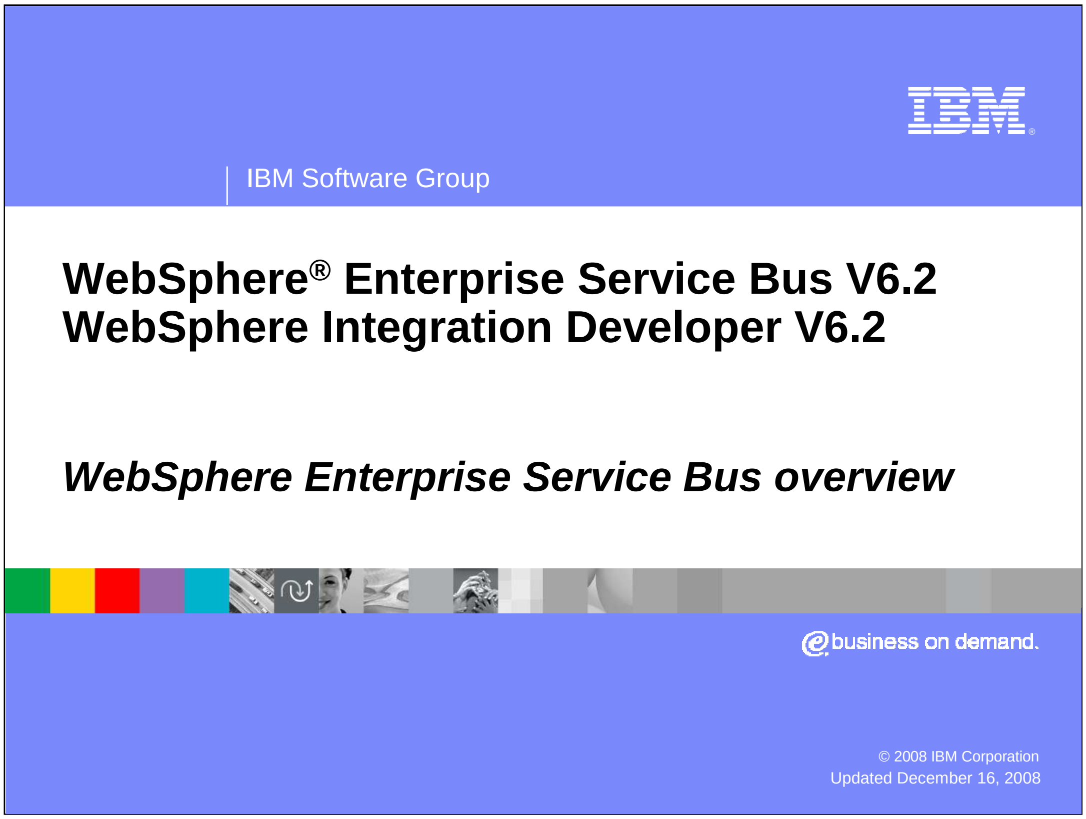


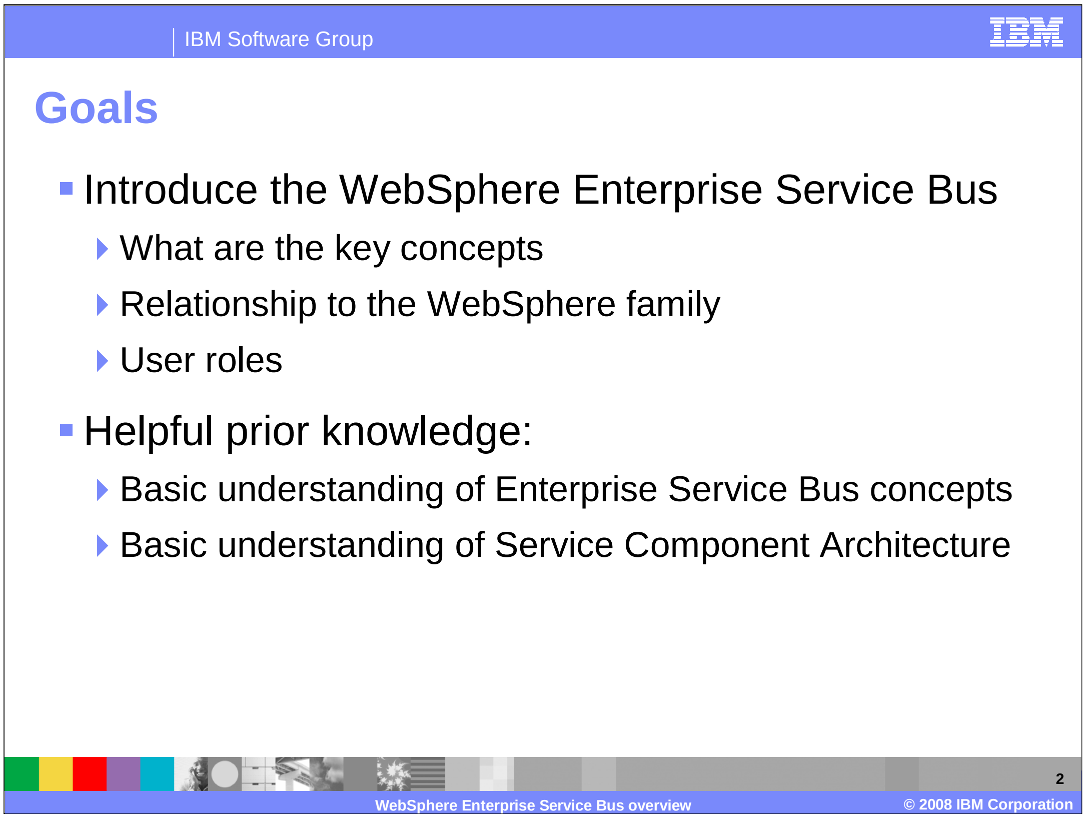
IBM Software Group

WebSphere® Enterprise Service Bus V6.2
WebSphere Integration Developer V6.2

WebSphere Enterprise Service Bus overview

Updated December 16, 2008

This presentation provides an overview of the WebSphere Enterprise Service Bus product.



Goals

- Introduce the WebSphere Enterprise Service Bus
  - What are the key concepts
  - Relationship to the WebSphere family
  - User roles
- Helpful prior knowledge:
  - Basic understanding of Enterprise Service Bus concepts
  - Basic understanding of Service Component Architecture

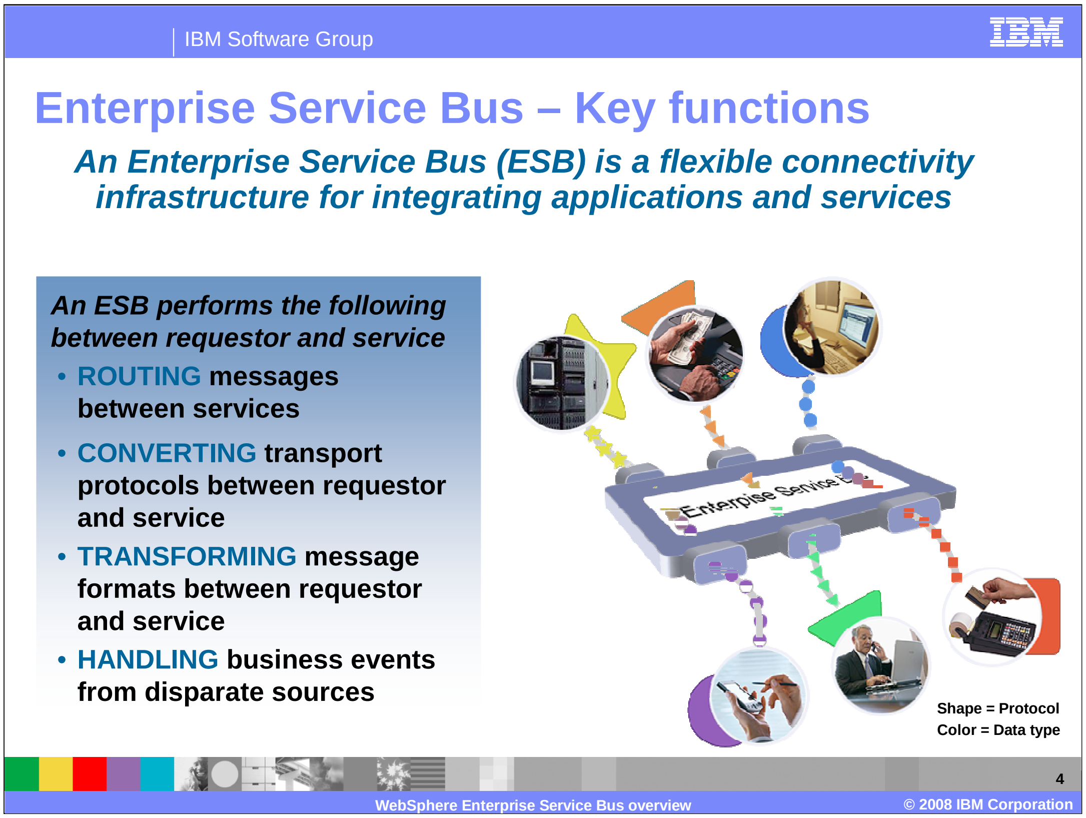
The goal of this presentation is to introduce you to the WebSphere Enterprise Service Bus product. You will learn the key concepts needed to understand the approach that is taken to implement mediation functionality. Then you will see how the WebSphere Enterprise Service Bus fits into the WebSphere family product stack. Finally, you will learn about the primary user roles that are associated with development and administration for WebSphere Enterprise Service Bus.

This presentation assumes you have at least a cursory knowledge of what an enterprise service bus is and understand the basic concepts of service component architecture, commonly referred to as SCA.

Section

WebSphere Enterprise Service Bus key concepts

In this section you will be given a quick reminder of the concepts associated with an enterprise service bus. With that background, the key concepts of how WebSphere Enterprise Service Bus implements ESB functionality will be presented.



IBM Software Group

Enterprise Service Bus – Key functions

An Enterprise Service Bus (ESB) is a flexible connectivity infrastructure for integrating applications and services

An ESB performs the following between requestor and service

- **ROUTING messages between services**
- **CONVERTING transport protocols between requestor and service**
- **TRANSFORMING message formats between requestor and service**
- **HANDLING business events from disparate sources**

Shape = Protocol
Color = Data type

4

WebSphere Enterprise Service Bus overview © 2008 IBM Corporation

You will probably recognize this slide from the enterprise service bus introduction presentation that is part of this education module. It provides a very good one slide view on the functions of an ESB. An enterprise service bus provides a flexible connectivity infrastructure for integrating applications and services, enabling composite applications to be built as a loose coupling of independent services. It is at the heart of your service oriented architecture, reducing the number, size, and complexity of interfaces and connections that must be defined and maintained.

