Message Broker Systems – Monitoring & Auditing

Dave Gorman (dave_gorman@uk.ibm.com)
IBM

2nd August 2010

Agenda

- What is business application monitoring
- History of monitoring support in WMB
- Configuring using the toolkit
- Configuring using the command line

What is Business Activity Monitoring?

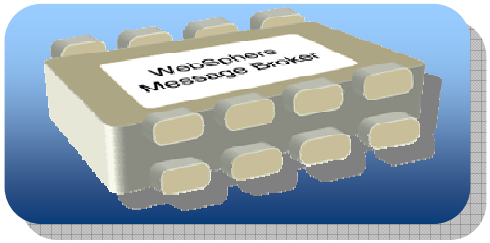
- Wikipedia:
  - “the aggregation, analysis, and presentation of real time information about activities inside organizations and involving customers and partners “
  - BAM is an enterprise solution primarily intended to provide a real-time summary of business activities to operations managers and upper management.

- A way of making the business more transparent
  - Allows evidence-based decision making
  - “X-Ray for business processes”

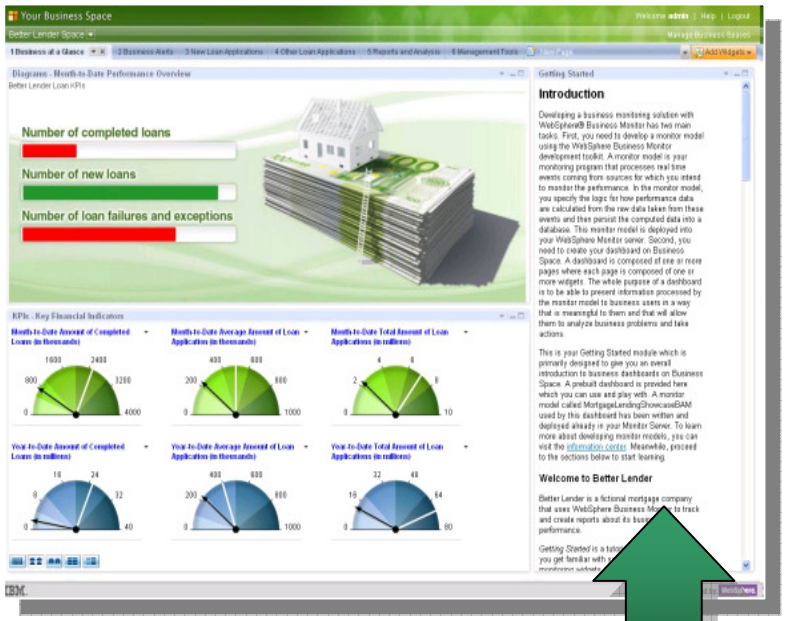
Business Activity Monitoring

- Why?
  - Decision-makers need Key Performance Indicators (KPIs)
- Where?
  - Best source of KPIs are the applications which run the business
  - The ESB has visibility of all of those applications
- How?
  - One way is to configure the ESB to produce monitoring events
  - Send the monitoring events to a monitoring application for analysis/display

Typical BAM scenario with Message Broker



MQ Pub/Sub topic



Message Driven Bean

CEI Repository

WebSphere Business Monitor

Monitor Model Application

WebSphere Application Server

Notes

- The message-driven bean is hosted within WAS.
  - So the WAS instance must have an MQ queue manager
- WebSphere Business Monitor can only receive events from the CEI component.
  - it would be difficult for WebSphere Message Broker to submit events directly to CEI
  - Better to publish in a flexible XML format and allow the MDB to make good for CEI/WBM if required.
- Picture at upper right is displaying Key Performance Indicators
  - ‘Business dashboard’

Typical BAM scenario details

- Events are published to an MQ topic
  - To allow multiple subscribers
  - To allow each subscriber to choose the level of granularity
    - Domain / Broker / Execution group / Message flow
- Event format is XML (published schema)
  - Designed to be compatible with CBE
  - Allows message broker to integrate with other monitoring applications
  - Allows entire message to be captured and logged to a database for audit purposes
- Events can be forwarded to WebSphere Business Monitor
  - Message driven bean provided with the monitoring sample
  - Fully supported offering
  - Wraps the WMB event in a CBE wrapper and submits to CEI

Monitoring support in v6.0 and earlier releases

- SupportPac IA9V
  - Subflow inserted into message flow to emit CBE event
  - Not a supported offering
  - Configured via XML files

- WBMTM
  - An IBM Services custom solution
  - Message capture/repair/replay facilities
  - Custom event repository
  - Custom user interface

- Custom message flow logic
  - Very flexible, but very costly in development effort

Notes

- IA9V
  - Payload data used ‘extended data elements’ feature of CBE
    - Not very good at dealing with complex payload data
- WBM Transaction Monitor
  - Aimed at transaction monitoring rather than BAM.
  - Customized for each customer/site.
    - Only one team of practitioners who know how to do this, so only available to a few large customers
- Custom message flow logic
  - Monitoring events are just XML messages…
  - …and broker is very good at XML messages
  - but much better to have a properly integrated offering.

