bus (destination)
  - Transform the message
  - Copy and/or reroute the message to a different destination, or sequence of destinations
  - Allow interaction with non-messaging resource managers (for example, databases)
- Mediation attached administratively to a destination
- Mediation construction scenarios:
  - Built from supplied mediation subcomponents (mediation beans)
    - Subcomponent implementations shipped with WPM
  - Mediation beans supplied by IBM or third-party
    - IBM-supplied mediation beans come in the future – not in V6



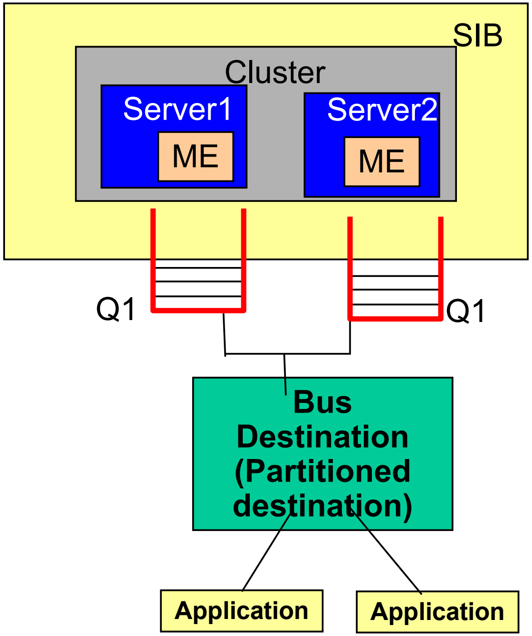
Clustering for High Availability

•Add cluster as a bus member – ME is automatically created
•Only one active ME at any given time – HA Mgr decides which server the ME runs on
•In case of active ME failure, HA Mgr fails over the ME to another standby server



Clustering for Scalability

- A single logical queue can be partitioned across the cluster
  - With each of *n* partitions perhaps holding an *n*th of the messages
- All MEs are active all the time
- Achieved by associating a bus destination to multiple MEs
- Messaging ordering not preserved

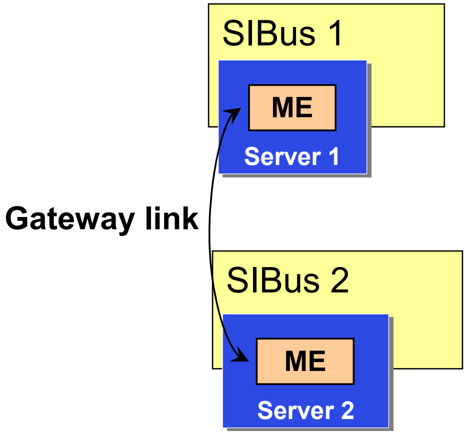


Messaging Engine Topology

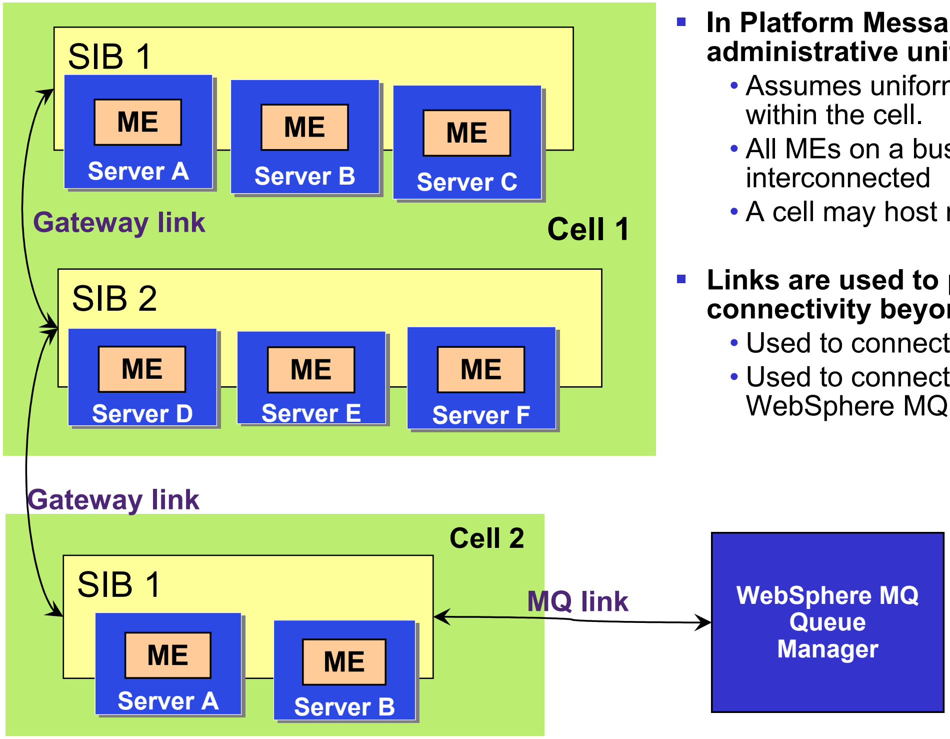
- The default topology consisting of just one messaging engine in a bus is adequate for many applications
- Advantages in deploying more than one messaging engine, and linking them together are:
  - Spreading messaging workload across multiple servers
  - Placing message processing close to the applications that are using it
  - Improving availability in the face of system or link failure
  - Accommodating firewalls or other network restrictions that limit the ability of network hosts all to connect to a single messaging engine

