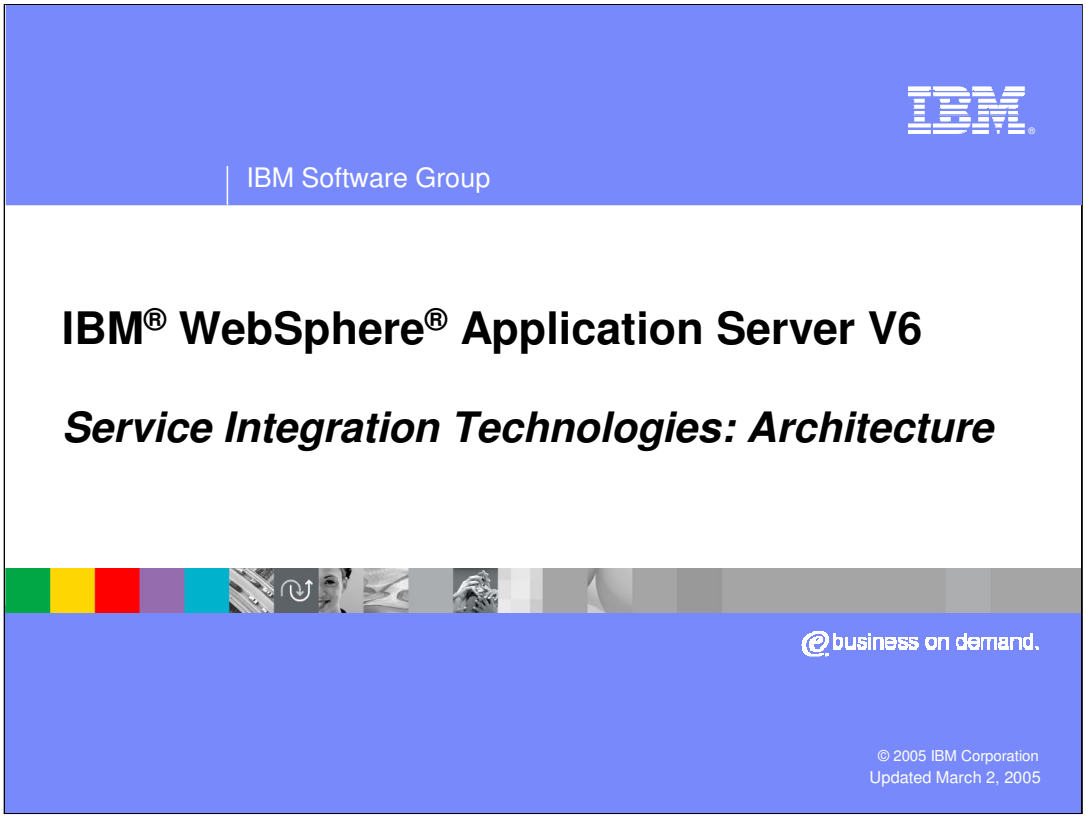


This presentation will focus on the architecture of the IBM Service Integration Technologies.

Goals

- Describe the core components of IBM Service Integration Technologies
- Focus on Java™ Messaging Service (JMS) messaging using Service Integration Bus
- Covered in another presentation:
  - Web Services integration with Service Integration Technologies

Agenda

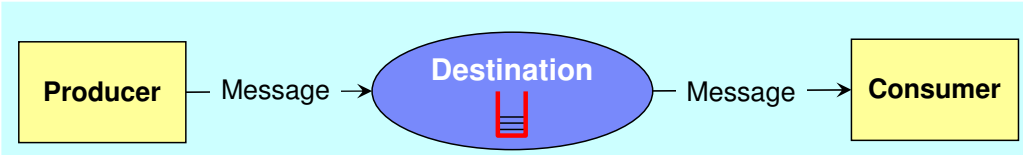
- Service Integration Technologies architecture
  - Service Integration Bus
  - Bus Member
  - Messaging Engine
  - Destinations
  - Message Store
  - Mediation
- Quality of service
- High Availability and scalability
- Topologies
- Interoperability
- Summary

