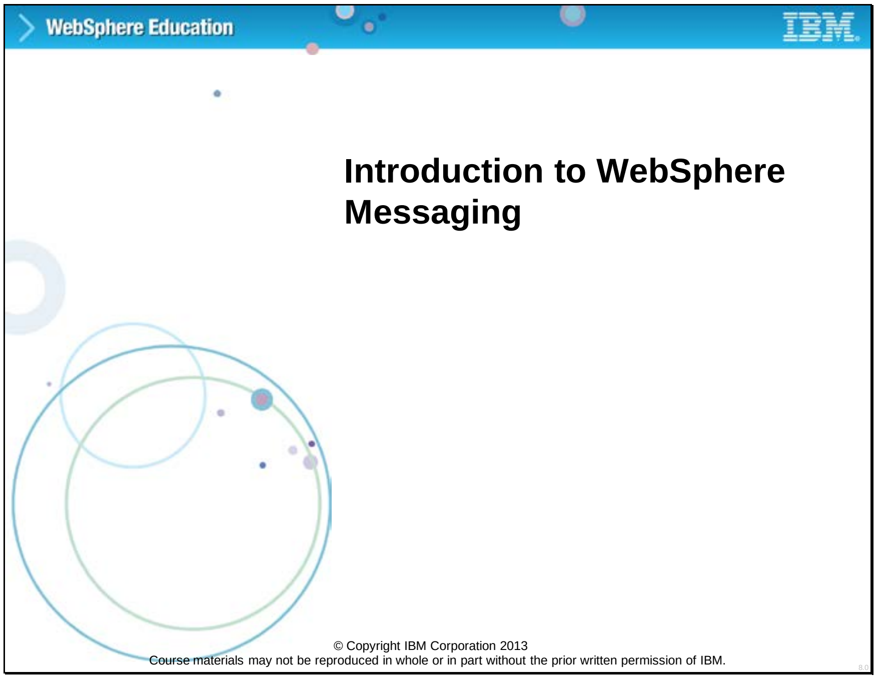
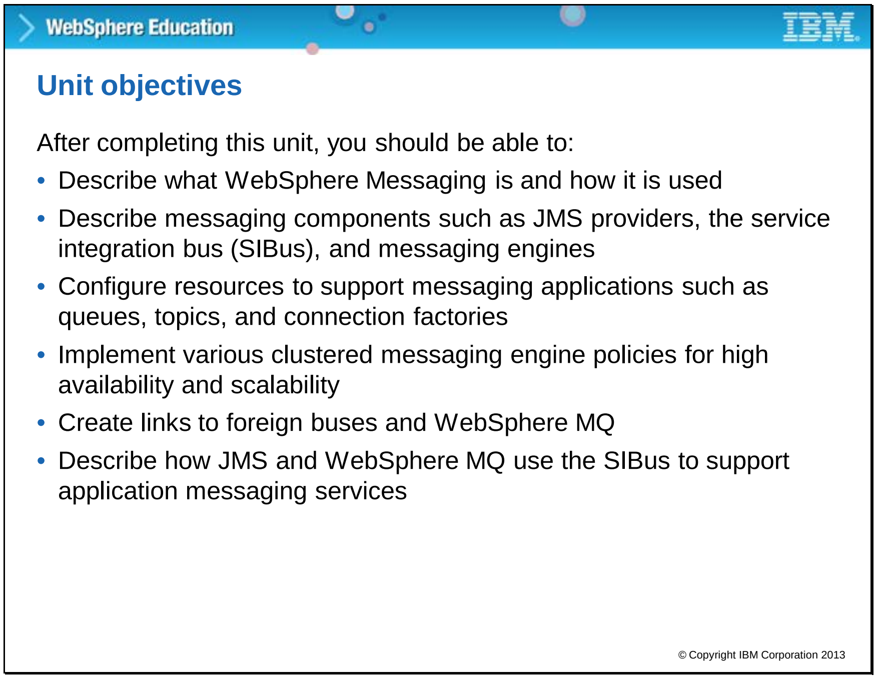
Slide 1



Introduction to WebSphere Messaging

Unit 15: Introduction to WebSphere Messaging

Slide 2



WebSphere Education

Unit objectives

After completing this unit, you should be able to:

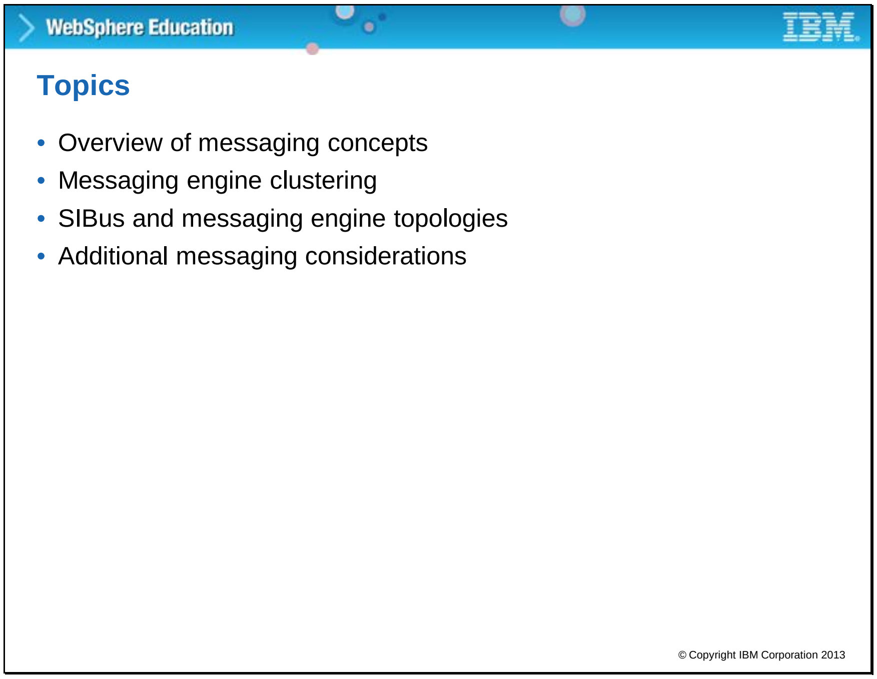
- Describe what WebSphere Messaging is and how it is used
- Describe messaging components such as JMS providers, the service integration bus (SIBus), and messaging engines
- Configure resources to support messaging applications such as queues, topics, and connection factories
- Implement various clustered messaging engine policies for high availability and scalability
- Create links to foreign buses and WebSphere MQ
- Describe how JMS and WebSphere MQ use the SIBus to support application messaging services

Title: Unit objectives

After completing this unit, you should be able to:

- Describe what WebSphere Messaging is and how it is used
- Describe messaging components such as JMS providers, the service integration bus (SIBus), and messaging engines
- Configure resources to support messaging applications such as queues, topics, and connection factories
- Implement different clustered messaging engine policies for high availability and scalability
- Create links to foreign buses and WebSphere MQ
- Describe how JMS uses the service integration bus (SIBus) to support application messaging services

Slide 3



WebSphere Education

Topics

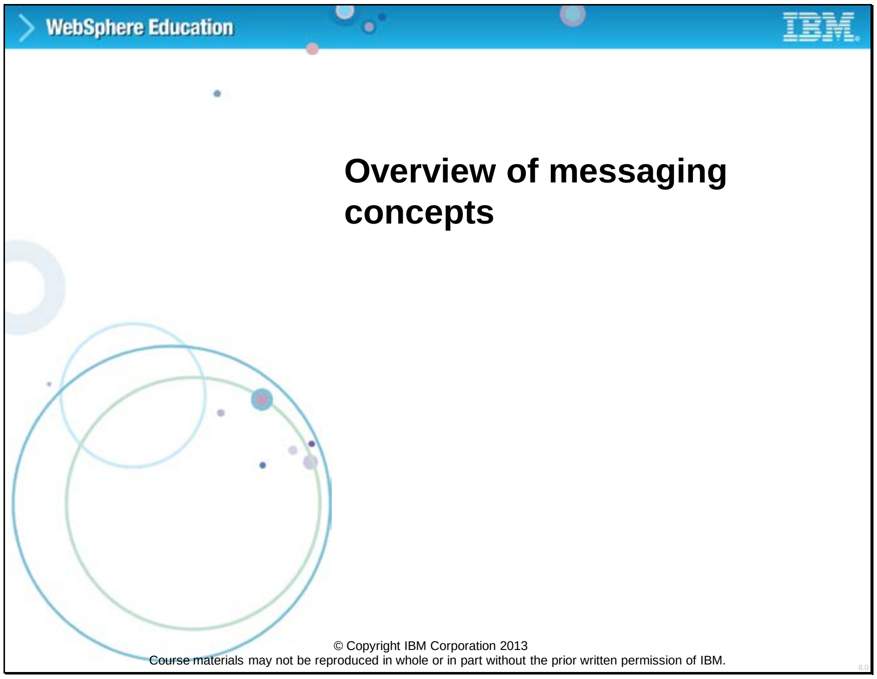
- Overview of messaging concepts
- Messaging engine clustering
- SIBus and messaging engine topologies
- Additional messaging considerations

Title: Topics

This unit covers the following topics:

- Overview of messaging concepts
- Messaging engine clustering
- SIBus and messaging engine topologies
- Other messaging considerations

Slide 4



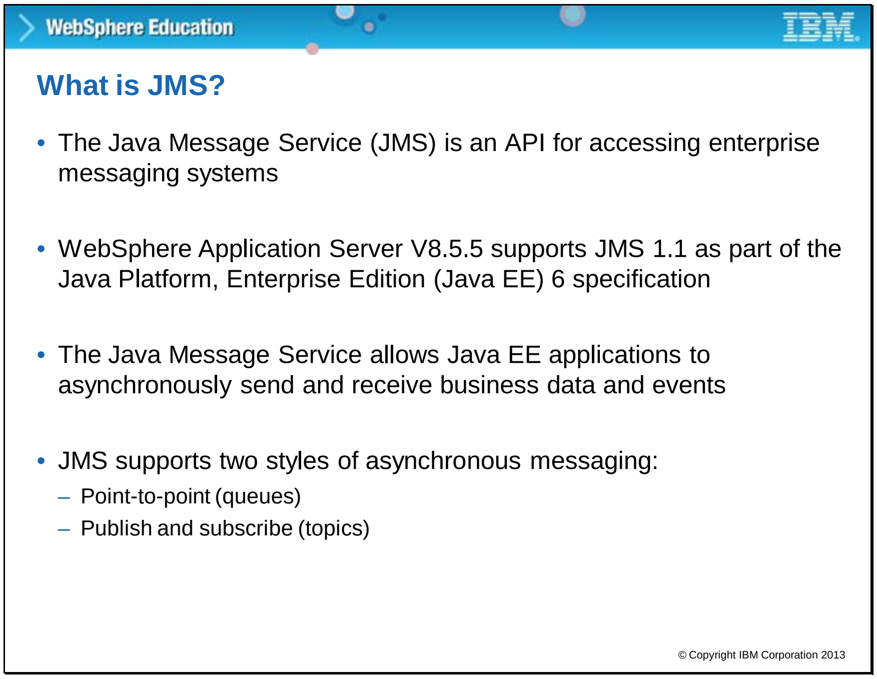
WebSphere Education

Overview of messaging concepts

Topic: Overview of messaging concepts

This topic describes basic Java messaging concepts.

Slide 5



WebSphere Education
IBM

What is JMS?

- The Java Message Service (JMS) is an API for accessing enterprise messaging systems
- WebSphere Application Server V8.5.5 supports JMS 1.1 as part of the Java Platform, Enterprise Edition (Java EE) 6 specification
- The Java Message Service allows Java EE applications to asynchronously send and receive business data and events
- JMS supports two styles of asynchronous messaging:
  - Point-to-point (queues)
  - Publish and subscribe (topics)

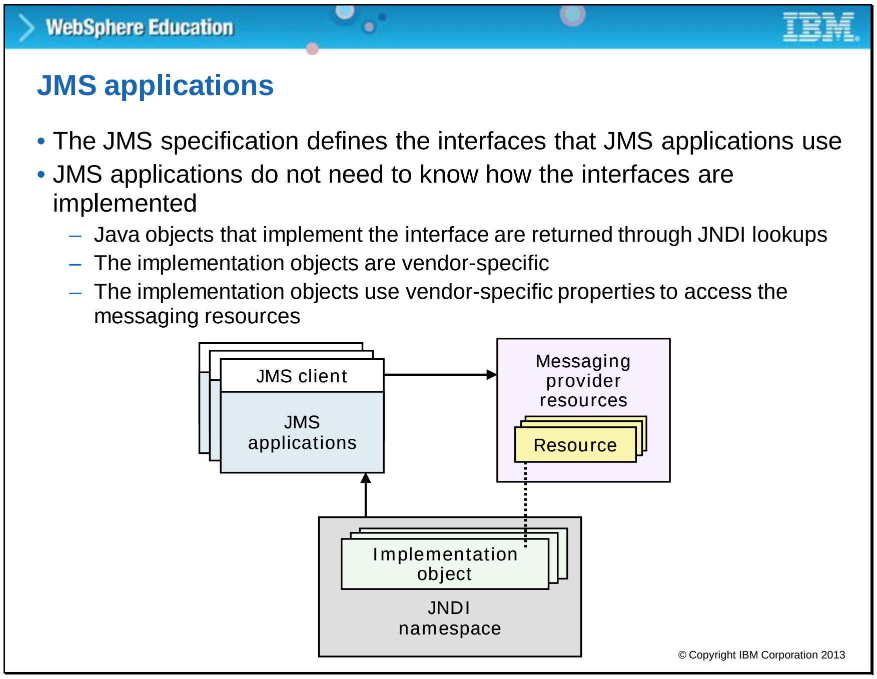
© Copyright IBM Corporation 2013

Title: What is JMS?

The Java Messaging Service, or JMS, is an API that defines some interfaces for applications to use. The developer must provide the implementation under those interfaces. The implementation is delivered as a JMS provider.

JMS allows Java EE applications to asynchronously send and receive business data and events. JMS supports two styles of asynchronous messaging: point-to-point or queues, and publish and subscribe, or topic spaces. WebSphere Application Server V8.5 supports JMS 1.1 as part of the Java EE 6 specification.

Slide 6



WebSphere Education
IBM

JMS applications

- The JMS specification defines the interfaces that JMS applications use
- JMS applications do not need to know how the interfaces are implemented
  - Java objects that implement the interface are returned through JNDI lookups
  - The implementation objects are vendor-specific
  - The implementation objects use vendor-specific properties to access the messaging resources

JMS client

JMS applications

Messaging provider resources

Resource

Implementation object

JNDI namespace

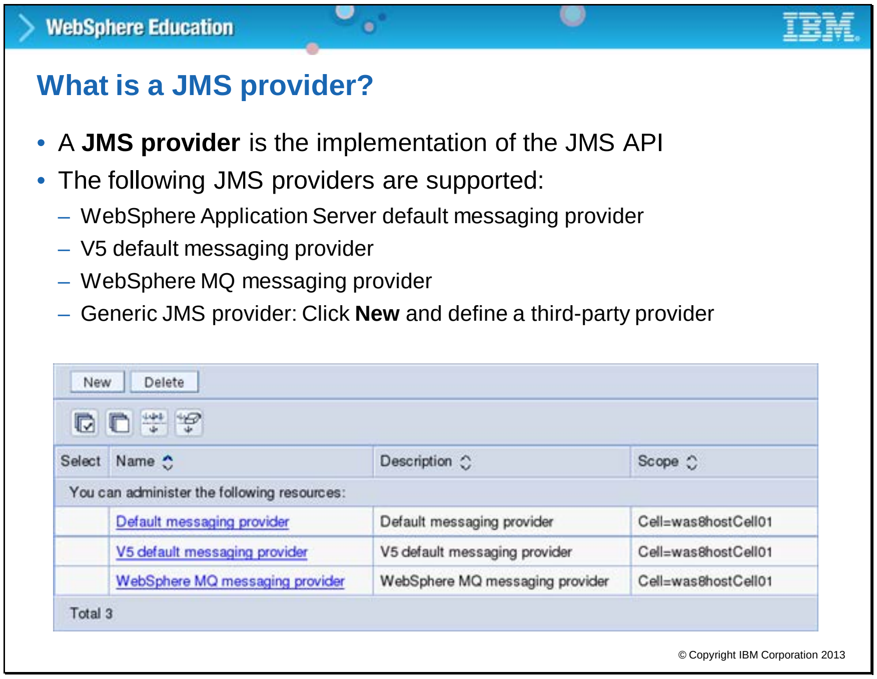
© Copyright IBM Corporation 2013

Title: JMS applications

This diagram shows how a JMS application works. Applications use the interfaces that are defined in the JMS specification. They do not require knowledge of how the interfaces are implemented. The JMS client, when using the JMS API, must get actual Java objects that implement the JMS interfaces. The objects are stored in JNDI and have vendor-specific properties and behaviors. After the object is retrieved from JNDI, the standard JMS APIs can be used to work with the vendor-specific resources.

If the client is deployed to a different JMS provider, the JNDI lookup remains the same, but the object that is returned is different. The client uses the same JMS API calls, but the object has different behaviors for working with the different vendor resources.

Slide 7



What is a JMS provider?