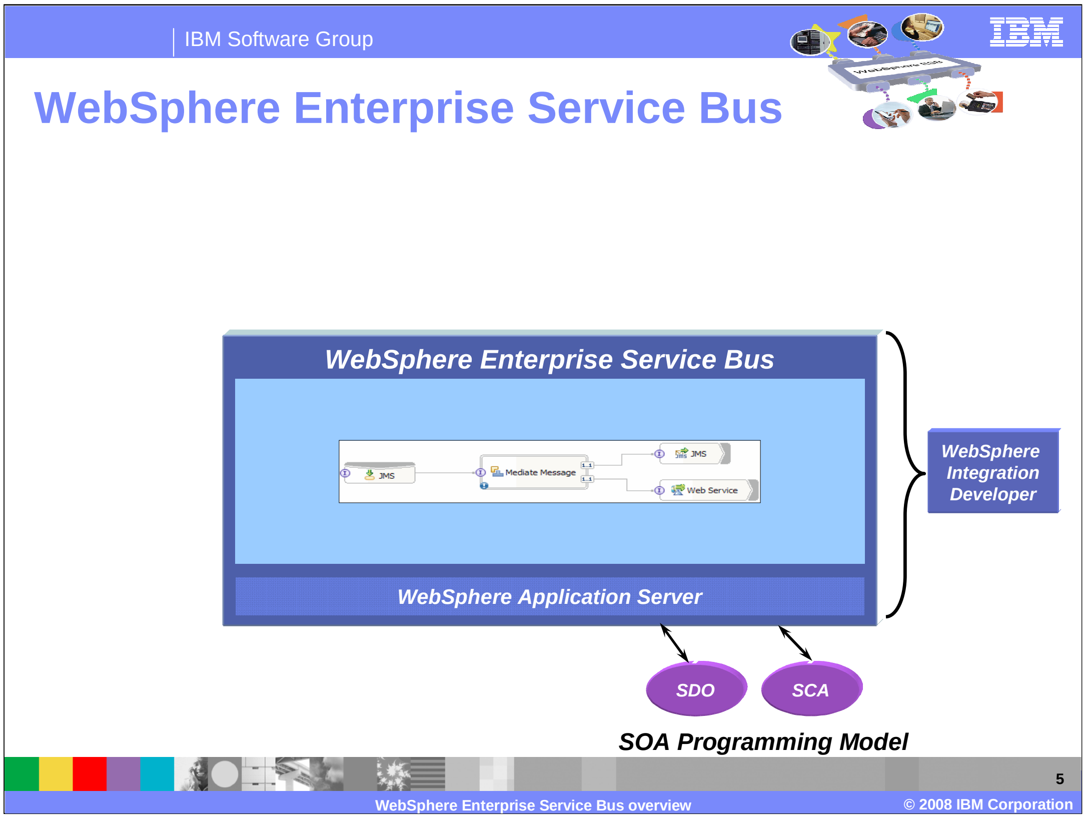
There are four primary functions provided by an enterprise service bus:

Its first responsibility is the routing of messages. Rather than the service requestor calling directly to the service provider, the requestor sends the request to the ESB, and the ESB then is responsible for making the call on the service provider.

Secondly, it is responsible for converting transport protocols. If the service requestor called directly to the service provider, they would need to use the same transport protocol. The ESB enables the service requestor to use one transport protocol while the service provider uses another.

Thirdly, it is responsible for transforming message formats. By eliminating the direct call from the service requestor to the service provider the ESB is capable of modifying the message so that the interfaces used by the requestor and provider do not have to be identical.

Finally, the ESB is capable of handling business events from disparate sources. Therefore, the same service provider responsible for performing some particular business function can be indirectly invoked from a variety of application contexts.



WebSphere Enterprise Service Bus

WebSphere Enterprise Service Bus

WebSphere Application Server

WebSphere Integration Developer

SDO

SCA

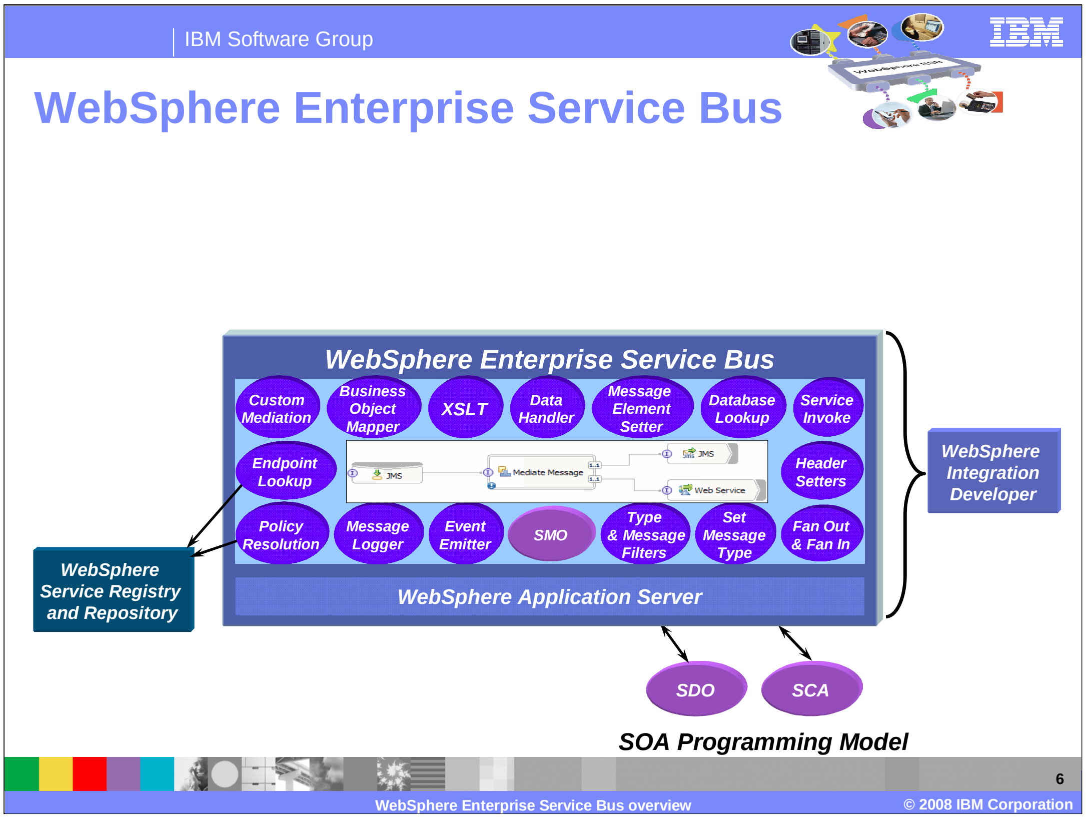
SOA Programming Model

This is the first of a series of diagrams which provides a fairly complete overview of what WebSphere Enterprise Service Bus is in terms of features and functions associated with the product.

At the center of the picture you can see that WebSphere Enterprise Service Bus is built on top of WebSphere Application Server. More detail on this aspect is provided in a subsequent section of this presentation.

The next thing to notice is that WebSphere Enterprise Service Bus is built around a service oriented architecture programming model. It is firmly based on service component architecture, commonly known as SCA, and on service data objects, known as SDO. In the center of the picture you will see an SCA assembly diagram which contains an SCA export, an SCA component and two SCA imports. An SCA export is used to receive service requests, an SCA import is used to make requests to service providers and an SCA component contains processing logic. In this case, it is a mediation flow component, the primary SCA component type for providing ESB functionality. It encapsulates the logic needed for message transformation and routing decisions. Protocol conversion happens because of the SCA exports and imports.

The last thing to notice on this diagram is WebSphere Integration Developer. It is the tool used to develop SCA based mediation applications that run in the WebSphere Enterprise Service Bus.



WebSphere Enterprise Service Bus

WebSphere Enterprise Service Bus

Custom Mediation
Business Object Mapper
XSLT
Data Handler
Message Element Setter
Database Lookup
Service Invoke
Endpoint Lookup
Header Setters
Policy Resolution
Message Logger
Event Emitter
SMO
Type & Message Filters
Set Message Type
Fan Out & Fan In

WebSphere Application Server

WebSphere Service Registry and Repository

WebSphere Integration Developer

SDO
SCA

SOA Programming Model

Added to the picture you can now see the mediation primitives and the service message object, commonly referred to as the SMO. The mediation primitives and the SMO are the primary entities involved in the definition of mediation logic within a mediation flow component. The SMO is a representation of the message passing through the bus. It contains a message body, which is the application data associated with the service request. It also contains headers with information relevant to the transport protocol, for example, the JMS properties associated with the message. The processing performed by the mediation primitives is centered on the SMO. Primitives can access and update data in the SMO and can modify the format of the SMO. The resulting content and format of the SMO defines what the outgoing message will be.

Mediation primitives perform some specific function and are customized through the use of configuration properties.

The message logger primitive enables a selected section of the SMO to be written to a message log database.