Section

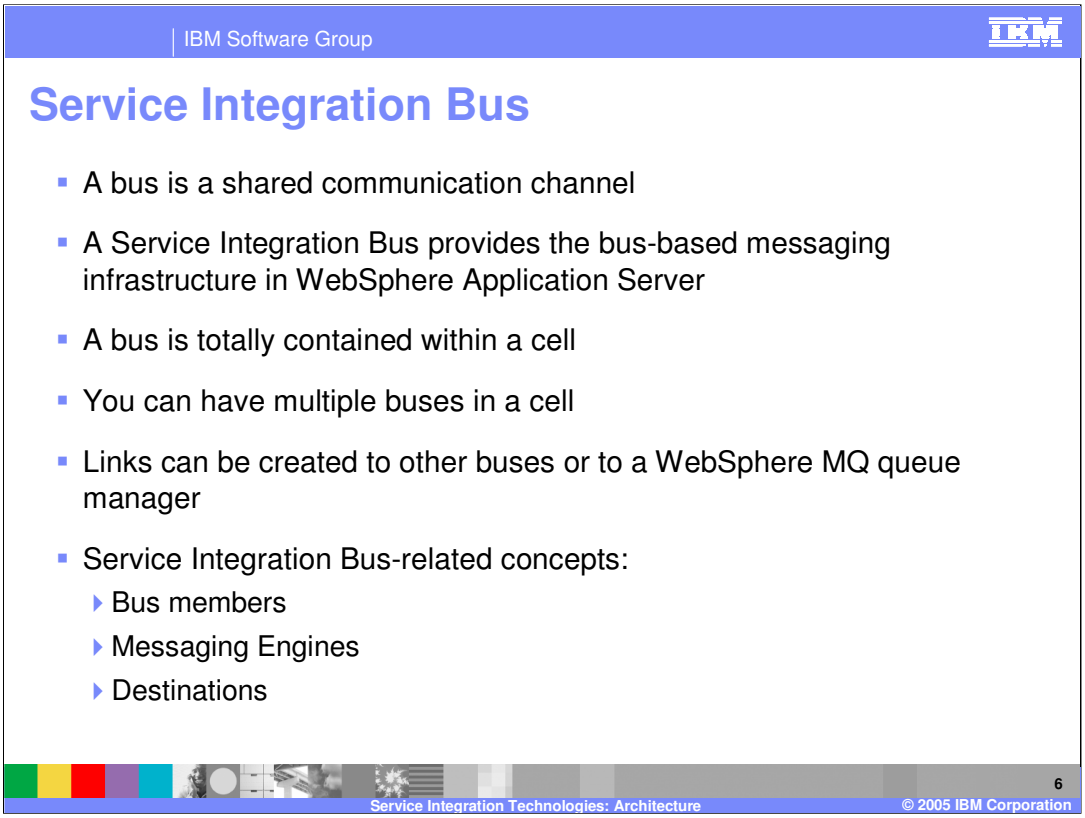
Service Integration Bus

Architecture and Components

Basic Message Flow



- Destinations are points of communication for messaging
  - Example: JMS Queues, topics, Web Service endpoints
- Producers send messages to destinations
- Consumers get messages from destinations



The following capabilities are provided by the bus:

•Any application can exchange messages with any other application by using a *destination* to which one application sends, and from which the other application receives.

•A message-producing application, that is, a *producer*, can produce messages for a destination regardless of which messaging engine the producer uses to connect to the bus.

•A message-consuming application, that is, a *consumer*, can consume messages from a destination (whenever that destination is available) regardless of which messaging engine the consumer uses to connect to the bus

The bus supports the following types of messaging:

•Sending messages synchronously (this requires the consuming application to be running and reachable). This is not supported by the JMS API.

•Sending messages asynchronously (possible whether the consuming application is running or not and whether or not the destination is reachable). Both point-to-point and publish/subscribe messaging are supported

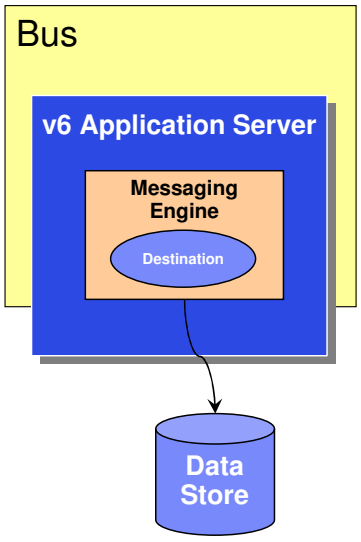
•Publishing events or other notifications. Notification messages can also be generated by the bus itself

Bus Members

- Adding a bus member associates a cluster or Application Server with a Service Integration Bus
- When a new Bus Member is defined, one Messaging Engine is automatically created on the corresponding Application Server or Cluster