- A **JMS provider** is the implementation of the JMS API
- The following JMS providers are supported:
  - WebSphere Application Server default messaging provider
  - V5 default messaging provider
  - WebSphere MQ messaging provider
  - Generic JMS provider: Click **New** and define a third-party provider

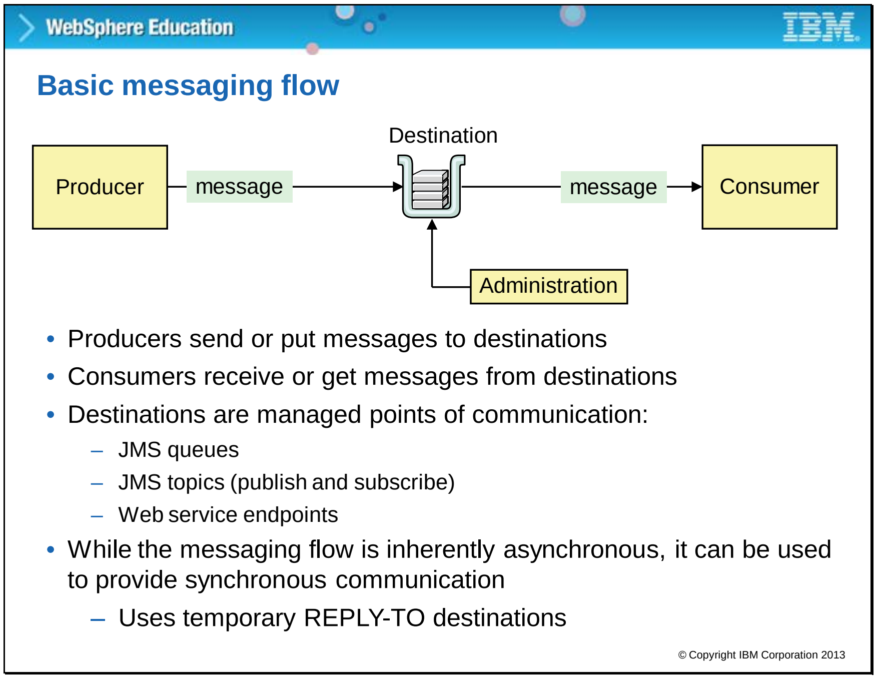
Select	Name	Description	Scope
You can administer the following resources:			
	Default messaging provider	Default messaging provider	Cell=was8hostCell01
	V5 default messaging provider	V5 default messaging provider	Cell=was8hostCell01
	WebSphere MQ messaging provider	WebSphere MQ messaging provider	Cell=was8hostCell01
Total 3			

Title: What is a JMS provider?

WebSphere Application Server default messaging provider is a fully featured messaging provider included with the WebSphere Application Server product. It is a robust and stable messaging platform that can handle point-to-point queues, topics in a publish and subscribe environment, and web service endpoints. Another supported JMS provider is WebSphere MQ messaging provider, which is a separate product.

It is suggested that you use WebSphere MQ when you require advanced messaging facilities and options. WebSphere MQ is an older product than the WebSphere Application Server default messaging provider and is available on many platforms, supporting many programming languages. It is fully JMS-compliant and has a large client base. Generic JMS providers are any external messaging providers other than WebSphere MQ. Although WebSphere Application Server works with any JMS-compliant messaging provider (after it is defined to WebSphere), there is limited administrative support in WebSphere. This approach is only suggested if you have an existing investment in a third-party messaging provider because much greater support is available in the WebSphere Application Server default messaging provider and WebSphere MQ messaging provider.

Slide 8



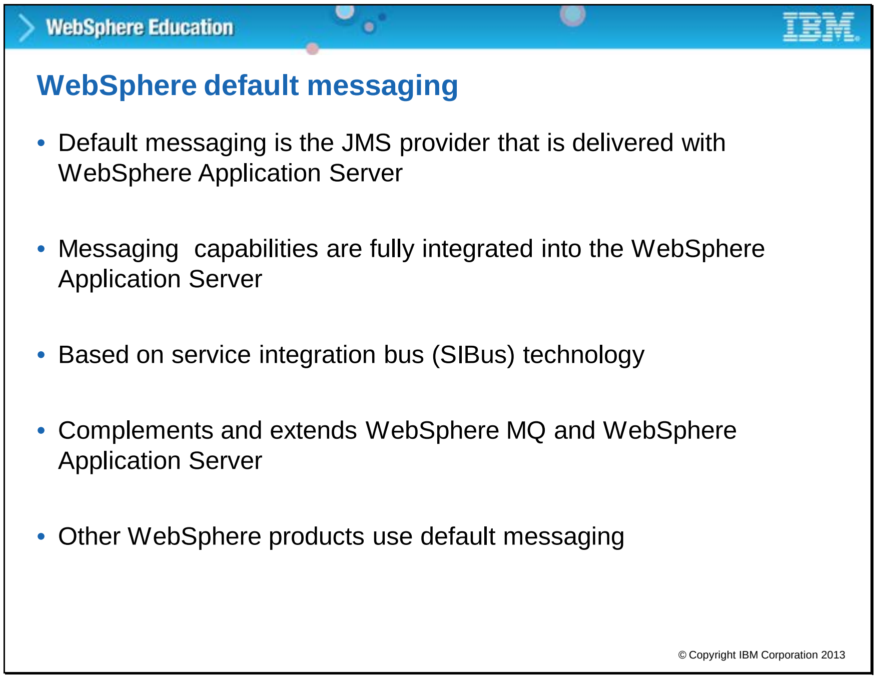
Basic messaging flow

- Producers send or put messages to destinations
- Consumers receive or get messages from destinations
- Destinations are managed points of communication:
  - JMS queues
  - JMS topics (publish and subscribe)
  - Web service endpoints
- While the messaging flow is inherently asynchronous, it can be used to provide synchronous communication
  - Uses temporary REPLY-TO destinations

Title: Basic messaging flow

This diagram shows the relationship between different components in a basic messaging flow. **Producers** *send* messages to destinations and **consumers** *get* messages from destinations. Destinations are managed points of communication exchange. Destinations can be JMS queues, JMS topics (publish and subscribe), or web service endpoints. System administrators might get involved with the configuration and management of destinations. Message flow is inherently asynchronous, but can also be used to provide synchronous communications.

Slide 9



WebSphere Education
IBM

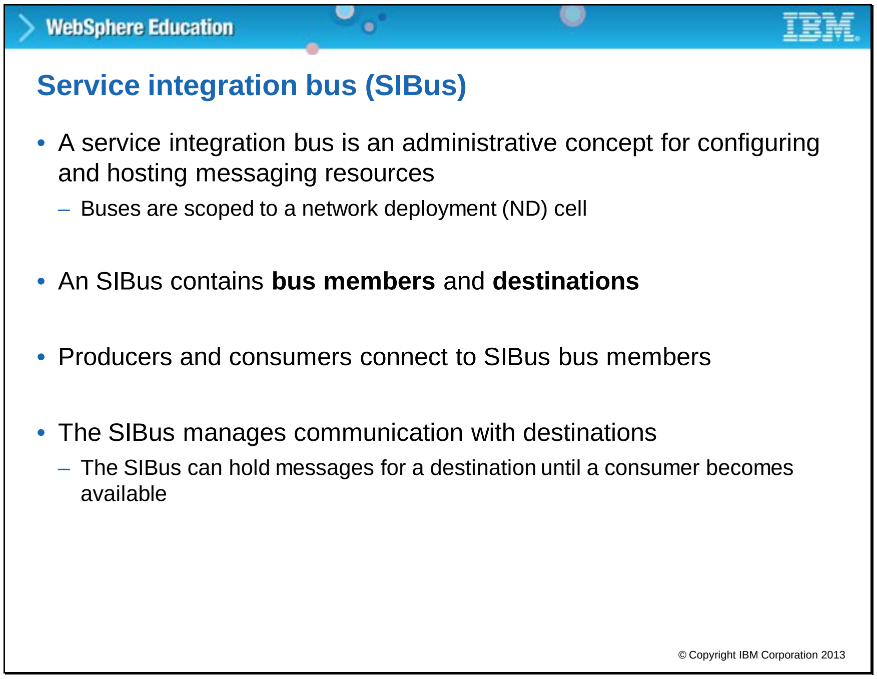
WebSphere default messaging

- Default messaging is the JMS provider that is delivered with WebSphere Application Server
- Messaging capabilities are fully integrated into the WebSphere Application Server
- Based on service integration bus (SIBus) technology
- Complements and extends WebSphere MQ and WebSphere Application Server
- Other WebSphere products use default messaging

Title: WebSphere default messaging

Messaging capabilities are fully integrated in the WebSphere Application Server platform, including security and high availability. They can be used to extend WebSphere MQ and WebSphere Application Server because you can share and extend messaging capabilities and interoperate with WebSphere MQ. Other WebSphere products use the default messaging such as WebSphere Enterprise Service Bus and WebSphere Process Server. The tools within WebSphere Integration Developer also support WebSphere messaging.

Slide 10



WebSphere Education
IBM

Service integration bus (SIBus)

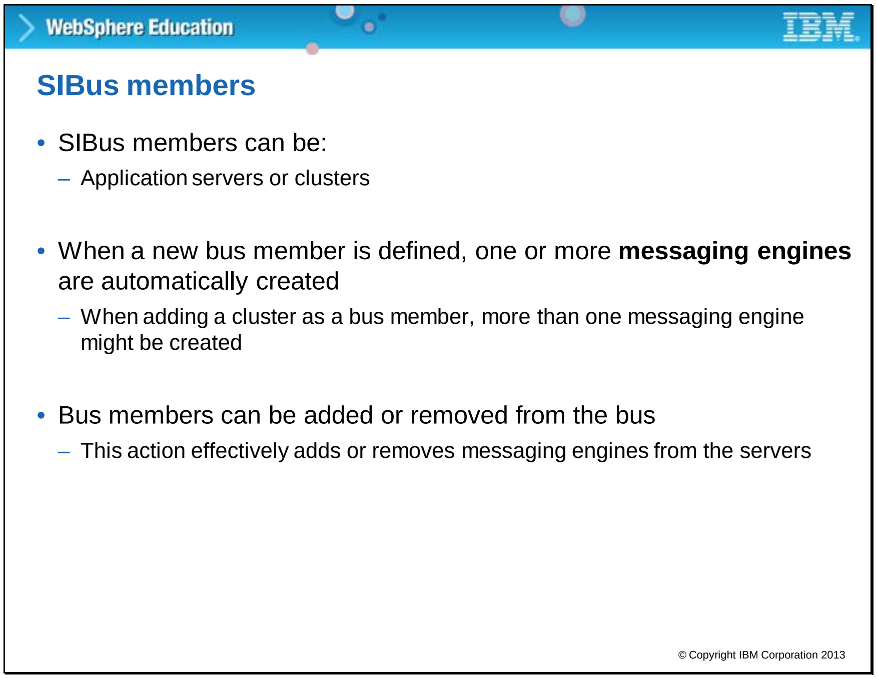
- A service integration bus is an administrative concept for configuring and hosting messaging resources
  - Buses are scoped to a network deployment (ND) cell
- An SIBus contains **bus members** and **destinations**
- Producers and consumers connect to SIBus bus members
- The SIBus manages communication with destinations
  - The SIBus can hold messages for a destination until a consumer becomes available

Title: Service integration bus (SIBus)

A service integration bus (SIBus) provides the underpinning for WebSphere Business Integration. An SIBus provides administration capabilities for hosting messaging resources, and can be scoped to a network deployment cell. An SIBus can contain bus members and destinations. Bus members host message producers and consumers. Producers and consumers can be external to WebSphere. The SIBus supports the following types of messaging: Sending messages synchronously (which requires the consuming application to be running and reachable) and sending messages asynchronously (where the consuming application might not be running or reachable).

The SIBus supports publishing events or other notifications. The SIBus also generates notification messages.

Slide 11



WebSphere Education

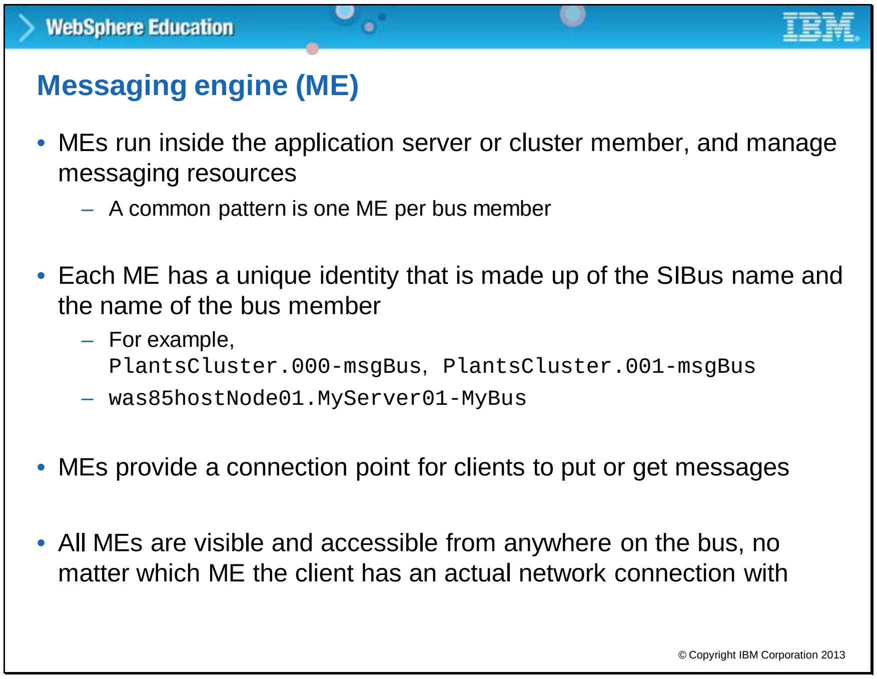
SIBus members

- SIBus members can be:
  - Application servers or clusters
- When a new bus member is defined, one or more **messaging engines** are automatically created
  - When adding a cluster as a bus member, more than one messaging engine might be created
- Bus members can be added or removed from the bus
  - This action effectively adds or removes messaging engines from the servers

Title: SIBus members

SIBus members can be application servers or clusters. When a new bus member is defined, a messaging engine (ME) is automatically created on the corresponding application server or cluster. If a cluster is chosen as the bus member, then the default behavior is that one cluster member has an active messaging engine. Bus members can be added or removed from the bus. This action effectively adds or removes messaging engines from the servers.

Slide 12


WebSphere Education

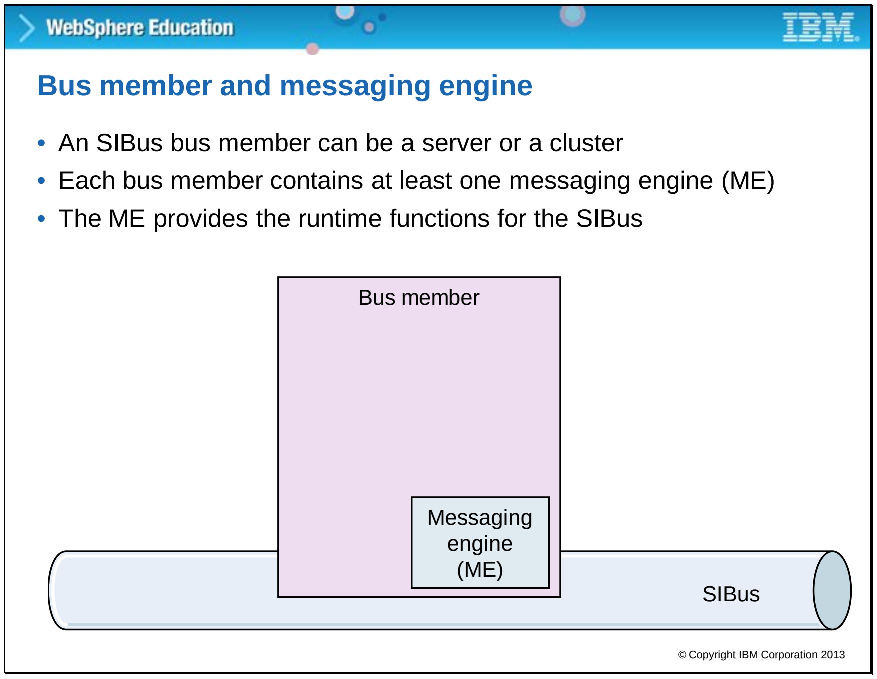
Messaging engine (ME)

- MEs run inside the application server or cluster member, and manage messaging resources
  - A common pattern is one ME per bus member
- Each ME has a unique identity that is made up of the SIBus name and the name of the bus member
  - For example, `PlantsCluster.000-msgBus`, `PlantsCluster.001-msgBus`
  - `was85hostNode01.MyServer01-MyBus`
- MEs provide a connection point for clients to put or get messages
- All MEs are visible and accessible from anywhere on the bus, no matter which ME the client has an actual network connection with

Title: Messaging engine (ME)

When you define a new bus member, or add an application server or cluster to a bus, one messaging engine (ME) is automatically created for the application server or the cluster. The messaging engine provides a connection point for clients to put or get messages. The messaging engines are visible and accessible from anywhere on the bus. Multiple messaging engines can be running within the same cluster, but the default is one active ME per cluster. Within an SIBus, each messaging engine has a unique identity that is made up of the SIBus name and the name of the bus member as shown in the example on this slide.