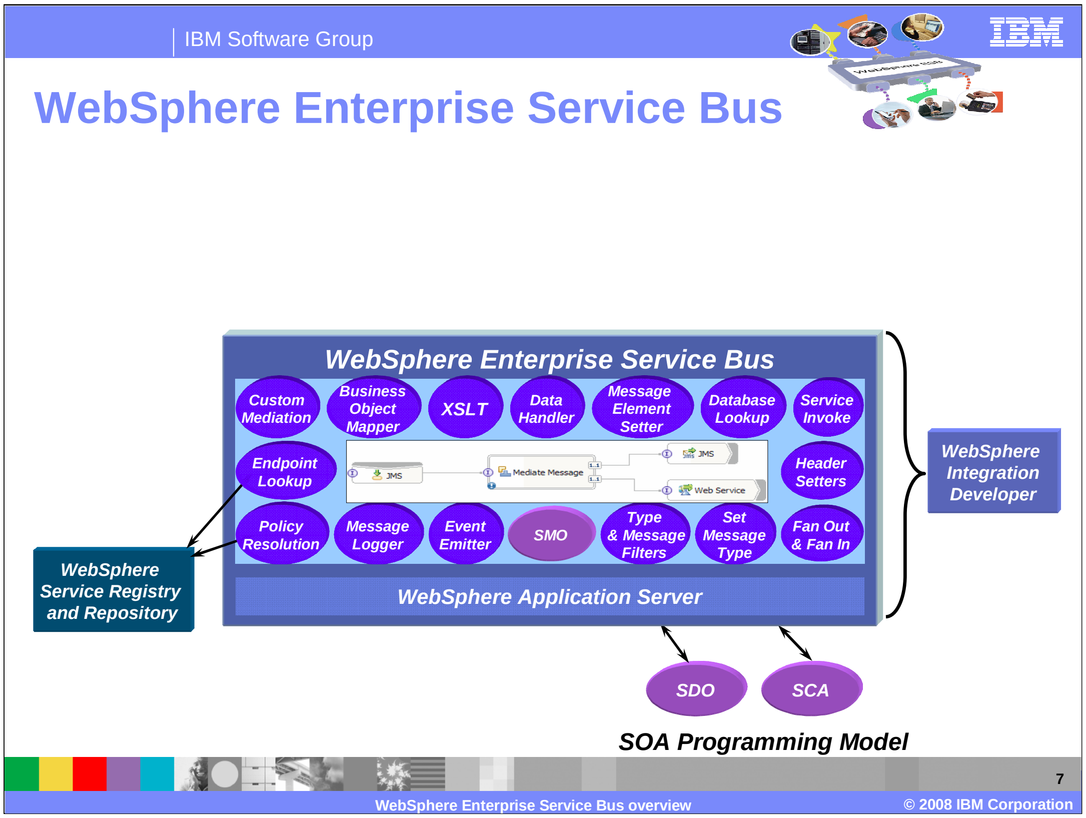
The event emitter primitive provides a way to generate a common base event containing information from the SMO. The event will be handled by the common event infrastructure.

The type filter and message filter primitives enable you to control the flow through the mediation based on what is contained in the SMO. The message filter controls the flow based on element values and the type filter controls the flow based on element types.

The set message type primitive is used to declare a more specific type for loosely typed data in the SMO, similar to a cast operation in a programming language.

The fan out and fan in primitives are used in conjunction with each other to provide aggregation support within mediation flows. The fan out splits a message to enable iterating through the individual elements of an array within the message. The fan in is used to aggregate the results of each iteration. If you consider the fan out primitive to be the beginning of a loop, the fan in primitive effectively is the end of the loop.

The header setter primitives are four protocol specific primitives used for accessing and manipulating the headers within the SMO. There are header setter primitives for JMS, HTTP, MQ and SOAP.



IBM Software Group

WebSphere Enterprise Service Bus

WebSphere Enterprise Service Bus

Custom Mediation

Business Object Mapper

XSLT

Data Handler

Message Element Setter

Database Lookup

Service Invoke

Endpoint Lookup

JMS

Mediate Message

JMS

Web Service

Header Setters

Policy Resolution

Message Logger

Event Emitter

SMO

Type & Message Filters

Set Message Type

Fan Out & Fan In

WebSphere Application Server

WebSphere Service Registry and Repository

WebSphere Integration Developer

SDO

SCA

SOA Programming Model

7

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

The service invoke primitive enables you to call other SCA services from within a mediation flow, including the ability to automatically retry service call failures to the same service at alternate endpoints.

The database lookup primitive enables you to use an element value from the SMO as a key for a database lookup. Values from the database which are returned from the lookup can then be used to update the SMO.

The message element setter primitive allows elements in the SMO to be set to a fixed value or to values copied from another part of the SMO.

The data handler primitive allows you to configure a data handler, similar to one used with an import or export, that converts between business object format and native data format.

The XSL transformation primitive performs an extensible stylesheet language transformation on the SMO. This capability enables the message to be modified so that the service requestor and service provider do not have to support the identical interface.

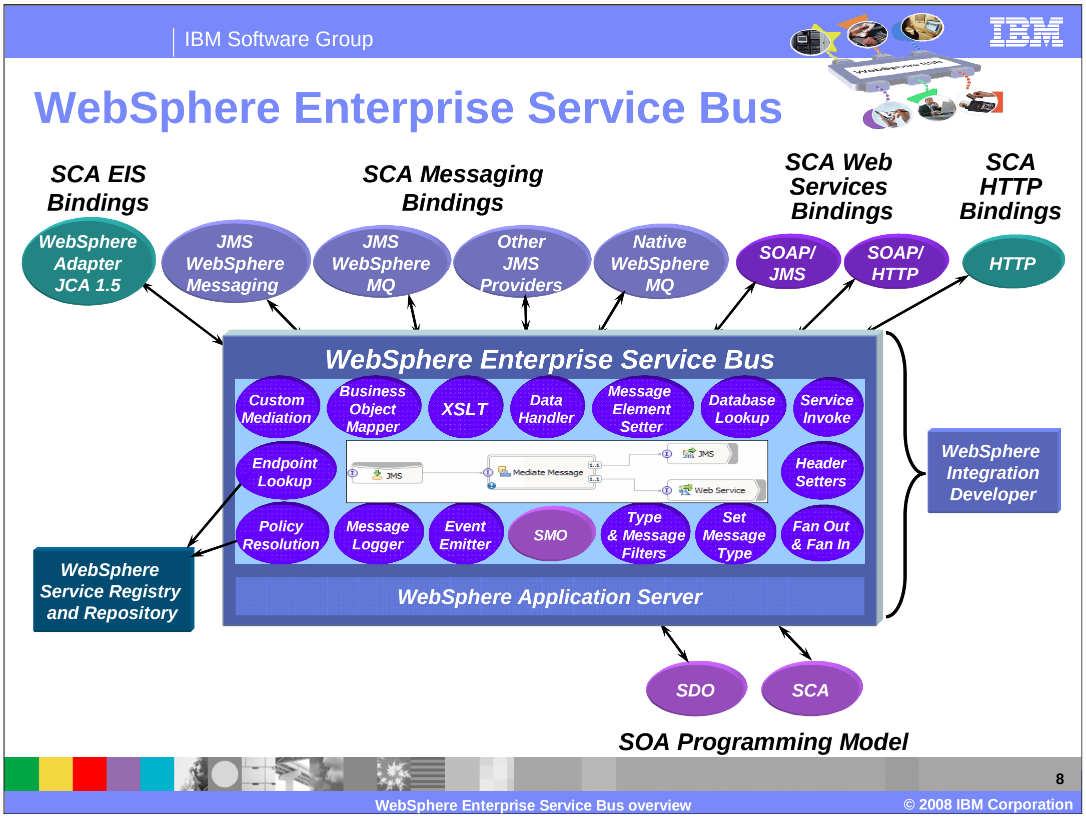
The business object mapper primitive enables the use of business object maps within a mediation flow. It is similar to the XSL transformation in that it enables the message to be modified so that the service requestor and service provider do not have to support the identical interface.

The custom mediation primitive enables you to drop into Java™ code to access and manipulate the SMO. It is provided to enable functionality that is not provided by the other mediation primitives.

The endpoint lookup primitive provides the capability to query the WebSphere Service Registry and Repository to locate service provider endpoints.

The policy resolution primitive also performs queries to the WebSphere Service Registry and Repository and is used to locate policies that dynamically control the behavior of the flow.

By wiring these mediation primitives together in a mediation flow component you define the logic of your mediation flow. You define one flow for each of the operations defined on the inbound interface.



IBM Software Group

WebSphere Enterprise Service Bus

SCA EIS Bindings

SCA Messaging Bindings

SCA Web Services Bindings

SCA HTTP Bindings

WebSphere Adapter JCA 1.5

JMS WebSphere Messaging

JMS WebSphere MQ

Other JMS Providers

Native WebSphere MQ

SOAP/ JMS

SOAP/ HTTP

HTTP

WebSphere Enterprise Service Bus

Custom Mediation

Business Object Mapper

XSLT

Data Handler

Message Element Setter

Database Lookup

Service Invoke

Endpoint Lookup

JMS

Mediate Message

JMS

Web Service

Header Setters

Policy Resolution

Message Logger

Event Emitter

SMO

Type & Message Filters

Set Message Type

Fan Out & Fan In

WebSphere Application Server

WebSphere Integration Developer

WebSphere Service Registry and Repository

SDO

SCA

SOA Programming Model

8

WebSphere Enterprise Service Bus overview

The SCA import and export binding types have now been added to the picture. These define the possible transport protocols that can be used with the WebSphere Enterprise Service Bus for receiving service requests and for making calls to service providers. The SCA bindings used with WebSphere Enterprise Service Bus are not unique but are the same SCA binding types available with WebSphere Process Server.

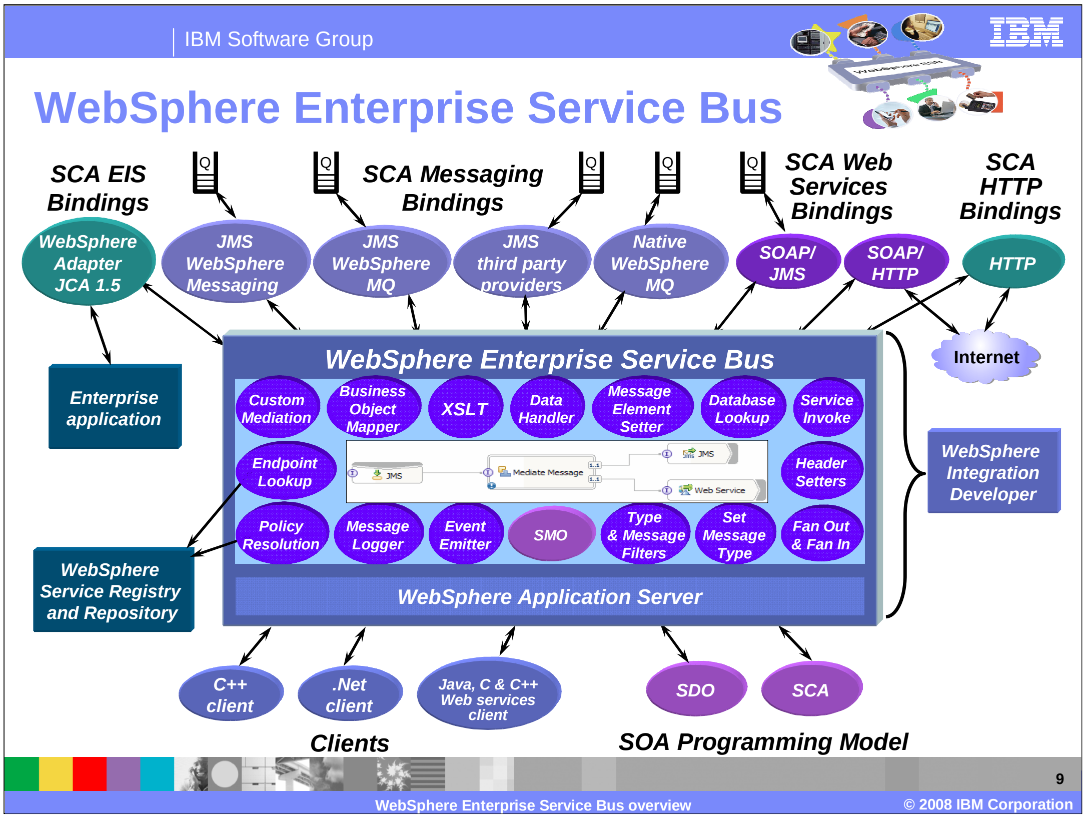
First, there is the HTTP binding which is used with HTTP protocols where the message can contain data in any format.