Messaging Engines

- An ME runs inside an Application Server and manages messaging resources
- Messaging Engines are transparent to application code
- Each server or cluster has at least one ME for each Bus with which it is associated
- The ME has an associated data store for message persistence and overflow



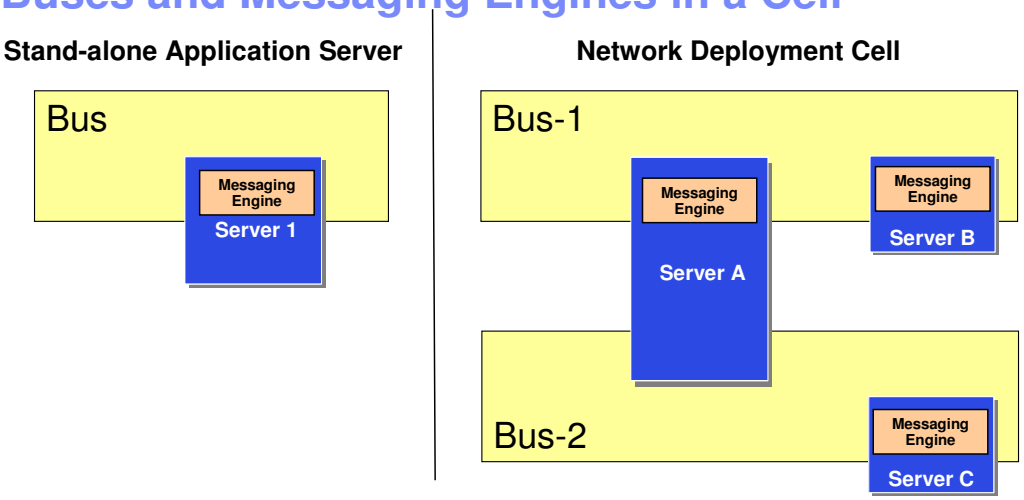
One Messaging Engine is automatically created for the application server or the cluster when defining a new bus member (Application Server or Cluster)

Can have multiple Messaging Engines running within the same cluster

Within a bus, each Messaging Engine has a unique identity

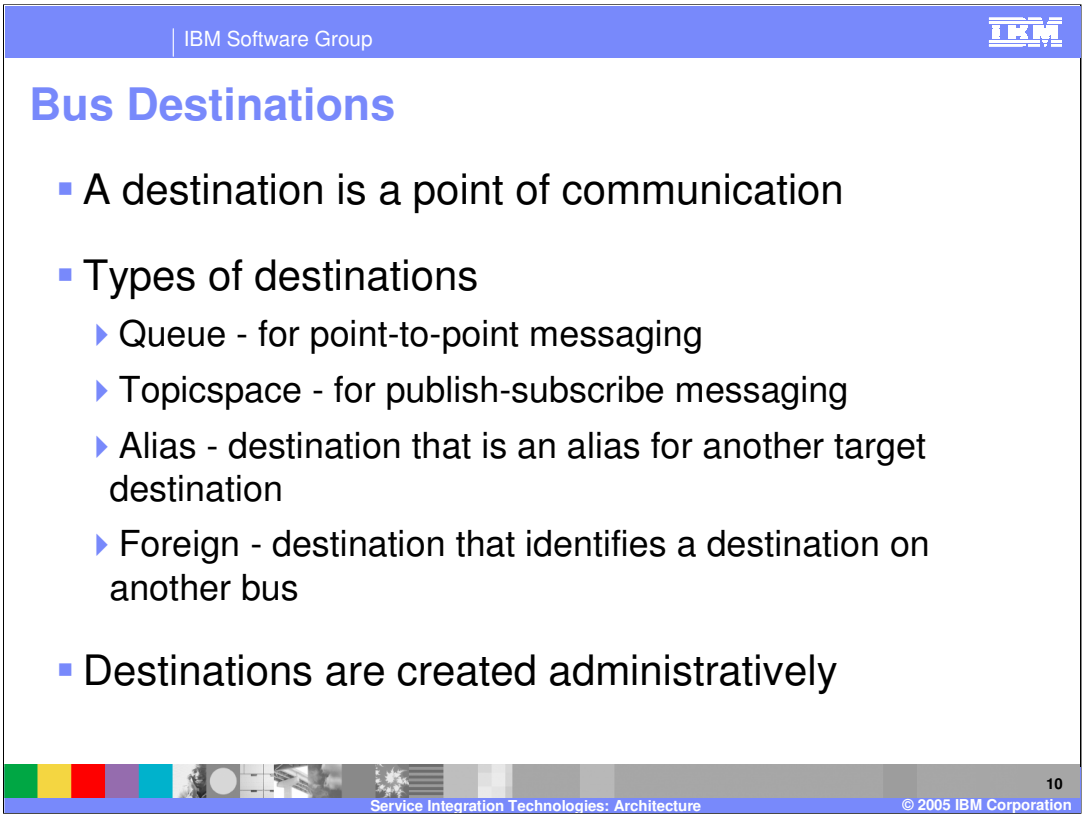
The data store preserves messages, subscriptions, and so on, so that they survive if the server or messaging engine is stopped and restarted. It is also used for the overflow of the non-persistent messages in some Quality of Service options