Slide 13



WebSphere Education

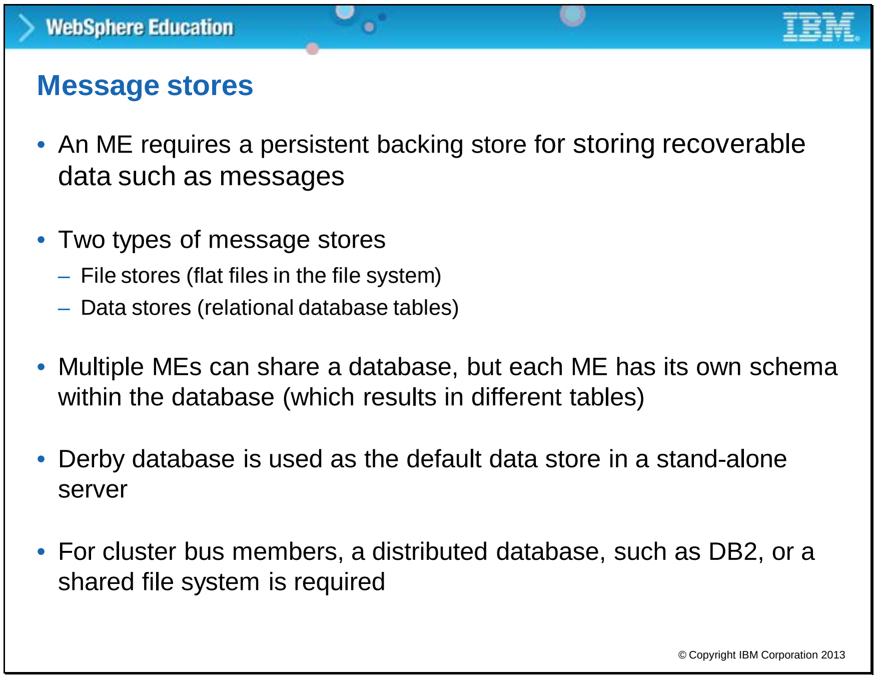
Bus member and messaging engine

- An SIBus bus member can be a server or a cluster
- Each bus member contains at least one messaging engine (ME)
- The ME provides the runtime functions for the SIBus

Title: Bus member and messaging engine

This diagram illustrates the relationship between a bus, a bus member, and the messaging engine. When you define a new bus member, one messaging engine is automatically created on the application server or cluster, represented in the diagram here as the bus member. The bus provides the architectural representation of the messaging environment. The messaging engine provides the runtime function for the bus.

Slide 14



WebSphere Education

Message stores

- An ME requires a persistent backing store for storing recoverable data such as messages
- Two types of message stores
  - File stores (flat files in the file system)
  - Data stores (relational database tables)
- Multiple MEs can share a database, but each ME has its own schema within the database (which results in different tables)
- Derby database is used as the default data store in a stand-alone server
- For cluster bus members, a distributed database, such as DB2, or a shared file system is required

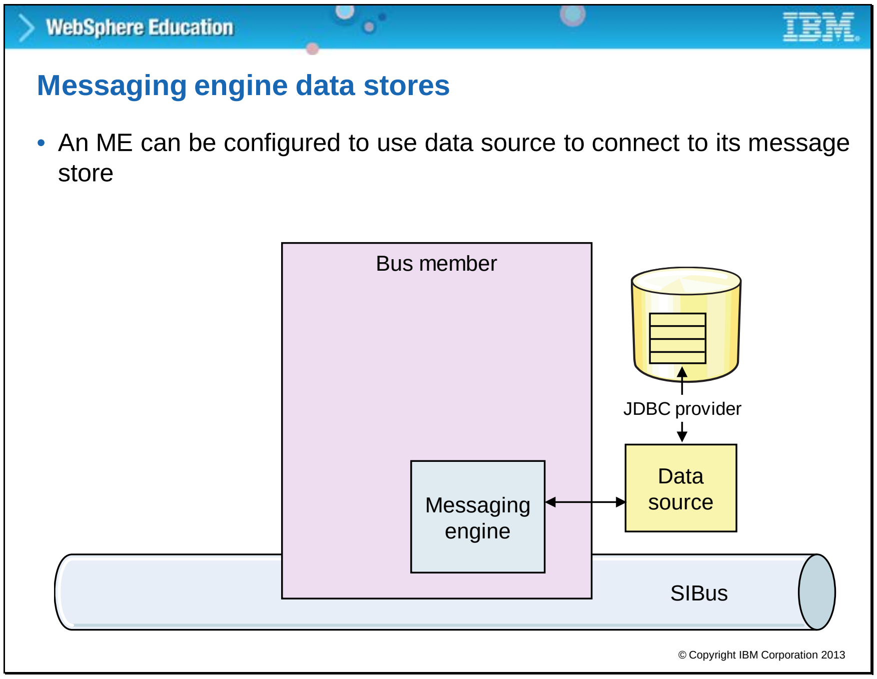
Title: Message stores

The message store is a subcomponent of the messaging engine that is used to buffer in-flight messages and hold other data, such as records of message delivery. A message store can be either persistent or nonpersistent. A persistent message store holds both persistent objects and volatile objects. Persistent objects contain state data that survives after an engine stops for some reason. Volatile objects contain state data that does not survive a failure, and might or might not survive an orderly shutdown of the messaging engine.

In contrast, a nonpersistent message store can hold only volatile objects; it cannot hold persistent objects. A messaging engine requires a persistent back-end data store, even for nonpersistent messages. It can be a file in the file system or a database.

Multiple messaging engines can share a database, but each one has its own schema within the database. Derby is the default database that is used for a stand-alone application server. A more robust, distributable database such as DB2 is required for a federated node.

Slide 15



WebSphere Education

Messaging engine data stores

- An ME can be configured to use data source to connect to its message store

Bus member

JDBC provider

Messaging engine

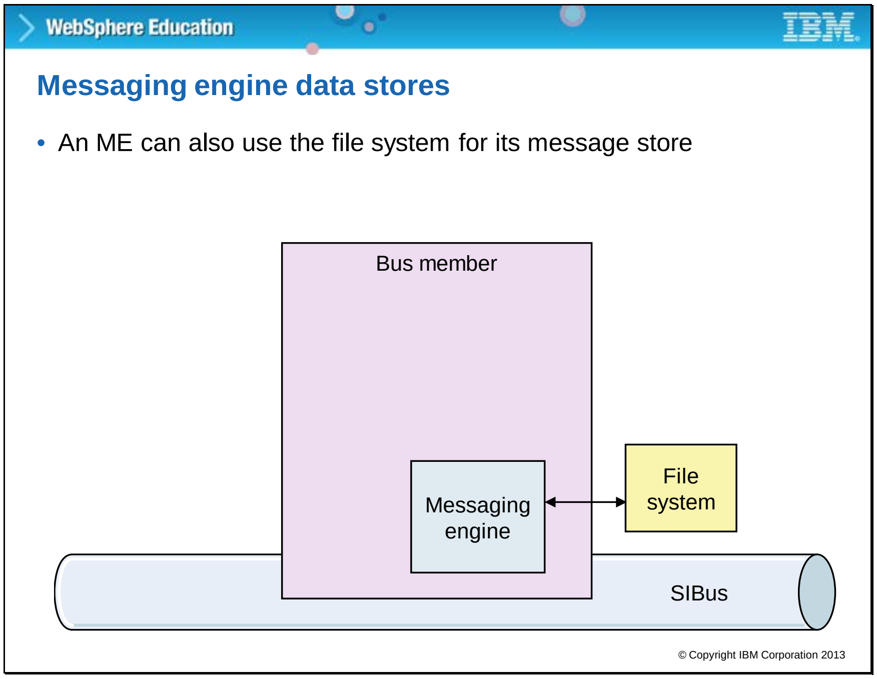
Data source

SIBus

Title: Messaging engine data stores

A file store or a data store (database) can provide persistent storage. The database preserves messages and subscriptions so that they survive if the server or messaging engine is stopped and restarted. It is also used for the overflow of the nonpersistent messages in some quality of service (QoS) options. This diagram adds a persistent storage to the simple topology. Also, a data source is needed to access a database.

Slide 16



WebSphere Education

Messaging engine data stores

- An ME can also use the file system for its message store

Bus member

Messaging engine

File system

SIBus

© Copyright IBM Corporation 2013

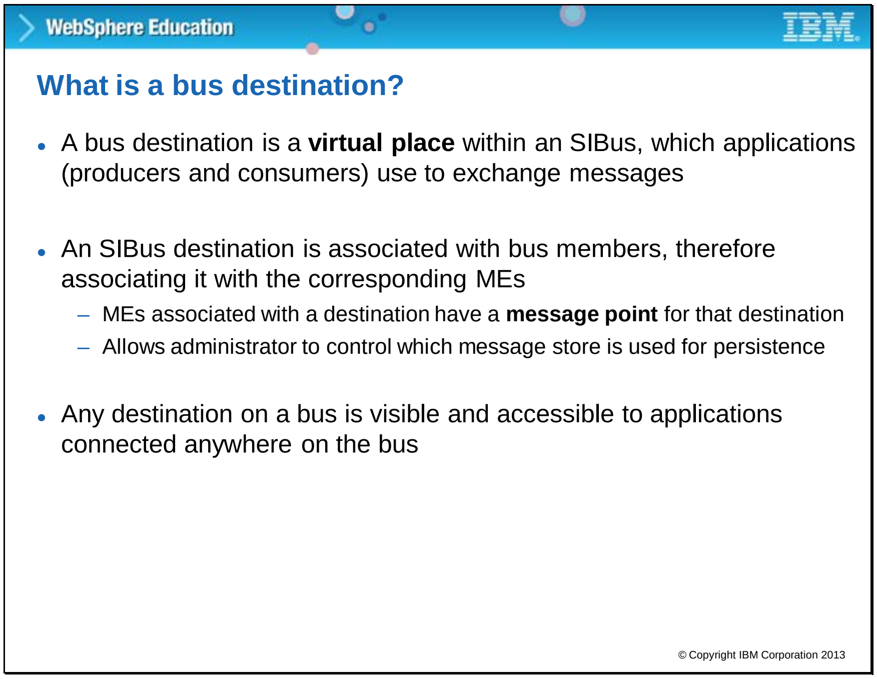
Title: Messaging engine data stores

An ME can also use the file system for its message store, which is known as a file store.

A file store provides an environment that is easier to manage and might perform better than a data store. A file store directly uses files in a file system through the operating system. The data storage in a file store is split into three levels: the log file, permanent store file, and temporary store file.

In a clustered environment, a shared file system must be used so all instances of messaging engines can have access for failover purposes.

Slide 17



WebSphere Education
IBM

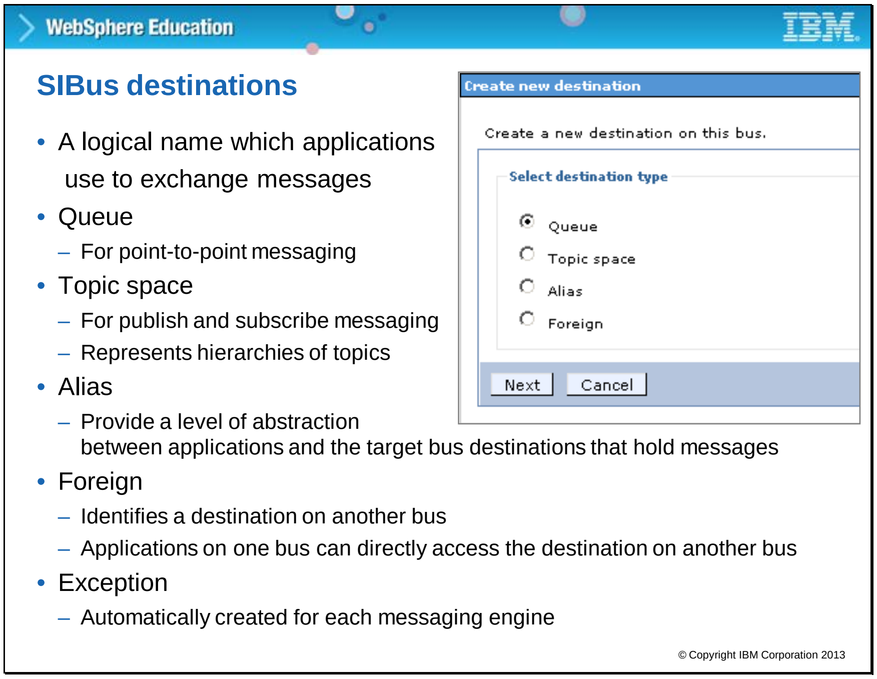
What is a bus destination?

- A bus destination is a **virtual place** within an SIBus, which applications (producers and consumers) use to exchange messages
- An SIBus destination is associated with bus members, therefore associating it with the corresponding MEs
  - MEs associated with a destination have a **message point** for that destination
  - Allows administrator to control which message store is used for persistence
- Any destination on a bus is visible and accessible to applications connected anywhere on the bus

Title: What is a bus destination?

A bus destination is a virtual location within an SIBus to which applications (producers, consumers, or both) attach to exchange messages. SIBus destinations are associated with bus members, which in turn, associates them with the corresponding messaging engines. Messaging engines that are associated with a destination have a message point for that destination. This message point allows the administrator to control which message store is used for persistence.

Slide 18



WebSphere Education

SIBus destinations

- A logical name which applications use to exchange messages
- Queue
  - For point-to-point messaging
- Topic space
  - For publish and subscribe messaging
  - Represents hierarchies of topics
- Alias
  - Provide a level of abstraction between applications and the target bus destinations that hold messages
- Foreign
  - Identifies a destination on another bus
  - Applications on one bus can directly access the destination on another bus
- Exception
  - Automatically created for each messaging engine

Create new destination

Create a new destination on this bus.

Select destination type

● Queue
○ Topic space
○ Alias
○ Foreign

Next Cancel

Title: SIBus destinations

Several types of destinations can be configured. These destinations include a queue, topic space, alias, foreign, and exception.

A **queue** is used for point-to-point messaging.

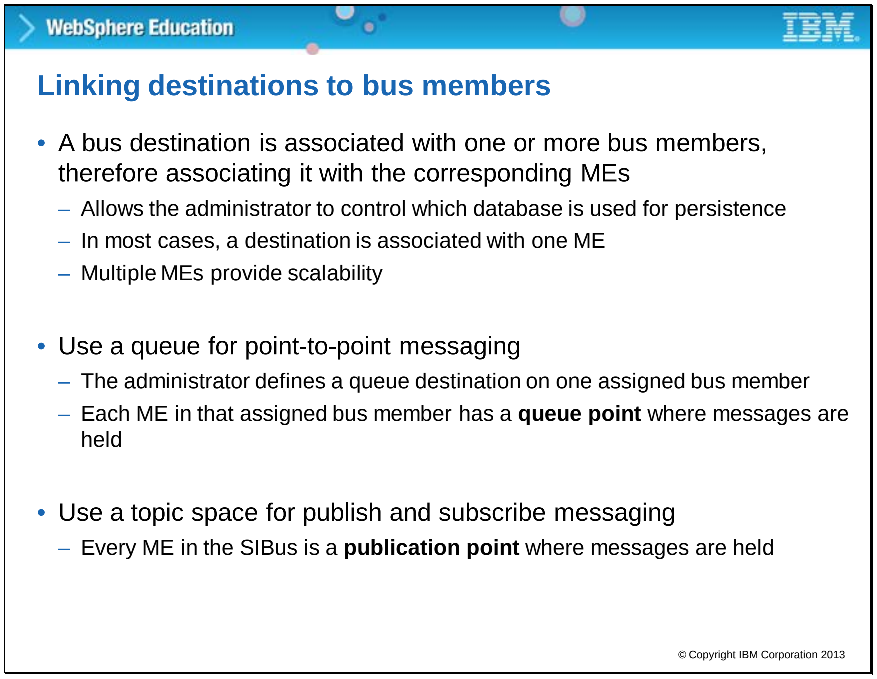
A **topic space** is a hierarchy of topics that are used for publish and subscribe messaging. Topics with the same name can exist in multiple topic spaces.

Alias destinations provide a level of abstraction between applications and the underlying target bus destinations that hold messages. Applications interact with the alias destination, so the target bus destination can be changed without changing the application. Each alias destination identifies a target bus destination and target SIBus. Applications can use an alias destination to route messages to a target destination in the same bus or to another (foreign) bus (including across an MQLink to a queue provided by WebSphere MQ).

The **foreign** destination identifies a destination on another bus, and enables applications on one bus to access directly the destination on another bus.

Each messaging engine has a default **exception** destination that can be used to handle undeliverable messages for all bus destinations that are localized to the messaging engine.

