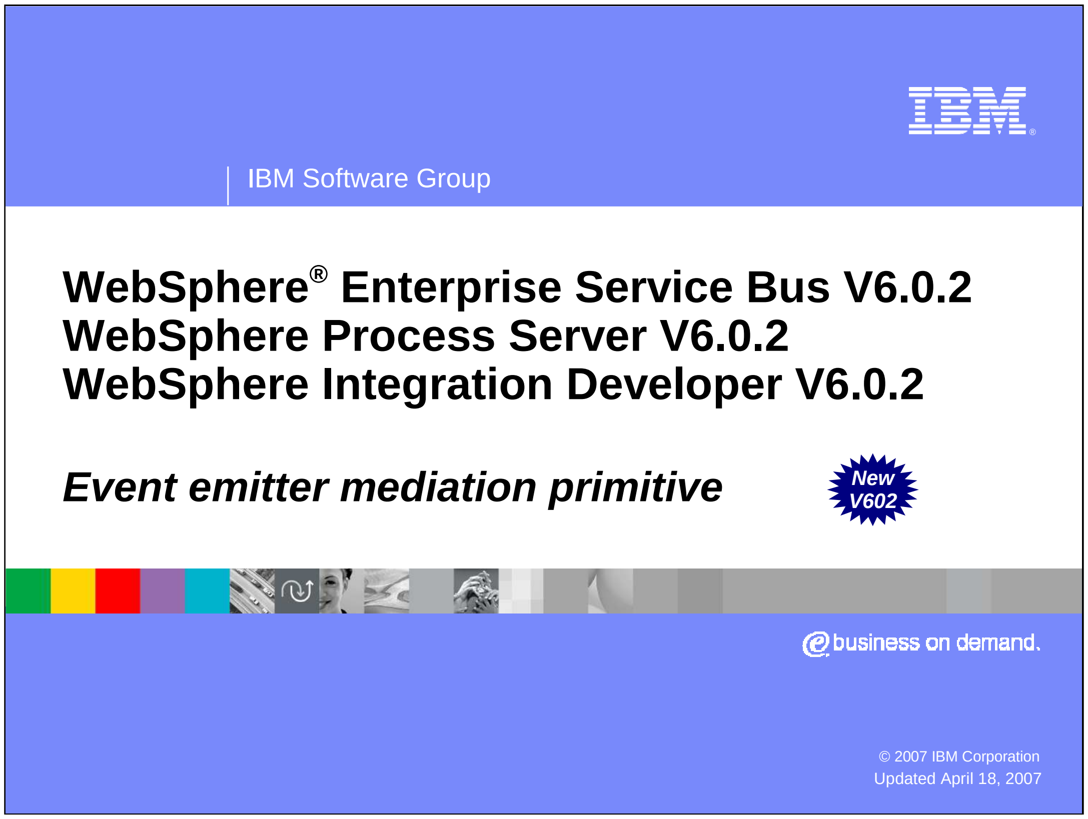


IBM Software Group

WebSphere® Enterprise Service Bus V6.0.2 WebSphere Process Server V6.0.2 WebSphere Integration Developer V6.0.2

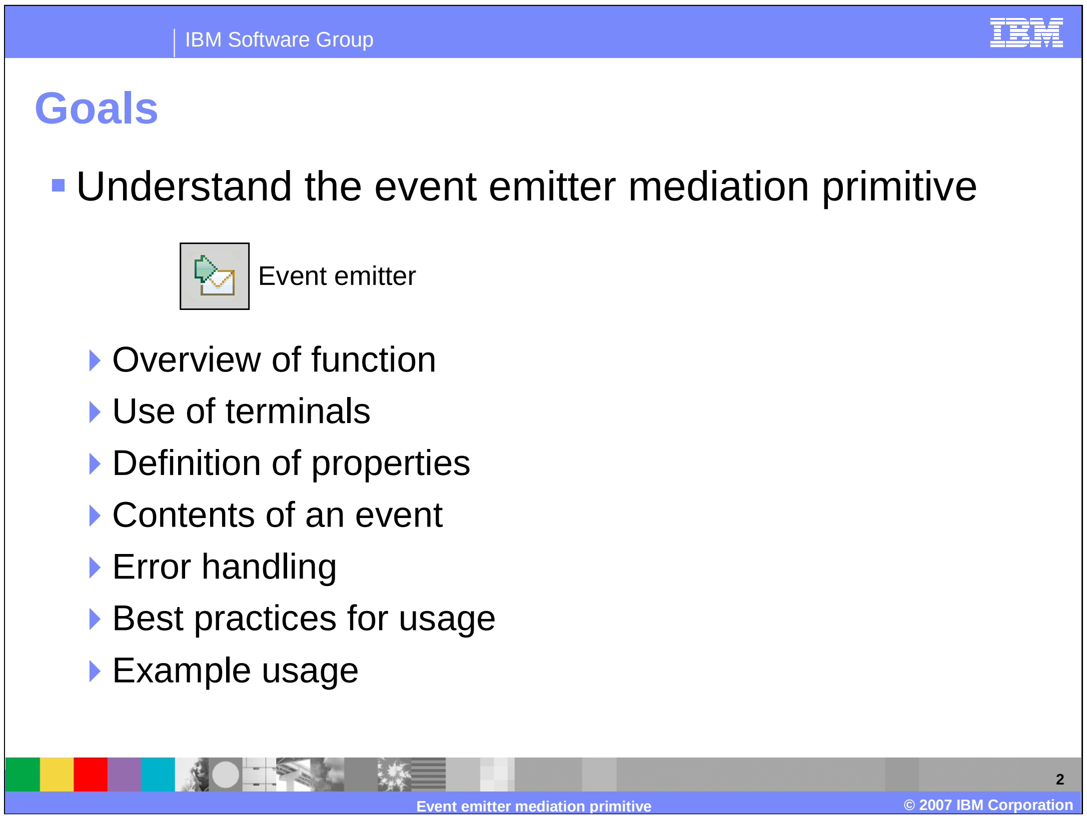
Event emitter mediation primitive

New V602

business on demand.

Updated April 18, 2007

This presentation provides a detailed look at the Event Emitter mediation primitive which was newly introduced in version 6.0.2.