Buses and Messaging Engines in a Cell



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

Old notes here were pretty inaccurate – ignore for script and rewrite.



Temporary destinations are bus destinations that are created and deleted automatically for API-specific temporary destinations. Temporary destinations appear on the list of destinations for a service integration bus, but normally need no administration.

A topicspace is a hierarchy of topics used for publish/subscribe messaging. Topics with the same name can exist in multiple topicspaces.

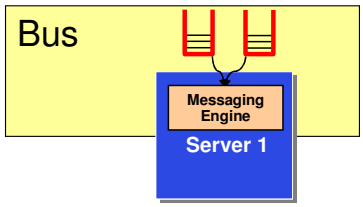
Alias destinations provide a level of abstraction between applications and the underlying target bus destinations that hold messages. Applications interact with the alias destination, so the target bus destination can be changed without changing the application. Each alias destination identifies a target bus destination and target service integration bus. 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).

Each *messaging engine* has a default exception destination, _SYSTEM.Exception.Destination.<*messaging_engine_name*>, that is used to handle undeliverable messages for all bus destinations that are localized to the messaging engine

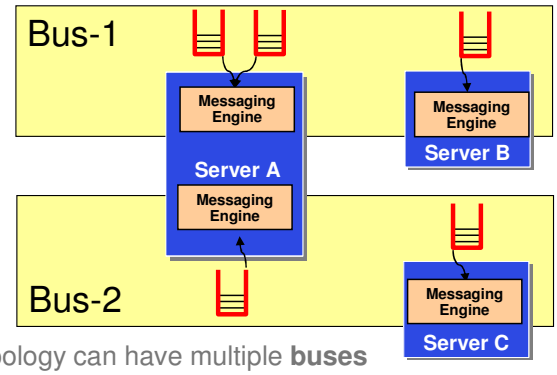
The foreign destination identifies a destination on another bus, and provides a mechanism for overriding system default for a particular destination.

Destinations and Buses

Stand-alone Application Server

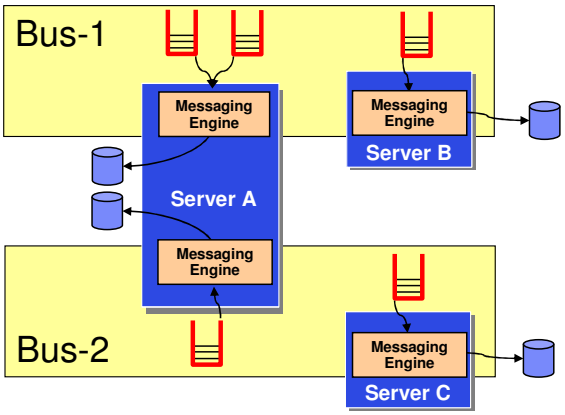


Network Deployment Cell



- WebSphere Application Server topology can have multiple **buses**
- Each bus can have servers or clusters or both as **bus members**
- When a server/cluster is made a bus member, a **Messaging Engine** is created
- **Destinations** are created on the bus and hosted on a Messaging Engine
  - A queue is hosted by a single Messaging Engine
  - TopicSpaces are hosted by all Messaging Engines on a bus

Data Store



- Each Messaging Engine has its own data store for storing messages, transactions states and delivery records

- A Messaging Engine requires a persistent data store
  - WebSphere Application Server uses JDBC for this
- Messaging Engines can share a database, but each Messaging Engine has its own schema within the database (which results in different tables)

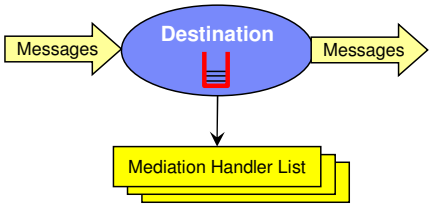
The Data Store is used to buffer in-flight messages and hold a number of other pieces of information (for example records of message delivery when delivering multiple copies of a single message).