Slide 19



WebSphere Education

Linking destinations to bus members

- A bus destination is associated with one or more bus members, therefore associating it with the corresponding MEs
  - Allows the administrator to control which database is used for persistence
  - In most cases, a destination is associated with one ME
  - Multiple MEs provide scalability
- Use a queue for point-to-point messaging
  - The 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
- Use a topic space for publish and subscribe messaging
  - Every ME in the SIBus is a **publication point** where messages are held

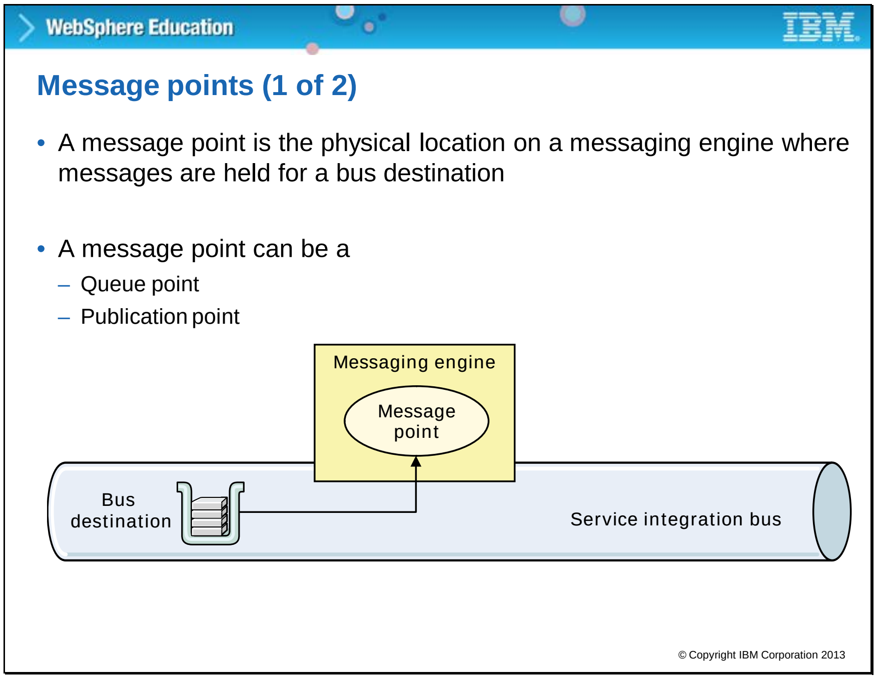
© Copyright IBM Corporation 2013

Title: Linking destinations to bus members

A bus is a virtual messaging environment that spans servers. If you want to send or receive messages through a bus, you must connect to it. Servers (for example, server1) can be added to a bus, and in doing so you create a messaging engine (ME) in that server. SIBus destinations are defined on the bus (rather than on an ME). Queues are localized to a particular ME (for example, messages sent to that queue are stored in a database on one particular ME within the bus), whereas topic spaces are localized on every ME. A **message point** is the general term for the location on a messaging engine where messages are held for a bus destination.

For point-to-point messaging, the administrator selects one bus member to implement the runtime state of a queue destination. This action automatically defines a **queue point** for each messaging engine in the assigned bus member. For a queue destination assigned to an application server, all messages sent to that destination are handled through the messaging engine of that server, and message order is preserved. For publish and subscribe messaging, the administrator configures the destination as a topic space, but does not need to select any assigned bus member for the topic space. A topic space has a **publication point** that is defined automatically for each messaging engine on the bus.

Slide 20



WebSphere Education

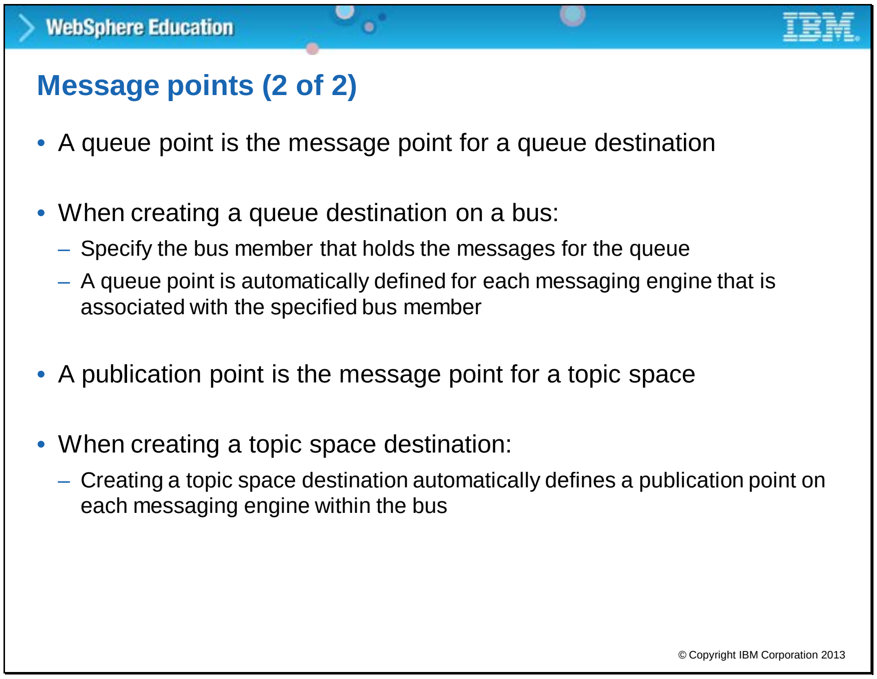
Message points (1 of 2)

- A message point is the physical location on a messaging engine where messages are held for a bus destination
- A message point can be a
  - Queue point
  - Publication point

Title: Message points (1 of 2)

If the bus member is an application server, a single queue point is created and associated with the messaging engine on that application server. The messaging engine handles all of the messages that are sent to the queue destination. In this configuration, the message order is maintained on the queue destination. If the bus member is a cluster of application servers, a queue point is created and associated with each messaging engine defined within the bus member. The queue destination is partitioned across the available messaging engines within the cluster. In this configuration, message order is not maintained on the queue destination.

Slide 21



WebSphere Education

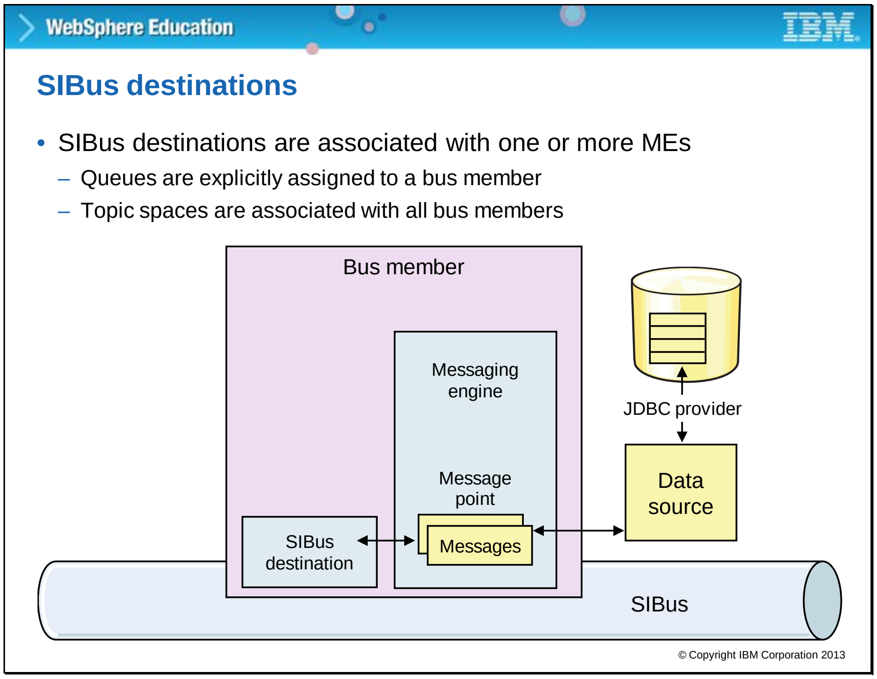
Message points (2 of 2)

- A queue point is the message point for a queue destination
- When creating a queue destination on a bus:
  - Specify the bus member that holds the messages for the queue
  - A queue point is automatically defined for each messaging engine that is associated with the specified bus member
- A publication point is the message point for a topic space
- When creating a topic space destination:
  - Creating a topic space destination automatically defines a publication point on each messaging engine within the bus

Title: Message points (2 of 2)

A **publication point** is the message point for a topic space. When creating a topic space destination, you specify a bus member to hold messages for the topic space. Creating a topic space destination automatically defines a publication point on each messaging engine within the bus.

Slide 22



WebSphere Education

SIBus destinations

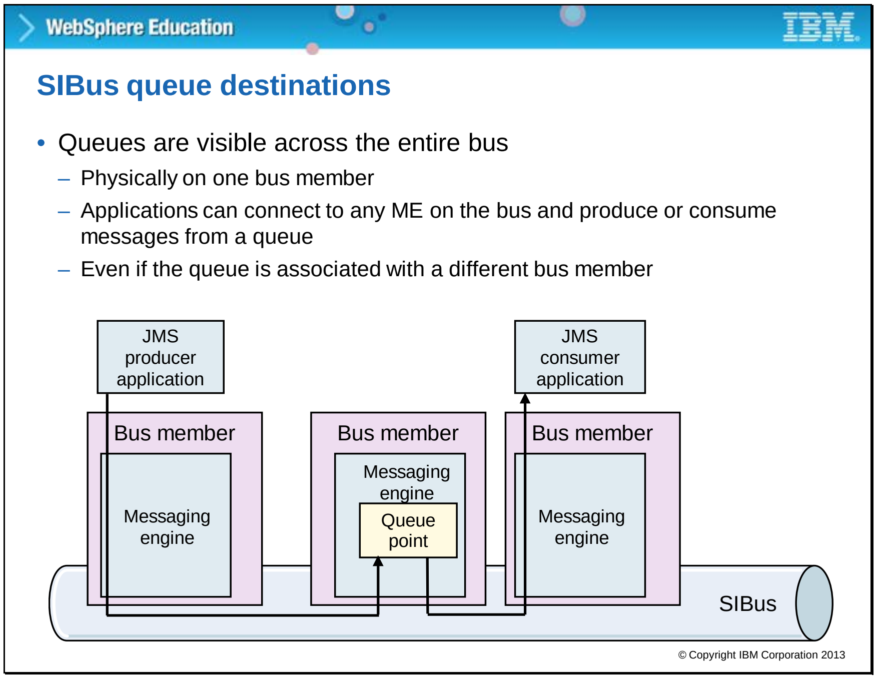
- SIBus destinations are associated with one or more MEs
  - Queues are explicitly assigned to a bus member
  - Topic spaces are associated with all bus members

Title: SIBus destinations

This diagram shows the relationship between SIBus destinations, the messaging engine, and message points. A message point is the location on a messaging engine where messages are held for a bus destination. A message point can be a queue point, a publication point, or a mediation point. A queue point is the message point for a queue destination. When creating a queue destination on a bus, an administrator specifies the bus member that holds the messages for the queue.

This action automatically defines a queue point for each messaging engine that is associated with the specified bus member. A publication point is the message point for a topic space. When creating a topic space destination, an administrator does not need to specify a bus member to hold messages for the topic space. Creating a topic space destination automatically defines a publication point on each messaging engine within the bus.

Slide 23



WebSphere Education

SIBus queue destinations

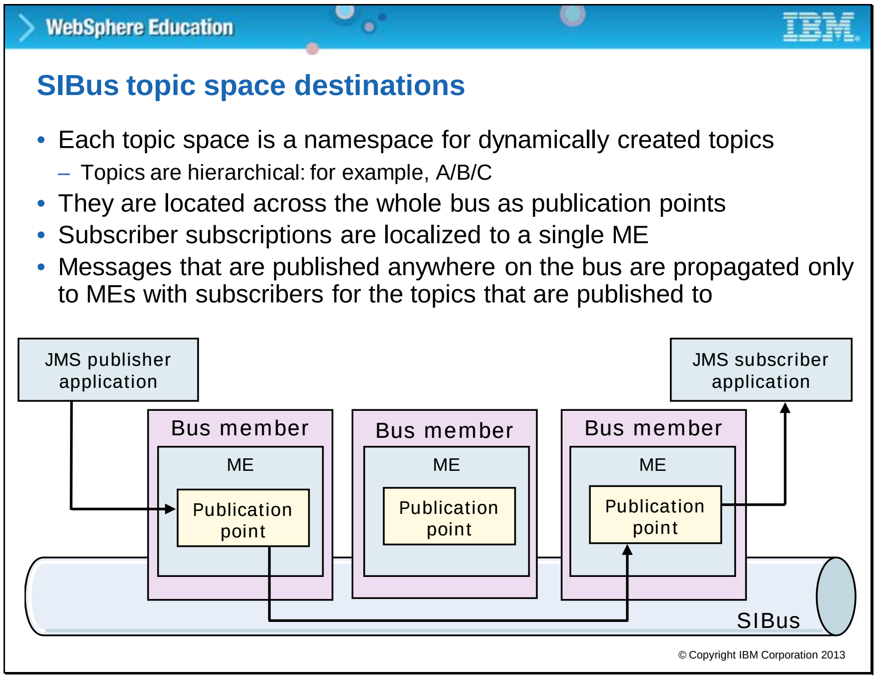
- Queues are visible across the entire bus
  - Physically on one bus member
  - Applications can connect to any ME on the bus and produce or consume messages from a queue
  - Even if the queue is associated with a different bus member

Title: SIBus queue destinations

This diagram illustrates how a SIBus queue destination works. A particular queue destination is associated with one SIBus, and is assigned to one of the bus members. If the bus member is an application server, a single ME is responsible for the queue.

If the bus member is a cluster, then each of the MEs in the cluster supports operations on the queue. Clients do not have to connect to the messaging engine responsible for the queue to access the queue. In this picture, although both the producer and the consumer are connected to different MEs from the one with the queue destination, both of the applications can access the queue.

Slide 24



WebSphere Education

SIBus topic space destinations

- Each topic space is a namespace for dynamically created topics
  - Topics are hierarchical: for example, A/B/C
- They are located across the whole bus as publication points
- Subscriber subscriptions are localized to a single ME
- Messages that are published anywhere on the bus are propagated only to MEs with subscribers for the topics that are published to

JMS publisher application

JMS subscriber application

Bus member

ME

Publication point

Bus member

ME

Publication point

Bus member

ME

Publication point

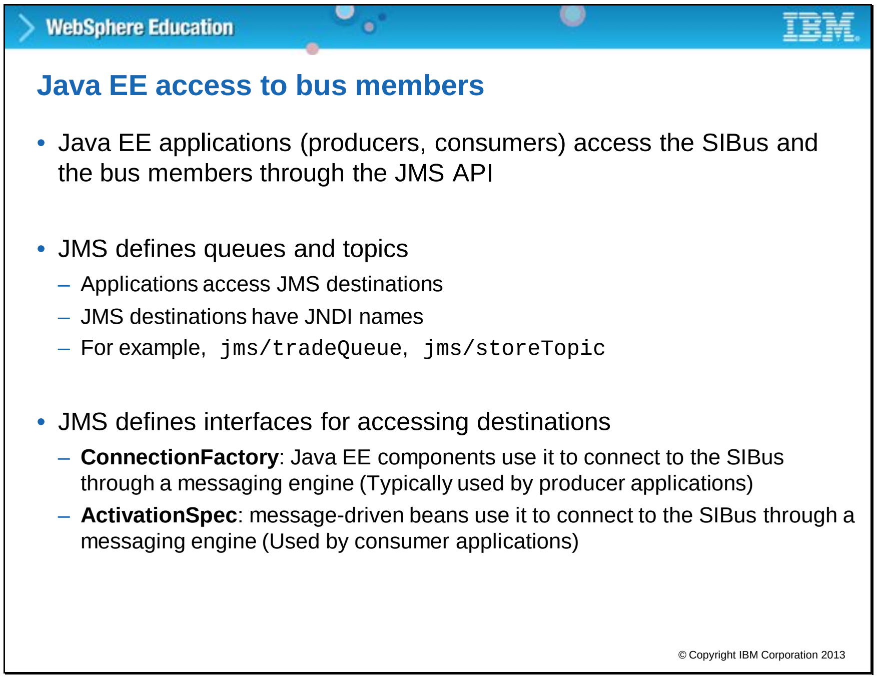
SIBus

© Copyright IBM Corporation 2013

Title: SIBus topic space destinations

This diagram shows how topic space destinations work. Topic space destinations provide support for publish and subscribe messaging. Every ME on the bus can handle operations for the topic spaces that are defined on the bus. Producer applications publish messages to particular topics through a topic space destination. Publishers can use any ME to publish topics. Subscriber applications create subscriptions on particular topics in a topic space. A subscription is associated with the ME that the subscriber application is connected to. Published messages are only propagated to MEs with subscribers. In this picture, there are two subscriber applications. When a publisher creates a message to which it is subscribed, the message is sent to the two MEs where there are subscribers. It is not sent to the ME without any connected subscribers.