Bus Topology – Stand-alone Node

- An enterprise might deploy multiple interconnected messaging buses for organizational reasons
  - For example, separately administered buses for each department
- A bus can connect to other buses which are known as **foreign buses**
  - The administrator creates a gateway link from a ME in the local bus to an ME in the foreign bus



Network Topologies – ND Cell



- **In Platform Messaging, the administrative unit is the cell**
  - Assumes uniform access to all MEs within the cell.
  - All MEs on a bus are fully interconnected
  - A cell may host multiple buses
- **Links are used to provide connectivity beyond a single bus**
  - Used to connect two different buses
  - Used to connect a PM bus and a WebSphere MQ network

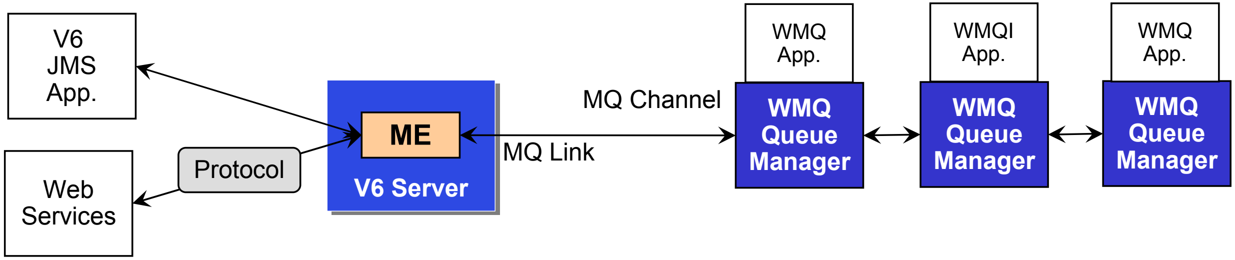
Platform Messaging: Interoperability

- Full interoperability with other SIBus in the same or different cell
- WebSphere V5 embedded JMS server interoperation
  - Existing WebSphere V5 embedded JMS clients can connect to V6 destinations
  - V6 JMS application to connect to an embedded JMS provider hosted in a V5 server
  - Note that it is not possible to connect a V5 embedded JMS server into a V6 SIBus
- **MQ Client Link** can be created to support any WebSphere V5 clients to talk to WebSphere V6 ME

Relationship to WebSphere MQ

- WebSphere MQ queue manager and/or a WebSphere MQ Integrator or Event Broker can coexist on the same machine as a ME
  - WebSphere MQ and Platform Messaging are separate products and do not share any modules or configuration data
- Connectivity between ME and MQ Queue Manager is established by defining a WebSphere MQLink
  - WebSphere MQLink converts between the formats and protocols used by WebSphere MQ and Platform Messaging
- Functions not supported in WebSphere V6
  - An MQ queue manager cannot attach to the bus using any communications protocol other than TCP/IP
  - A messaging engine cannot participate in a WebSphere MQ cluster

Interoperability with MQ



- Tight integration with WebSphere Platform Messaging and WebSphere MQ
- WebSphere MQ thinks that the V6 messaging engine is another queue manager
- WebSphere MQ applications can send messages to queues hosted on V6 messaging
- WebSphere V6 messaging applications can send messages to WebSphere MQ queues

Usage Scenarios for Platform Messaging

- Use Platform Messaging:
  - Customers and J2EE developers currently using WAS V5 embedded JMS provider for intra-WAS messaging
  - Messaging between WAS and existing MQ backbone and its applications
- Use WebSphere MQ:
  - Customers currently using WebSphere MQ may continue to use it
  - Access is required to heterogeneous non-JMS applications, WebSphere MQ clustering, or other WebSphere MQ functions

V6 Support for External JMS Providers

- WebSphere Application Server V6 supports external JMS 1.1 providers
- V6 supports the following JMS providers
  - Default Messaging Provider (Platform Messaging)
  - WebSphere MQ V5.4
  - Generic JMS Providers

Unit Summary

Having completed this unit, you should be able to discuss:

- WebSphere platform messaging providing a JMS V1.1-compliant JMS provider
- SIBus as the communication layer for WebSphere platform messaging
- Messaging engines managing messaging resources
- How MQLinks can be created to communicate with WebSphere MQ queue managers