A Messaging Engine requires a persistent back end data store, even for non-persistent messages (e.g., for message overflow).

Out of the box support for persistence using any supported database.

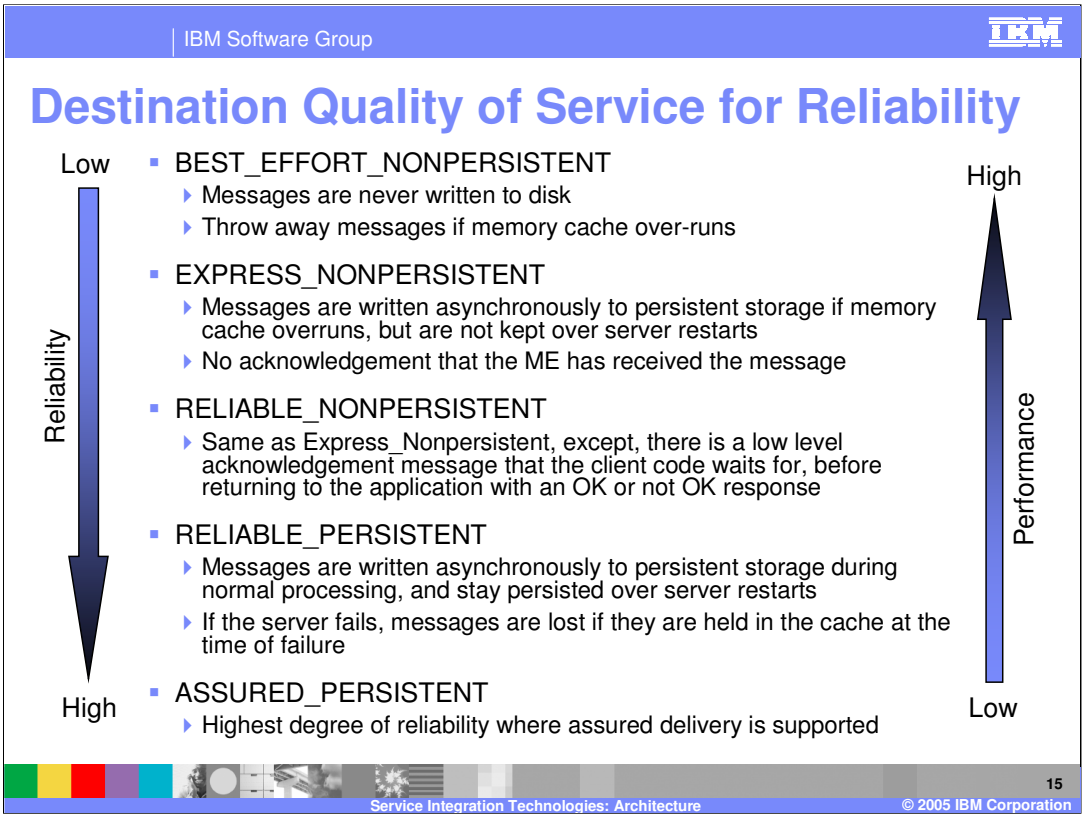
Mediation

- The ability to manipulate a message at a destination
  - Transform or reroute the message
  - Send copies of the message to additional destinations
  - Allow interaction with non-messaging resource managers (e.g. databases)
- Mediation attached administratively to a Destination



Section

Quality of Service



Express non-persistent and reliable-persistent are defaults for non-persistent and persistent.

