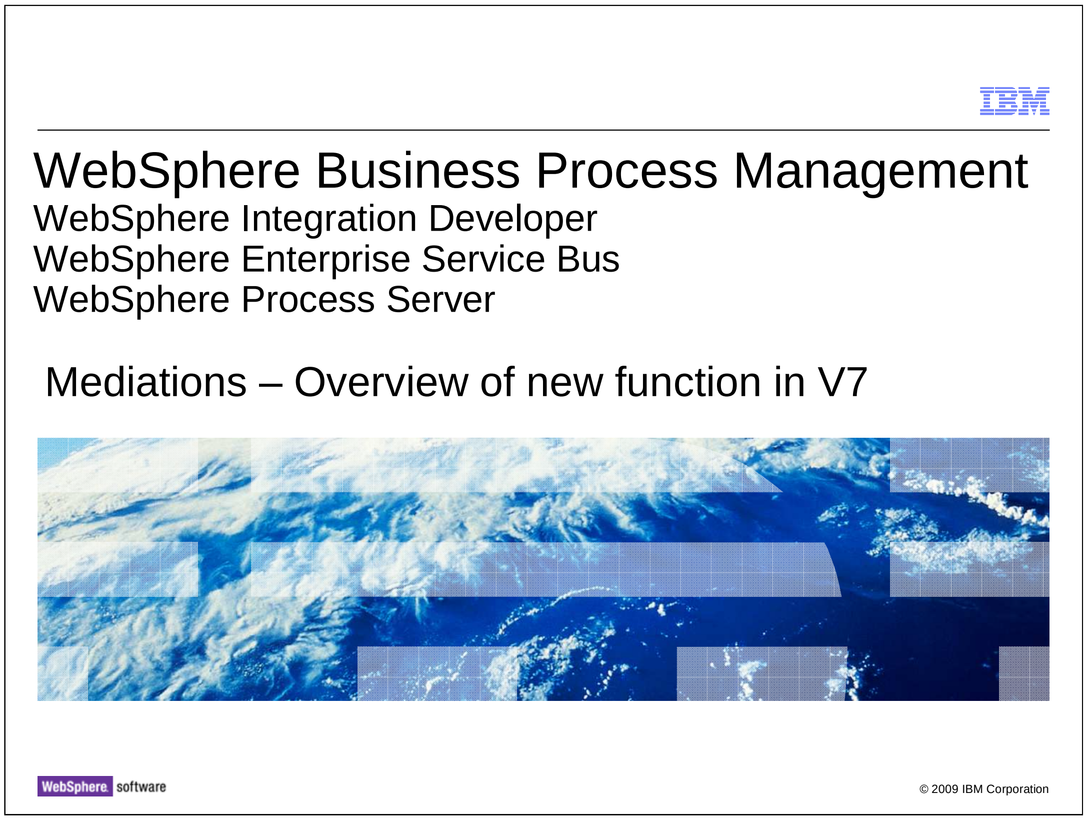
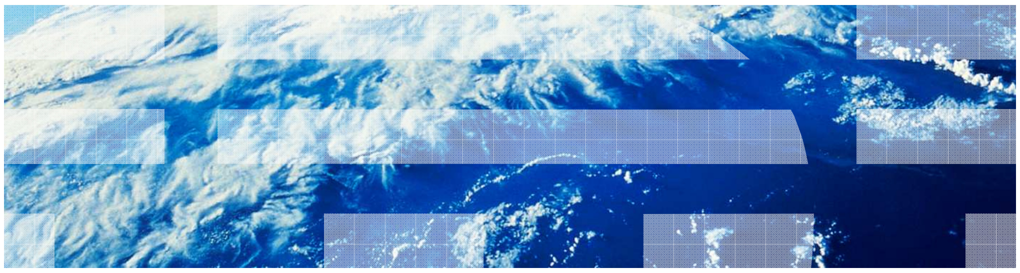


WebSphere Business Process Management

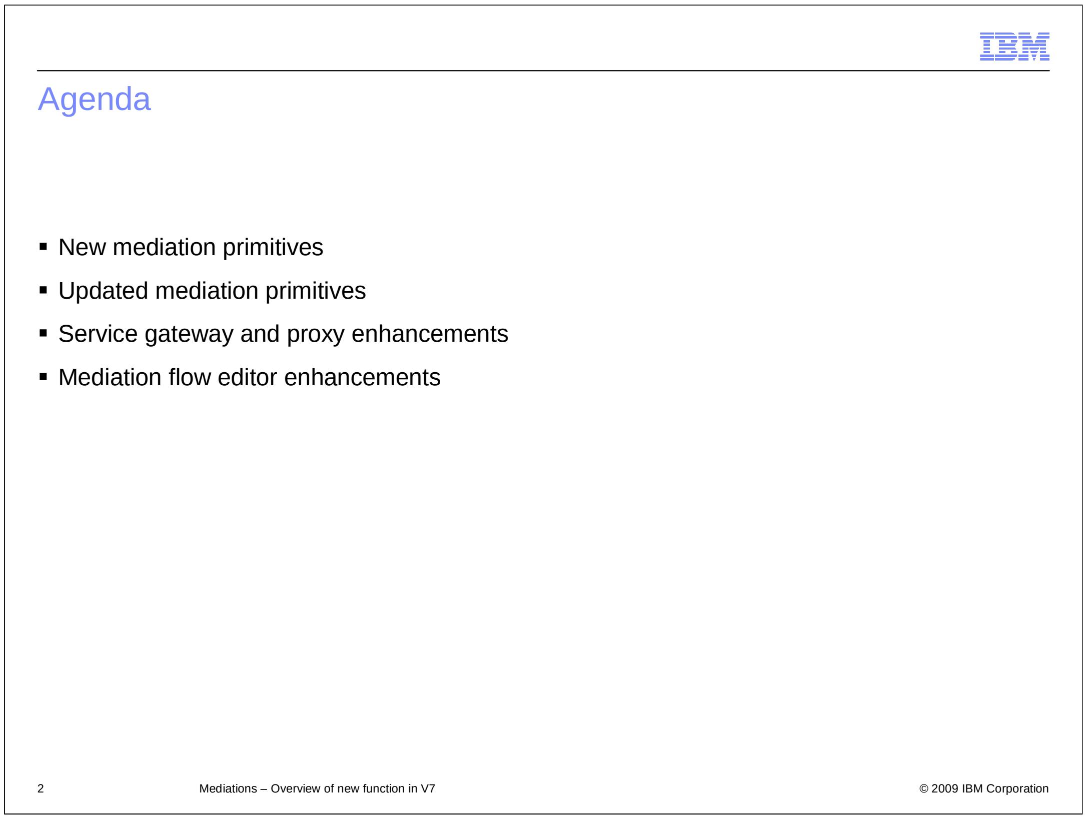
WebSphere Integration Developer
WebSphere Enterprise Service Bus
WebSphere Process Server

Mediations – Overview of new function in V7



WebSphere software

This presentation looks at the new mediation functionality added to the WebSphere® Business Process Management products in version seven. The products to which this applies are the WebSphere Integration Developer, the WebSphere Enterprise Service Bus and the WebSphere Process Server.



Agenda

- New mediation primitives
- Updated mediation primitives
- Service gateway and proxy enhancements
- Mediation flow editor enhancements

2 Mediations – Overview of new function in V7 © 2009 IBM Corporation

Several new mediation primitives are added to enhance the capabilities of mediation flows. Additionally, there are some existing mediation primitives that are updated to provide additional functionality. Enhancements to mediation flow capabilities are in the service gateway and proxy patterns. To enhance the experience of creating mediation flows, the mediation flow editor has improved usability and functionality. This presentation provides an overview of these new capabilities.

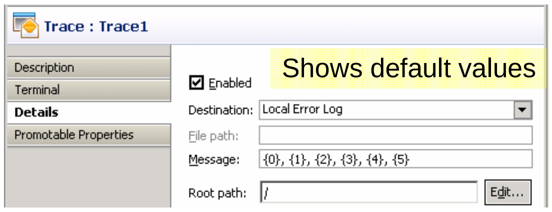
Section

New mediation primitives

The new mediation primitives are introduced in this section.

Trace primitive

- Enables writing a user defined trace record
  - User specified format with inserts
  - All or part of the SMO can be included
- Trace written to a configurable file destination
- Useful during development and debug of a mediation flow
  - Eliminates need to use custom mediation primitive with Java™ code
- Tracing can be toggled on/off
  - Promoting the enabled property allows this to be dynamic



☑ Enabled
Destination: Local Error Log
File path:
Message: {0}, {1}, {2}, {3}, {4}, {5}
Root path: /

4 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The trace primitive enables you to define a trace record to be written from within a mediation flow. You can configure a message for the trace record that includes several configurable inserts, such as a timestamp and the name of the mediation flow. The portion of the SMO to be included in the message is defined using an XPath expression.