Next there are the Web services bindings. These are used with messages conforming to SOAP and can be configured in a number of ways. You can use JAX-WS with SOAP 1.1 or SOAP 1.2 over HTTP. Alternatively, you can have JAX-RPC with SOAP 1.1 over either JMS or HTTP.

The next set of bindings are the messaging bindings. There are JMS bindings that can be used with the messaging support that is built directly into WebSphere Enterprise Service Bus and is also available with WebSphere Application Server and WebSphere Process Server. There are JMS bindings that can be used with a WebSphere MQ JMS provider. Additionally, there are also generic JMS bindings, which can be used with other JMS providers that conform to the JMS 1.1 specification. Finally, there are messaging bindings, which can be used directly with native WebSphere MQ without JMS.

Another kind of binding is the SCA EIS binding, which is used in conjunction with WebSphere Adapters that are built as JCA 1.5 resource adapters.

Not shown but also supported is the SCA binding type, which is based on the SCA messaging model and can be used between SCA imports and exports.



IBM Software Group

WebSphere Enterprise Service Bus

SCA EIS Bindings

SCA Messaging Bindings

SCA Web Services Bindings

SCA HTTP Bindings

WebSphere Adapter JCA 1.5

JMS WebSphere Messaging

JMS WebSphere MQ

JMS third party providers

Native WebSphere MQ

SOAP/ JMS

SOAP/ HTTP

HTTP

Internet

Enterprise application

WebSphere Enterprise Service Bus

Custom Mediation

Business Object Mapper

XSLT

Data Handler

Message Element Setter

Database Lookup

Service Invoke

Endpoint Lookup

JMS

Mediate Message

JMS

Web Service

Header Setters

Policy Resolution

Message Logger

Event Emitter

SMO

Type & Message Filters

Set Message Type

Fan Out & Fan In

WebSphere Application Server

WebSphere Integration Developer

WebSphere Service Registry and Repository

C++ client

.Net client

Java, C & C++ Web services client

Clients

SDO

SCA

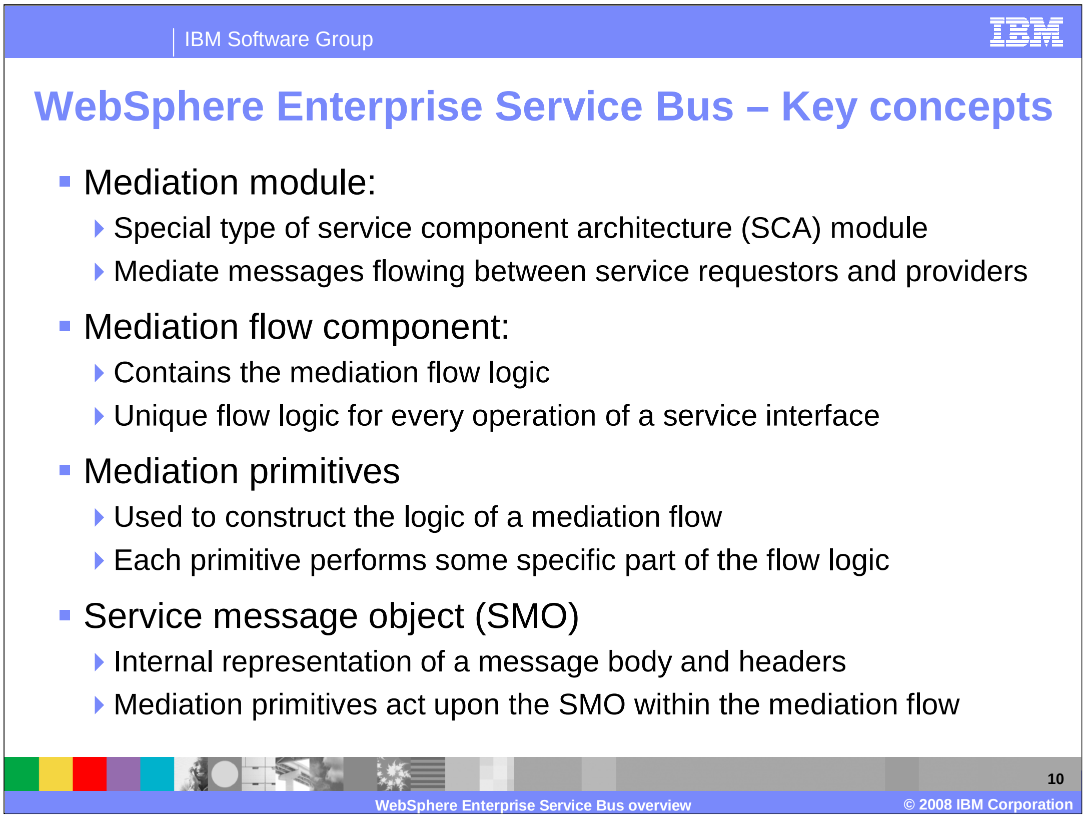
SOA Programming Model

9

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

In this completed picture you will now see those things that are external to the WebSphere Enterprise Service Bus. There are queues associated with the various messaging bindings. The information defining queues, topic, queue managers and so forth are part of the configuration information for each of the bindings. You can see that the HTTP and SOAP over HTTP bindings are connected to the internet or an intranet. The SCA EIS bindings with their associated WebSphere Adapters talk to whatever backend system is supported by the adapter configured for that binding. Finally, you can see that calls can be made to the WebSphere Enterprise Service Bus using various types of clients.



WebSphere Enterprise Service Bus – Key concepts

- Mediation module:
  - Special type of service component architecture (SCA) module
  - Mediate messages flowing between service requestors and providers
- Mediation flow component:
  - Contains the mediation flow logic
  - Unique flow logic for every operation of a service interface
- Mediation primitives
  - Used to construct the logic of a mediation flow
  - Each primitive performs some specific part of the flow logic
- Service message object (SMO)
  - Internal representation of a message body and headers
  - Mediation primitives act upon the SMO within the mediation flow

The previous slides showed and explained to you the functionality of the WebSphere Enterprise Service Bus. This slide summarizes the key concepts already presented that are needed to understand mediations as implemented in WebSphere Enterprise Service Bus.

Starting at the highest level of abstraction, there are mediation modules, which are a special type of SCA module. They make use of SCA exports and imports to communicate with service requestors and service providers. These provide the key to handling protocol conversions within the bus. The mediation module also contains a mediation flow component.

It is the mediation flow component where the overall logic for the mediation is defined. For every operation defined on an input interface there is unique mediation flow logic defined for the operation’s request and response. It is in this mediation flow logic that message transformation and dynamic routing decisions take place.

The flow logic is defined within the mediation flow component using mediation primitives. Each mediation primitive provides some specific portion of the logic. The overall logic is defined by wiring these mediation primitives together into a logical flow.

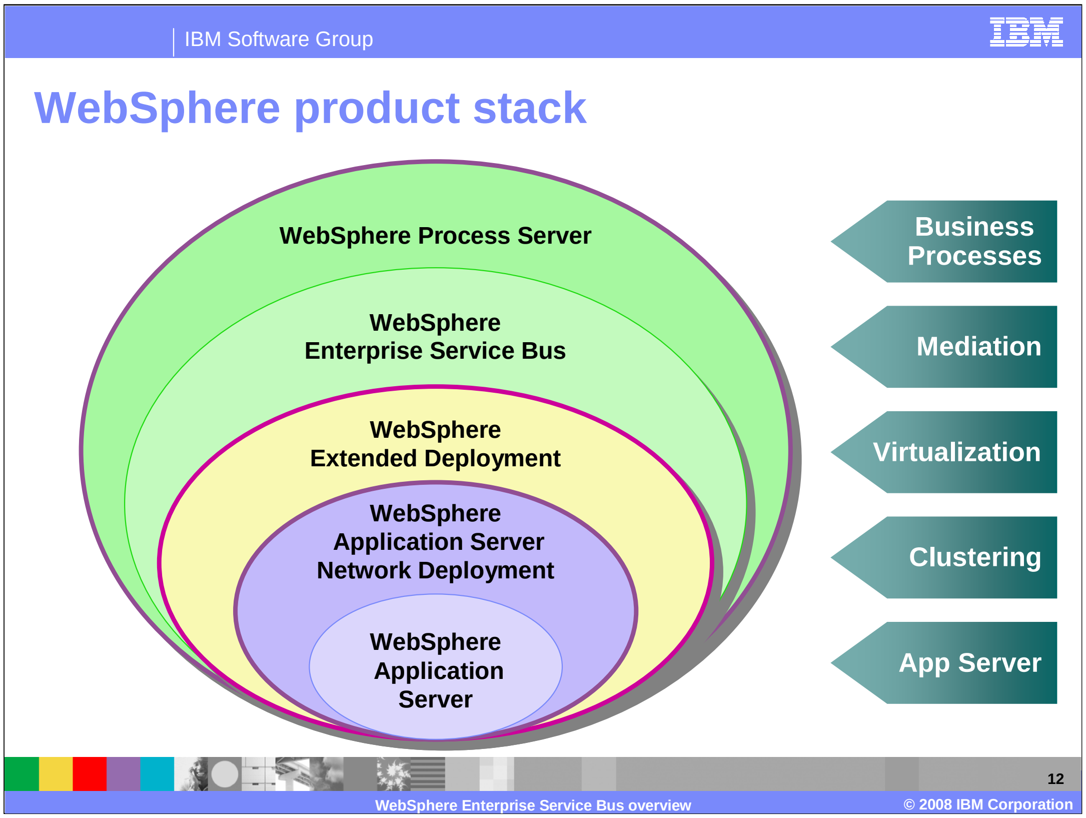
The data that is acted upon by the mediation primitives is contained in a representation of the message called a service message object. The SMO contains a body, which is the application data and headers, which provide protocol and control data.