Goals

- Understand the event emitter mediation primitive

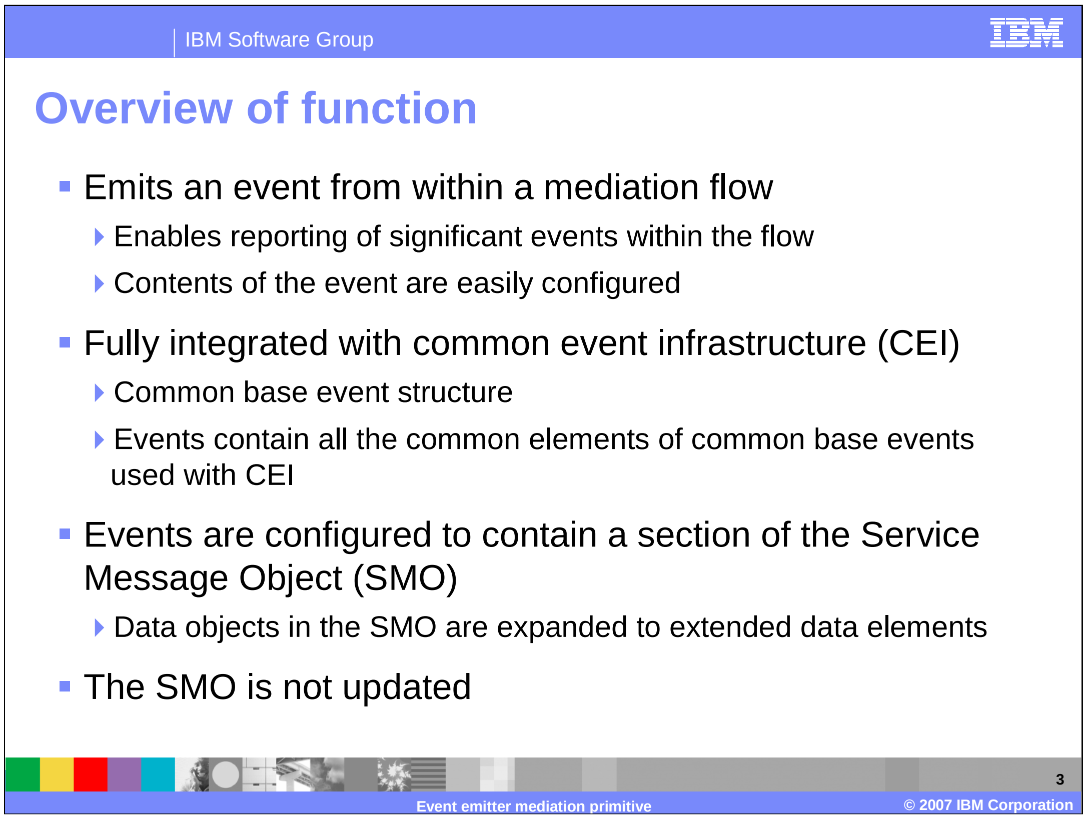
  Event emitter

  - Overview of function
  - Use of terminals
  - Definition of properties
  - Contents of an event
  - Error handling
  - Best practices for usage
  - Example usage

The goal of this presentation is to provide you with a full understanding of the Event Emitter mediation primitive.

The presentation assumes that you are already familiar with the material presented in the **Mediation Primitive Common Details** presentation and the **Common Details – Promoted Properties** presentation. These two presentations serve as a base for understanding mediation primitives in general.

In this presentation, an overview of the Event Emitter mediation primitive is provided, along with information about the primitive's use of terminals and its properties. There is a section which discusses the content of an event and how a formal definition of an event can be exported for use by monitoring applications. The error handling characteristics are then covered along with some information regarding best practices for the use of Event Emitters. Finally, an example usage of a Event Emitter is provided.



Overview of function

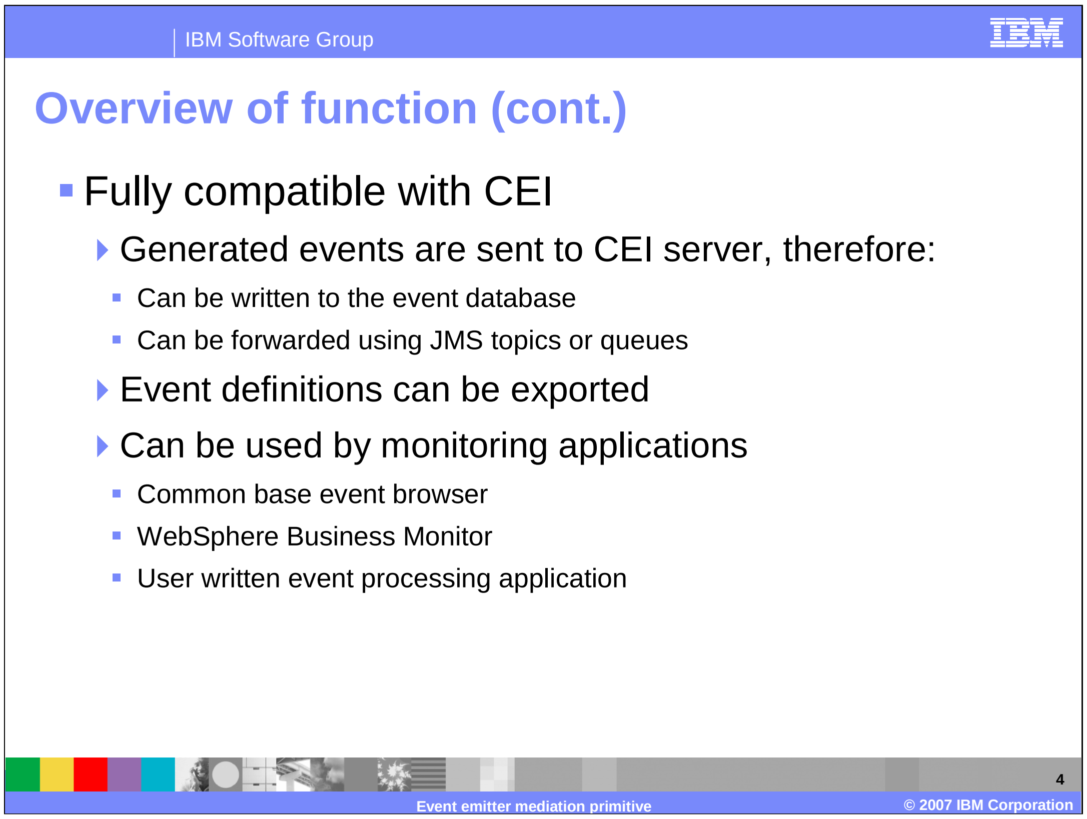
- Emits an event from within a mediation flow
  - Enables reporting of significant events within the flow
  - Contents of the event are easily configured
- Fully integrated with common event infrastructure (CEI)
  - Common base event structure
  - Events contain all the common elements of common base events used with CEI
- Events are configured to contain a section of the Service Message Object (SMO)
  - Data objects in the SMO are expanded to extended data elements
- The SMO is not updated

The Event Emitter primitive provides an simple and easily configured mechanism that can be used to report significant events which have occurred within a mediation flow.

The event that is emitted is fully compatible with the common event infrastructure, which is part of the Service Oriented Architecture core within WebSphere Enterprise Service Bus and WebSphere Process Server. The structure of the event is known as a common base event. Events emitted by this primitive contain all the common base event elements that the common event infrastructure expects it to contain.

When specifying the properties for the Event Emitter, you configure the portion of the Service Message Object (SMO) which you want the event to contain. Data objects within that portion of the SMO are expanded and placed into the common base event as extended data elements. Examples of this are shown later in the presentation.

The Event Emitter primitive does not update the SMO.



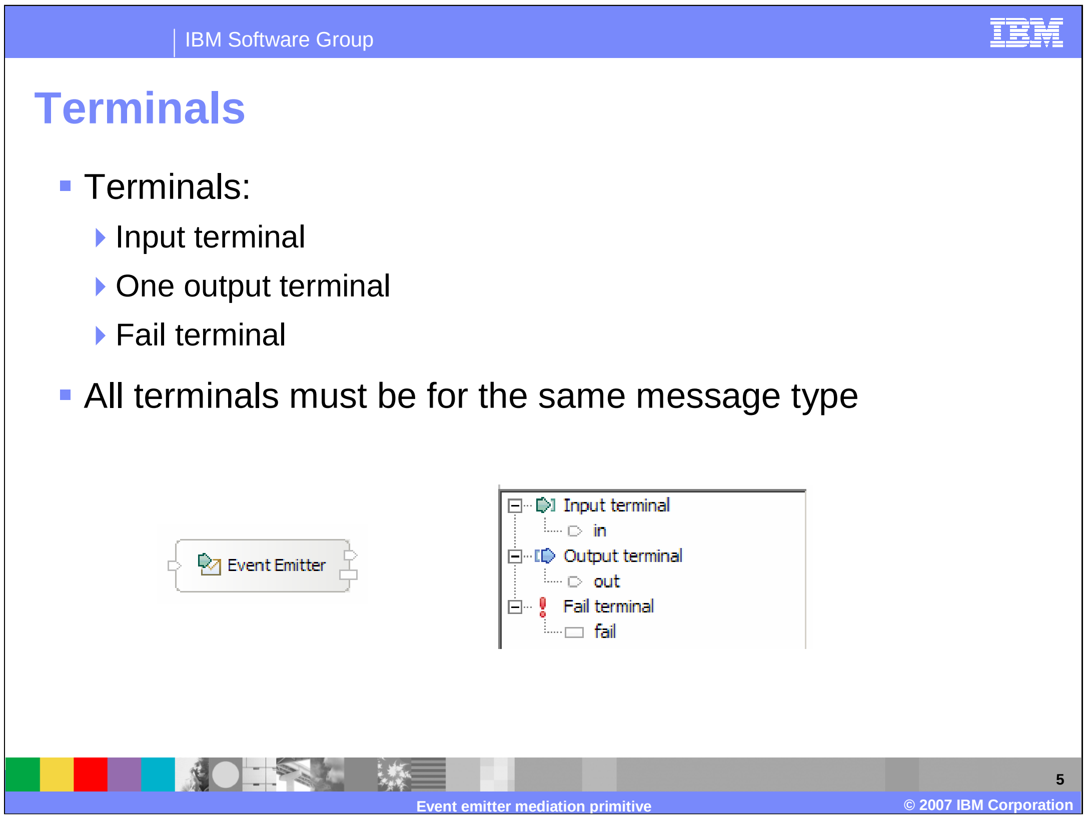
Overview of function (cont.)