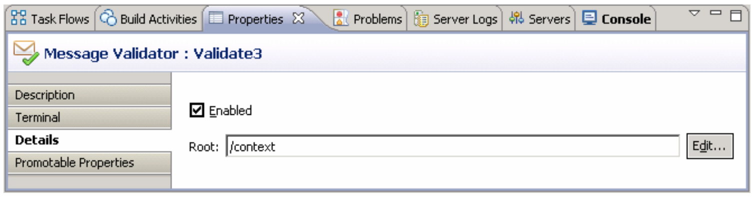
The file to which the trace record is written is configurable. This can be the system log file for the server, a designated trace file or file that you name with either a relative or absolute path.

The trace primitive is primarily useful during the development and debugging of a mediation flow to enable you to see portions of the SMO at selected points in the flow. In previous releases, it was common to do this with a custom mediation primitive. This necessitated the authoring of Java code to write the SMO to the system log or to a file. The trace primitive eliminates the need to write Java code to accomplish this.

The enabled property, which is promotable, allows you to keep trace primitives in your flow and toggle the use of them on and off as needed.

Message validator primitive

- Validates SMO against schema and assertions
  - SMO schema, schema of SMO body (message type), configured schema of SMO contexts
  - Downcast type assertions of weakly typed elements (such as those defined with the set message type primitive)
- Configurable to specify which part of SMO to validate
- Behavior
  - Validation successful
    - Unchanged message propagated through the out terminal
  - Validation fails
    - If fail terminal is not wired → Message written to the system log file
    - If fail terminal is wired → failInfo element contains exception information



Message Validator : Validate3

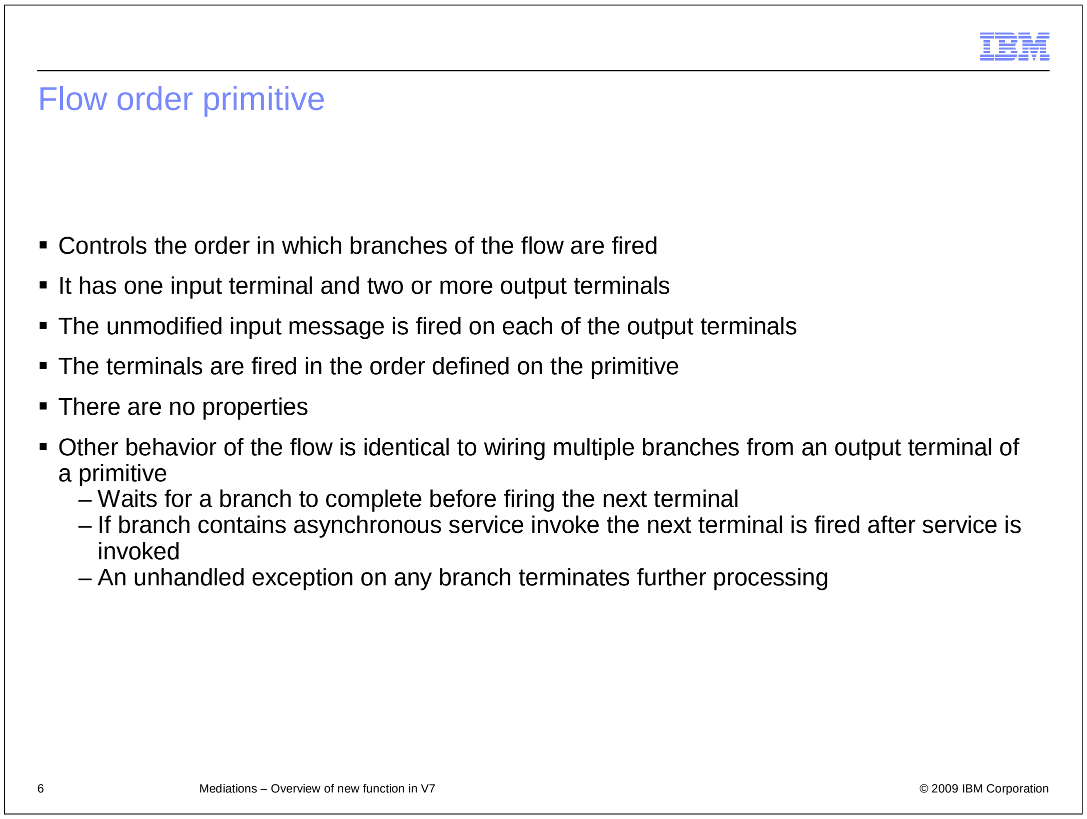
☑ Enabled

Root: /context

The message validator primitive allows you to perform a validation check of the SMO at selected points in a mediation flow. There are various portions that are checked to make sure the schema definitions and actual SMO contents are consistent with each other. There is the SMO schema itself that defines the overall structure of the SMO and parts of the context and most of the headers. There is the body of the SMO, also referred to as the message type, defined by the operation. There are user defined contexts that are defined by business object schema specified as part of the flow. Finally, there are weakly typed fields that have been downcast to more specific types.

The primitive is configurable using an XPath expression to specify what part of the SMO should be validated. When the validation is successful, the SMO is propagated unchanged. When the validation fails, the path from the fail terminal is followed if it is wired, otherwise a message is written to the system log and the flow is terminated.

The message validator is configurable with the enabled property which is promotable. This allows you to keep them in your flow without validation being performed, which avoids the overhead of the validation checking. However, they can be enabled when required to do debugging of a problem.



Flow order primitive

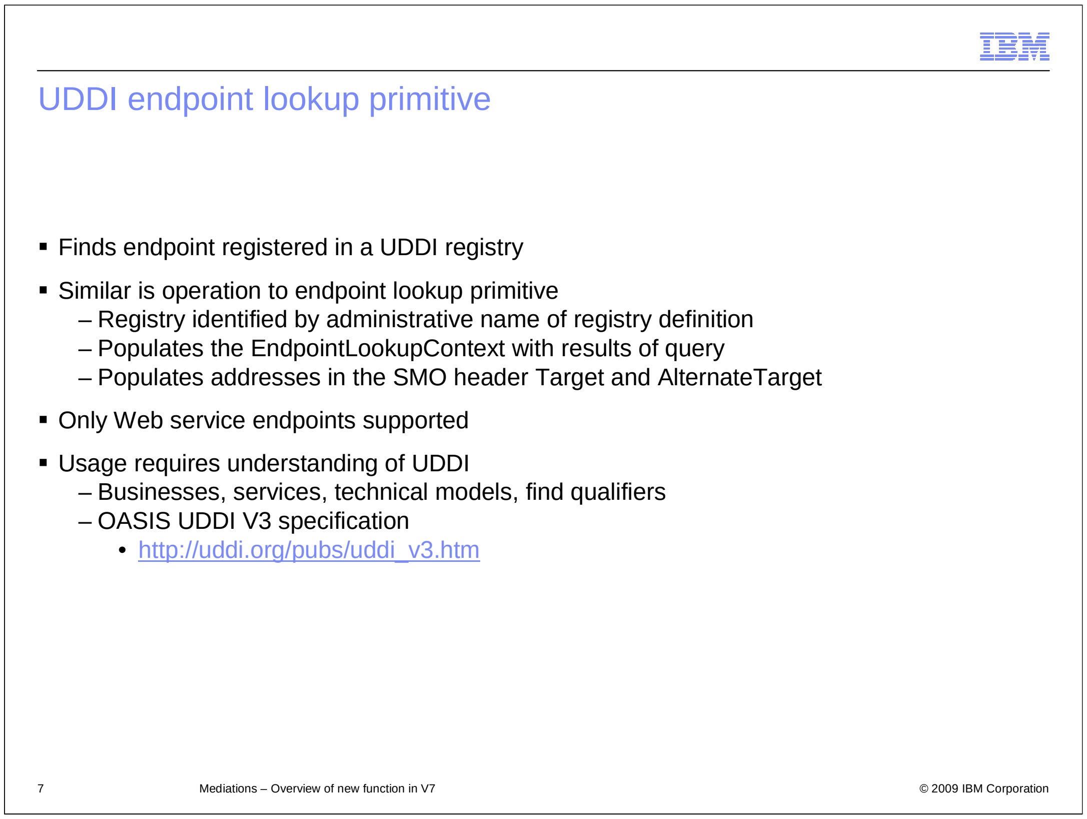
- Controls the order in which branches of the flow are fired
- It has one input terminal and two or more output terminals
- The unmodified input message is fired on each of the output terminals
- The terminals are fired in the order defined on the primitive
- There are no properties
- Other behavior of the flow is identical to wiring multiple branches from an output terminal of a primitive
  - Waits for a branch to complete before firing the next terminal
  - If branch contains asynchronous service invoke the next terminal is fired after service is invoked
  - An unhandled exception on any branch terminates further processing

6 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The flow order primitive is used to control the order in which branches of a flow are taken. It has a single input terminal and two or more output terminals. The unmodified message is propagated on each branch of the flow by firing the output terminals in the order that they are defined on the primitive. There are no properties associated with this primitive.