To summarize, the highest level of a mediation is the mediation module, which contains a mediation flow component, which contains mediation flows defined using mediation primitives which act upon service message objects.

Section

WebSphere Enterprise Service Bus product family

In this section you will learn about the relationship between WebSphere Enterprise Service Bus and other products that are part of the WebSphere server stack.



IBM Software Group

WebSphere product stack

WebSphere Process Server

WebSphere
Enterprise Service Bus

WebSphere
Extended Deployment

WebSphere
Application Server
Network Deployment

WebSphere
Application
Server

Business Processes

Mediation

Virtualization

Clustering

App Server

12

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

This slide shows the product stack of the WebSphere servers.

The core of the stack is the WebSphere Application Server which provides the base J2EE application hosting environment.

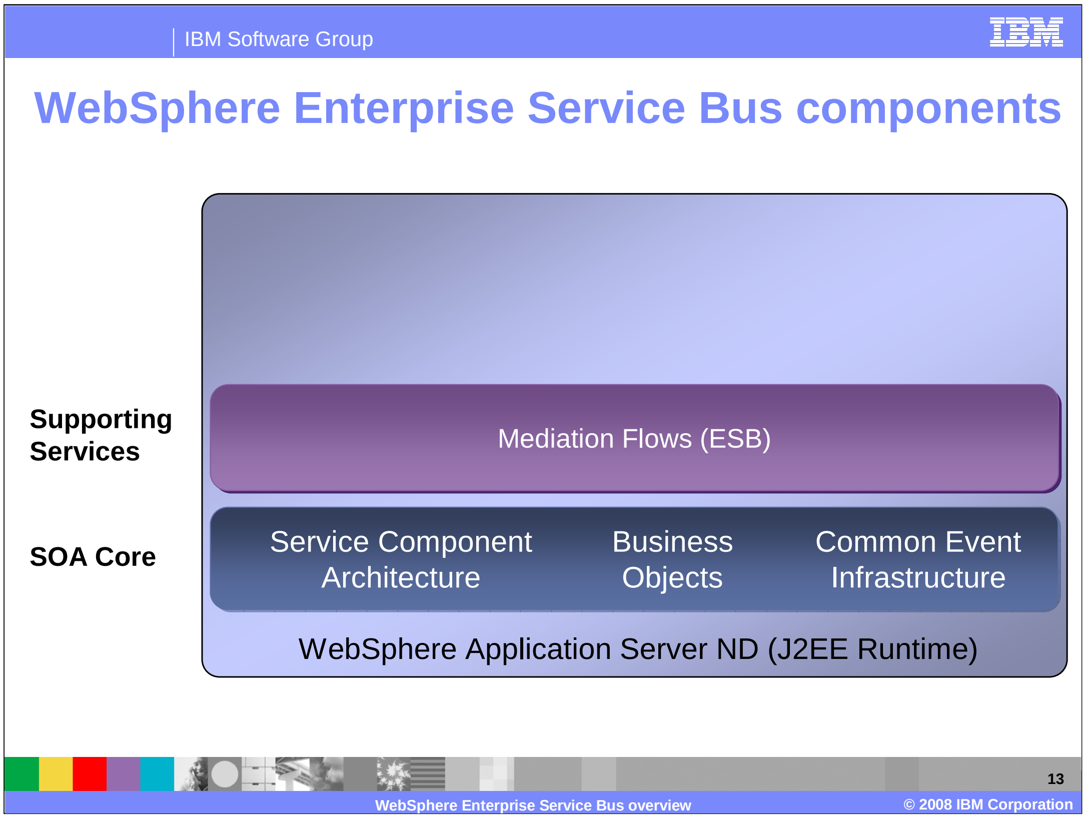
The WebSphere Application Server Network Deployment product then adds the ability to cluster application servers for scalability and high availability and centralizes server administration.

The next part of the stack is WebSphere Extended Deployment which is optional. It provides for virtualization, which helps to balance hardware resources and reduce the hardware requirements needed at times of peak demand.

WebSphere Enterprise Service bus is built on top of WebSphere Application Server Network Deployment with or without WebSphere Extended Deployment being present. As you have seen in this presentation, WebSphere Enterprise Service Bus provides the service oriented architecture and enterprise service bus functionality to be able to mediate message flows between service requestors and providers.

The top of the stack is WebSphere Process Server which focuses on enabling business processes. It provides the facilities for business process automation and integration to enable business process management applications.

Built as a stackable architecture, each product makes use of the functionality of the products on which they are built, easily allowing the extension of capabilities as needed.



IBM Software Group

WebSphere Enterprise Service Bus components

Supporting Services

SOA Core

Mediation Flows (ESB)

Service Component Architecture

Business Objects

Common Event Infrastructure

WebSphere Application Server ND (J2EE Runtime)

13

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

In this slide you can see the components that make up the WebSphere Enterprise Service Bus. As has been previously mentioned, it is built on WebSphere Application Server Network Deployment. This provides a robust J2EE application server runtime with capabilities that the ESB server implementation can exploit, such as JMS messaging, Web services support and enterprise Java beans. It can also make use of the application server qualities of service such as transactions, security and clustering. Overall, this provides a well proven and scaleable runtime environment for WebSphere Enterprise Service Bus.

The service oriented architecture core is the foundation for the WebSphere Enterprise Service Bus. The main components of the SOA core are the service component architecture, business objects and the common event infrastructure.

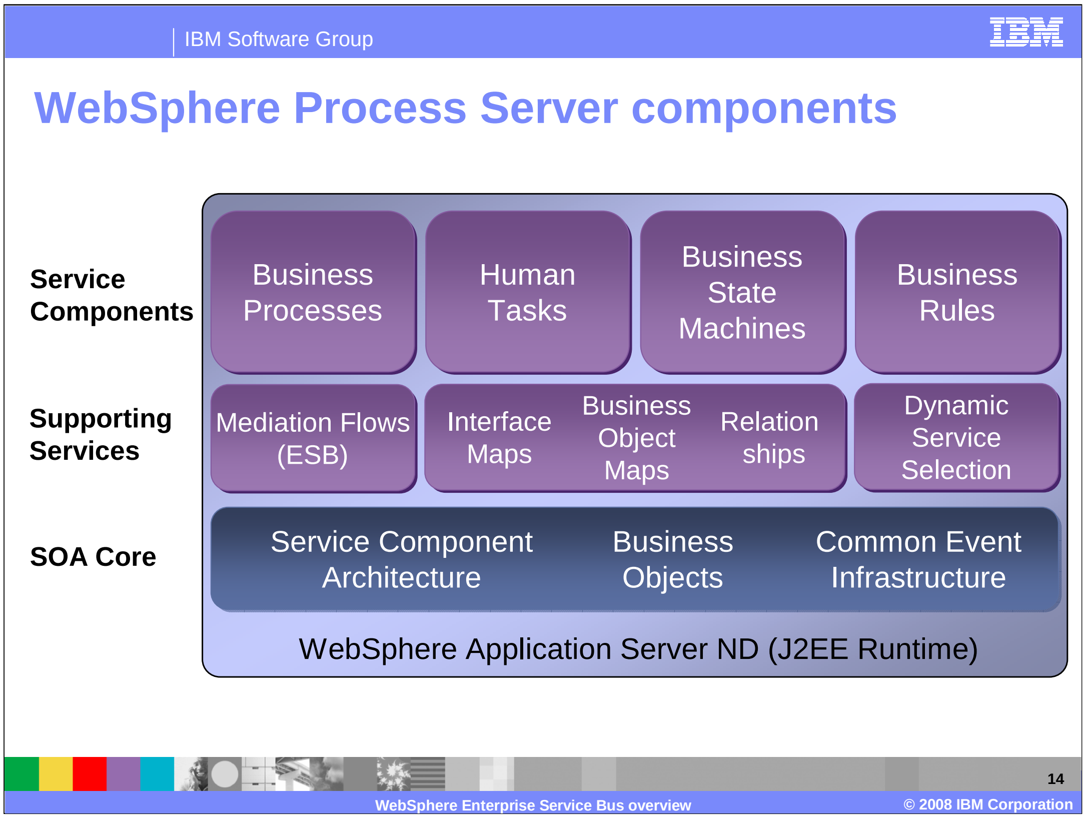
SCA is the uniform programming and invocation model for business services that publish or operate on business data. It is one of the key components of the new SOA programming model.

Business objects represent the data that is passed within that framework. Business objects are extensions to service data objects, carrying additional information needed for some integration scenarios. SDO in the form of business objects is another of the key components of the new SOA programming model.

The common event infrastructure, also known as CEI, provides the basis for the management and handling of events. It provides facilities for the generation, propagation, persistence and consumption of events.

On top of the SOA core, the heart of the WebSphere Enterprise Service Bus is provided by mediation flows. These are defined within the mediation flow component which has been fully described in previous slides of this presentation.

With SCA and mediation flows, the WebSphere Enterprise Service Bus includes all the functions and services necessary to provide the ESB services in any given integration solution.



WebSphere Process Server components

Service Components

Business Processes | Human Tasks | Business State Machines | Business Rules

Supporting Services

Mediation Flows (ESB) | Interface Maps | Business Object Maps | Relationships | Dynamic Service Selection

SOA Core

Service Component Architecture | Business Objects | Common Event Infrastructure

WebSphere Application Server ND (J2EE Runtime)

This slide expands on the previous slide to show the additional components that WebSphere Process Server introduces in addition to what WebSphere Enterprise Service Bus provides. As you can see, all the components of WebSphere Enterprise Service Bus are included in the WebSphere Process Server.

WebSphere Process Server introduces some additional supporting services to the mediation flows provided by WebSphere Enterprise Service Bus.

Interface maps, business object maps and relationships combine to handle transformation essential to enabling the integration and synchronization of disparate backend systems within an enterprise.

Dynamic service selection provides the capability to invoke different component implementations of the same interface based on a date and time criteria.