- Fully compatible with CEI
  - Generated events are sent to CEI server, therefore:
    - Can be written to the event database
    - Can be forwarded using JMS topics or queues
  - Event definitions can be exported
  - Can be used by monitoring applications
    - Common base event browser
    - WebSphere Business Monitor
    - User written event processing application

As stated on the previous slide, the Event Emitter primitive produces events which are fully compatible with the common event infrastructure. This implies some capabilities which are worth mentioning here.

The events are sent to the CEI server and what happens with an event depends upon how the CEI server is configured. The possibilities are that the event is written to an event database, sent as a message on a JMS queue and published to a JMS topic. These capabilities make the event available to applications which query the database or receive the event through JMS.

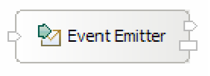
Because applications may be interested in interpreting the contents of the event, the WebSphere Integration Developer provides the capability to export a definition for the event. Applications which browse, interrogate or consume the events include the common base event browser, the WebSphere Business Monitor or any user written event processing application.

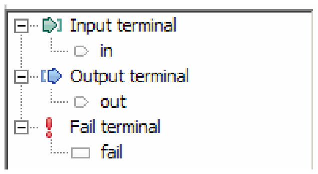


IBM Software Group

Terminals

- Terminals:
  - Input terminal
  - One output terminal
  - Fail terminal
- All terminals must be for the same message type



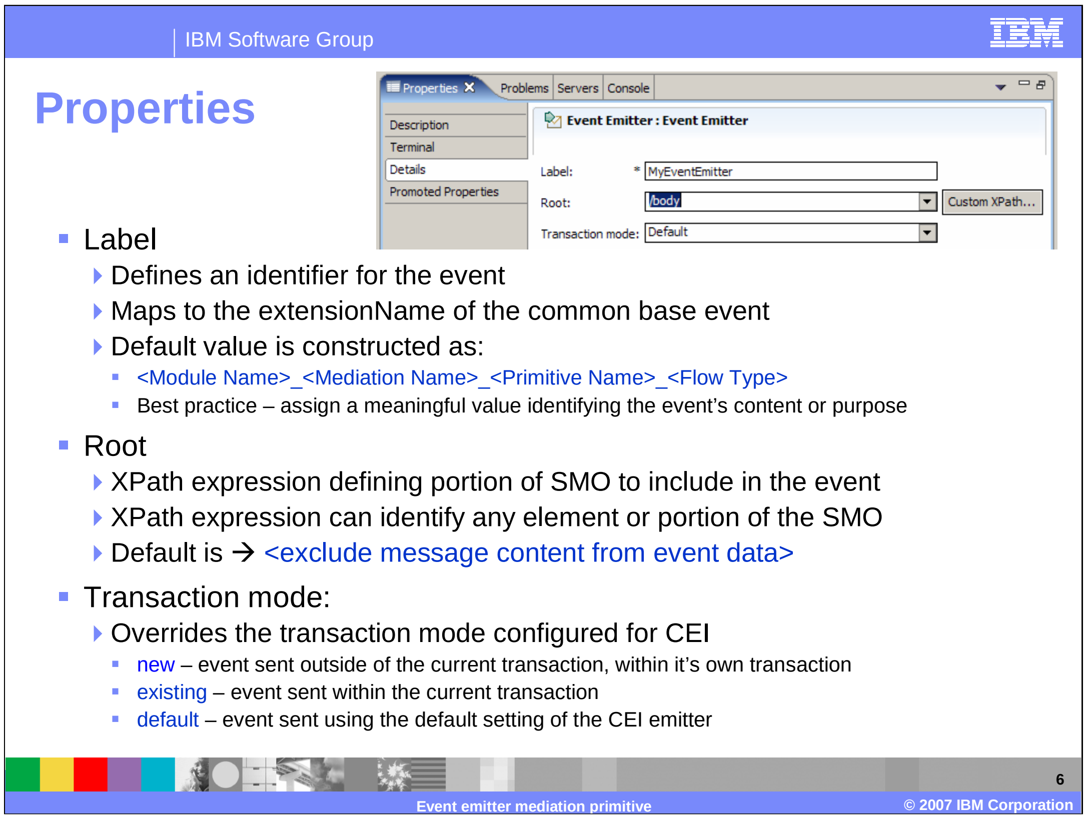


5

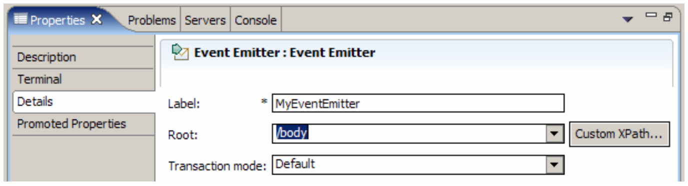
Event emitter mediation primitive

© 2007 IBM Corporation

The Event Emitter primitive has one input terminal, one output terminal and a fail terminal. The output terminal must be for the same message type as the input terminal because the Event Emitter primitive does not modify the message body. Shown here is an Event Emitter primitive with its terminals and the terminals as seen in the properties view.



Properties



Event Emitter : Event Emitter

Label: * MyEventEmitter

Root: /body

Transaction mode: Default

- Label
  - Defines an identifier for the event
  - Maps to the extensionName of the common base event
  - Default value is constructed as:
    - <Module Name>_<Mediation Name>_<Primitive Name>_<Flow Type>
    - Best practice – assign a meaningful value identifying the event's content or purpose
- Root
  - XPath expression defining portion of SMO to include in the event
  - XPath expression can identify any element or portion of the SMO
  - Default is → <exclude message content from event data>
- Transaction mode:
  - Overrides the transaction mode configured for CEI
    - new – event sent outside of the current transaction, within it's own transaction
    - existing – event sent within the current transaction
    - default – event sent using the default setting of the CEI emitter

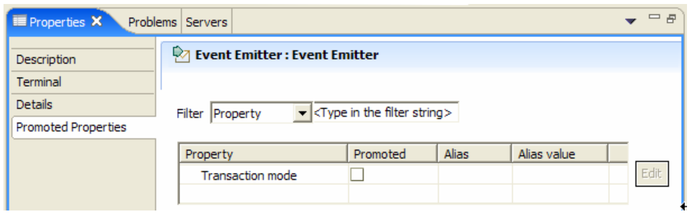
In the upper right portion of the slide is a screen capture showing the Details tab of the Properties view for an Event Emitter. An Event Emitter has the following properties:

The **Label** property provides an identifier for the event. The value for this property is placed into the **extensionName** field, which is a part of the standard common base event format. If you do not assign a specific value to this property, WebSphere Integration Developer provides a default value, as shown in the slide. It is constructed from the Mediation Module name, the Mediation Flow name, the Event Emitter primitive name and the flow type, all separated using underbars. It is likely that this default won't be meaningful to whomever is consuming the events, so it is recommended that you assign some meaningful value to this property.

The **Root** property contains an XPath expression and is used to define what portion of the Service Message Object is included in the event. It can define a leaf element or a data object within the SMO. The **Custom XPath…** button can be used to access the XPath Expression Builder dialog to allow you to easily construct the XPath expression. If you do not specify this property, the default is to exclude all SMO content from the event.