To understand the use of this primitive, it is useful to understand the behavior for any primitive when an output terminal has more than one wire coming from it. When the output terminal is fired, one of the branches is followed and runs until it completes or an asynchronous service invoke is called. After that, another branch is followed in the same way. This continues until all the wires coming from the output terminal have been processed. The order in which they are processed is indeterminate. If an unhandled exception occurs on any of the branches, the mediation flow is terminated.

What the flow order primitive provides is a way for this same behavior to be exhibited, but to control the order in which the branches are processed.



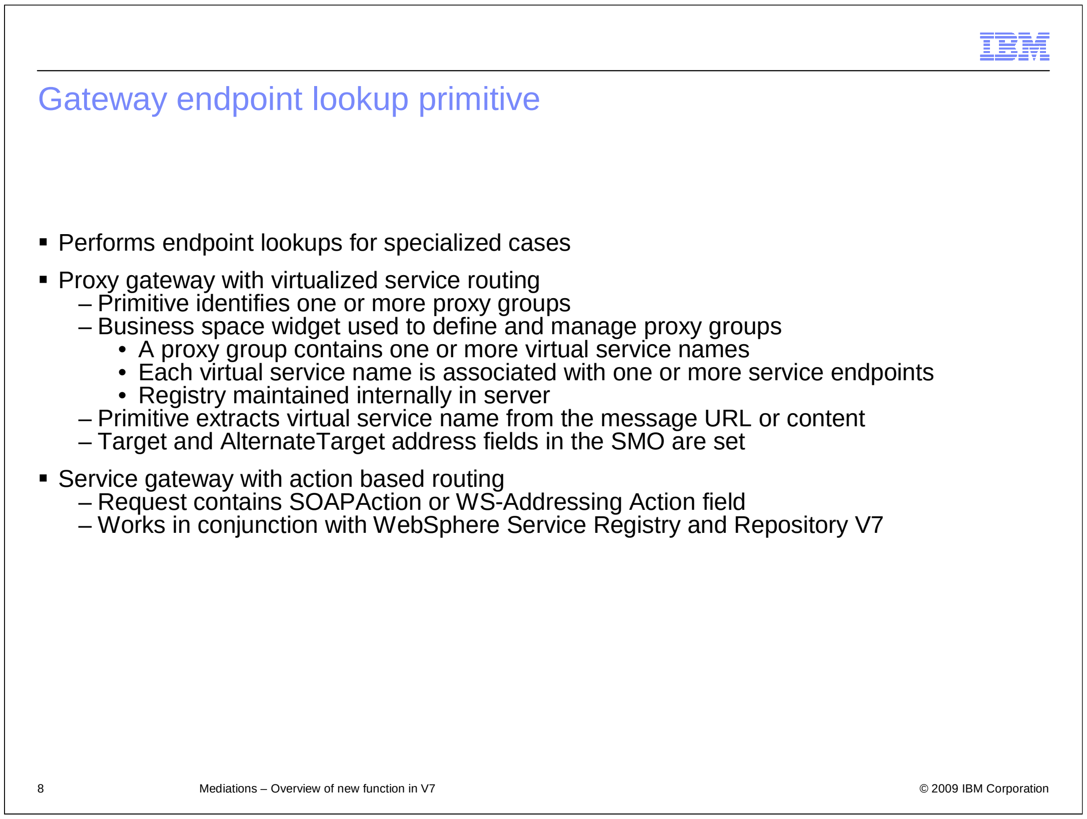
UDDI endpoint lookup primitive

- Finds endpoint registered in a UDDI registry
- Similar is operation to endpoint lookup primitive
  - Registry identified by administrative name of registry definition
  - Populates the EndpointLookupContext with results of query
  - Populates addresses in the SMO header Target and AlternateTarget
- Only Web service endpoints supported
- Usage requires understanding of UDDI
  - Businesses, services, technical models, find qualifiers
  - OASIS UDDI V3 specification
    - http://uddi.org/pubs/uddi_v3.htm

7 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The UDDI endpoint lookup primitive is used to perform queries against a UDDI registry to find registered endpoints that satisfy the query. The operation of this primitive is similar to the operation of the endpoint lookup primitive, with these similarities. The endpoint for the registry is administratively assigned a name and the primitive uses the name to identify the registry to be used for the query. The result of the query is used to populate the EndpointLookupContext in the SMO. Additionally, the endpoint addresses are populated into the SMO header fields Target and AlternateTarget, which makes them useful for dynamic addressing. However, unlike the endpoint lookup primitive, the UDDI endpoint lookup only supports Web service endpoints.

To make use of this primitive, you have to understand how a UDDI registry classifies endpoints. This includes the concepts of businesses, services, technical models, and find qualifiers. All of these can be specified when configuring the primitive to define the query that is made by the primitive to the registry. To understand more about UDDI, the OASIS UDDI V3 specification should be consulted. A link to the specification is contained on the slide.



Gateway endpoint lookup primitive

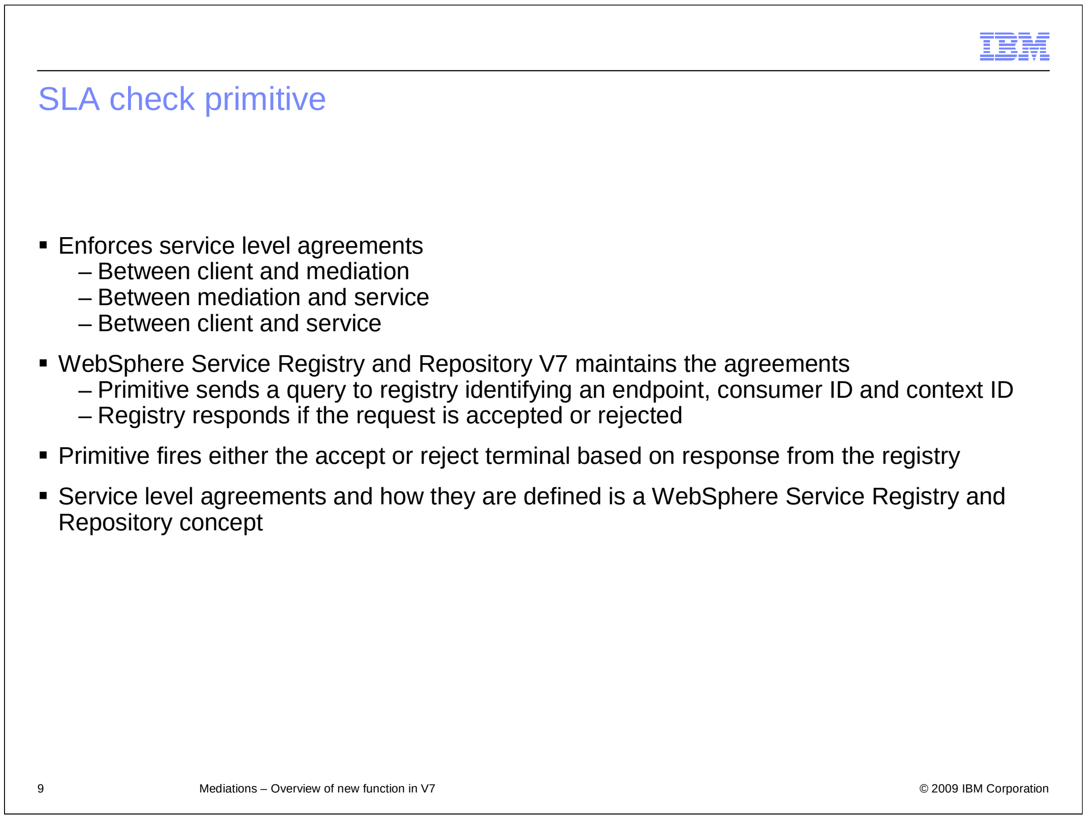
- Performs endpoint lookups for specialized cases
- Proxy gateway with virtualized service routing
  - Primitive identifies one or more proxy groups
  - Business space widget used to define and manage proxy groups
    - A proxy group contains one or more virtual service names
    - Each virtual service name is associated with one or more service endpoints
    - Registry maintained internally in server
  - Primitive extracts virtual service name from the message URL or content
  - Target and AlternateTarget address fields in the SMO are set
- Service gateway with action based routing
  - Request contains SOAPAction or WS-Addressing Action field
  - Works in conjunction with WebSphere Service Registry and Repository V7

8 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The gateway endpoint lookup primitive is used in specialized cases to locate endpoints from a registry. The primary usage is for the proxy gateway pattern introduced in version seven. It can also be used in service gateway patterns when SOAPAction or WS_Addressing Action fields are used in the request. In both cases, only JAX-WS Web service bindings are supported.

In the case of the proxy gateway, service endpoints that are identified by a virtual service name are looked up from an internal registry. The primitive itself is configured to identify one or more proxy groups. The proxy groups are defined and managed in the internal registry and are administered using a business space widget. Each proxy group can identify one or more virtual service names, and each virtual service name can identify one or more endpoints. The primitive extracts the virtual service name from the message based on how the primitive is configured. It can be obtained from either the URL or as identified by an XPath expression from the message header or body. The endpoints associated with the virtual service name are populated into the Target and AlternateTarget fields of the SMO, where they can then be used for dynamic addressing.