Slide 25



WebSphere Education

Java EE access to bus members

- Java EE applications (producers, consumers) access the SIBus and the bus members through the JMS API
- JMS defines queues and topics
  - Applications access JMS destinations
  - JMS destinations have JNDI names
  - For example, `jms/tradeQueue`, `jms/storeTopic`
- JMS defines interfaces for accessing destinations
  - **ConnectionFactory**: Java EE components use it to connect to the SIBus through a messaging engine (Typically used by producer applications)
  - **ActivationSpec**: message-driven beans use it to connect to the SIBus through a messaging engine (Used by consumer applications)

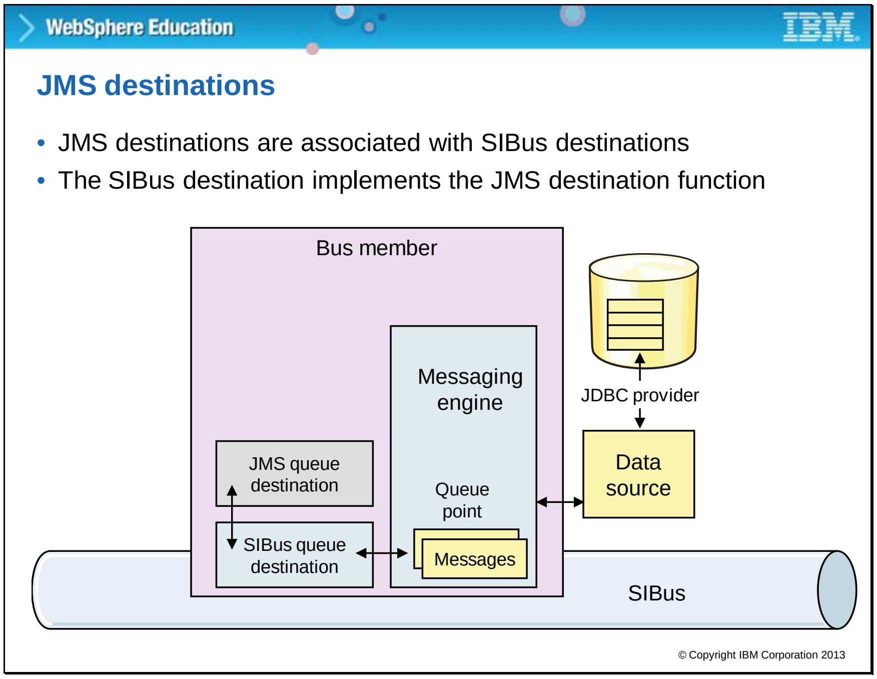
© Copyright IBM Corporation 2013

Title: Java EE access to bus members

Java EE applications, the producers and consumers, access the SIBus and bus members through the JMS API. JMS destinations have JNDI names. JMS defines the interfaces for accessing those destinations. Session EJBs use the connection factory to connect to a JMS provider, in this case, to connect to the SIBus through a messaging engine. The connection factory generates connection objects. The generated connection object holds details of how to create a connection: for example, an IP address, port, and SSL configuration. After connecting to a messaging engine, the session EJB can access destinations on the bus. The destination is not associated with the messaging engine. After connecting, the session EJB can connect to any destination on the bus.

Message-driven beans use the ActivationSpec to connect to the SIBus. The ActivationSpec is a resource adapter. It requires the JNDI name of the JMS destination where messages are consumed. The ActivationSpec calls the `onMessage()` method of the MDB when a message becomes available on the destination. An ActivationSpec can be created for a queue or a topic space destination.

Slide 26



WebSphere Education

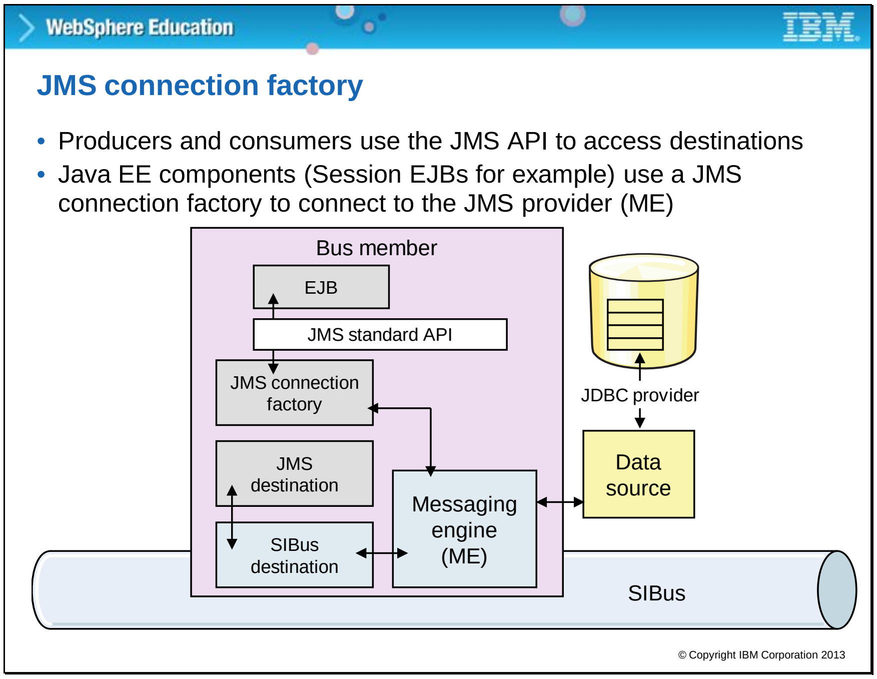
JMS destinations

- JMS destinations are associated with SIBus destinations
- The SIBus destination implements the JMS destination function

Title: JMS destinations

An administrator must create both the JMS destination and the SIBus destination, as shown in the diagram. As mentioned earlier, the message point is created automatically when the SIBus destination is created.

Slide 27



WebSphere Education

JMS connection factory

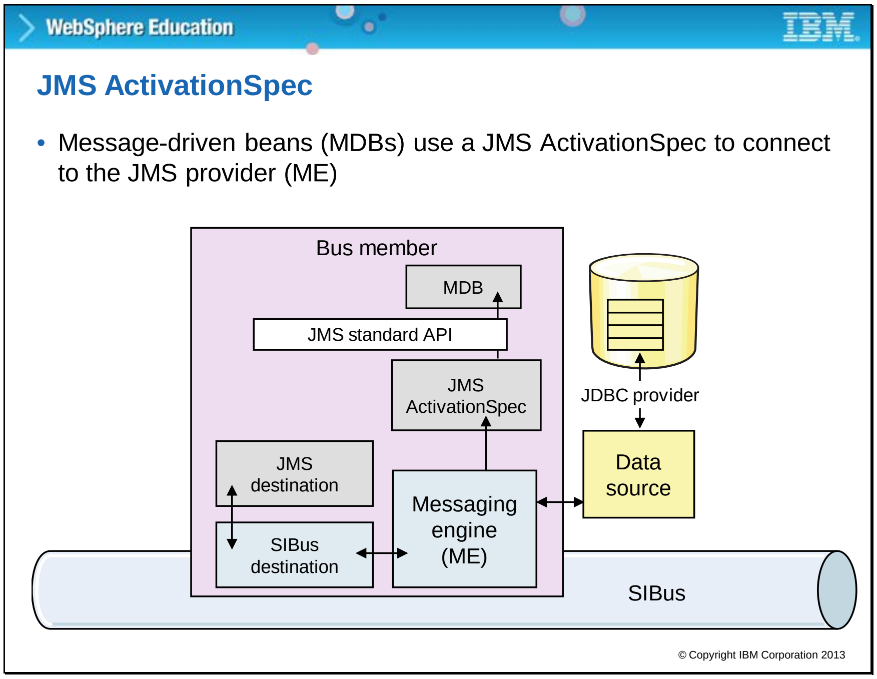
- Producers and consumers use the JMS API to access destinations
- Java EE components (Session EJBs for example) use a JMS connection factory to connect to the JMS provider (ME)

© Copyright IBM Corporation 2013

Title: JMS connection factory

This diagram shows how session EJBs use JMS to access destinations. Session EJBs can be producers or consumers. Here is a typical message flow. A session EJB references a JMS connection factory. The EJB uses the connection factory to create a connection to the message point. The messaging engine and SIBus destination enable the EJB to access the JMS destination.

Slide 28



WebSphere Education

JMS ActivationSpec

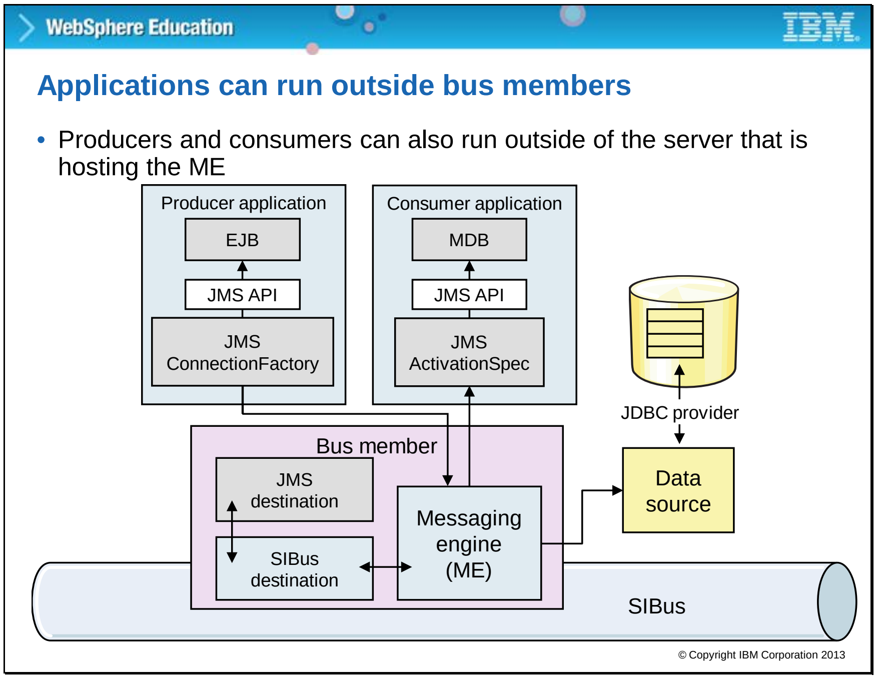
- Message-driven beans (MDBs) use a JMS ActivationSpec to connect to the JMS provider (ME)

Title: JMS ActivationSpec

This diagram shows the relationship between a message driven bean and a destination. Message driven beans use a JMS ActivationSpec to connect to the JMS provider or messaging engine. Message driven beans are associated with an ActivationSpec by using the ejb deployment descriptor or by using annotations.

Administrators create the configuration for an ActivationSpec and associate it with a message destination. A message-driven bean is only a *consumer* of messages, hence the one-way arrows back to the bean in the diagram.

Slide 29



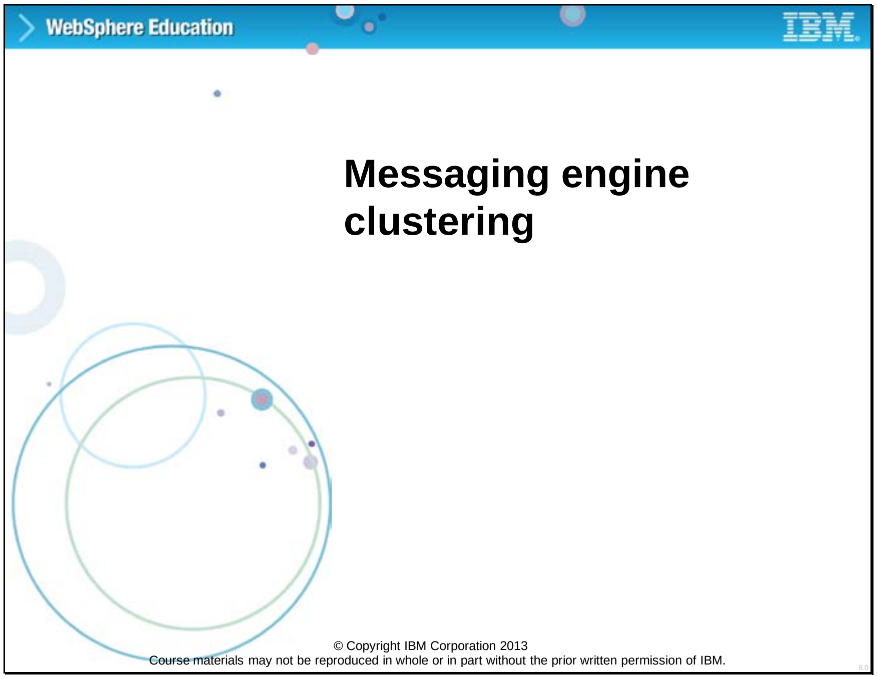
Applications can run outside bus members

- Producers and consumers can also run outside of the server that is hosting the ME

Title: Applications can run outside bus members

As shown in this diagram, both producers and consumers can also run outside of the server that hosts the messaging engine.

Slide 30



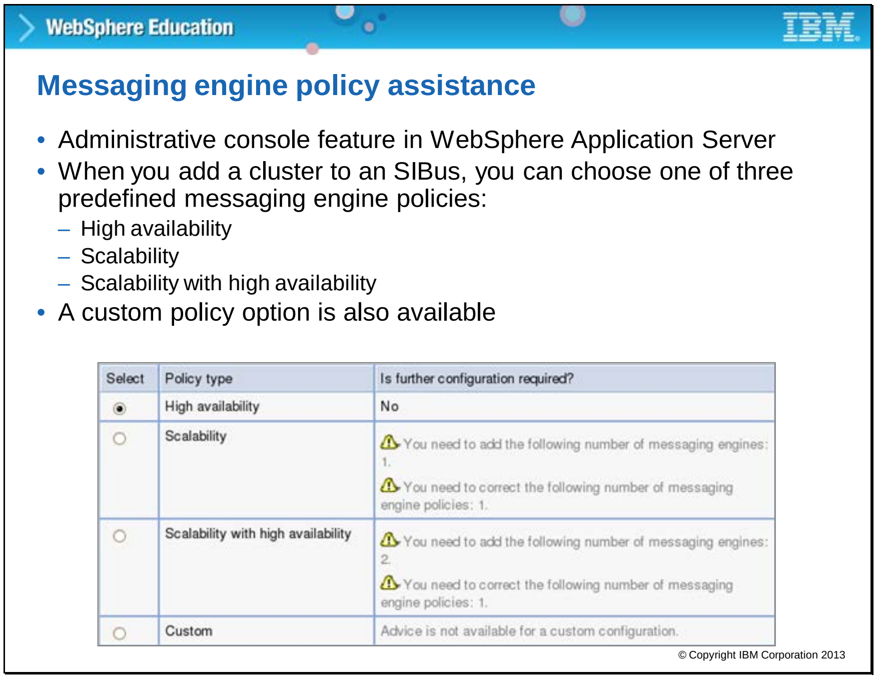
WebSphere Education

Messaging engine clustering

Topic: Messaging engine clustering

This topic describes how to configure a cluster of messaging engines.

Slide 31



Messaging engine policy assistance

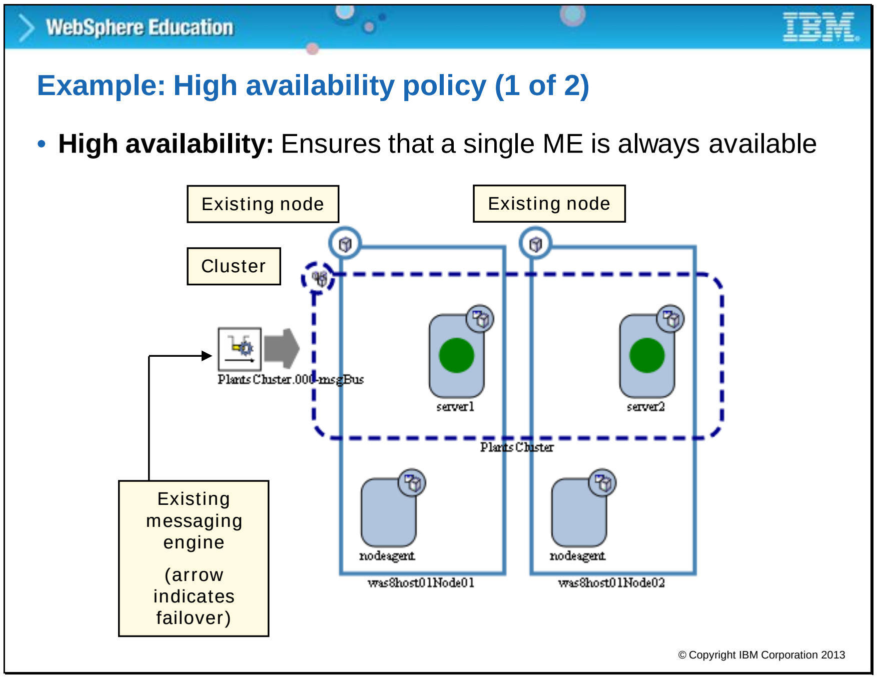
- Administrative console feature in WebSphere Application Server
- When you add a cluster to an SIBus, you can choose one of three predefined messaging engine policies:
  - High availability
  - Scalability
  - Scalability with high availability
- A custom policy option is also available

Select	Policy type	Is further configuration required?
●	High availability	No
○	Scalability	⚠ You need to add the following number of messaging engines: 1. ⚠ You need to correct the following number of messaging engine policies: 1.
○	Scalability with high availability	⚠ You need to add the following number of messaging engines: 2. ⚠ You need to correct the following number of messaging engine policies: 1.
○	Custom	Advice is not available for a custom configuration.

Title: Messaging engine policy assistance

The administrative console provides messaging engine policy assistance whenever you add a cluster as a member of a SIBus. Three predefined policies can be configured: high availability, scalability, and scalability with high availability. There is also an option to select a custom policy, but there is no console assistance for this option.