The **Transaction mode** property is used to define transaction handling for the writing of the event. A setting of **new** means that the event is emitted within the context of its own transaction. A setting of **existing** means that the event participates in the current transaction. Finally, a setting of **default** indicates to use whatever setting has been configured for CEI.

Promoted properties



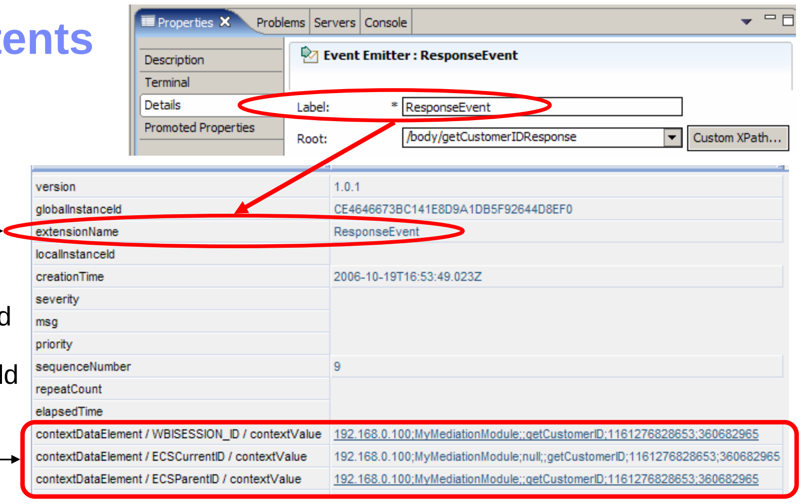
Filter Property <Type in the filter string>

Property	Promoted	Alias	Alias value
Transaction mode	☐		

- Promotable
  - Transaction mode
- Not promotable
  - Label
  - Root

This slide shows the Promoted Properties panel for the Event Emitter primitive. As you can see, only the Transaction mode property is promotable. The Label and Root properties are not promotable. This is because any change to them would cause the event to not match an exported event definition which external event monitoring applications might be relying on.

Event contents



Properties | Problems | Servers | Console

Description / Terminal / Details / Promoted Properties

Event Emitter : ResponseEvent

Label: * ResponseEvent

Root: /body/getCustomerIDResponse [Custom XPath...]

version	1.0.1
globalInstanceId	CE4646673BC141E8D9A1DB5F92644D8EF0
extensionName	ResponseEvent
localInstanceId	
creationTime	2006-10-19T16:53:49.023Z
severity	
msg	
priority	
sequenceNumber	9
repeatCount	
elapsedTime	
contextDataElement / WBISESSION_ID / contextValue	192.168.0.100;MyMediationModule;;getCustomerID;1161276828653;360682965
contextDataElement / ECSCurrentID / contextValue	192.168.0.100;MyMediationModule;null;;getCustomerID;1161276828653;360682965
contextDataElement / ECSParentID / contextValue	192.168.0.100;MyMediationModule;;getCustomerID;1161276828653;360682965

Label property used to set the extensionName field

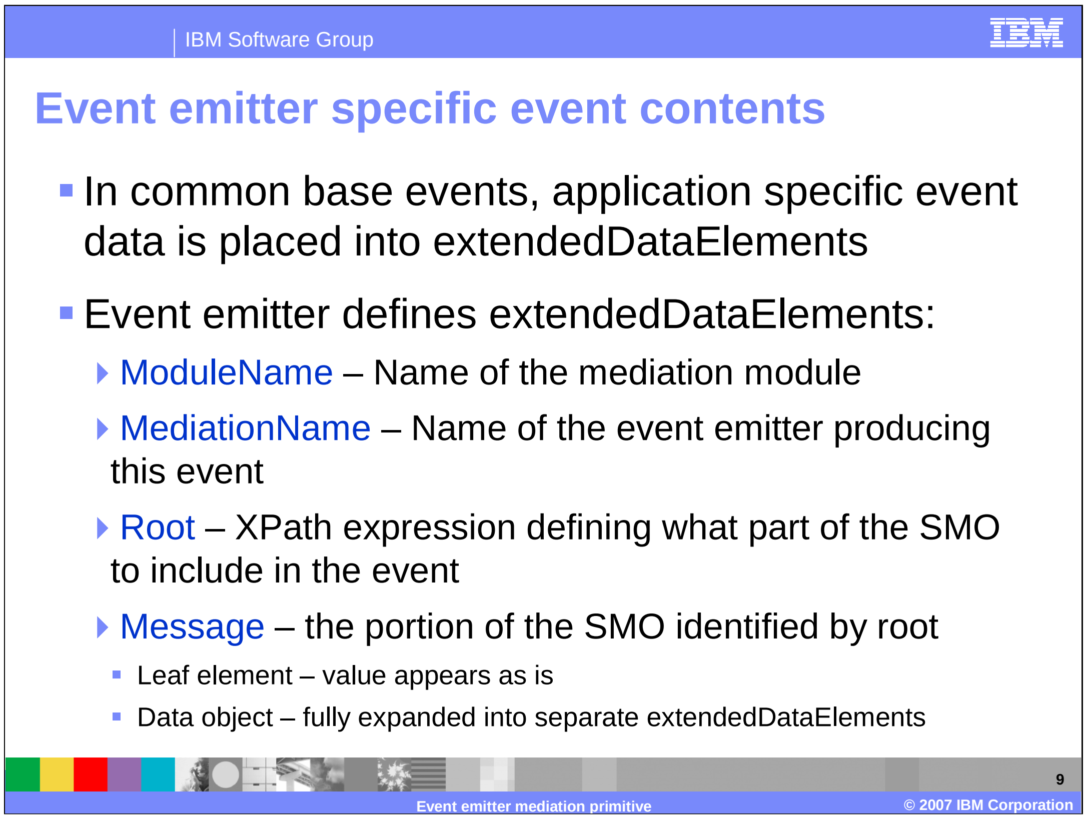
- contextDataElements contain IDs used for correlation
  - Enables correlation with other events from the same flow
    - Predefined events, such as events from SCA components or BPEL processes
    - Application defined events
  - Session ID, Current Correlation ID and Parent Correlation ID

The next several slides take a look at the contents of an event.

The screen capture of the event in this slide was taken from an event displayed in the common base event browser. The common base event browser is an application built into WebSphere Enterprise Service Bus and WebSphere Process Server.

The first thing to notice is the placement of the Label property value into the extensionName field of the event. This is a standard field in any common base event.

The next thing to notice is the contextDataElements. This is an extensible portion of the common base event where context information can be included. For events emitted from WebSphere Enterprise Service Bus and WebSphere Process Server, this context data contains session and correlation IDs. Using these IDs, this event can be associated with other events that occurred as part of the same overall flow. These other events could be predefined events such as are emitted by SCA and BPEL, or they could also be application defined events.


IBM Software Group

Event emitter specific event contents

- In common base events, application specific event data is placed into extendedDataElements
- Event emitter defines extendedDataElements:
  - ModuleName – Name of the mediation module
  - MediationName – Name of the event emitter producing this event
  - Root – XPath expression defining what part of the SMO to include in the event
  - Message – the portion of the SMO identified by root
    - Leaf element – value appears as is
    - Data object – fully expanded into separate extendedDataElements