In the case of the service gateway, the primitive supports requests that contain an action specified as a SOAPAction or WS-Addressing action. A query is made to the WebSphere Service Registry and Repository to obtain the appropriate endpoints. Version seven of WebSphere Service Registry and Repository is required to support these queries.



SLA check primitive

- Enforces service level agreements
  - Between client and mediation
  - Between mediation and service
  - Between client and service
- WebSphere Service Registry and Repository V7 maintains the agreements
  - Primitive sends a query to registry identifying an endpoint, consumer ID and context ID
  - Registry responds if the request is accepted or rejected
- Primitive fires either the accept or reject terminal based on response from the registry
- Service level agreements and how they are defined is a WebSphere Service Registry and Repository concept

9 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The SLA check primitive is used to enable the enforcement of service level agreements. The service level agreements can be made between a client and the mediation, between the mediation and a service or between a client and a service. WebSphere Service Registry and Repository version seven is used to maintain the agreements. The primitive sends a query to the registry, passing an endpoint, a consumer ID and a context ID. The registry then responds with an indication if the request should be accepted or rejected based on the SLAs configured in the registry. The primitive then fires either the accept or reject output terminal. Service level agreements, how they are defined and what they represent, is a concept of WebSphere Service Registry and Repository version seven. To understand them you should consult the WebSphere Service Registry and Repository information center.

Section

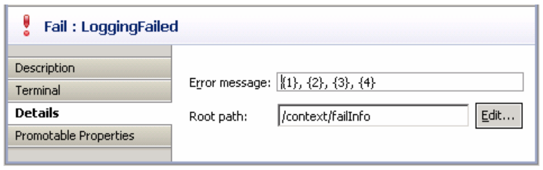
Updated mediation primitives

10 Mediations – Overview of new function in V7 © 2009 IBM Corporation

This section discusses the enhancements to existing mediation primitives.

Fail primitive

- The fail primitive has been enhanced:
  - In version six only a static message can be specified
  - Version seven now allows inserts from the SMO into the message

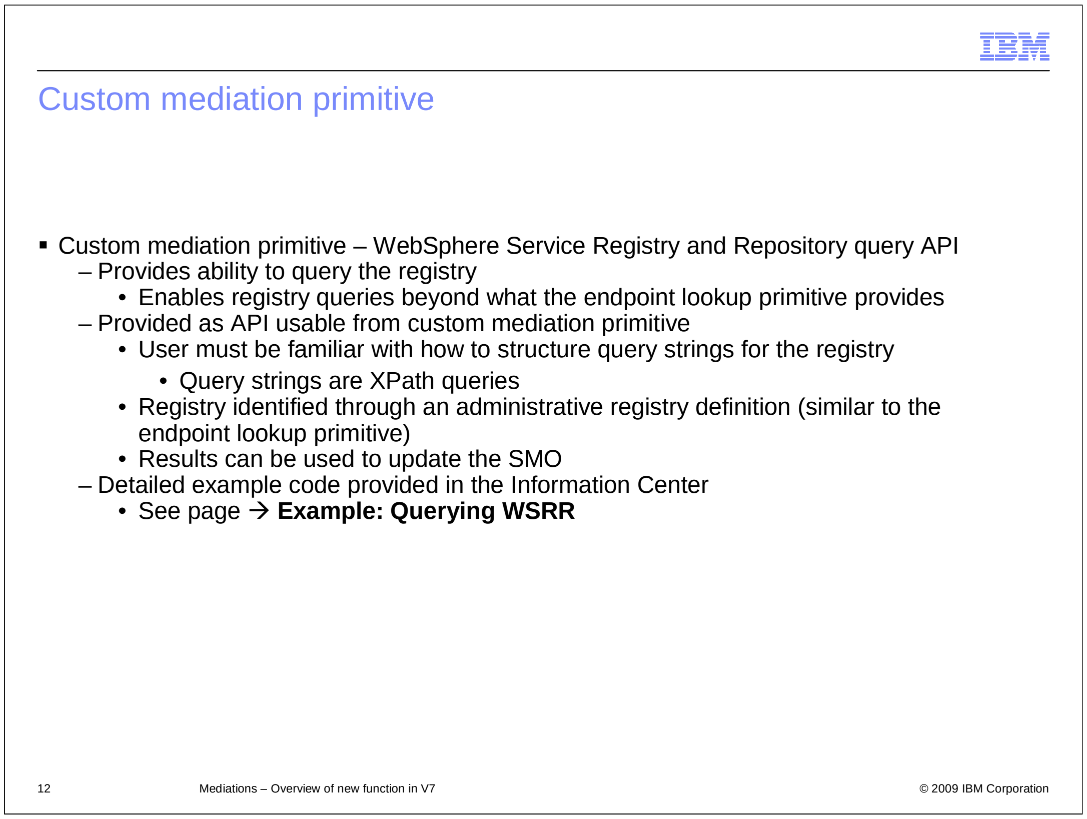


Error message: {1}, {2}, {3}, {4}

Root path: /context/failInfo

  - Inserts defined by {n} – (identical to the trace and message logger primitives)
    - {0} – Time stamp – indicates when the message was logged
    - {1} – Message ID – the message ID from the SMO
    - {2} – Mediation name – the name of the fail primitive that logged the message
    - {3} – Module name – the name of the mediation module containing the message logger
    - {4} – Message – the message as defined by the XPath specified in the root property
    - {5} – Version – the version of the SMO

The fail primitive in version six allowed you to specify a static message. In version seven, you can now specify a message to be formatted with selected inserts. The inserts that can be specified are the same ones that can be used with the message logger primitive and the trace primitive. These include a time stamp, message ID, name of the fail mediation primitive, name of the mediation module and SMO version. You use an XPath expression to identify a portion of the SMO to be inserted.



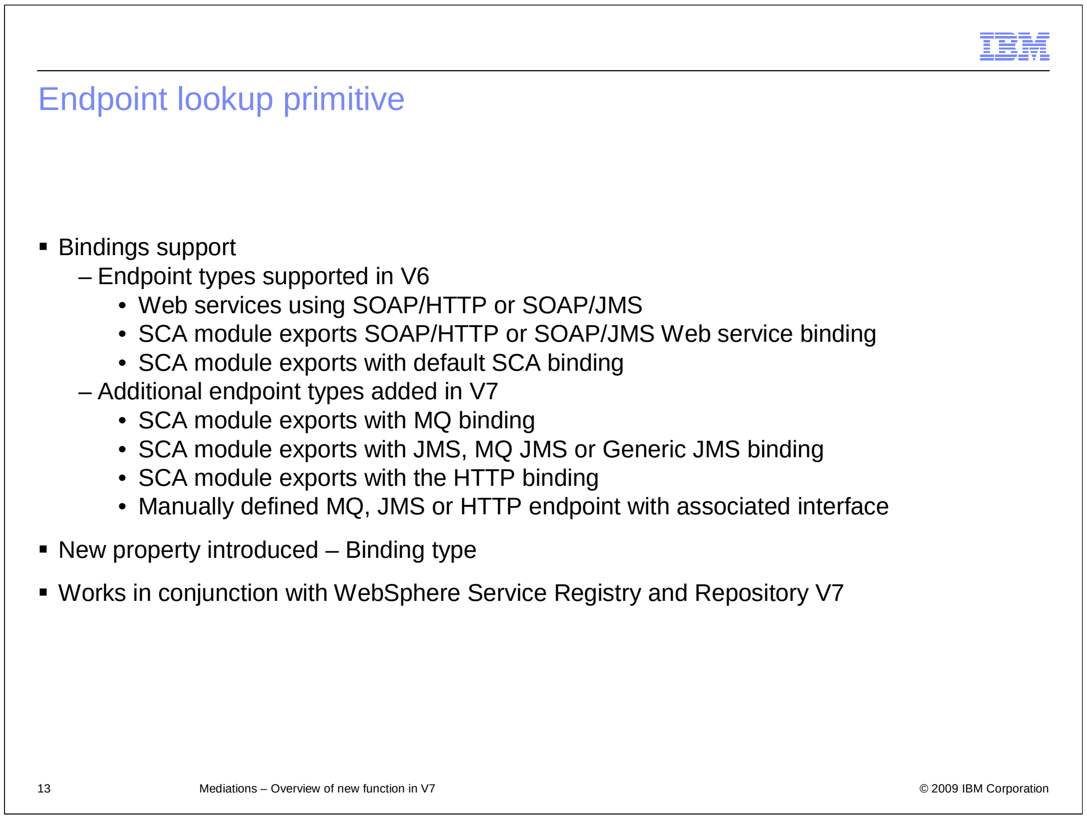
Custom mediation primitive

- Custom mediation primitive – WebSphere Service Registry and Repository query API
  - Provides ability to query the registry
    - Enables registry queries beyond what the endpoint lookup primitive provides
  - Provided as API usable from custom mediation primitive
    - User must be familiar with how to structure query strings for the registry
      - Query strings are XPath queries
    - Registry identified through an administrative registry definition (similar to the endpoint lookup primitive)
    - Results can be used to update the SMO
  - Detailed example code provided in the Information Center
    - See page → **Example: Querying WSRR**

12 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The custom mediation primitive itself has not been enhanced, but a new API has been introduced that allows you to invoke a query to the WebSphere Service Registry and Repository. This enables you to do any query to the registry from the custom mediation primitive. In order to use the API, you must be familiar with how to use XPath queries to properly structure a query string to send to the registry. The registry to use is identified through a named registry definition that is administratively maintained. This is the same registry definition used by an endpoint lookup primitive. Within your code in the custom mediation primitive, you can use the query results to update the SMO.