Slide 32



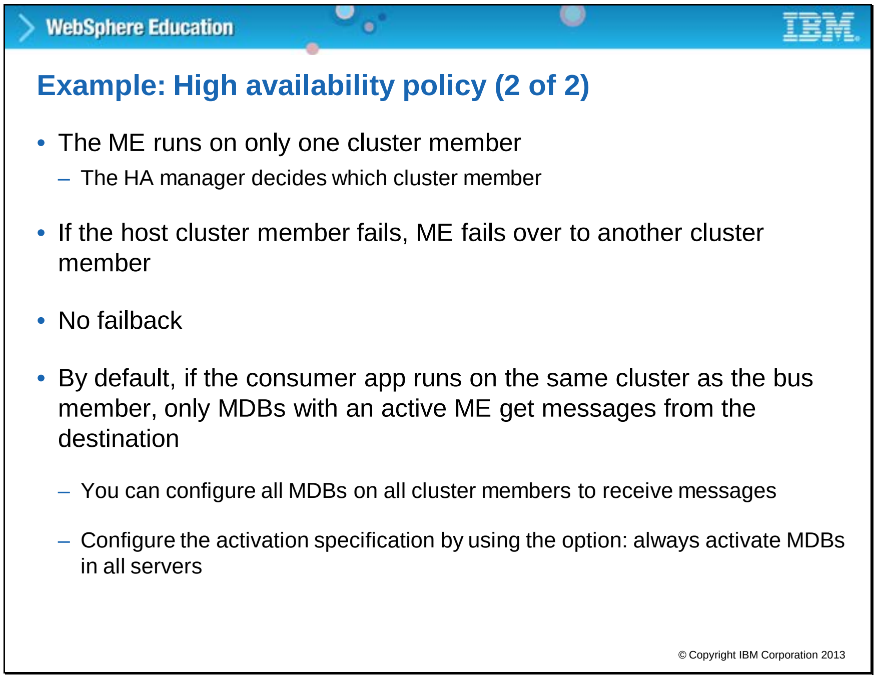
Example: High availability policy (1 of 2)

- **High availability:** Ensures that a single ME is always available

Title: Example: High availability policy (1 of 2)

This diagram illustrates the high availability policy. It ensures that only one messaging engine is always available. By default, only one application server in a cluster has an active messaging engine on a bus. If the server fails, the messaging engine on another server in the cluster is activated. This method provides failover, but no workload management. The server with the active messaging engine has local access to the bus, but the rest of the servers in the cluster access the bus remotely by connecting to the active messaging engine. Servers accessing the bus remotely can use asynchronous messages from a remote messaging engine. However, an instance of a message-driven bean (MDB) deployed to the cluster can consume only from a local messaging engine. Because everything is funneled through one messaging engine, that messaging engine can become a performance bottleneck.

Slide 33



WebSphere Education

Example: High availability policy (2 of 2)

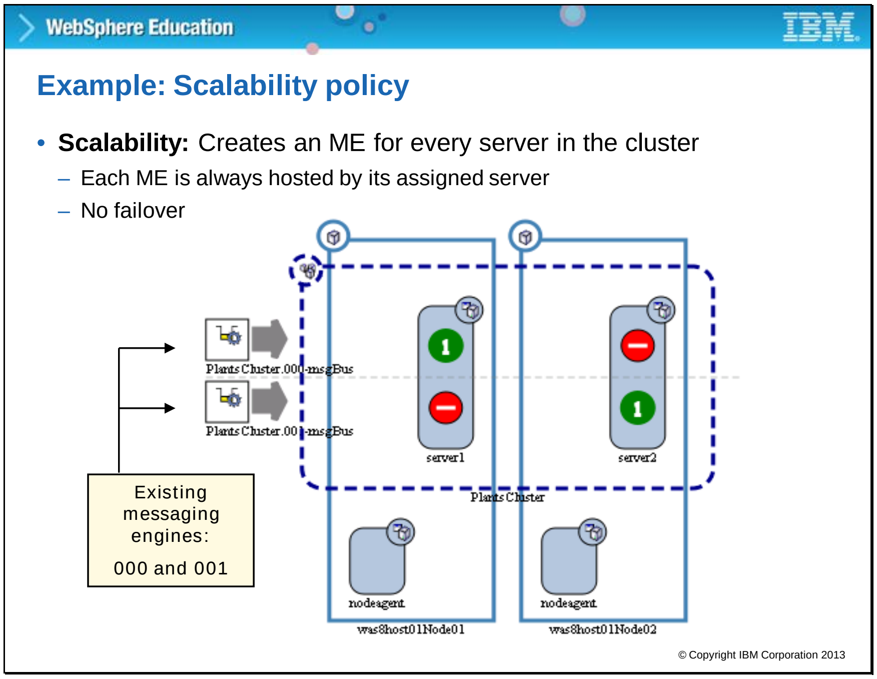
- The ME runs on only one cluster member
  - The HA manager decides which cluster member
- If the host cluster member fails, ME fails over to another cluster member
- No failback
- By default, if the consumer app runs on the same cluster as the bus member, only MDBs with an active ME get messages from the destination
  - You can configure all MDBs on all cluster members to receive messages
  - Configure the activation specification by using the option: always activate MDBs in all servers

© Copyright IBM Corporation 2013

Title: Example: High availability policy (2 of 2)

To review, for the high availability policy, the ME runs on only one cluster member; the high availability manager decides which one. If the host cluster member fails, the ME fails over to another cluster member, which continues to host the ME even when the original cluster member is running again. There is no failback. By default only MDBs on the host cluster member can get messages from the destination, but you can configure all consumers on all cluster members to receive messages. You must enable the option that is called “Always activate MDBs in all servers”, for the activation specification.

Slide 34



WebSphere Education

Example: Scalability policy

- **Scalability:** Creates an ME for every server in the cluster
  - Each ME is always hosted by its assigned server
  - No failover

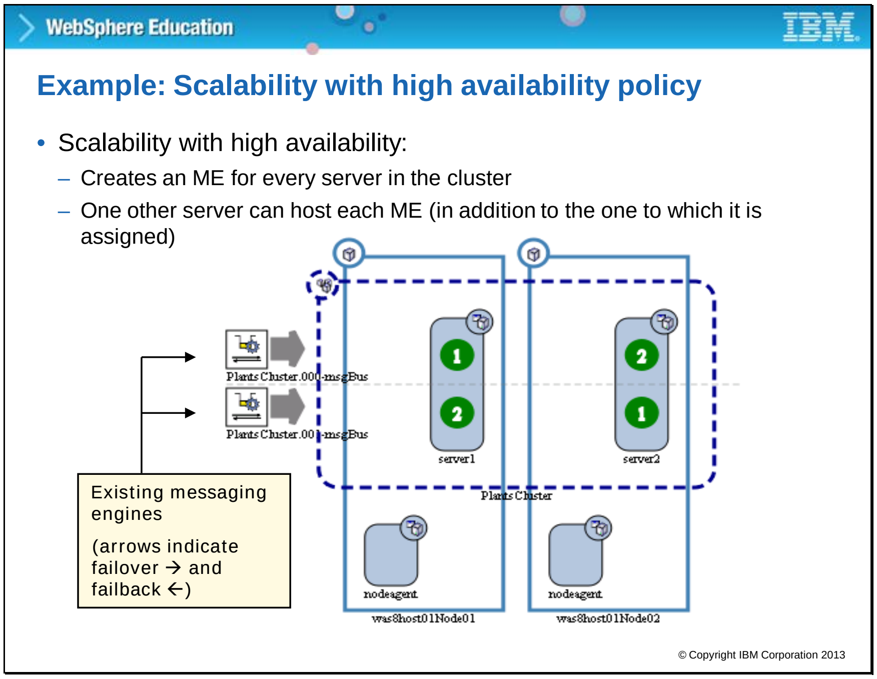
Existing messaging engines:

000 and 001

Title: Example: Scalability policy

This diagram illustrates the scalability policy. Each cluster member has its own ME. If one cluster member fails, its ME does not fail over to another cluster member. This policy is good for scalability, but there is no failover.

Slide 35



WebSphere Education

Example: Scalability with high availability policy

- Scalability with high availability:
  - Creates an ME for every server in the cluster
  - One other server can host each ME (in addition to the one to which it is assigned)

Existing messaging engines

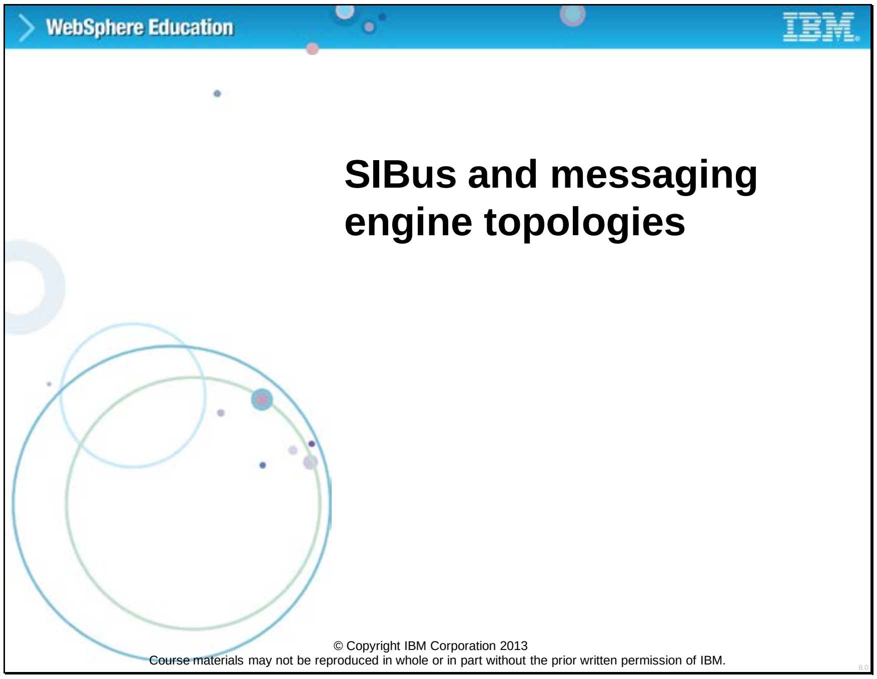
(arrows indicate failover → and failback ←)

© Copyright IBM Corporation 2013

Title: Example: Scalability with high availability policy

This policy combines scalability with high availability. Each server in the cluster is able to host at most two MEs, its own and one for another cluster member. If a cluster member fails, there is failover to another cluster member.

Slide 36



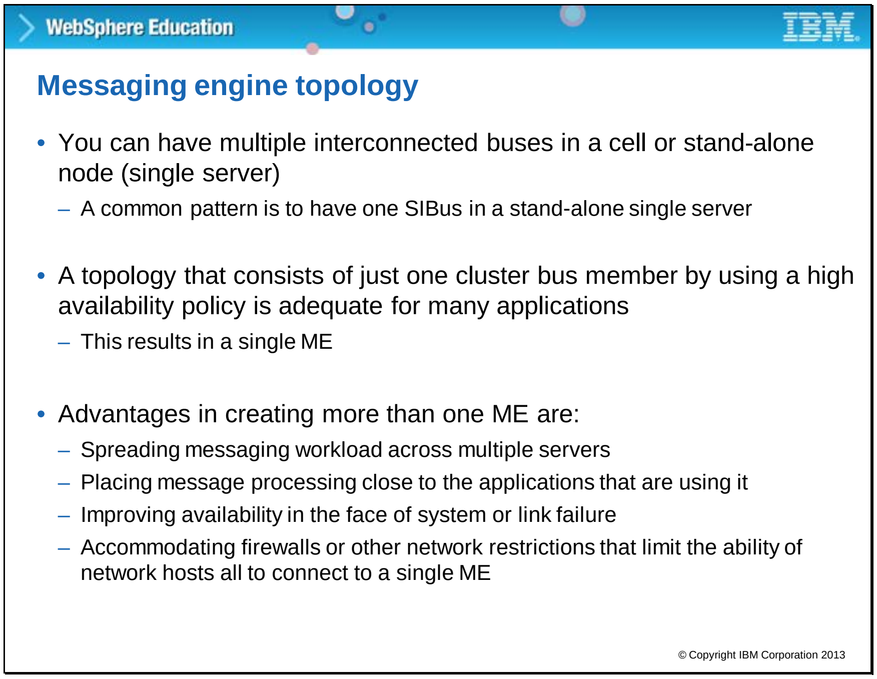
WebSphere Education

SIBus and messaging engine topologies

Topic: SIBus and messaging engine topologies

This topic covers various topologies for messaging engines, SIBuses, and the WebSphere Network Deployment cell.

Slide 37



WebSphere Education

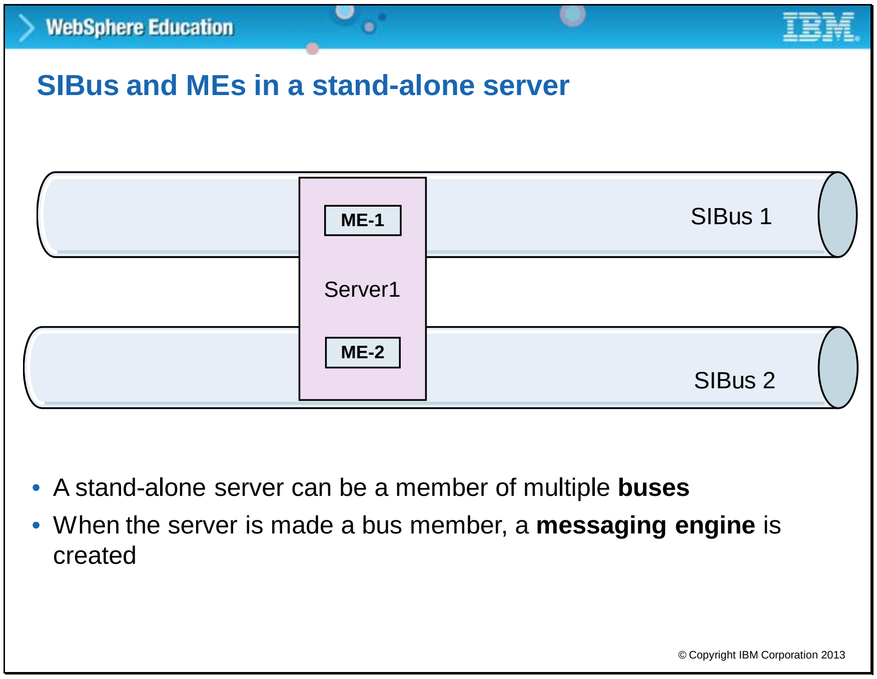
Messaging engine topology

- You 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
- A topology that consists of just one cluster bus member by using a high availability policy is adequate for many applications
  - This results in a single ME
- Advantages in creating more than one ME 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 ME

Title: Messaging engine topology

Messaging engine topologies can have multiple interconnected buses in a cell or stand-alone application server. A common pattern is to have one SIBus in a stand-alone application server. The default topology of just one ME on a bus is adequate for many applications. There are several advantages in deploying more than one ME and linking them together. This topology spreads messaging workload across multiple servers, and places message processing close to the applications that are using it. Therefore, availability is improved in the face of system or link failures, where firewalls or other network restrictions limit the ability of network hosts to connect to a single ME.

Slide 38



WebSphere Education

SIBus and MEs in a stand-alone server

- A stand-alone server can be a member of multiple **buses**
- When the server is made a bus member, a **messaging engine** is created

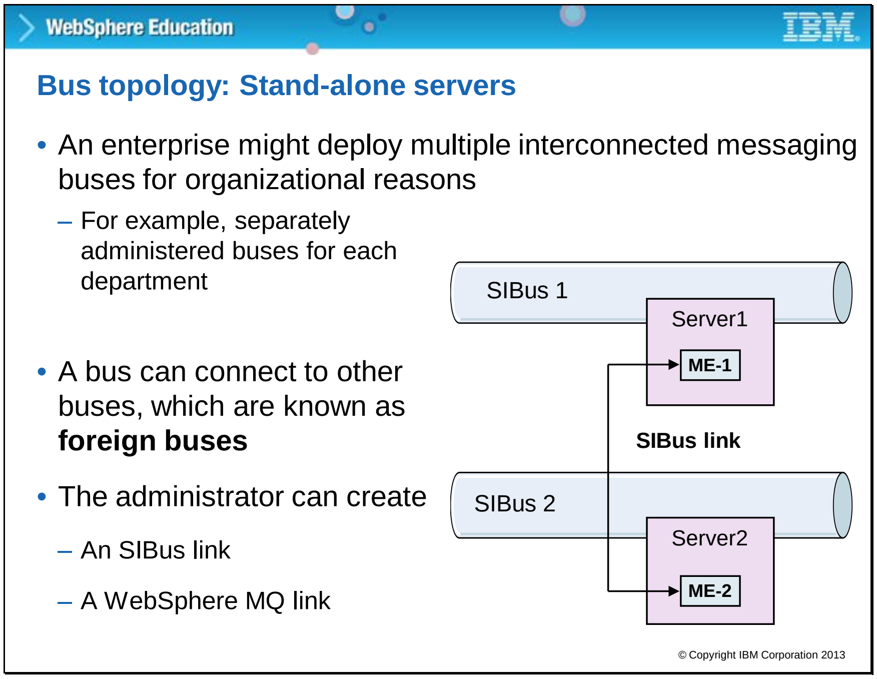
Title: SIBus and MEs in a stand-alone server

A typical WebSphere Application Server configuration is to define a single bus for each WebSphere cell and to run a default ME in each application server that is a member of this bus. This member serves as the embedded JMS provider and provides SIBus capabilities for web service applications. Messaging engines are running inside an application server together with Java EE applications and their containers.