9

Event emitter mediation primitive © 2007 IBM Corporation

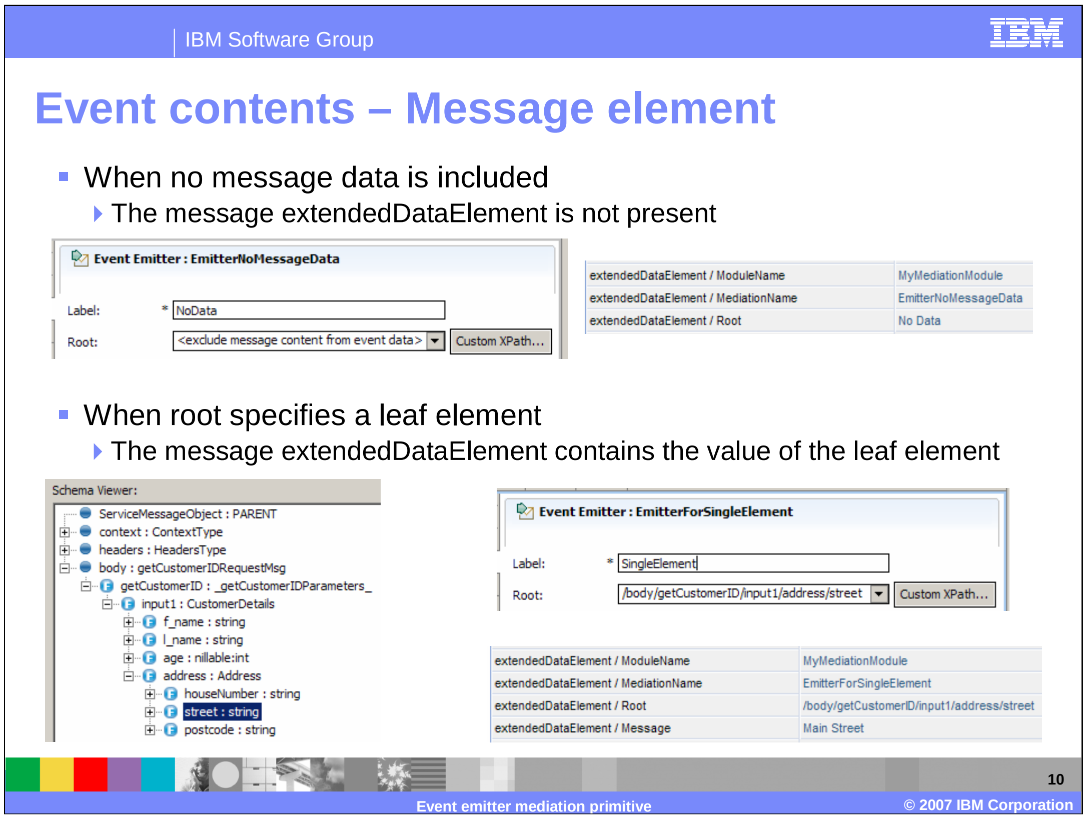
Common base events normally contain some amount of application specific data. Common base events define the extendedDataElement as the location for this application specific data. All events that are produced by the Event Emitter primitive contain a common set of application data as shown in this slide.

The **ModuleName** is the name of the Mediation Module in which the Event Emitter exists.

The **MediationName** is the name of the Event Emitter primitive which generated the event.

The **Root** is the XPath expression which defines what portion of the SMO is included in the event.

The **Message** is that portion of the SMO that is identified by the Root. If the Root identifies a leaf element in the SMO, it is just the value of that element that appears in the event. However, it the Root identifies a data object, the data object is expanded to show all of its individual elements. This is shown in more detail on the next couple of slides.



IBM Software Group

Event contents – Message element

- When no message data is included
  - The message extendedDataElement is not present

Event Emitter : EmitterNoMessageData

Label: * NoData

Root: <exclude message content from event data> Custom XPath...

extendedDataElement / ModuleName	MyMediationModule
extendedDataElement / MediationName	EmitterNoMessageData
extendedDataElement / Root	No Data

- When root specifies a leaf element
  - The message extendedDataElement contains the value of the leaf element

Schema Viewer:

```
ServiceMessageObject : PARENT
context : ContextType
headers : HeadersType
body : getCustomerIDRequestMsg
  getCustomerID : _getCustomerIDParameters_
    input1 : CustomerDetails
      f_name : string
      l_name : string
      age : nillable:int
      address : Address
        houseNumber : string
        street : string
        postcode : string
```

Event Emitter : EmitterForSingleElement

Label: * SingleElement

Root: /body/getCustomerID/input1/address/street Custom XPath...

extendedDataElement / ModuleName	MyMediationModule
extendedDataElement / MediationName	EmitterForSingleElement
extendedDataElement / Root	/body/getCustomerID/input1/address/street
extendedDataElement / Message	Main Street

10

Event emitter mediation primitive © 2007 IBM Corporation

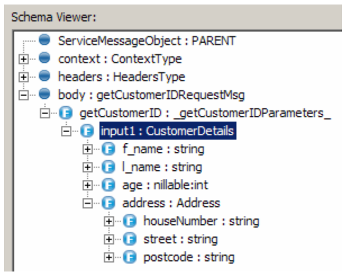
This slide looks at the Message element placed into the event.

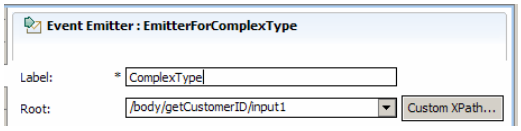
The top portion of this slide illustrates what appears in an event when the Root property specifies the default value, indicating that no message content is placed into the event. The screen capture in the upper left shows the contents of the Root property. The screen capture on the upper right shows the content of the event. Notice that all the fields are qualified by extendedDataElement, which is the application data portion of the event. The ModuleName, MediationName and Root are all present, but the Message is not present.

In the bottom portion of the slide is an illustration of what the event looks like when the Root property specifies a leaf element in the SMO. On the lower left is a screen capture from the Schema Viewer panel of the XPath Expression Builder, showing that the leaf element named street is selected. On the right you can see that the Root property contains the XPath expression identifying the element street. In the event, the Message is present. It contains the value of the element street taken from the SMO, but does not identify the element by its name, street.

Event contents – Message element (cont.)

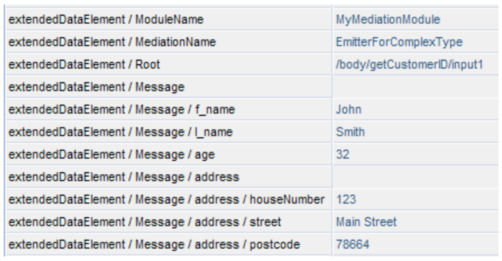
- When root specifies a data object (complex type)
  - Multiple message extendedDataElements showing values for all contained elements




Event Emitter : EmitterForComplexType

Label: * ComplexType

Root: /body/getCustomerID/input1



extendedDataElement / ModuleName	MyMediationModule
extendedDataElement / MediationName	EmitterForComplexType
extendedDataElement / Root	/body/getCustomerID/input1
extendedDataElement / Message	
extendedDataElement / Message / f_name	John
extendedDataElement / Message / l_name	Smith
extendedDataElement / Message / age	32
extendedDataElement / Message / address	
extendedDataElement / Message / address / houseNumber	123
extendedDataElement / Message / address / street	Main Street
extendedDataElement / Message / address / postcode	78664

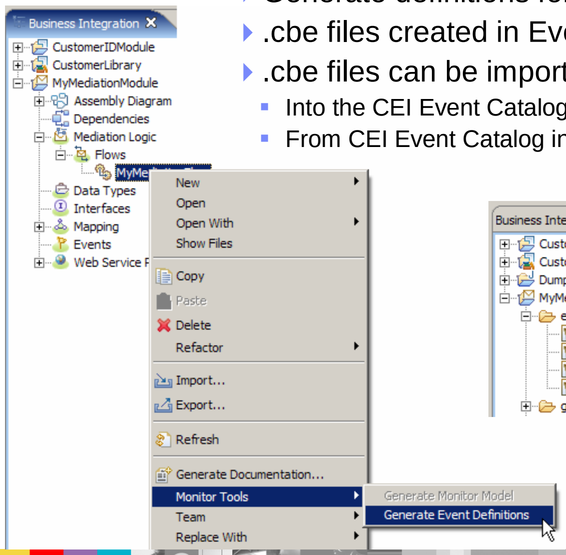
This slide continues the look at the Message element portion of the event. In this case, the portion of the SMO identified by the Root property is a data object.