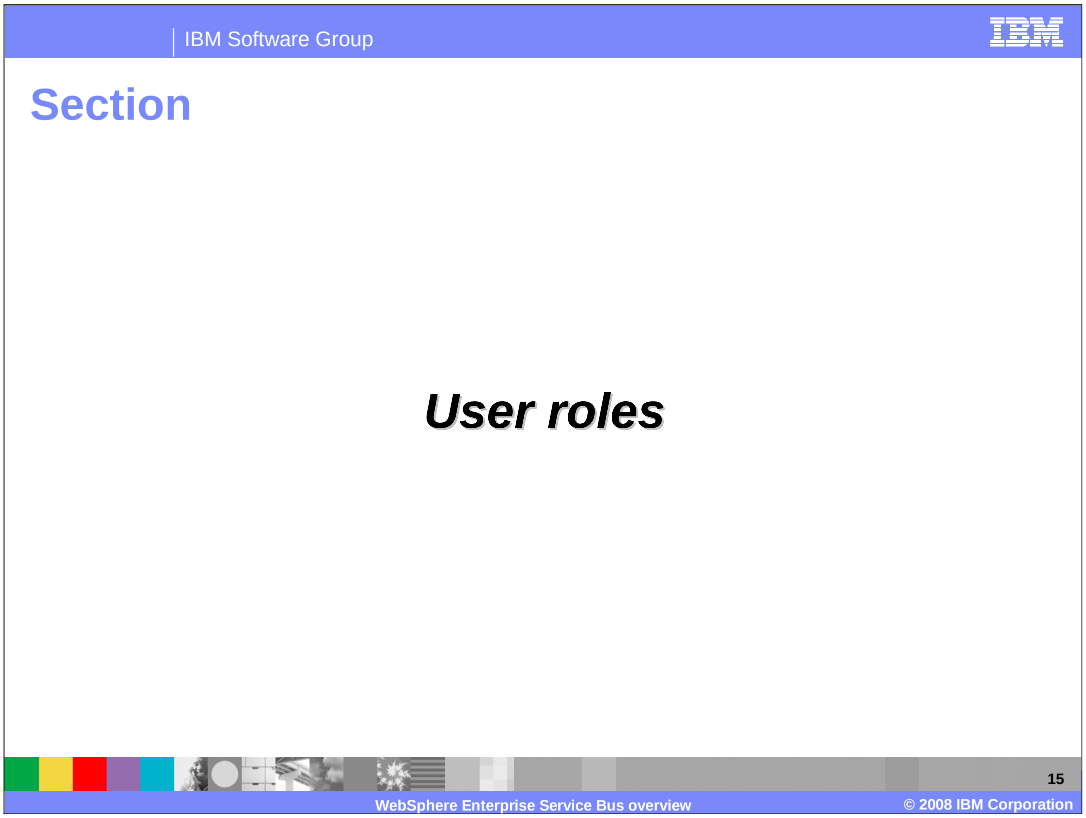
On top of the supporting services are the service components, which are the primary SCA components for enabling business process application function.

Business processes provide an implementation of a process model that describes the logical order in which the different activities of the process take place, making calls out to the individual SCA services that implement the specific activities.

Human tasks allow people to participate in a business process. A human task may integrate into the overall business process in a machine-to-human scenario, a human-to-machine scenario and in a human-to-human scenario.

The business state machine service component provides another way of modeling a business process. It is used when the business process can be easily thought of in terms of a state transition diagram.

Business rules provide a means for implementing and enforcing business policy through externalization of business function. Externalization enables the business rules to be managed independently from other aspects of an application.



IBM Software Group

Section

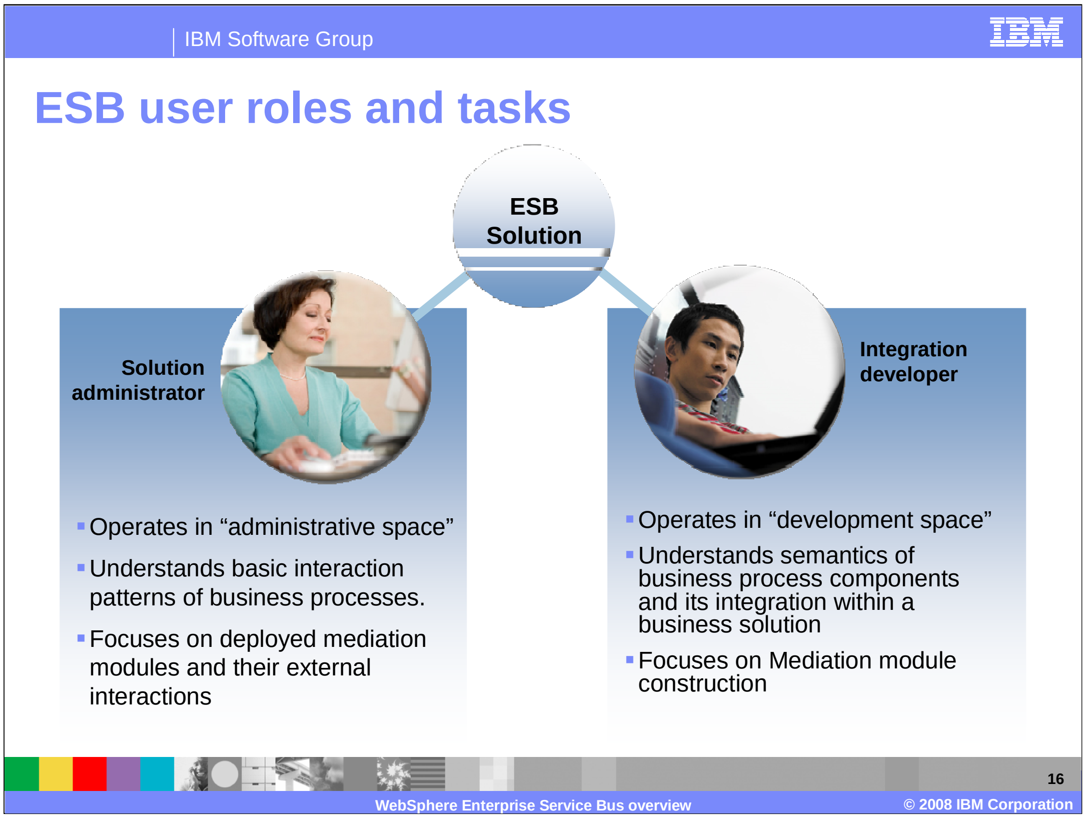
User roles

15

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

This section will take a look at the user roles that are typically associated with the WebSphere Enterprise Service Bus.



IBM Software Group

ESB user roles and tasks

ESB Solution

Solution administrator

- Operates in “administrative space”
- Understands basic interaction patterns of business processes.
- Focuses on deployed mediation modules and their external interactions

Integration developer

- Operates in “development space”
- Understands semantics of business process components and its integration within a business solution
- Focuses on Mediation module construction

16

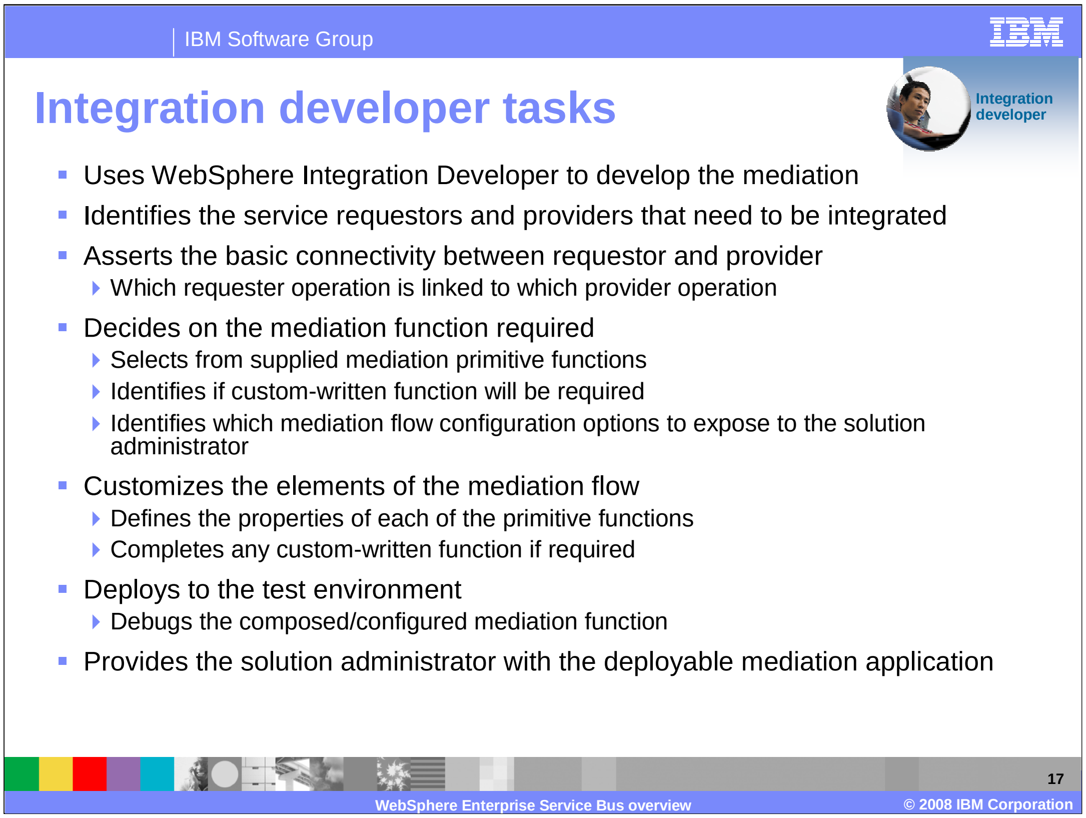
WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

There are two primary user roles that are associated with the WebSphere Enterprise Service Bus.

The first of these is the integration developer. They use the WebSphere Integration Developer tool and they focus on creating the mediation module and defining the mediation flows. Integration developers need to understand the semantics of the various services available and how they can be integrated together within a business solution.

The solution administrator uses WebSphere Enterprise Service Bus or WebSphere Process Server to manage the mediation module within the server runtime environment. In some situations, the solution administrator may need to understand the basic interaction patterns of the business processes to change the routing of the business process if needed.



IBM Software Group

Integration developer tasks

Integration developer

- Uses WebSphere Integration Developer to develop the mediation
- Identifies the service requestors and providers that need to be integrated
- Asserts the basic connectivity between requestor and provider
  - Which requester operation is linked to which provider operation
- Decides on the mediation function required
  - Selects from supplied mediation primitive functions
  - Identifies if custom-written function will be required
  - Identifies which mediation flow configuration options to expose to the solution administrator
- Customizes the elements of the mediation flow
  - Defines the properties of each of the primitive functions
  - Completes any custom-written function if required
- Deploys to the test environment
  - Debugs the composed/configured mediation function
- Provides the solution administrator with the deployable mediation application

17

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

This slide takes a closer look at some of the tasks performed by the integration developer for developing a mediation flow using the WebSphere Integration Developer tool.