The information center provides a detailed code example under the topic heading "Example: Querying WSRR"



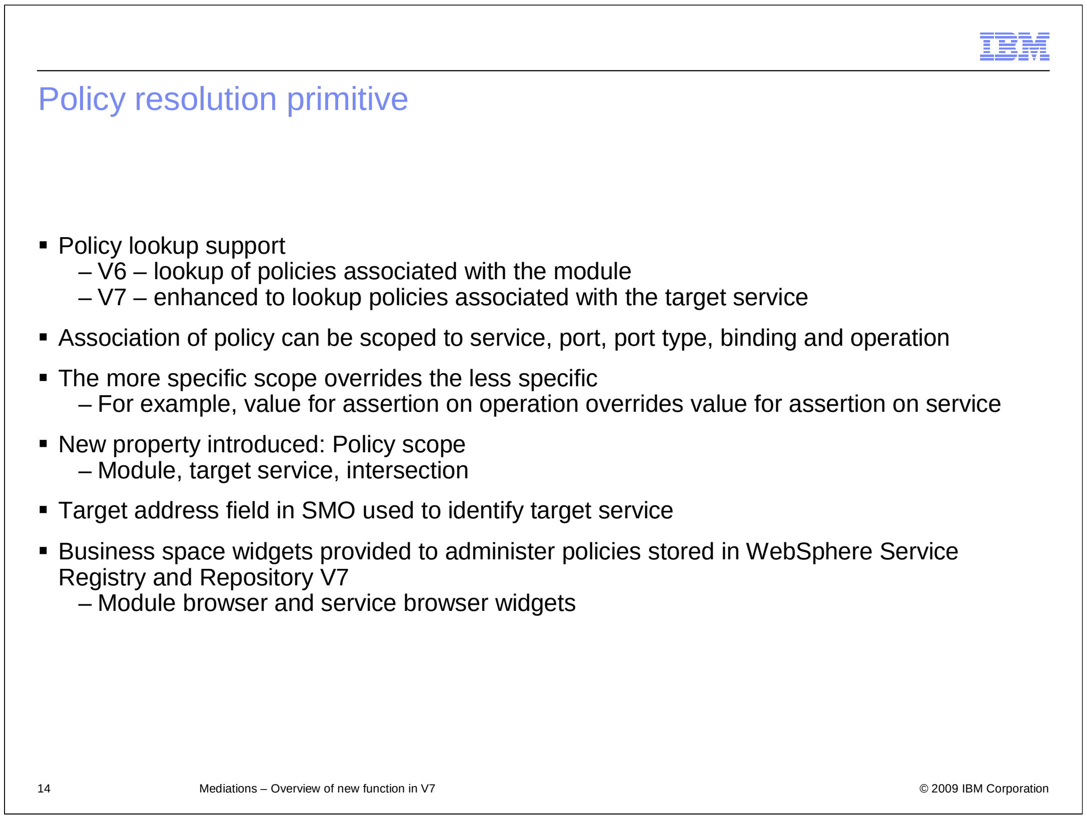
Endpoint lookup primitive

- Bindings support
  - Endpoint types supported in V6
    - Web services using SOAP/HTTP or SOAP/JMS
    - SCA module exports SOAP/HTTP or SOAP/JMS Web service binding
    - SCA module exports with default SCA binding
  - Additional endpoint types added in V7
    - SCA module exports with MQ binding
    - SCA module exports with JMS, MQ JMS or Generic JMS binding
    - SCA module exports with the HTTP binding
    - Manually defined MQ, JMS or HTTP endpoint with associated interface
- New property introduced – Binding type
- Works in conjunction with WebSphere Service Registry and Repository V7

13 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The endpoint lookup primitive has been enhanced to support additional endpoint types. In version six, the primitive only supported Web service endpoints and endpoints defined by SCA exports with Web service bindings or SCA default bindings. In version seven, the primitive now also supports endpoints defined by SCA exports with messaging or HTTP bindings. It also supports manually defined MQ, JMS, and HTTP endpoints. A new binding type property has been introduced which allows you to specify a particular binding type the requested endpoint must support.

The support for these new binding types requires use of the version seven WebSphere Service Registry and Repository.



Policy resolution primitive

- Policy lookup support
  - V6 – lookup of policies associated with the module
  - V7 – enhanced to lookup policies associated with the target service
- Association of policy can be scoped to service, port, port type, binding and operation
- The more specific scope overrides the less specific
  - For example, value for assertion on operation overrides value for assertion on service
- New property introduced: Policy scope
  - Module, target service, intersection
- Target address field in SMO used to identify target service
- Business space widgets provided to administer policies stored in WebSphere Service Registry and Repository V7
  - Module browser and service browser widgets

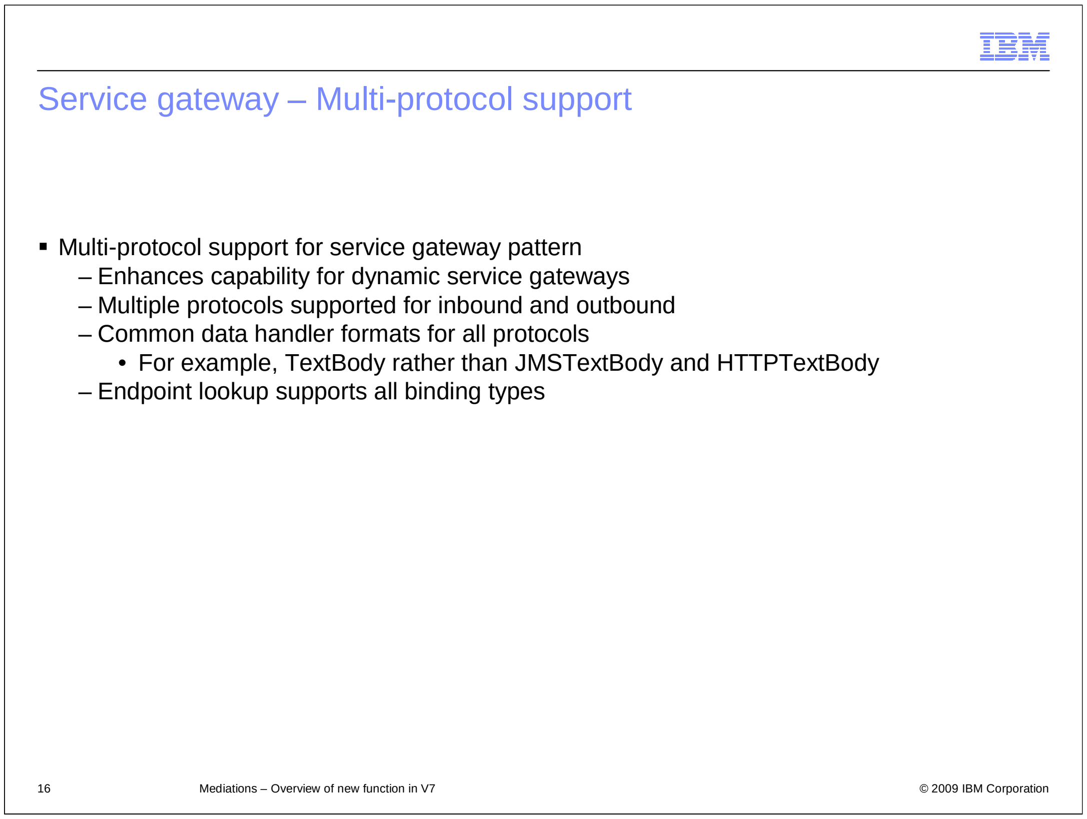
The policy resolution primitive is enhanced in version seven so that policies can now be associated with target services in addition to modules. Policies associated with services can be scoped to the service, port, port type, binding, and operation. When applicable policies exist at more than one scope the policy for the more specific scope overrides the policy for the less specific scope. For example, the policy value of operation overrides the policy value on service. In order to support target service based policies, a new policy scope property has been added to the primitive. The value in the target address field of the SMO header is used to identify the target service for which to obtain applicable policies. Business space widgets, specifically the module browser and service browser widgets, are provided. These allow you to administer the policy definitions in WebSphere Service Registry and Repository. Version seven of the registry is required to support this.