In express non-persistent, messages are sent from a producer to the ME, but there is never an acknowledgement flow (at the low level communications layer) to indicate that the ME has the message. The application resumes immediately and assumes that all is well. In the reliable non-persistent there is a low level acknowledgement message that the client code waits for before returning to the application with an OK, Not OK response. So - express runs faster, but with a slightly lower level of reliability

Quality of Service (QOS) specification

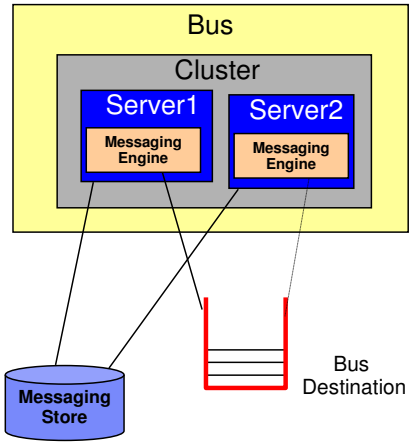
- QOS can be specified on
  - A Destination
    - Maximum QOS can be defined on a Destination (for example, a Destination that can only accept non-persistent messages)
  - A message
    - By individual Producers, typically under application control via a combination of API calls (e.g. JMS Persistent or Non-persistent) and the Connection Factory used (which defines the Persistent/NonPersistent Mapping)

Section

High Availability and Scalability

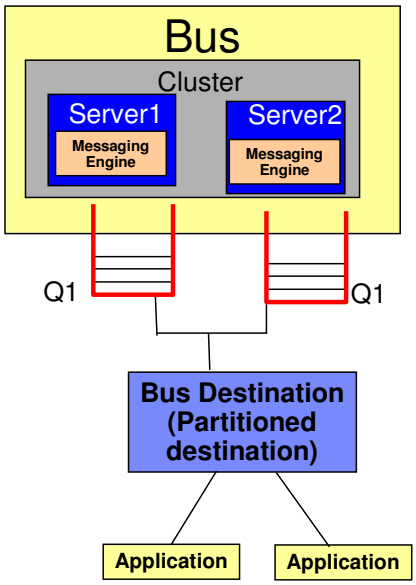
High Availability

- Destinations can be associated with a Bus Member of type “cluster”
- An ME can failover to another cluster member
- For messaging to continue, the message store must be accessible from the server to which the ME fails over
  - Ensure that the message store is accessible from all cluster members



Scalability

- Scalability is achieved by deploying *multiple messaging engines per bus* to a cluster
- With this flexibility, there are **one or more** concurrently active messaging engines running in a cluster on the same bus
  - When a destination is hosted on such a cluster, then the destination is partitioned
- Failover is handled by the HAManager
- Message order not preserved



This case provides scalability. The destination has been localized to a Cluster ME. It is therefore partitioned across the Messaging Engines within the cluster. With a partitioned Destination, the recoverable objects associated with the destination are split between separate Data Stores and hence separate Data stores. This configuration has the disadvantage that message order cannot be preserved, but has advantages. One is that multiple consumers (or producers) can be deployed across the same Cluster to provide high messaging bandwidth. Messaging operations would always be locally fulfilled.

Scalability can be increased by adding additional cluster members to run additional messaging engines.

Section

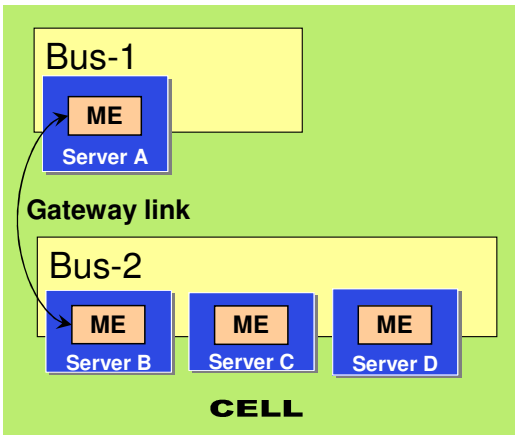
Topologies

Messaging Engine Topology

- One messaging engine per bus member is adequate for many applications
  - This is the default behavior when adding a bus member
- Advantages of deploying more than one messaging engine:
  - Spreading messaging workload across multiple servers
  - Performance is better when a messaging engine is hosted in the same server as the applications that access it

Bus Topology

- 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**
  - By way of a gateway link
  - Gateway links are created administratively



A bus can connect to other buses, which are referred to as foreign buses.

The inter-bus links might reflect the distribution of buses across organizations, or across departments within organizations.

To create a link to a foreign bus, the administrator first creates a virtual link from the local bus to the foreign bus, then creates a physical gateway link from a messaging engine in the local bus to the foreign bus.