Monitoring support in v6.1.0.2

- Monitoring events from message flows
  - Any input node can optionally emit **transactionStart**, **transactionEnd** and **transactionRollback** events.
    - But *only* on input nodes
    - Events can contain simple fields from input message payload
    - Not possible to capture “output” data from the flow
- Configuration
  - Only via the command line
    - A ‘monitoring profile’ held in a configurable service
    - Monitoring profile is an XML file which conforms to a published schema
  - All input nodes share the same monitoring profile
  - New commands `mqsichangeflowmonitoring` and `mqsireportflowmonitoring.`

Notes

- Transaction events
  - ‘Rollback’ is only issued if the catch/failure terminals are not wired up on the input node
    - So should not be integral to a properly-designed monitoring solution
- Configuration
  - This slide is setting up the story for v6.1.0.3, where configuration is much more powerful, and these v6.1.0.2 command-line facilities are upgraded to become a complete alternative to the toolkit facilities.

Monitoring Information

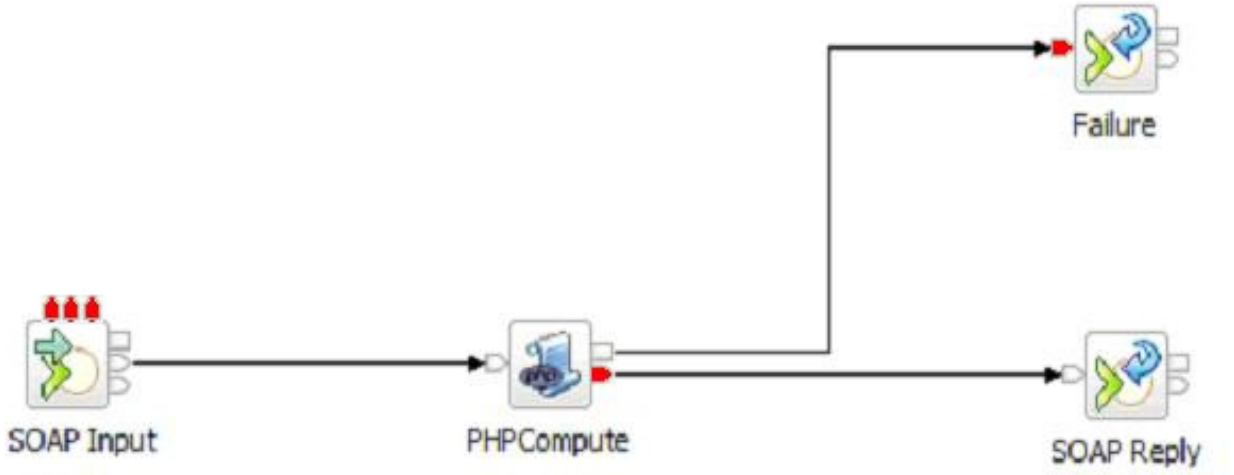
- Monitoring Events from message flows
  - Any input node can optionally emit transactionStart, transactionEnd and transactionRollback events.
  - Any terminal can emit an event as the message passes through
  - All events are optional, and fully configurable
  - Events can contain simple or complex data from message payload
- Configuration
  - Via the message flow editor
    - Excellent support in message flow editor via new Monitoring page on all nodes.
  - Via the command line
    - Monitoring profile upgraded to support the new facilities
    - `mqsichangeflowmonitoring` and `mqsireportflowmonitoring` updated to support the new facilities

Notes

- Terminal events
  - Event is only emitted if the message passes through the terminal in a forward direction
  - Events for error conditions should be configured on the nodes attached to the failure/catch terminals of the input node(s)

- Configuration
  - Purposely designed to be administered without the toolkit
    - So command-line / monitoring profile can do anything that the toolkit can do
    - …and toolkit config can be exported as monitoring profile to ease the transition (see later slide re: `mqsireportflowmonitoring`)

Monitoring support



Brokers | Properties

Default Values for Message Flow Properties - MessageProcessor

Description
Monitoring

Events

5 events defined. Events are defined via the Monitoring tab of a selected node in the message flow.