Section

Service gateway and proxy enhancements

15 Mediations – Overview of new function in V7 © 2009 IBM Corporation

This section introduces the enhancements to service gateway and the introduction of the proxy gateway.

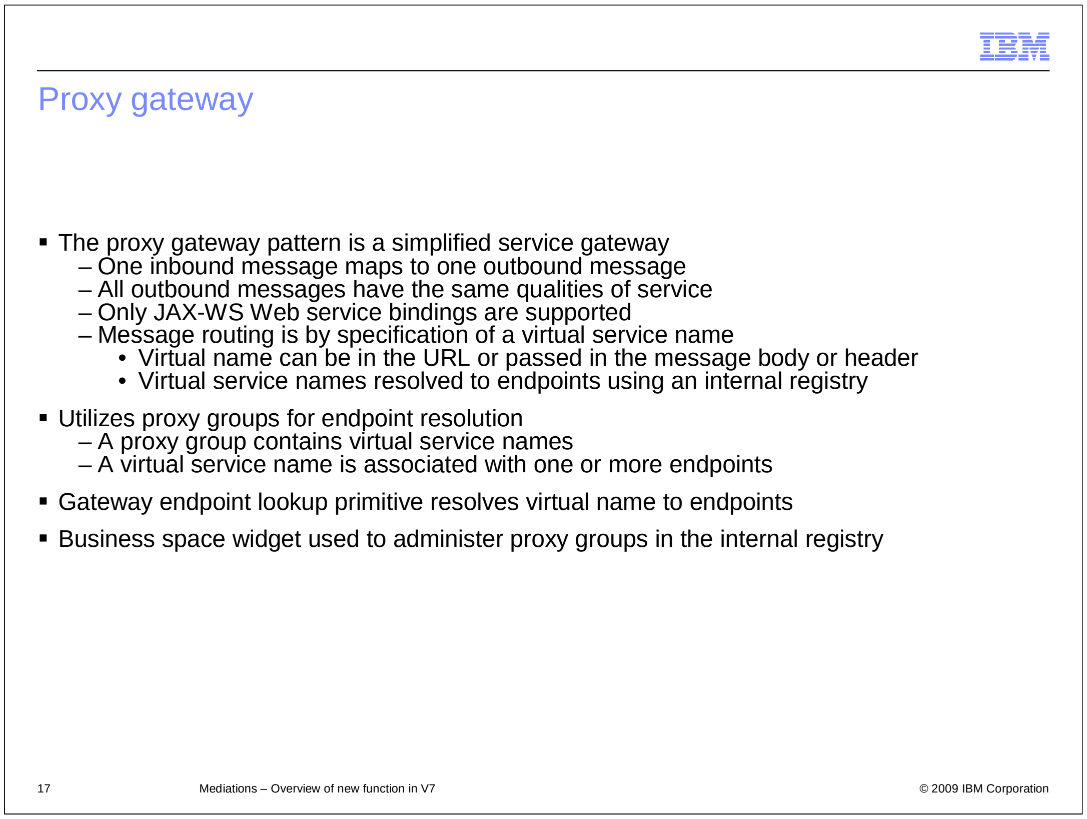


Service gateway – Multi-protocol support

- Multi-protocol support for service gateway pattern
  - Enhances capability for dynamic service gateways
  - Multiple protocols supported for inbound and outbound
  - Common data handler formats for all protocols
    - For example, TextBody rather than JMSTextBody and HTTPTextBody
  - Endpoint lookup supports all binding types

16 Mediations – Overview of new function in V7 © 2009 IBM Corporation

Multi-protocol support has been added to the service gateway pattern, both for inbound and outbound messages. This enhances the capabilities provided by the dynamic service gateway. In order to simplify the flow, common data handler formats that support all protocols are provided. For example, in version six, unique object types are provided for a text message from JMS and a text message from HTTP, whereas in version seven there is a single object type for a text message. Also, the enhancement to the endpoint lookup to support all binding types makes this capability useful for a broader set of scenarios.

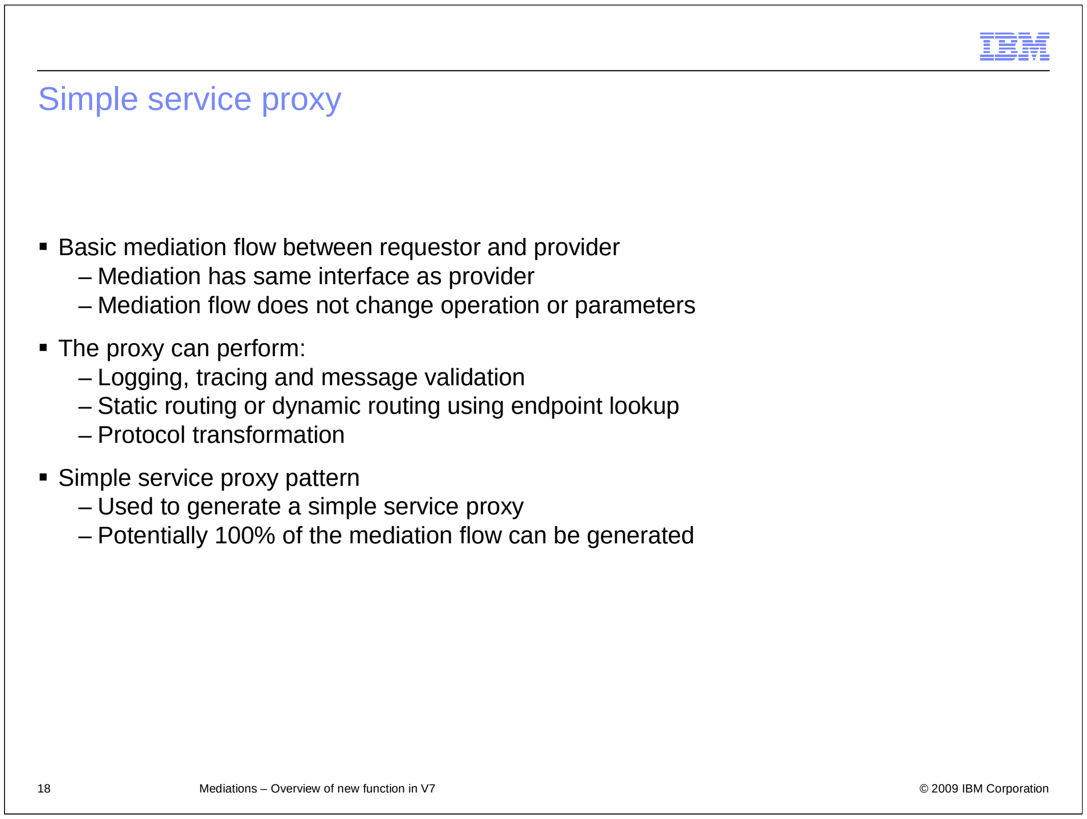


Proxy gateway

- The proxy gateway pattern is a simplified service gateway
  - One inbound message maps to one outbound message
  - All outbound messages have the same qualities of service
  - Only JAX-WS Web service bindings are supported
  - Message routing is by specification of a virtual service name
    - Virtual name can be in the URL or passed in the message body or header
    - Virtual service names resolved to endpoints using an internal registry
- Utilizes proxy groups for endpoint resolution
  - A proxy group contains virtual service names
  - A virtual service name is associated with one or more endpoints
- Gateway endpoint lookup primitive resolves virtual name to endpoints
- Business space widget used to administer proxy groups in the internal registry

17 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The proxy gateway pattern provides a service gateway pattern based on some simplifying assumptions that enables more of the mediation to be generated and provides an administrative capability for managing endpoints. With a proxy gateway, one inbound message maps to one outbound message and all of the outbound messages have the same qualities of service. Only JAX-WS bindings are supported with the proxy gateway. The routing is determined using a virtual service name that is obtained in the request URL or passed in the message headers or body. The virtual service name is resolved to an endpoint using an internal registry. The internal registry is based on proxy groups, where a proxy group contains virtual service names. Each virtual service name is associated with one or more endpoints. The gateway endpoint lookup primitive is configured with proxy group names. It finds the virtual service name in the configured proxy groups and populates the SMO Target and AlternateTarget address fields with the endpoints. There is a business space widget provided to administer the proxy groups in the internal registry.