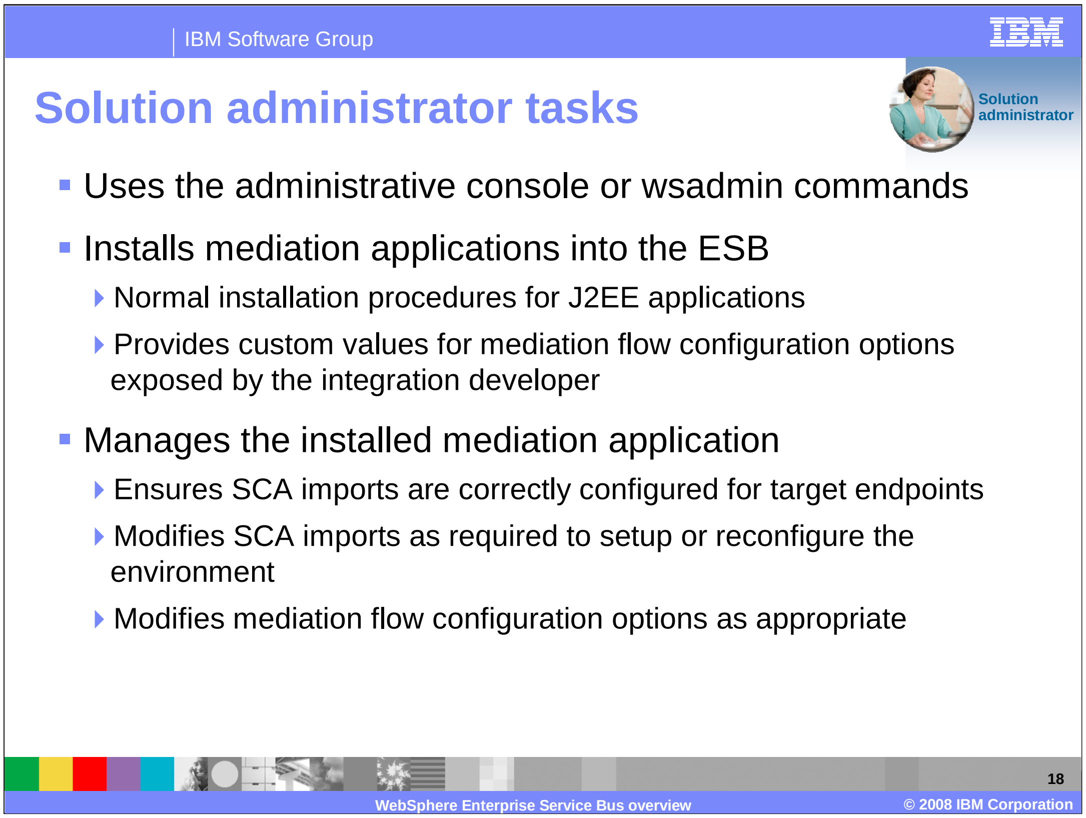
First they need to identify the service requestors and providers that will be interacting with the bus. They need to understand the interface and the protocols used by the requestors and providers. The mediation module is then started by defining the appropriate SCA exports and imports needed to communicate with the requestors and providers.

For each operation on the requestors interface, flow logic must be defined for the request and the response. If the service provider supports a different interface, the first thing the integration developer must do is determine which operations on the provider interface will be used to satisfy incoming requests. Then each of the flows must be constructed by selecting the appropriate mediation primitives, wiring them together and customizing their properties to perform the required functions. This may also include the need to write Java code for a custom mediation primitive.

Another decision the integration developer must make is to determine which of the configuration properties of the mediation flow should be exposed at runtime, this allowing a level of runtime configuration of the flow’s behavior.

Finally, the integration developer can use the WebSphere Integration Developer unit test environment servers to test and debug the message flow through the mediation module.

When it has been fully tested, it can be exported from the WebSphere Integration Developer as a J2EE application and provided to the solution administrator.



IBM Software Group

IBM

Solution administrator tasks

Solution administrator

- Uses the administrative console or wsadmin commands
- Installs mediation applications into the ESB
  - Normal installation procedures for J2EE applications
  - Provides custom values for mediation flow configuration options exposed by the integration developer
- Manages the installed mediation application
  - Ensures SCA imports are correctly configured for target endpoints
  - Modifies SCA imports as required to setup or reconfigure the environment
  - Modifies mediation flow configuration options as appropriate

18

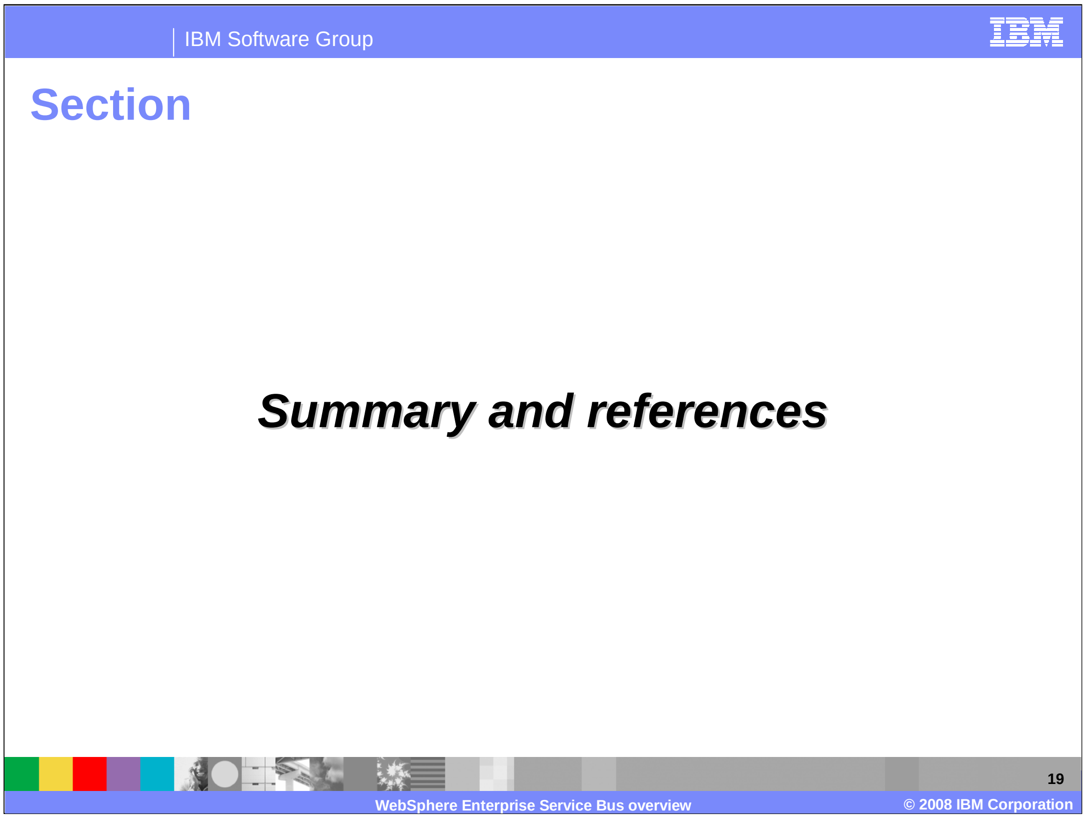
WebSphere Enterprise Service Bus overview

The solution administrator operates in the runtime environment for WebSphere Enterprise Service Bus, using the administrative console or the wsadmin commands to administer the ESB applications.

The first task is to install the mediation application. Mediation applications are installed in the same way any other J2EE application would be installed into WebSphere Application Server. However, the mediation application may contain mediation flow configuration properties which the integration developer choose to expose to the solution administrator. If so, the default values specified at development time may need to be changed during the installation of the application.

Once the application is installed, the solution administrator needs to make sure that the SCA imports are correctly configured for the service provider endpoints they should be associated with. If not, these can be administratively changed at this time.

Finally, while the mediation application is running, there may be a need to modify configuration options if the behavior of the flow needs to be changed.