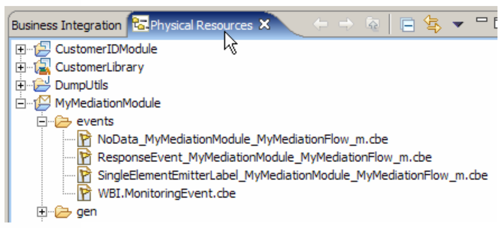
Looking on the left you can see that the input1 data object has been selected in the Schema Viewer of the XPath Expression Builder dialog. This is also reflected in the Root property shown on the right. Looking at the event itself, there are several extendedDataElements for Message. The first represents input1, but the name itself does not show. It is composed of leaf elements f_name, l_name, age, and the data object address, which is composed of the leaf elements houseNumber. street and postcode. All of these are identified by name along with the leaf element values.

IBM Software Group

Export event definitions

- Event Emitter event definitions can be exported
  - Generate definitions for all event emitters in a flow
  - .cbe files created in Events folder of Mediation Module
  - .cbe files can be imported for monitoring
    - Into the CEI Event Catalog
    - From CEI Event Catalog into the WebSphere Business Monitor

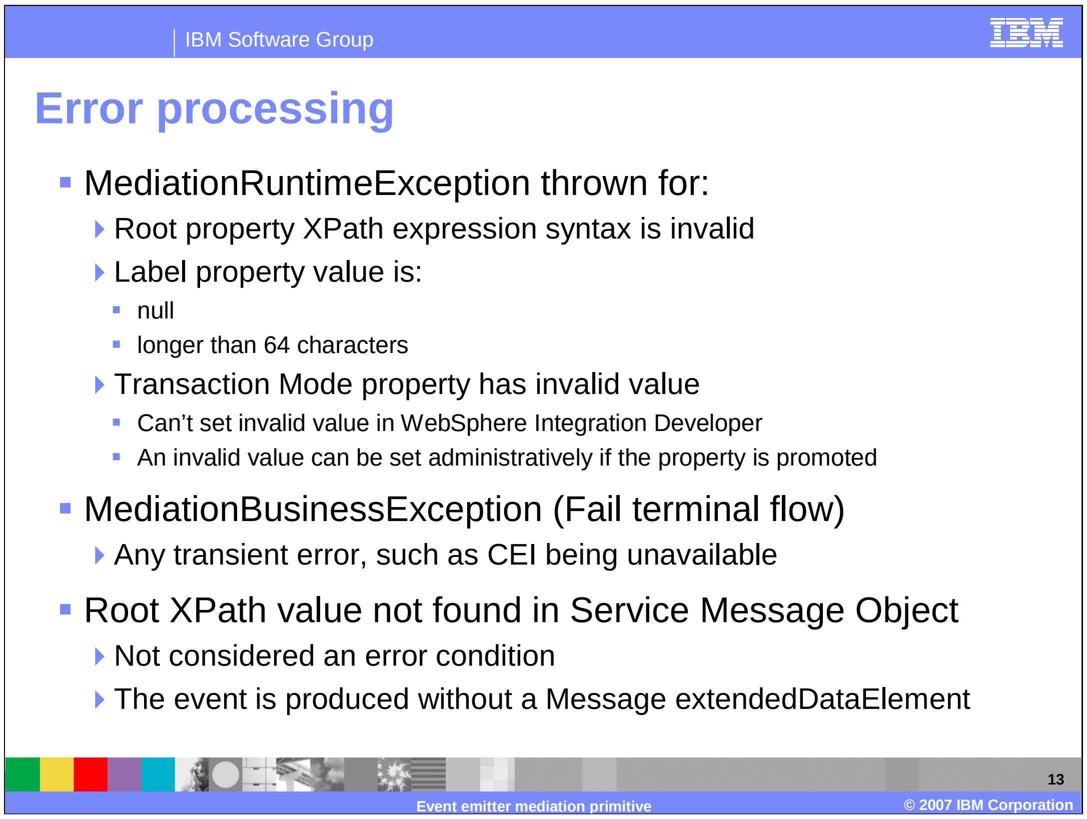



Business Integration
Physical Resources
CustomerIDModule
CustomerLibrary
DumpUtils
MyMediationModule
events
NoData_MyMediationModule_MyMediationFlow_m.cbe
ResponseEvent_MyMediationModule_MyMediationFlow_m.cbe
SingleElementEmitterLabel_MyMediationModule_MyMediationFlow_m.cbe
WBI.MonitoringEvent.cbe
gen

12

Event emitter mediation primitive © 2007 IBM Corporation

Event monitoring applications need to understand the contents on the events they are processing. WebSphere Integration Developer allows you to export event definitions that can be used by the applications. The screen capture on the left shows that using the pop-up menu for a mediation flow, you can select the Monitor Tools menu item and then the Generate Event Definitions menu item. This causes an Events folder to be created in your mediation module. It is populated with .cbe files representing the events defined by the Event Emitter contained in the module, as shown on the right. These .cbe files can then be imported into the CEI Event Catalog. If you are using the WebSphere Business Monitor, it can import the event definitions from the CEI Event Catalog.



Error processing

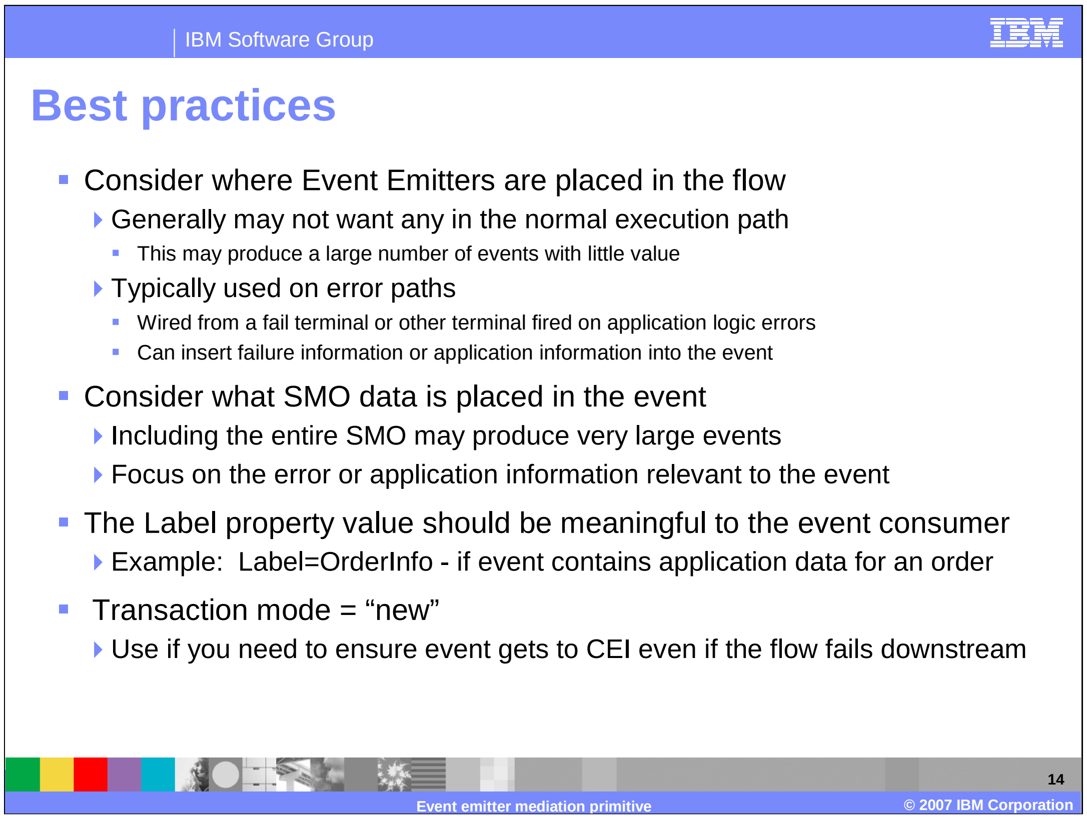
- MediationRuntimeException thrown for:
  - Root property XPath expression syntax is invalid
  - Label property value is:
    - null
    - longer than 64 characters
  - Transaction Mode property has invalid value
    - Can't set invalid value in WebSphere Integration Developer
    - An invalid value can be set administratively if the property is promoted
- MediationBusinessException (Fail terminal flow)
  - Any transient error, such as CEI being unavailable
- Root XPath value not found in Service Message Object
  - Not considered an error condition
  - The event is produced without a Message extendedDataElement

The error processing details and considerations are examined in this slide.