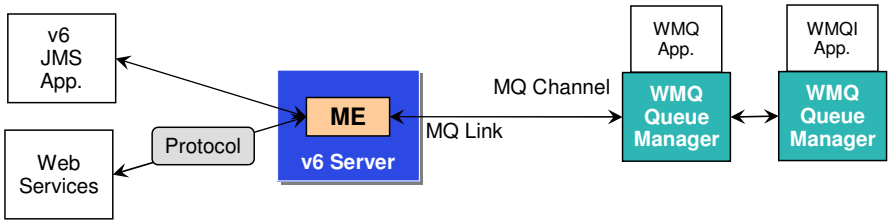
Section

Interoperability

Interoperability

- Full interoperability with other Buses in the same or different Cell or Stand-alone Application Server
- WebSphere Application Server V5 Embedded JMS Server interoperation
  - v5 embedded JMS clients can connect to V6 destinations by way of an MQ Client Link
    - MQ Client Link is created administratively
  - A JMS 1.0 application running in a V6 server can connect to a V5 server
    - v5 does not support JMS 1.1

Interoperability with WebSphere MQ



- Tight integration between V6 Buses and WebSphere MQ
- WebSphere MQ thinks that the V6 Messaging Engine (ME) is another Queue Manager
- WebSphere MQ applications can send messages to queues hosted on V6 Bus
- v6 Messaging clients can send messages to WebSphere MQ queues

You can have multiple MQLinks out of a bus, but each link goes to a different queue manager, and further these queue managers should not be interconnected. The link engine can be part of a cluster, but the issue is in handling failover. The ME hosting the MQLink must keep a fixed host and port because that is what MQ expects, and so you have to marry the new WebSphere Application Server HA support with more traditional HACMP® like HA solutions.

Section

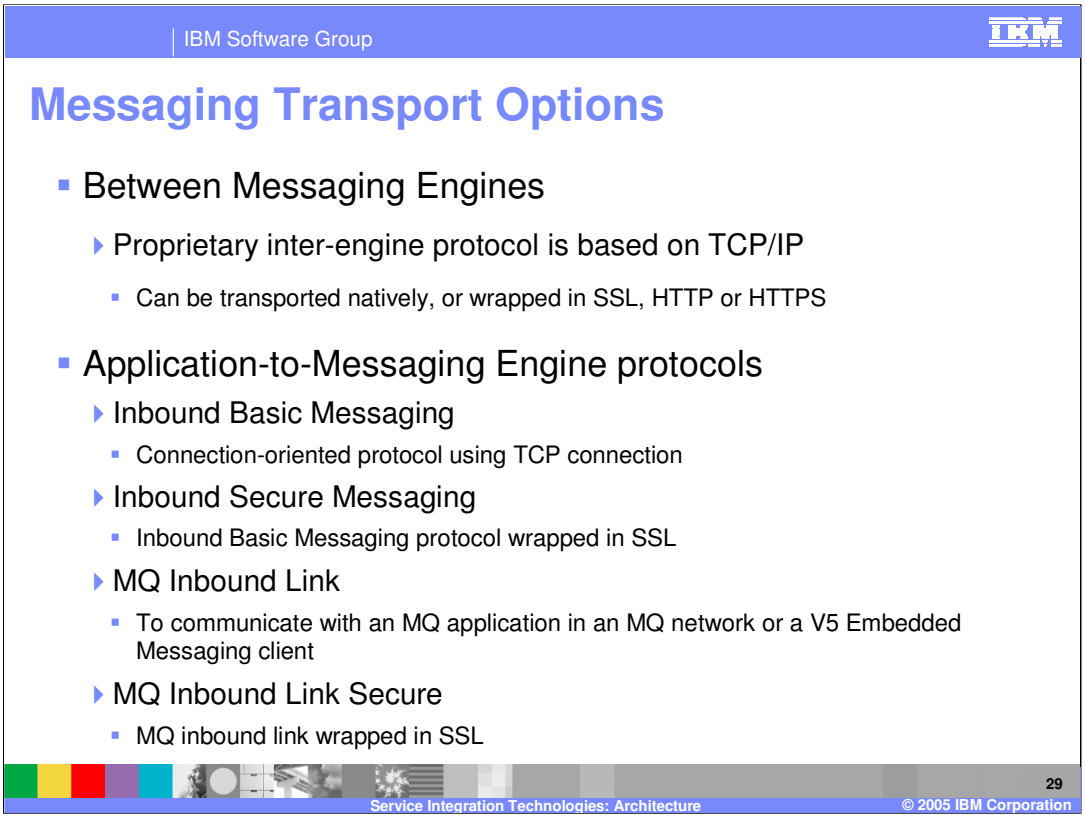
Summary

Summary

- IBM Service Integration Technologies provide a integrated messaging infrastructure for WebSphere Application Server
  - Fully integrated with runtime services such as HA Manager
- The “Default Messaging Provider” is implemented using Service Integration Technologies
  - JMS 1.1 compliant
- Messaging Engines manage messaging resources
- Messaging destination types include queues, topics
- Gateway Links connect multiple buses

Section

Appendix: Service Integration Technologies Protocols



Formats and Protocols (FAP) :The WebSphere MQ FAPs define how queue managers communicate with one another, and also how WebSphere MQ clients communicate with server queue managers.