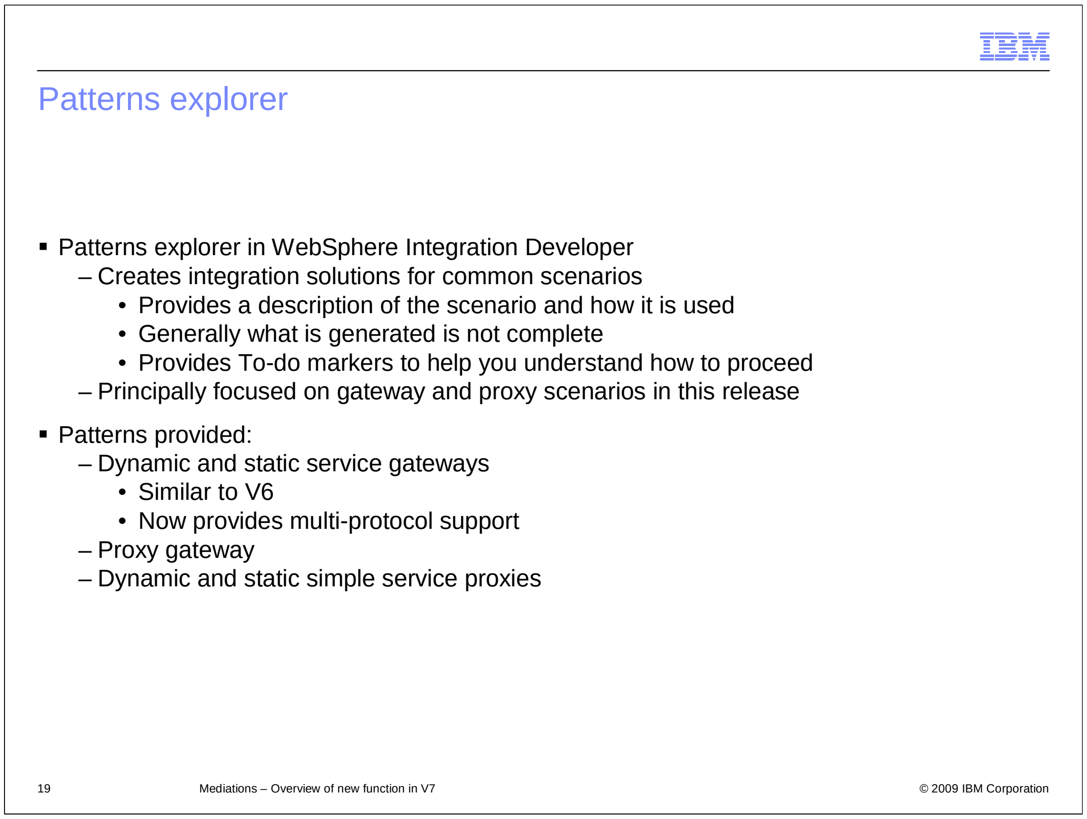


Simple service proxy

- Basic mediation flow between requestor and provider
  - Mediation has same interface as provider
  - Mediation flow does not change operation or parameters
- The proxy can perform:
  - Logging, tracing and message validation
  - Static routing or dynamic routing using endpoint lookup
  - Protocol transformation
- Simple service proxy pattern
  - Used to generate a simple service proxy
  - Potentially 100% of the mediation flow can be generated

18 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The simple service proxy pattern is a basic mediation flow between a service requestor and service provider. The mediation has the same interface as the service provider and the mediation flow does not change any of the operations or parameters. The proxy enables a mediation that can perform logging, tracing and message validation functions. Routing can be static or can be dynamic using and endpoint lookup primitive. Protocol transformation can be done as the inbound and outbound messages do not need to be of the same protocol. There is a simple service proxy pattern provided by WebSphere Integration Developer that generates the mediation. Depending upon your requirements, there is a possibility that the entire mediation can be generated without requiring you to do additional work to complete it.



Patterns explorer

- Patterns explorer in WebSphere Integration Developer
  - Creates integration solutions for common scenarios
    - Provides a description of the scenario and how it is used
    - Generally what is generated is not complete
    - Provides To-do markers to help you understand how to proceed
  - Principally focused on gateway and proxy scenarios in this release
- Patterns provided:
  - Dynamic and static service gateways
    - Similar to V6
    - Now provides multi-protocol support
  - Proxy gateway
  - Dynamic and static simple service proxies

19 Mediations – Overview of new function in V7 © 2009 IBM Corporation

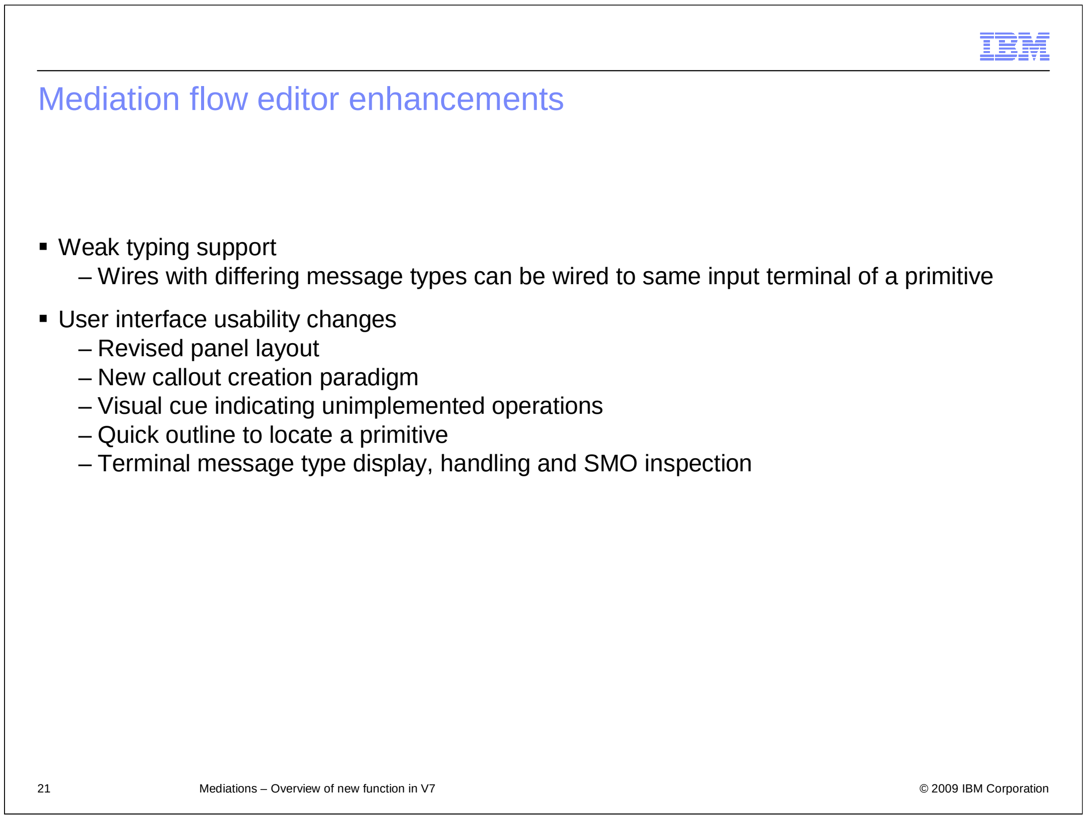
WebSphere Integration Developer has introduced a patterns explorer in version seven. It is used to create integration solutions for common scenarios. A description of each scenario along with information about how it should be used is provided within the explorer. Then a pattern is generated, it generally is not complete and needs to be further developed and refined. To-do markers are placed into the generated mediations to provide hints about where you might need to do further work to complete the flow. In version seven, the patterns explorer is primarily focused on patterns for service gateway and proxy scenarios. The patterns provided include dynamic and static service gateways which are similar to those provided in version six with the addition of multi-protocol support. The new patterns are for the proxy gateway and for the dynamic and static simple service proxies.

Section

Mediation flow editor enhancements

20 Mediations – Overview of new function in V7 © 2009 IBM Corporation

This section provides information about enhancements to the mediation flow editor.