A MediationRuntimeException is thrown for a number of reasons. One case is where the Root property contains an XPath expression with an invalid syntax. Another cause is when the Label property is null or if it is longer than 64 characters. The MediationRuntimeException also occurs when there is an invalid value in the Transaction mode property. This would only occur in the event that the Transaction mode property were promoted, and the invalid value was set administratively. The WebSphere Integration Developer prevents you from setting an invalid value at development time.

A MediationBusinessException occurs for any transient problems, such as CEI not being available. These exceptions cause the Fail terminal flow to be taken.

It is possible for the Root property to have a syntactically correct XPath express which does not identify an element within the SMO. This could occur do to a mistake in specifying the property or from an optional element not being present in the SMO. This is considered a normal situation by the Event Emitter primitive, and it produces an event without the Message extendedDataElement being present.



Best practices

- Consider where Event Emitters are placed in the flow
  - Generally may not want any in the normal execution path
    - This may produce a large number of events with little value
  - Typically used on error paths
    - Wired from a fail terminal or other terminal fired on application logic errors
    - Can insert failure information or application information into the event
- Consider what SMO data is placed in the event
  - Including the entire SMO may produce very large events
  - Focus on the error or application information relevant to the event
- The Label property value should be meaningful to the event consumer
  - Example: Label=OrderInfo - if event contains application data for an order
- Transaction mode = “new”
  - Use if you need to ensure event gets to CEI even if the flow fails downstream

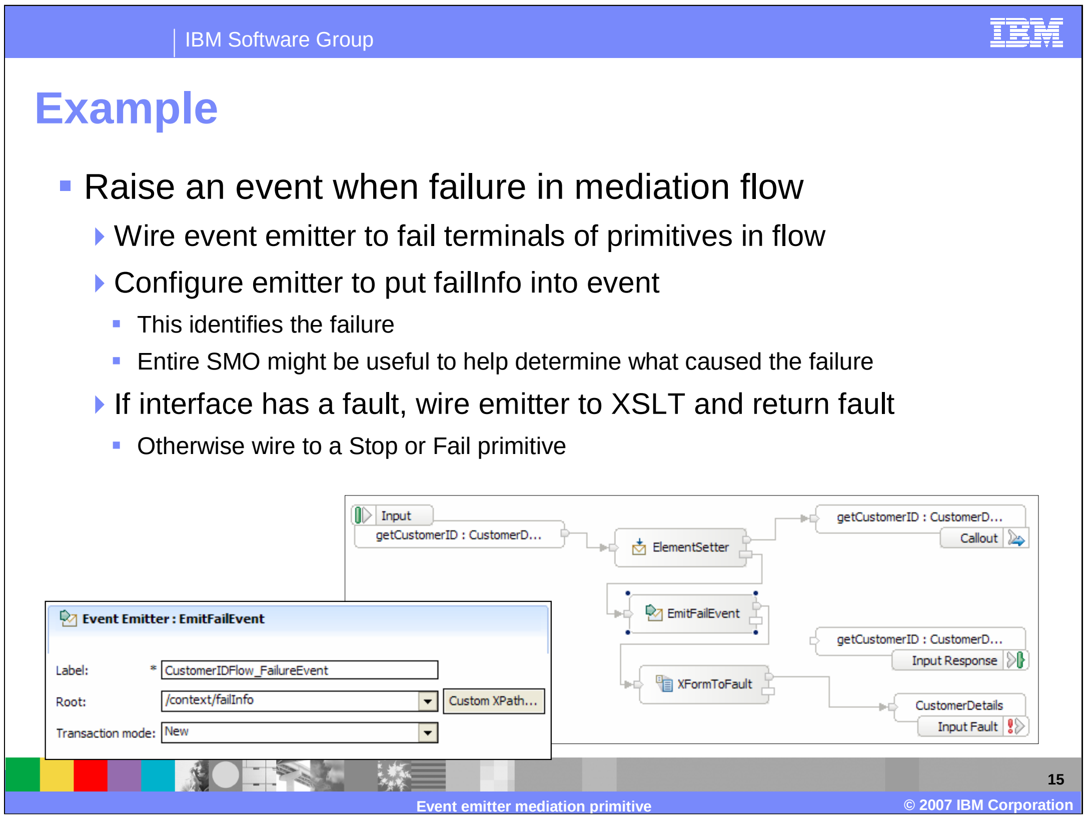
When designing your flow, there are some best practices which you can consider regarding the use of Event Emitter primitives.

The first of these is to be judicious about where in the flow you place them. Generally speaking, unless you have a specific application requirement, you would not want to place an Event Emitter on the normal path of a flow. This has the potential to raise a large number of events with potentially little value. A typical use would be to put an Event Emitter on an error path, such as wired to a Fail terminal or some other output terminal that represents an application error. Such an event might contain the failure information from the SMO header or some application specific data from the body of the SMO.

Placing the entire SMO into an event may be appropriate in some instances. However, a smaller event is generated if you can be selective about the specific data you need to see for the situation you are reporting with the event.

Give the Label property a value which has meaning to the consumer of the event. This might be something like OrderInfo for an event that contains application data for an order.

Finally, using a Transaction mode setting of new ensures the event gets emitted to CEI even if there is a failure in your flow downstream from the Event Emitter.



Example

- Raise an event when failure in mediation flow
  - Wire event emitter to fail terminals of primitives in flow
  - Configure emitter to put failInfo into event
    - This identifies the failure
    - Entire SMO might be useful to help determine what caused the failure
  - If interface has a fault, wire emitter to XSLT and return fault
    - Otherwise wire to a Stop or Fail primitive

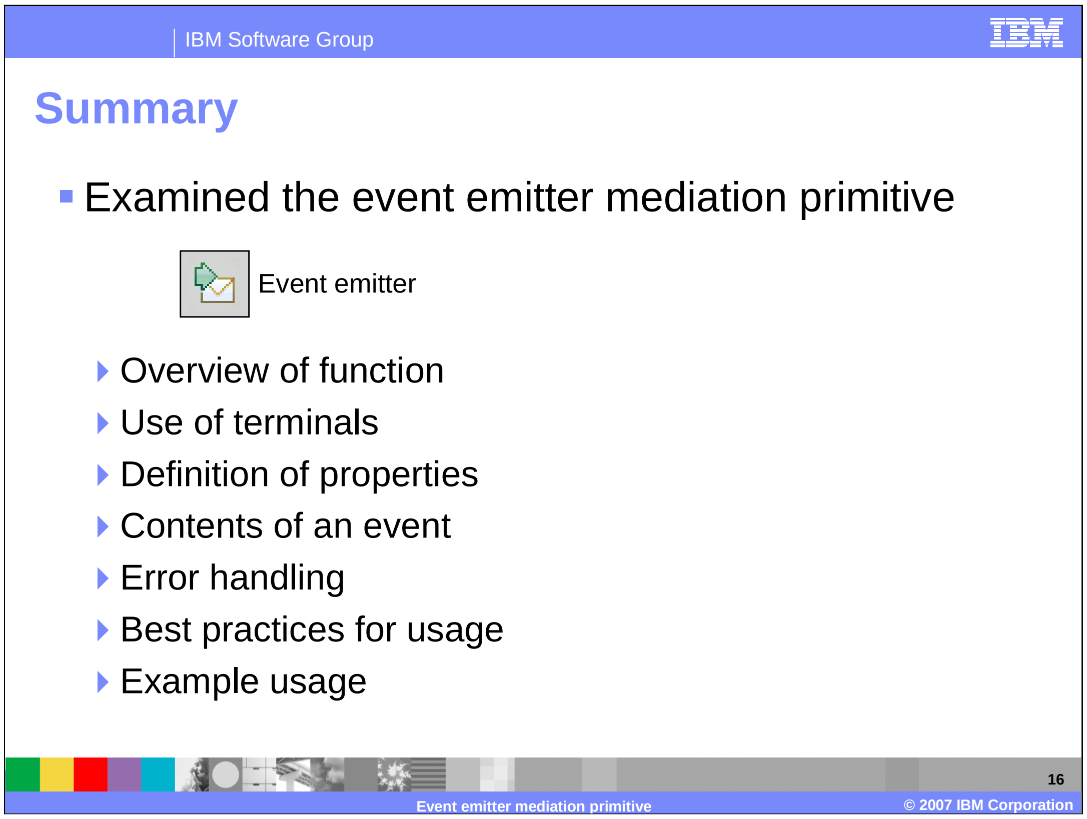
Event Emitter : EmitFailEvent

Label:	* CustomerIDFlow_FailureEvent
Root:	/context/failInfo
Transaction mode:	New

In this example, an Event Emitter is being used to report the occurrence of a failure in the mediation flow. The flow contains a Message Element Setter primitive which is the only primitive on a normal flow with no failures.