JFAP :The formats and protocols used to communicate between messaging engines in WebSphere Application Server V6.

Messaging Engines communicate with each other using an efficient proprietary protocol. This carries the actual message data, along with control information that allows the work of the bus to be distributed across the various engines that make it up.

InboundBasicMessaging

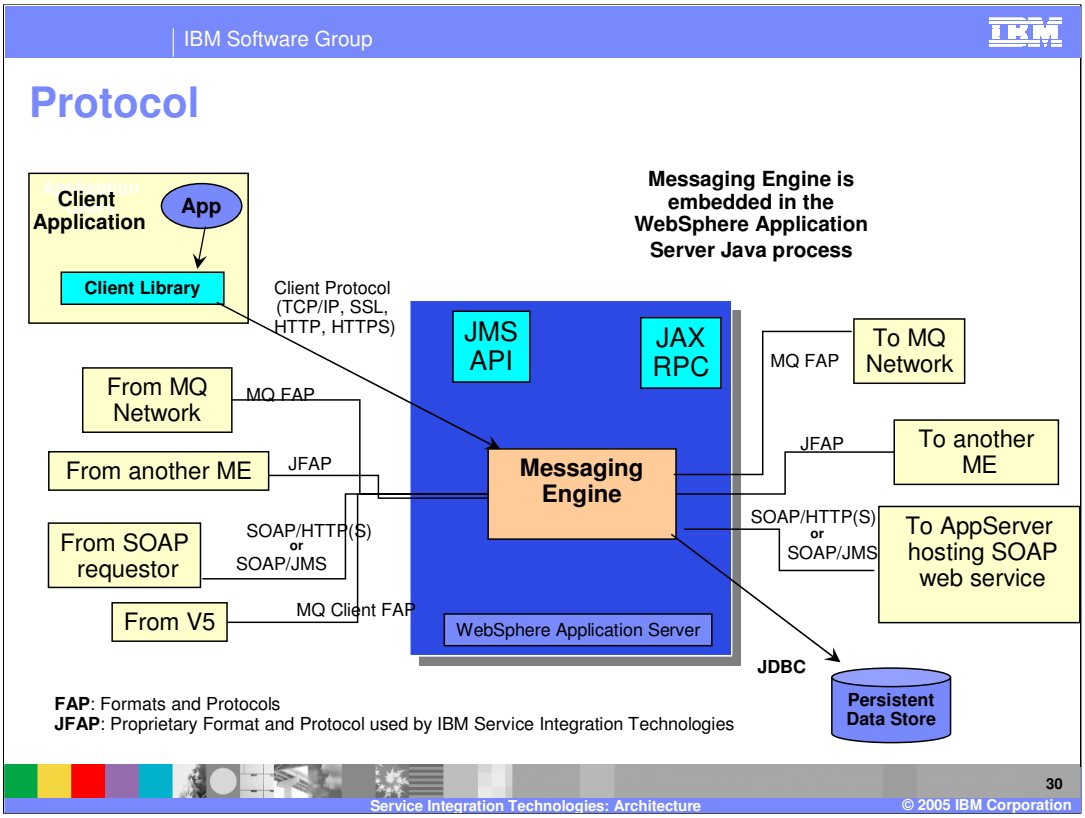
This is a connection-oriented protocol, using a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging

This is the InboundBasicMessaging protocol wrapped in SSL.

InboundBasic and Secure Messaging settings are made when creating the connection factory

MQ Inbound Link



Applications can attach to a bus using the JMS API. IBM Service Integration Technologies provide API libraries that connect the application to a messaging engine, either via an in-process call, or across a network using a remote client. A remote client may run in the J2EE Application Client environment and the J2EE Application Server environment.

Template Revision: 11/02/2004 5:50 PM

Trademarks, Copyrights, and Disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements and/or changes in the product(s) and/or program(s) described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (e.g., IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.