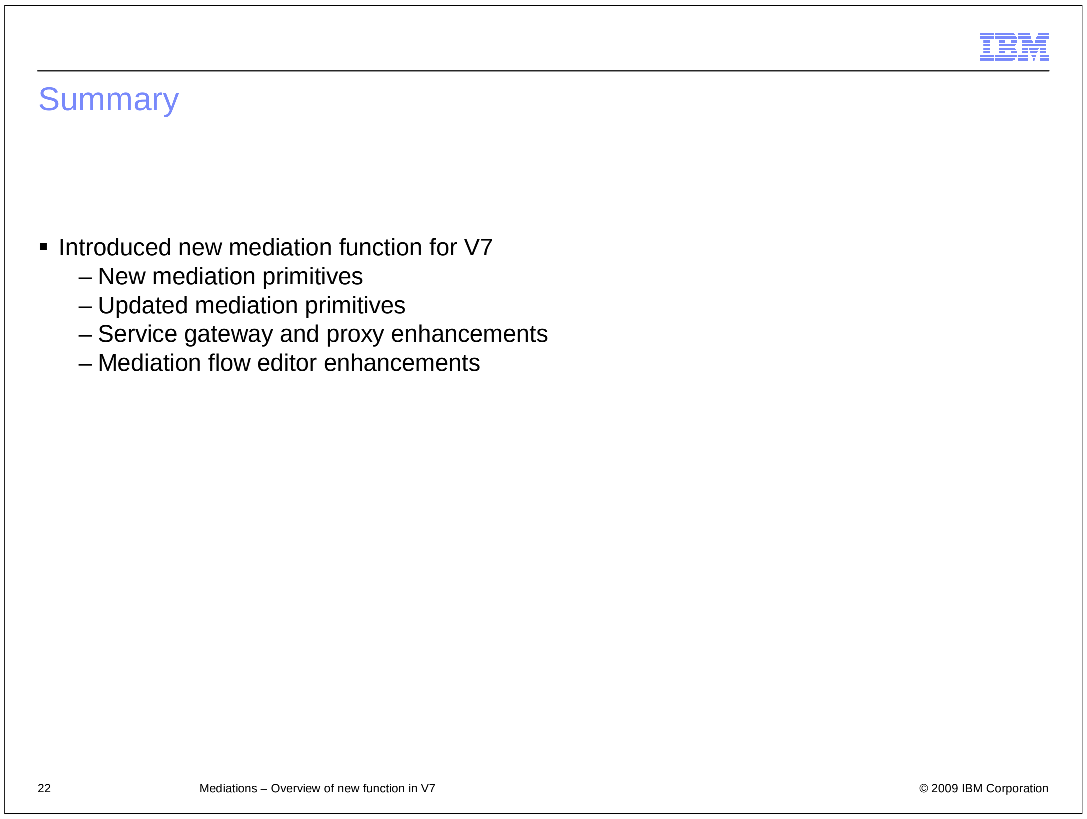
Mediation flow editor enhancements

- Weak typing support
  - Wires with differing message types can be wired to same input terminal of a primitive
- User interface usability changes
  - Revised panel layout
  - New callout creation paradigm
  - Visual cue indicating unimplemented operations
  - Quick outline to locate a primitive
  - Terminal message type display, handling and SMO inspection

21 Mediations – Overview of new function in V7 © 2009 IBM Corporation

The mediation flow editor now supports weak typing. What this means is that wires representing different message types can be wired to the same input terminal of a primitive. This is useful in cases where the primitive does not access or manipulate elements within the body of the message.

There have been several user interface changes that improve the usability of the mediation flow editor. The panel layout has been revised so that the operations connections and mediation flow canvas panels are no longer separate panels, but rather are tabs on a shared panel. The steps taken to add a callout to a mediation flow have changed so that it is done on the mediation flow canvas similar to adding a primitive, rather than in the operations connections panel. The operations connections panel now contains visual cues to indicate if an operation is implemented or not. For large mediation flows, a quick outline has been added to enable you to easily locate a specific primitive in the flow. Finally, there is a new display to show terminal message type that allows you to expand the SMO for inspection and enables you to easily change the message type.

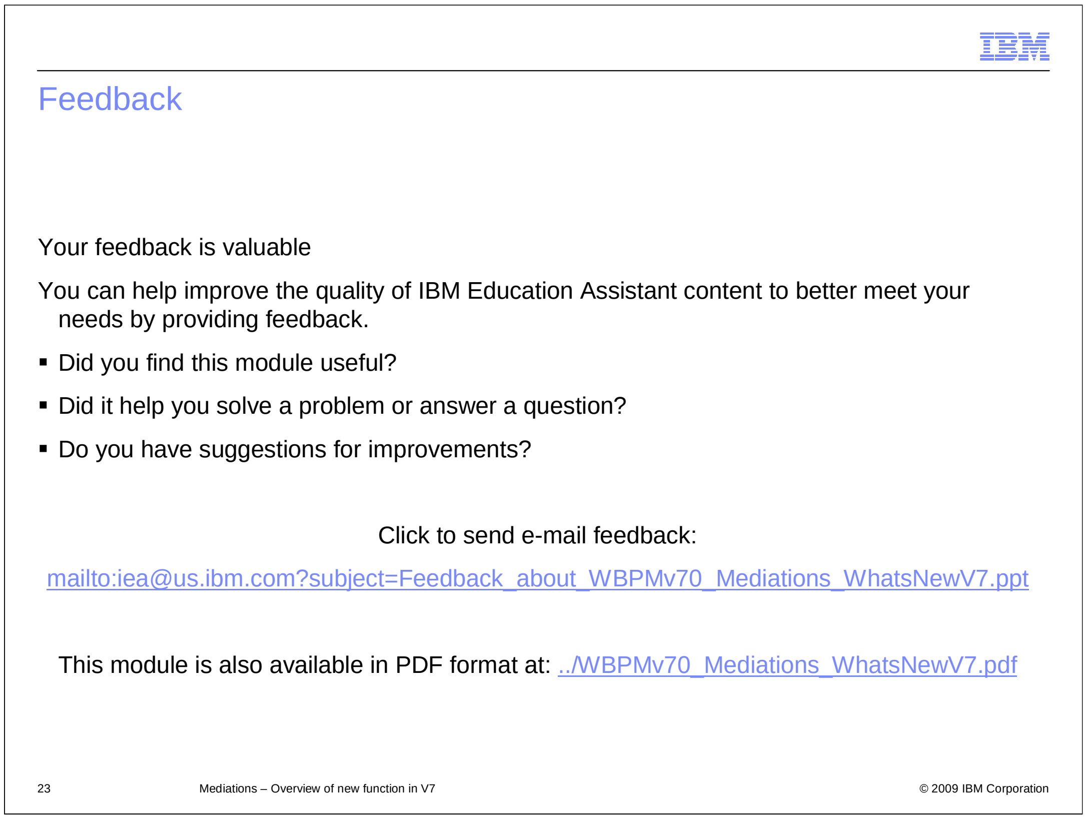


Summary

- Introduced new mediation function for V7
  - New mediation primitives
  - Updated mediation primitives
  - Service gateway and proxy enhancements
  - Mediation flow editor enhancements

22 Mediations – Overview of new function in V7 © 2009 IBM Corporation

In summary, this presentation provided you with a quick introduction to the new features for mediations in version seven. This includes introducing the new mediation primitives and describing the enhancements to existing mediation primitives. The enhancements to service gateway and the addition of proxy gateway were described. Finally, a short list of enhancements to the mediation flow editor was provided.



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?

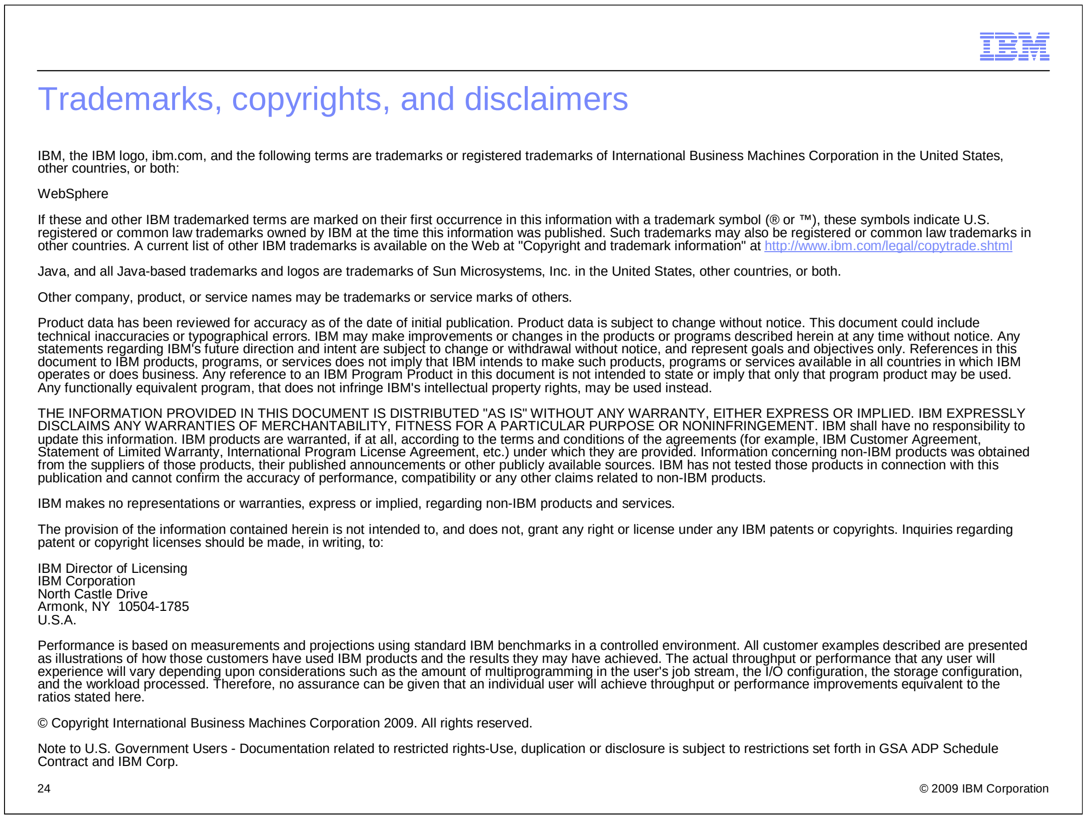
Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WBPMv70_Mediations_WhatsNewV7.ppt

This module is also available in PDF format at: ../WBPMv70_Mediations_WhatsNewV7.pdf

23 Mediations – Overview of new function in V7 © 2009 IBM Corporation

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



Trademarks, copyrights, and disclaimers