The Event Emitter is wired to the Fail terminal of the Message Element Setter. You can see in the properties, the Root specifies that the failInfo from the context of the SMO is to be inserted into the event, thus providing detail of why the failure occurred. If it was felt that the application data would be helpful in this case, the entire SMO could be placed into the event so that the failInfo and the SMO body would both be available.

Notice that in this particular flow, the Event Emitter is wired to an XSL Transformation primitive, which transforms the input message into a fault message that can be returned using the Input Fault node. An alternative would be to wire the Event Emitter to a Stop or Fail primitive.



IBM Software Group

Summary

- Examined the event emitter mediation primitive

  Event emitter

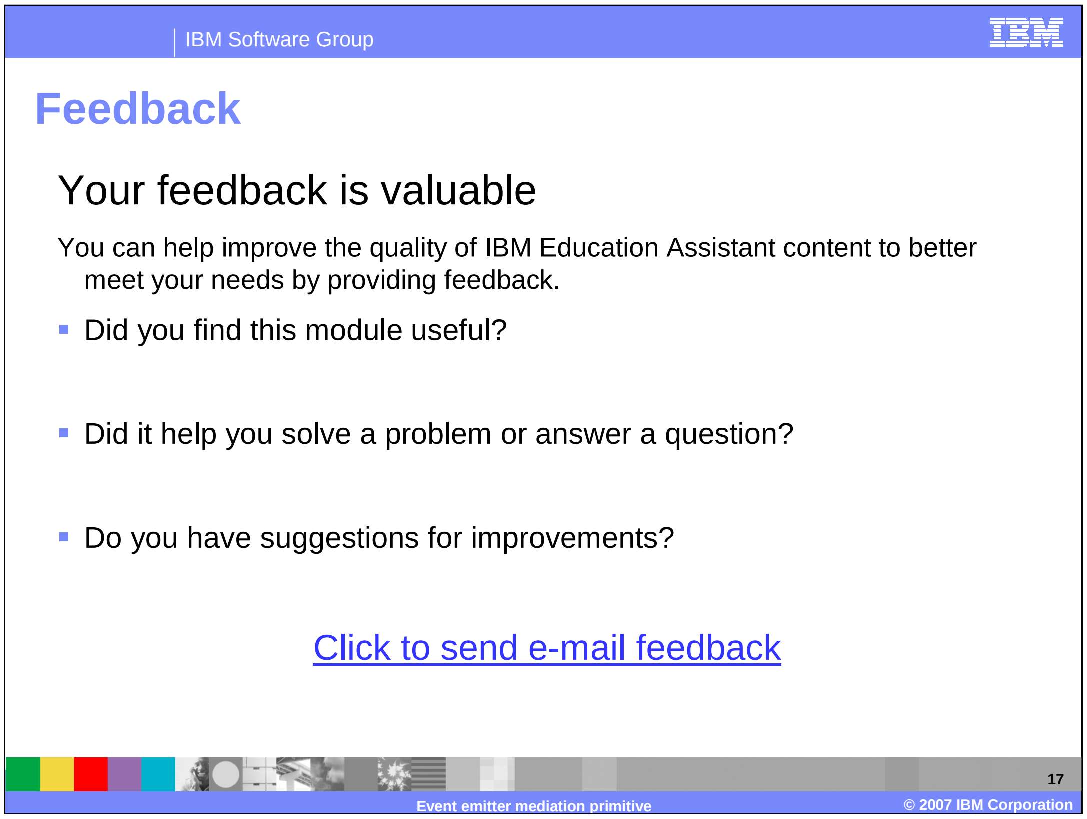
  - Overview of function
  - Use of terminals
  - Definition of properties
  - Contents of an event
  - Error handling
  - Best practices for usage
  - Example usage

16

Event emitter mediation primitive

© 2007 IBM Corporation

In summary, this presentation presented an overview of the Event Emitter mediation primitive, along with information about the primitive's use of terminals and its properties. There was a section which discussed the content of an event and how a formal definition of an event can be exported for use by monitoring applications. The error handling characteristics were then covered along with some information regarding best practices for the use of Event Emitters. The presentation concluded with an example usage of a Event Emitter primitive.



Feedback

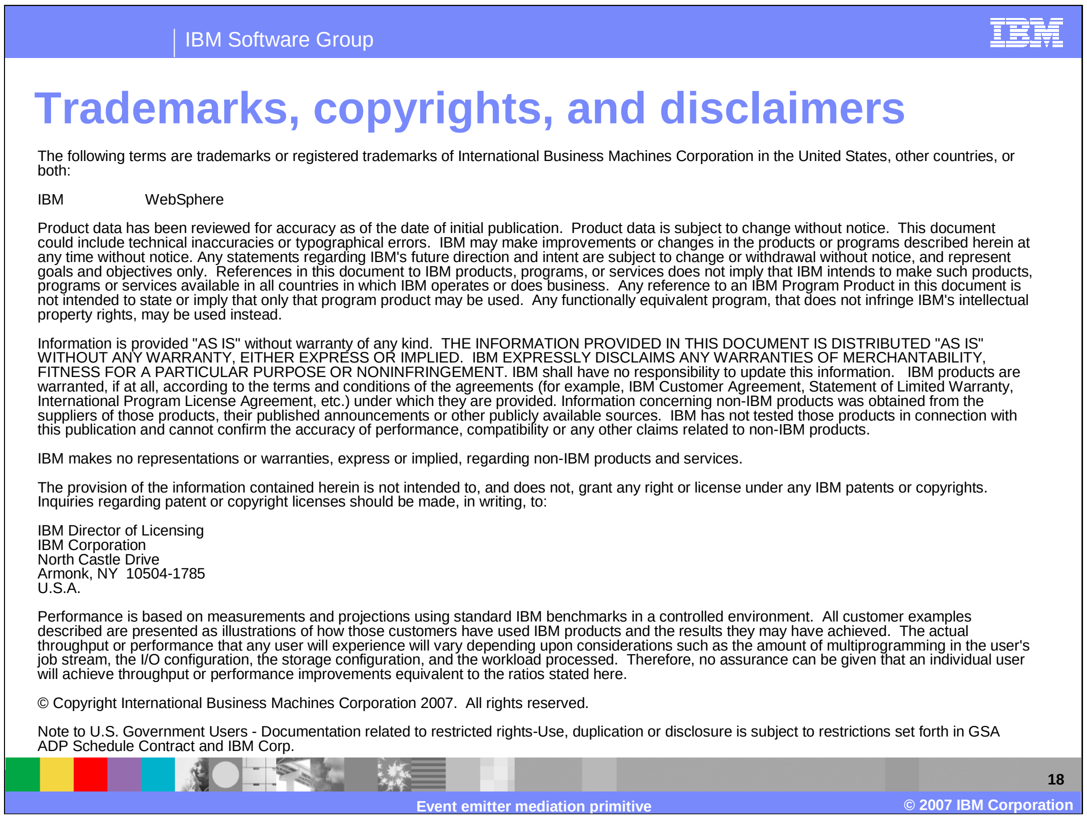
Your feedback is valuable

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

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



Trademarks, copyrights, and disclaimers