IBM Software Group

Section

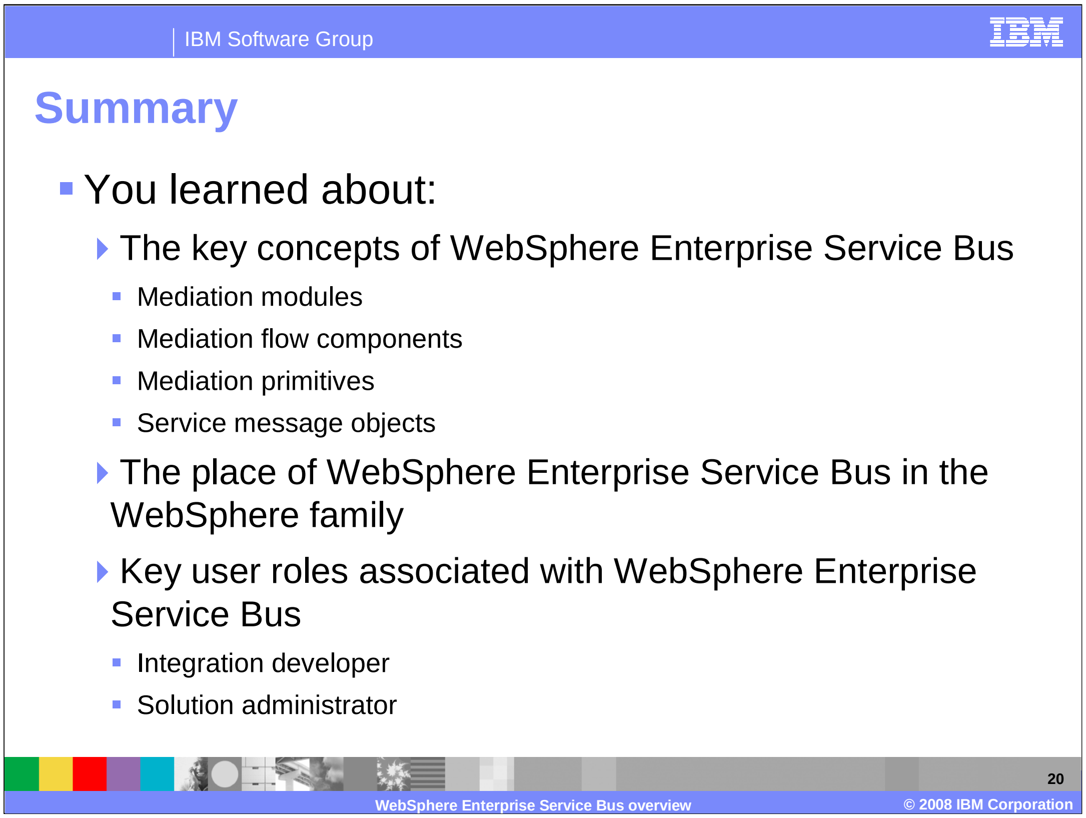
Summary and references

19

WebSphere Enterprise Service Bus overview

© 2008 IBM Corporation

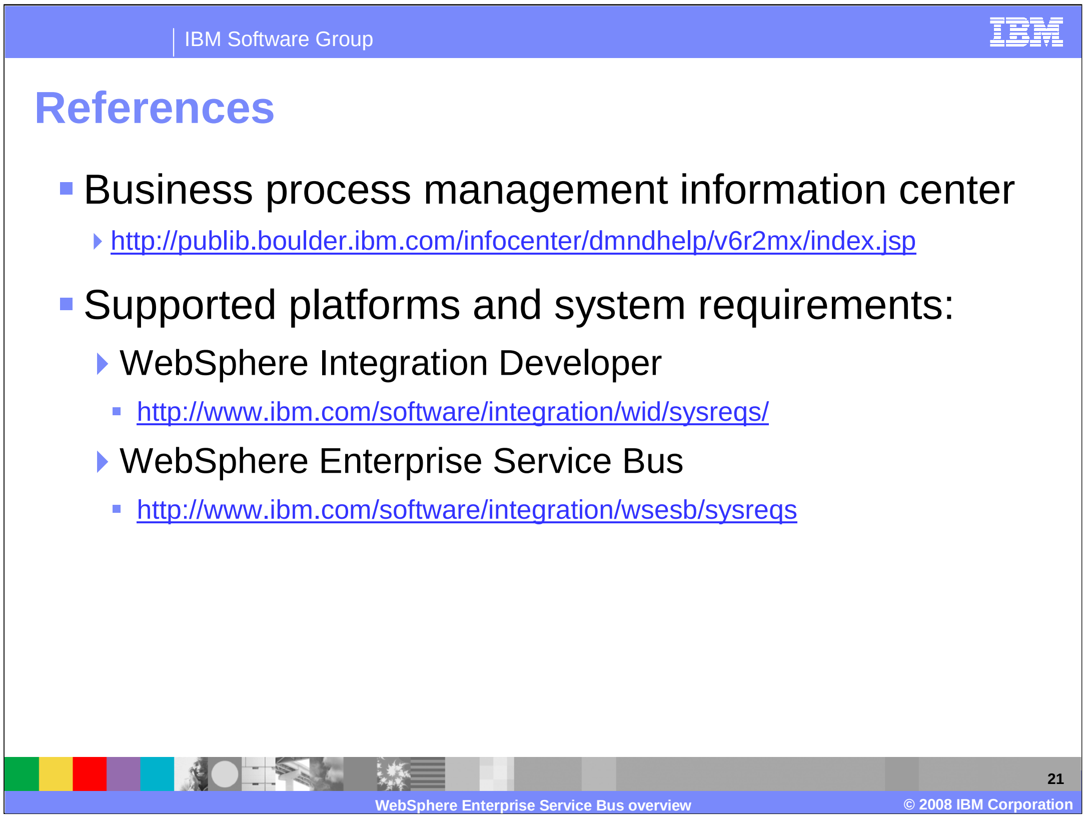
The last section of the presentation provides a summary and some references.



Summary

- You learned about:
  - The key concepts of WebSphere Enterprise Service Bus
    - Mediation modules
    - Mediation flow components
    - Mediation primitives
    - Service message objects
  - The place of WebSphere Enterprise Service Bus in the WebSphere family
  - Key user roles associated with WebSphere Enterprise Service Bus
    - Integration developer
    - Solution administrator

In this presentation you learned about the key concepts of the WebSphere Enterprise service bus, specifically the mediation module, mediation flow component, mediation primitives and service message objects. Then the place of the WebSphere Enterprise Service Bus in the WebSphere application server stack was examined. Finally, you learned that the integration developer and the solution administrator are the primary user roles that are associated with the WebSphere Enterprise Service Bus.

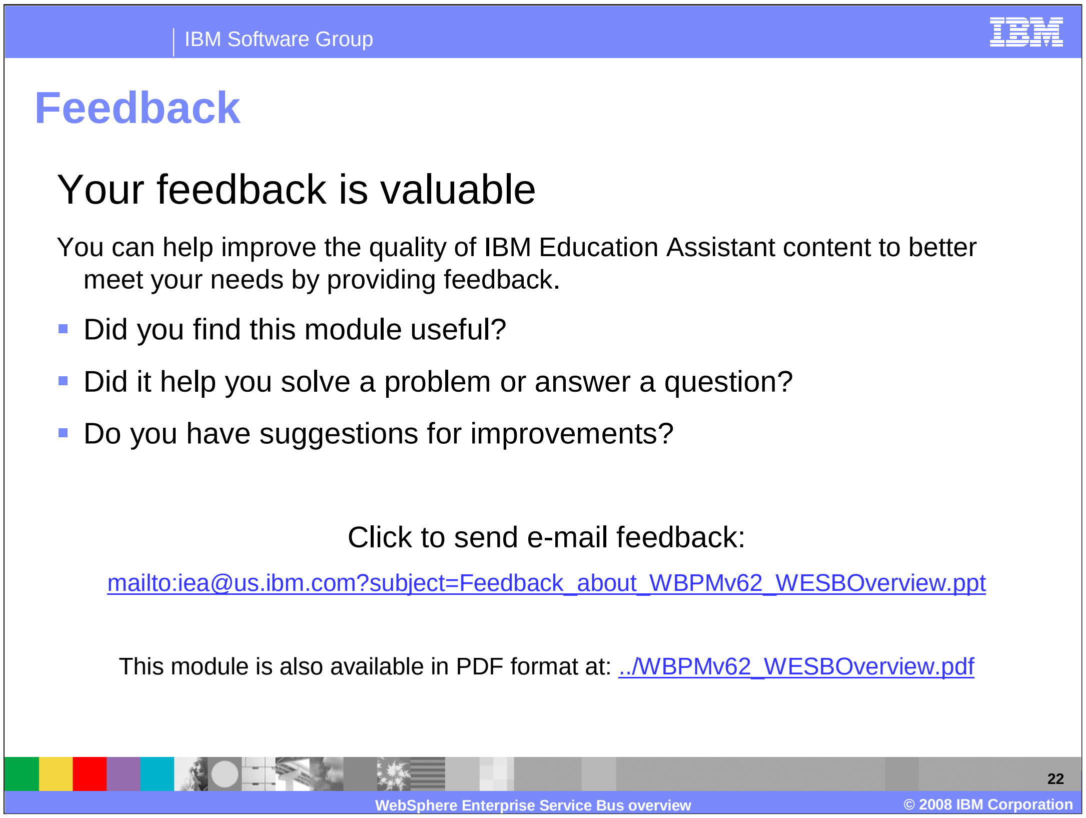


References

- Business process management information center
  - http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp
- Supported platforms and system requirements:
  - WebSphere Integration Developer
    - http://www.ibm.com/software/integration/wid/sysreqs/
  - WebSphere Enterprise Service Bus
    - http://www.ibm.com/software/integration/wsesb/sysreqs

On this slide you will find a link to the IBM business process management information center which includes the documentation for the WebSphere Enterprise Service Bus and the WebSphere Integration Developer.

There are also links to the IBM software pages defining the platforms supported and system requirements for WebSphere Integration Developer and WebSphere Enterprise Service Bus.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

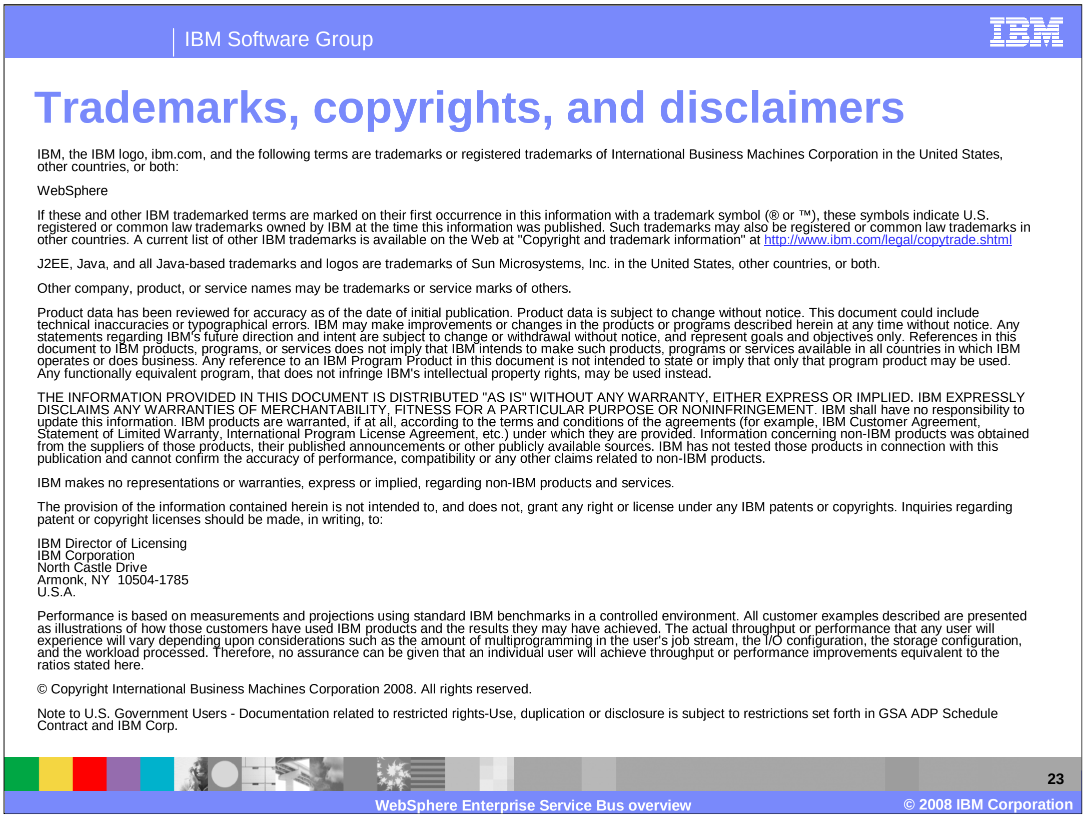
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv62_WESBOverview.ppt

This module is also available in PDF format at: ../WBPMv62_WESBOverview.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

J2EE, Java, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or 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 or changes in the products or programs 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.

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 (for example, 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.