You can have multiple MEs running in the same application server.

This configuration allows messaging applications to have a dedicated messaging engine and bus each with its own topology and set of resources. The diagram shows one application server that is connected to two different SIBuses. The application server is running two messaging engines, each connected to a different SIBus.

Slide 39



WebSphere Education

Bus topology: Stand-alone servers

- 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 can create
  - An SIBus link
  - A WebSphere MQ link

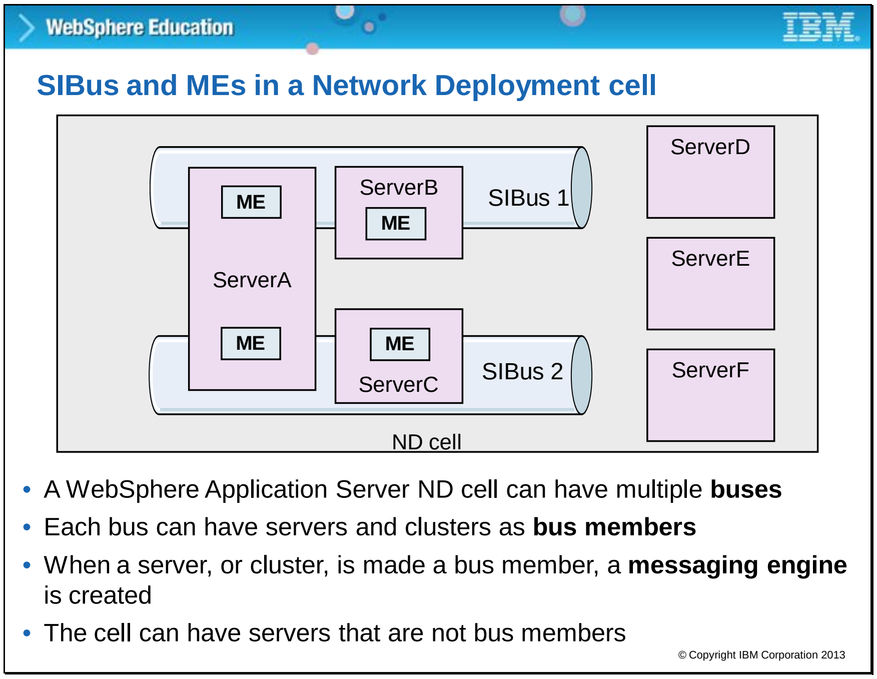
© Copyright IBM Corporation 2013

Title: Bus topology: Stand-alone servers

You can use multiple interconnected buses for organization reasons. A bus can connect to other buses, which are referred to as foreign buses. If messaging engines are on different buses, applications can use those different buses, each with its own topology and set of resources. The inter-bus links might reflect the distribution of buses across organizations, across departments within organizations, or perhaps the separation of test and production facilities.

To create a link to a foreign bus, the administrator first creates a virtual link from the local bus to the foreign bus. The administrator then creates a physical gateway link from a messaging engine in the local bus to the foreign bus. The diagram shows two stand-alone application servers, each connected to a different SIBus. The application servers can connect through the inter-bus link.

Slide 40



WebSphere Education

IBM

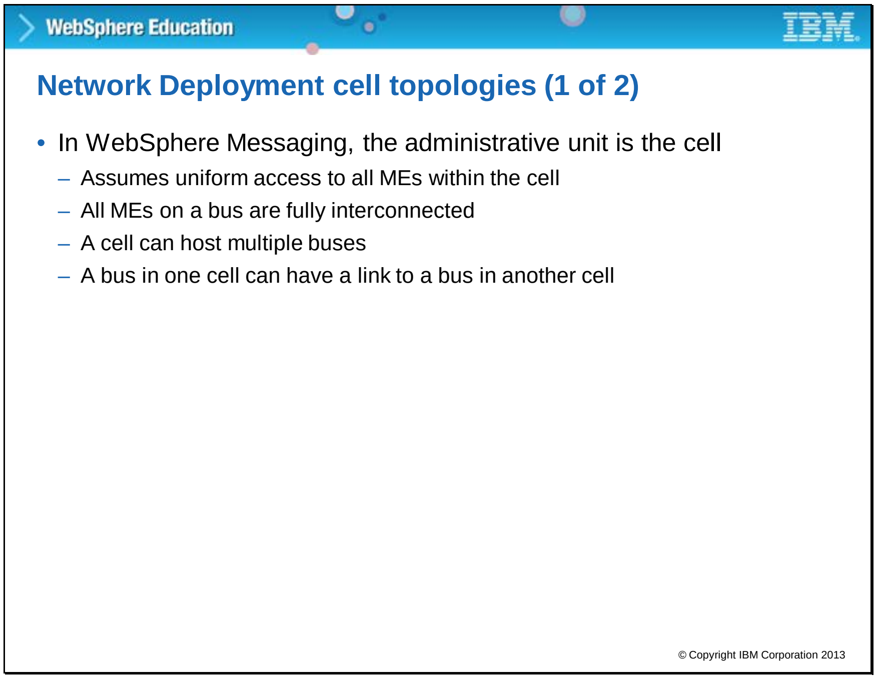
SIBus and MEs in a Network Deployment cell

- A WebSphere Application Server ND 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, a **messaging engine** is created
- The cell can have servers that are not bus members

Title: SIBus and MEs in a network deployment cell

This diagram shows the relationships between messaging engines within a network deployment cell. You can have multiple buses in cell, and each bus can have different servers or clusters as bus members. The diagram shows one cell, containing two buses, and three federated nodes or application servers, each with a messaging engine. Server A is a member of both SIBus 1 and 2. Server B is member of SIBus 1. Server C is a member of SIBus 2.

Slide 41



WebSphere Education

Network Deployment cell topologies (1 of 2)

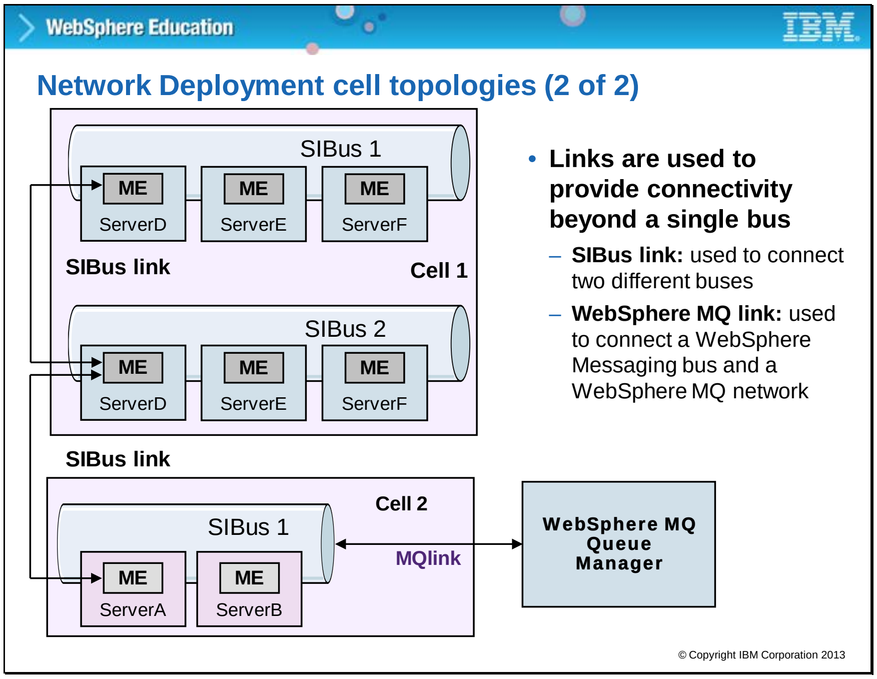
- In WebSphere Messaging, the administrative unit is the cell
  - Assumes uniform access to all MEs within the cell
  - All MEs on a bus are fully interconnected
  - A cell can host multiple buses
  - A bus in one cell can have a link to a bus in another cell

© Copyright IBM Corporation 2013

Title: Network deployment cell topologies (1 of 2)

In a network deployment cell, all messaging engines are fully connected and it assumes uniform access within the cell. It is possible that a bus within one cell, has a link to a bus in another cell. When you have a bus, every destination on that bus must have a unique name, so to a certain extent, the bus is a namespace. If you have a second bus, you can have another destination with the same name as one in the first bus. When you go to the level of the JMS proxies in the JNDI namespace, a JMS destination specifies only the destination name, not the bus name. The connection factory specifies the name of the bus to connect to. It would be possible to use the same JMS destination with two different connection factories, specifying different bus names, to connect to two different destinations.

Slide 42



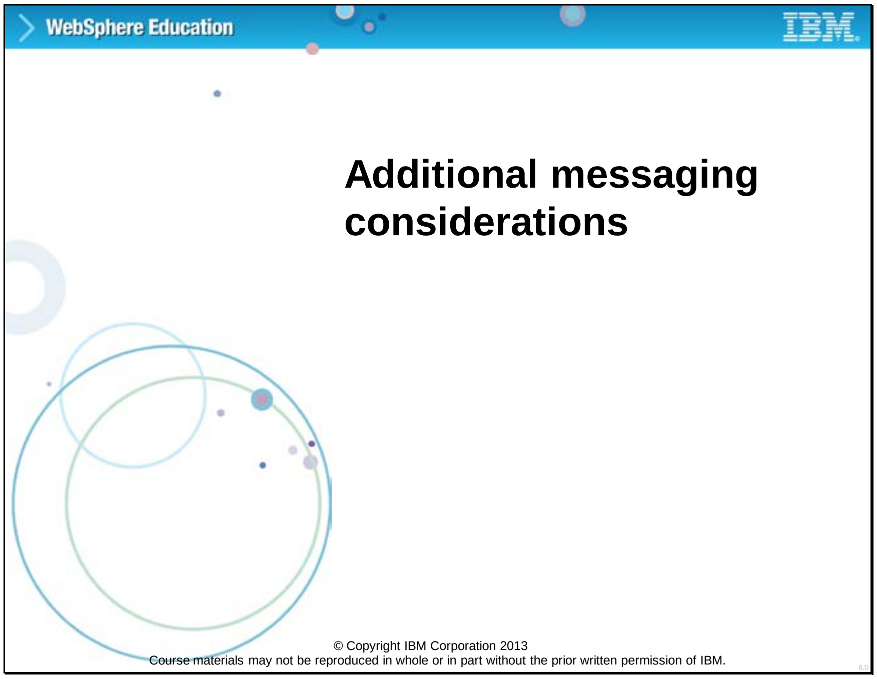
Network Deployment cell topologies (2 of 2)

- **Links are used to provide connectivity beyond a single bus**
  - **SIBus link:** used to connect two different buses
  - **WebSphere MQ link:** used to connect a WebSphere Messaging bus and a WebSphere MQ network

Title: Network deployment cell topologies (2 of 2)

Besides, using the inter-bus link as described previously, you can also link to WebSphere MQ. Connectivity between a messaging engine and an WebSphere MQ queue manager is established by defining an MQLink. One of the primary functions of the MQLink is to convert between the formats and protocols that WebSphere MQ uses and WebSphere messaging uses. A component that is called the JMS/MQ protocol adapter, handles the conversion. The diagram shows two network deployment cells. The first cell contains two SIBuses linked together by an inter-bus link. The second cell contains one SIBus, which is linked to one of the buses in the first cell, and is also linked to WebSphere MQ by using MQLink.

Slide 43

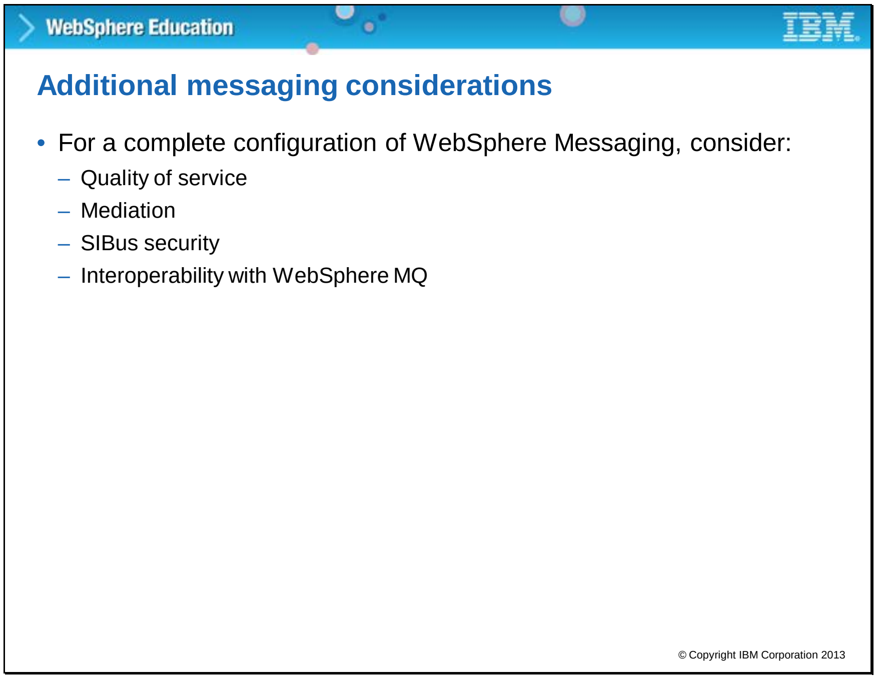


Additional messaging considerations

Topic: Other messaging considerations

This topic covers issues such as quality of service, mediation, SIBus security, and interoperability with WebSphere MQ.

Slide 44



WebSphere Education IBM

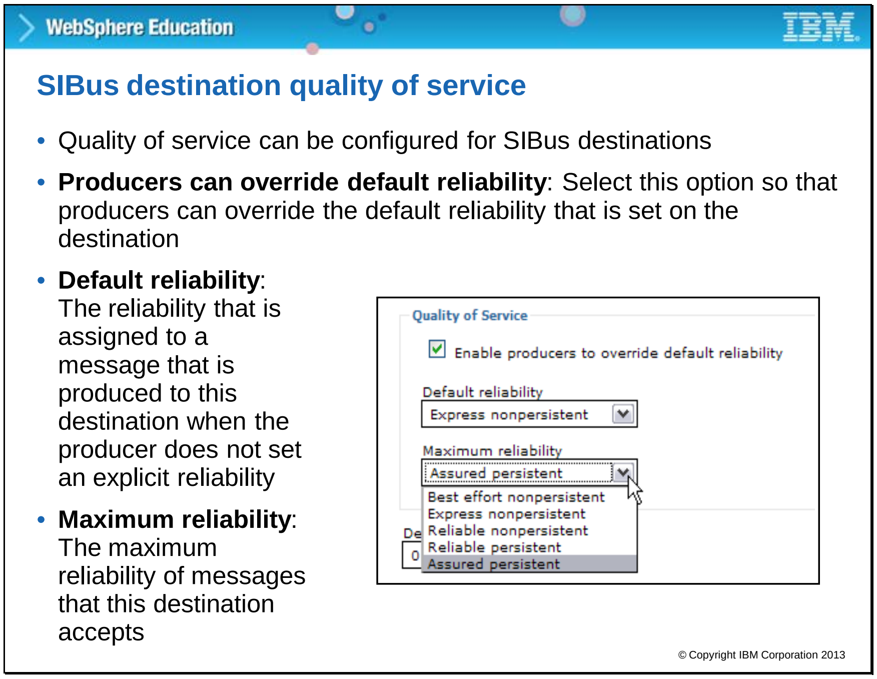
Additional messaging considerations

- For a complete configuration of WebSphere Messaging, consider:
  - Quality of service
  - Mediation
  - SIBus security
  - Interoperability with WebSphere MQ

Title: Other messaging considerations

To complete the configuration of your messaging environment, consider implementing these additional options: Quality of service, mediation, SIBus security, and interoperability with WebSphere MQ.

Slide 45



SIBus destination quality of service

- Quality of service can be configured for SIBus destinations
- **Producers can override default reliability**: Select this option so that producers can override the default reliability that is set on the destination
- **Default reliability**: The reliability that is assigned to a message that is produced to this destination when the producer does not set an explicit reliability
- **Maximum reliability**: The maximum reliability of messages that this destination accepts

Quality of Service

☑ Enable producers to override default reliability

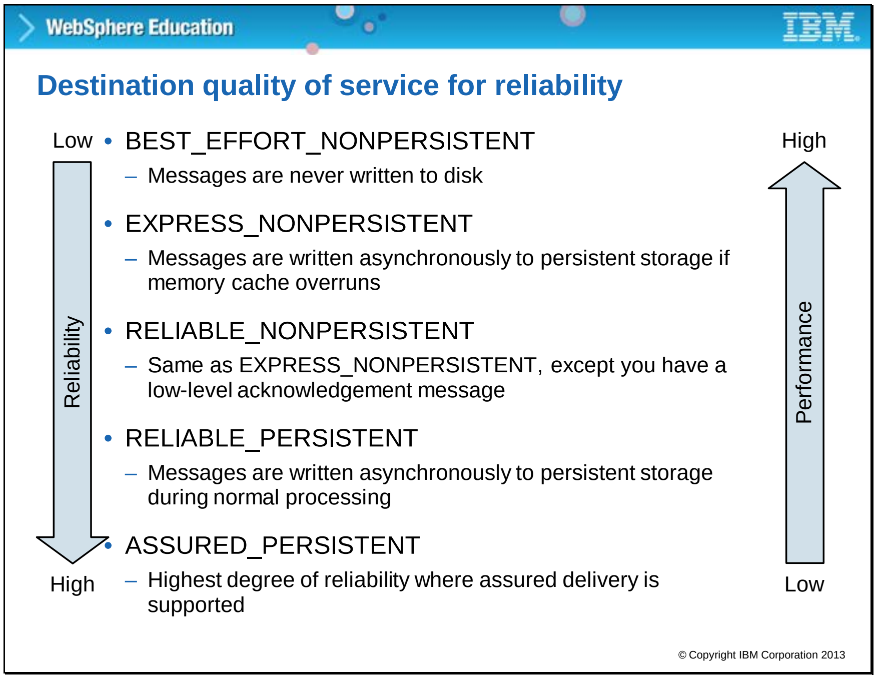
Default reliability: Express nonpersistent

Maximum reliability: Assured persistent

Title: SIBus Destination quality of service

You can define quality of service to determine the level of message persistence and optimize for failover or performance, whichever is most appropriate for the application. Producers are able to override the quality of service that is defined for a destination. The default reliability is assigned to a message produced to the destination. Maximum reliability defines the maximum QoS for this destination.

Slide 46



WebSphere Education

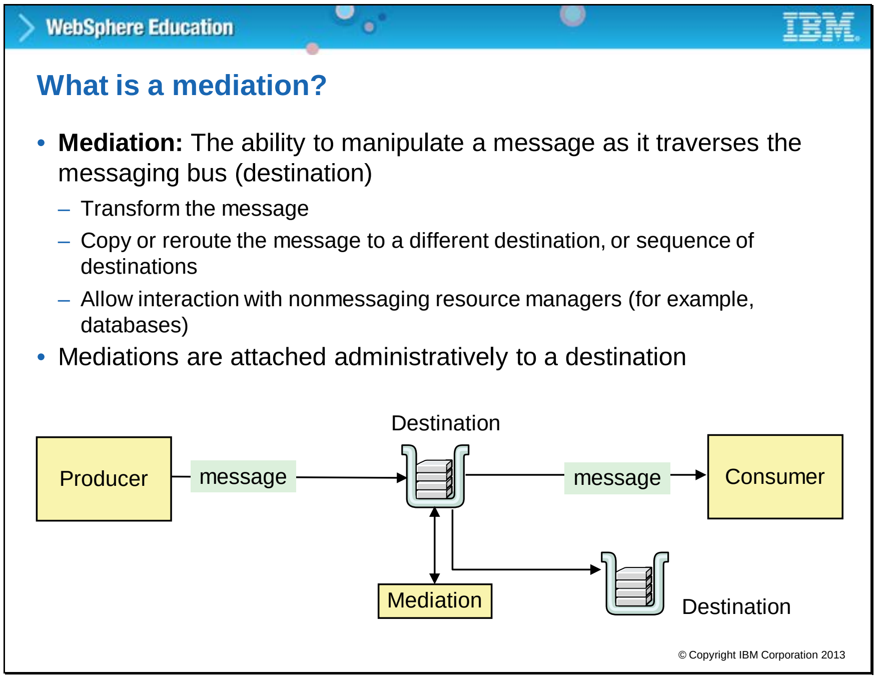
Destination quality of service for reliability

- BEST_EFFORT_NONPERSISTENT
  - Messages are never written to disk
- EXPRESS_NONPERSISTENT
  - Messages are written asynchronously to persistent storage if memory cache overruns
- RELIABLE_NONPERSISTENT
  - Same as EXPRESS_NONPERSISTENT, except you have a low-level acknowledgement message
- RELIABLE_PERSISTENT
  - Messages are written asynchronously to persistent storage during normal processing
- ASSURED_PERSISTENT
  - Highest degree of reliability where assured delivery is supported

Title: Destination quality of service for reliability

Quality of service is configured through the administrative console under **Service Integration > Buses > Destinations** page. You can also specify quality of service within the application. The settings range **from best effort nonpersistent**, which is low on the reliability scale, but high on the performance scale, to **assured persistent**, which is high on reliability and low on performance.

Slide 47



WebSphere Education

IBM

What is a mediation?

- **Mediation:** The ability to manipulate a message as it traverses the messaging bus (destination)
  - Transform the message
  - Copy or reroute the message to a different destination, or sequence of destinations
  - Allow interaction with nonmessaging resource managers (for example, databases)
- Mediations are attached administratively to a destination

Producer

message

Destination

message

Consumer

Mediation

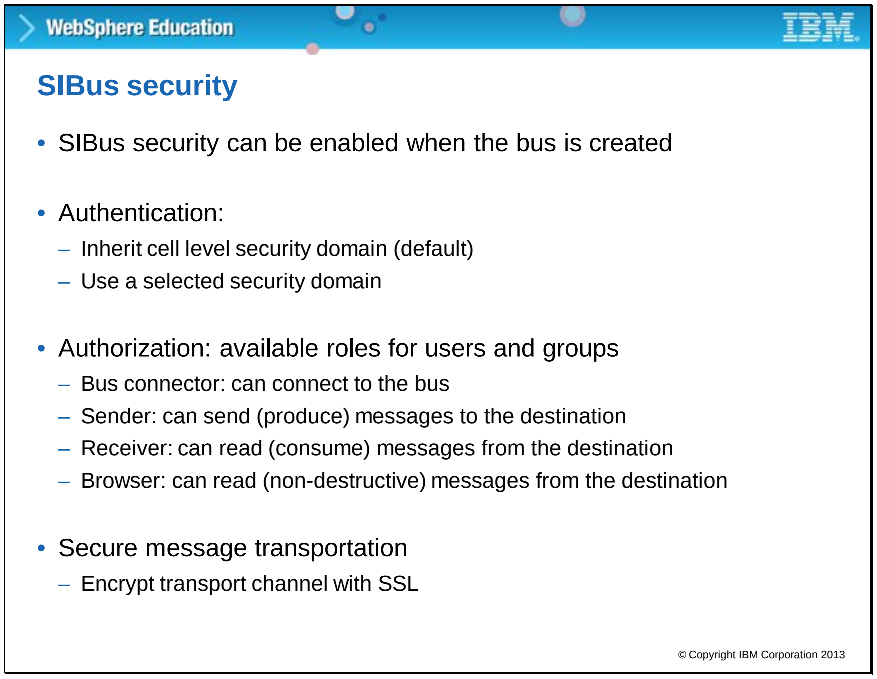
Destination

© Copyright IBM Corporation 2013

Title: What is a mediation?

A mediation is a programmable extension to the messaging capabilities of WebSphere Application Server that can simplify connecting systems, services, applications, or components that use messaging. It can be used to interact with non-messaging resources, such as a database. A mediation is used to process in-flight messages. The type of processing a mediation can handle includes: Modifying or transforming a message, copying or routing a message to other destinations, and allowing or disallowing a message to be delivered based on some conditional logic in the mediation. Mediation is attached administratively to a destination.

Slide 48


WebSphere Education
IBM

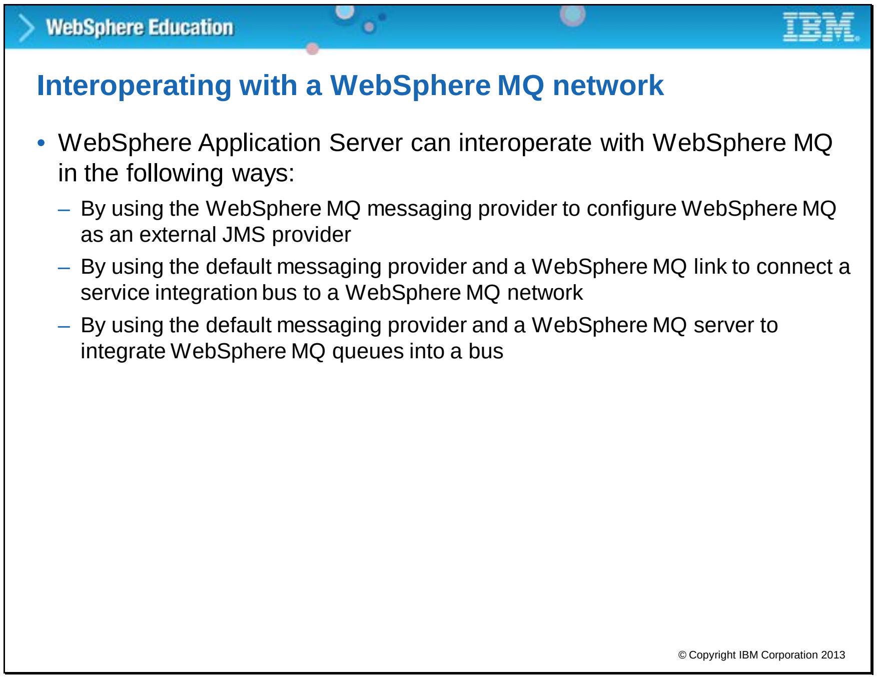
SIBus security

- SIBus security can be enabled when the bus is created
- Authentication:
  - Inherit cell level security domain (default)
  - Use a selected security domain
- Authorization: available roles for users and groups
  - Bus connector: can connect to the bus
  - Sender: can send (produce) messages to the destination
  - Receiver: can read (consume) messages from the destination
  - Browser: can read (non-destructive) messages from the destination
- Secure message transportation
  - Encrypt transport channel with SSL

Title: SIBus security

Several new security panels are provided in the administrative console to help configure role-based access to bus resources. Previously, the only way to administer the role-based authorization mechanism was through several commands that are started in wsadmin. You can enable security when you create a SIBus to control access to the SIBus. You can allow the SIBus to inherit the cell level security or select a security domain. The roles that you can assign to users and groups include bus connector, sender, receiver, and browser. Users in the bus connector role can connect to a bus. Senders can produce messages to a destination; receivers can use messages from a destination, and browsers can read messages from a destination. You can also apply SSL encryption to the transport channel.

Slide 49



WebSphere Education

IBM

Interoperating with a WebSphere MQ network

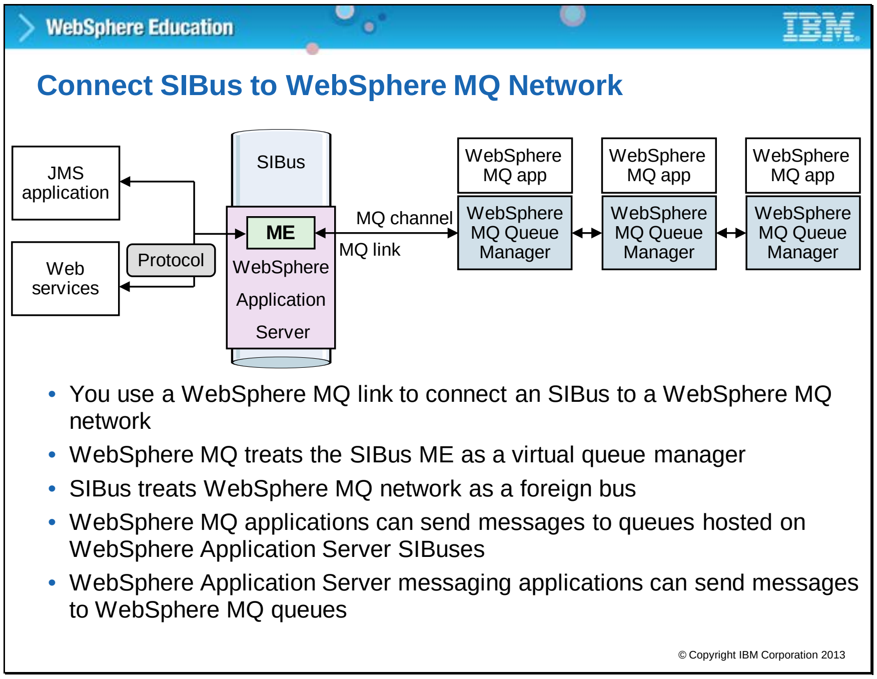
- WebSphere Application Server can interoperate with WebSphere MQ in the following ways:
  - By using the WebSphere MQ messaging provider to configure WebSphere MQ as an external JMS provider
  - By using the default messaging provider and a WebSphere MQ link to connect a service integration bus to a WebSphere MQ network
  - By using the default messaging provider and a WebSphere MQ server to integrate WebSphere MQ queues into a bus

© Copyright IBM Corporation 2013

Title: Interoperating with a WebSphere MQ network

As previously described, WebSphere messaging can interact with WebSphere MQ by using an WebSphere MQ link. You can also interoperate with WebSphere MQ by configuring WebSphere MQ as an external JMS provider or by using a WebSphere MQ server. Alternatively, you can use WebSphere MQ as your messaging provider. Each type of connectivity is designed for different situations, and provides different advantages. Choose the most appropriate interoperation method for each of your messaging applications.

Slide 50



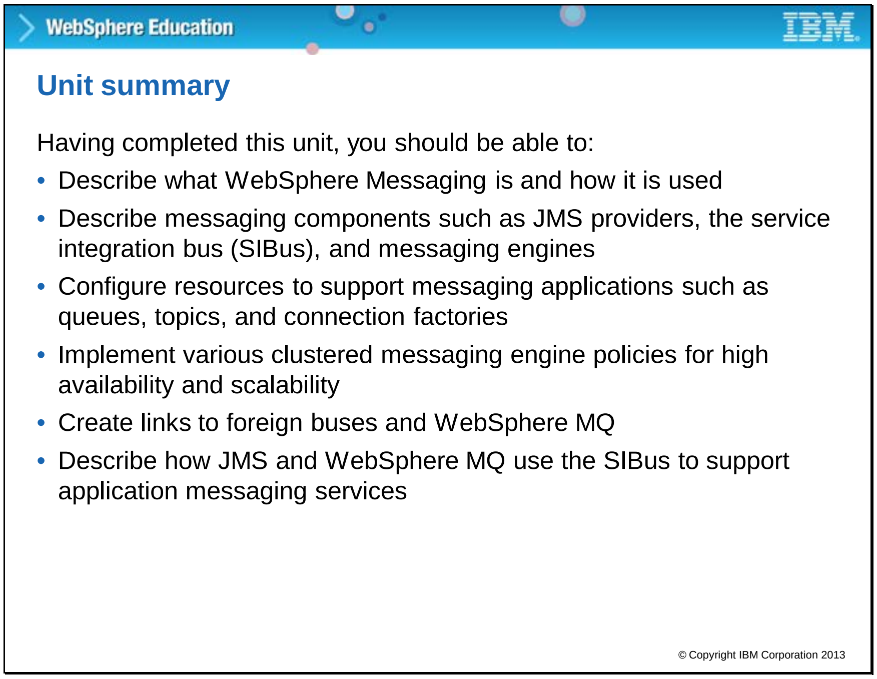
Connect SIBus to WebSphere MQ Network

- You use a WebSphere MQ link to connect an SIBus to a WebSphere MQ network
- WebSphere MQ treats the SIBus ME as a virtual queue manager
- SIBus treats WebSphere MQ network as a foreign bus
- WebSphere MQ applications can send messages to queues hosted on WebSphere Application Server SIBuses
- WebSphere Application Server messaging applications can send messages to WebSphere MQ queues

Title: Connect SIBus to WebSphere MQ network

A WebSphere MQ link provides a server-to-server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network. When you use a WebSphere MQ link, the messaging engine appears to the WebSphere MQ network as a virtual queue manager, and the WebSphere MQ network looks like a foreign bus to the service integration bus. A WebSphere MQ link allows WebSphere Application Server applications to send point-to-point messages to WebSphere MQ queues, which are defined as destinations in the service integration bus. A WebSphere MQ link also allows WebSphere MQ applications to send point-to-point messages to destinations in the service integration bus, which are defined as remote queues in WebSphere MQ. The link also allows WebSphere Application Server applications to subscribe to messages published by WebSphere MQ applications, and WebSphere MQ applications to subscribe to messages published by WebSphere Application Server applications. The link ensures that messages are converted between the formats that WebSphere Application Server uses and those formats that WebSphere MQ uses.

Slide 51



WebSphere Education

Unit summary

Having completed this unit, you should be able to:

- Describe what WebSphere Messaging is and how it is used
- Describe messaging components such as JMS providers, the service integration bus (SIBus), and messaging engines
- Configure resources to support messaging applications such as queues, topics, and connection factories
- Implement various clustered messaging engine policies for high availability and scalability
- Create links to foreign buses and WebSphere MQ
- Describe how JMS and WebSphere MQ use the SIBus to support application messaging services

Title: Unit summary

Having completed this unit, you should be able to:

- Describe what WebSphere Messaging is and how it is used
- Describe messaging components such as JMS providers, the service integration bus (SIBus), and messaging engines
- Configure resources to support messaging applications such as queues, topics, and connection factories
- Implement different clustered messaging engine policies for high availability and scalability
- Create links to foreign buses and WebSphere MQ
- Describe how JMS uses the service integration bus (SIBus) to support application messaging services