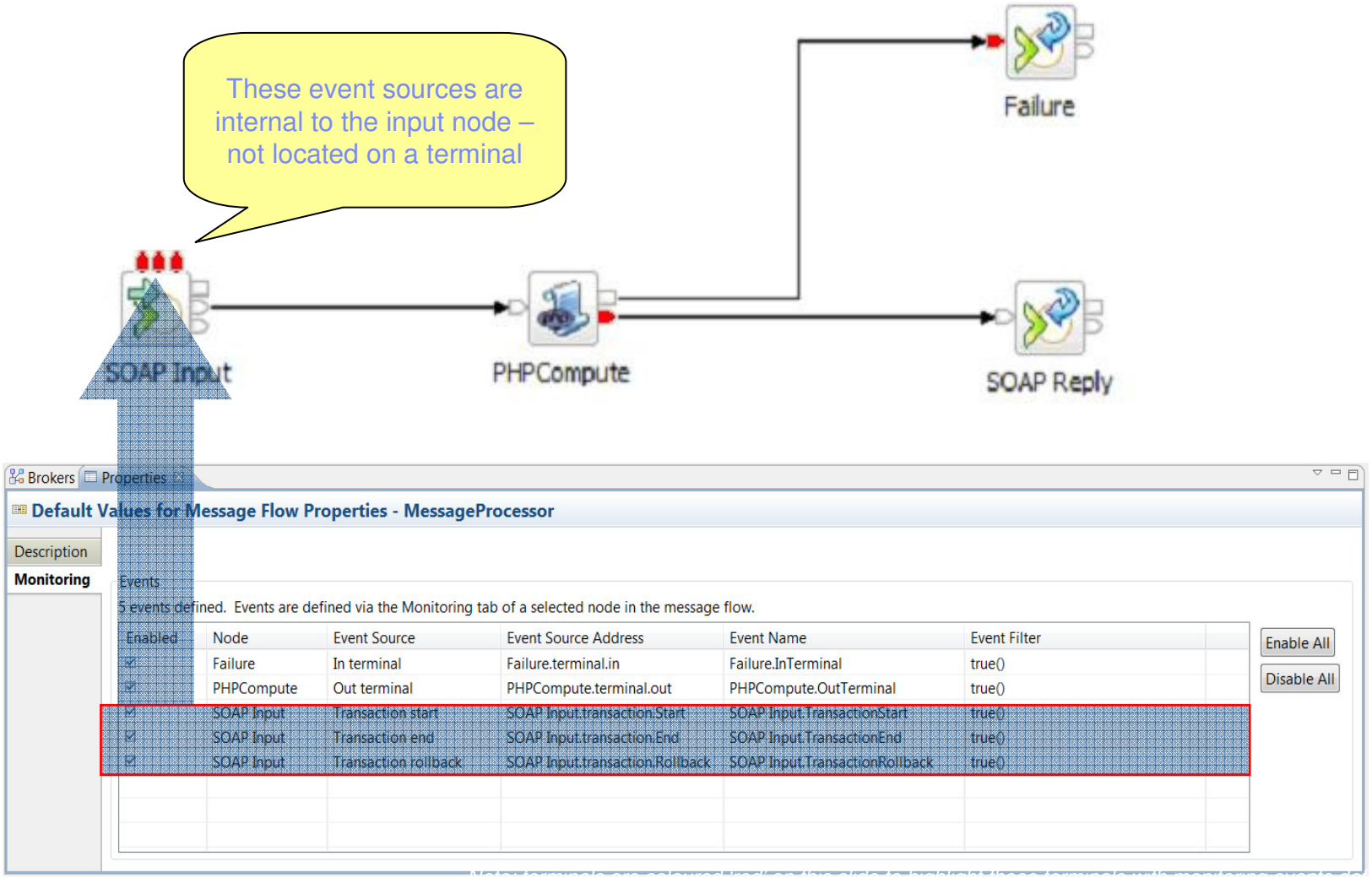
Enabled	Node	Event Source	Event Source Address	Event Name	Event Filter
☑	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()
☑	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()
☑	SOAP Input	Transaction start	SOAP Input.transaction.Start	SOAP Input.TransactionStart	true()
☑	SOAP Input	Transaction end	SOAP Input.transaction.End	SOAP Input.TransactionEnd	true()
☑	SOAP Input	Transaction rollback	SOAP Input.transaction.Rollback	SOAP Input.TransactionRollback	true()

Enable All
Disable All

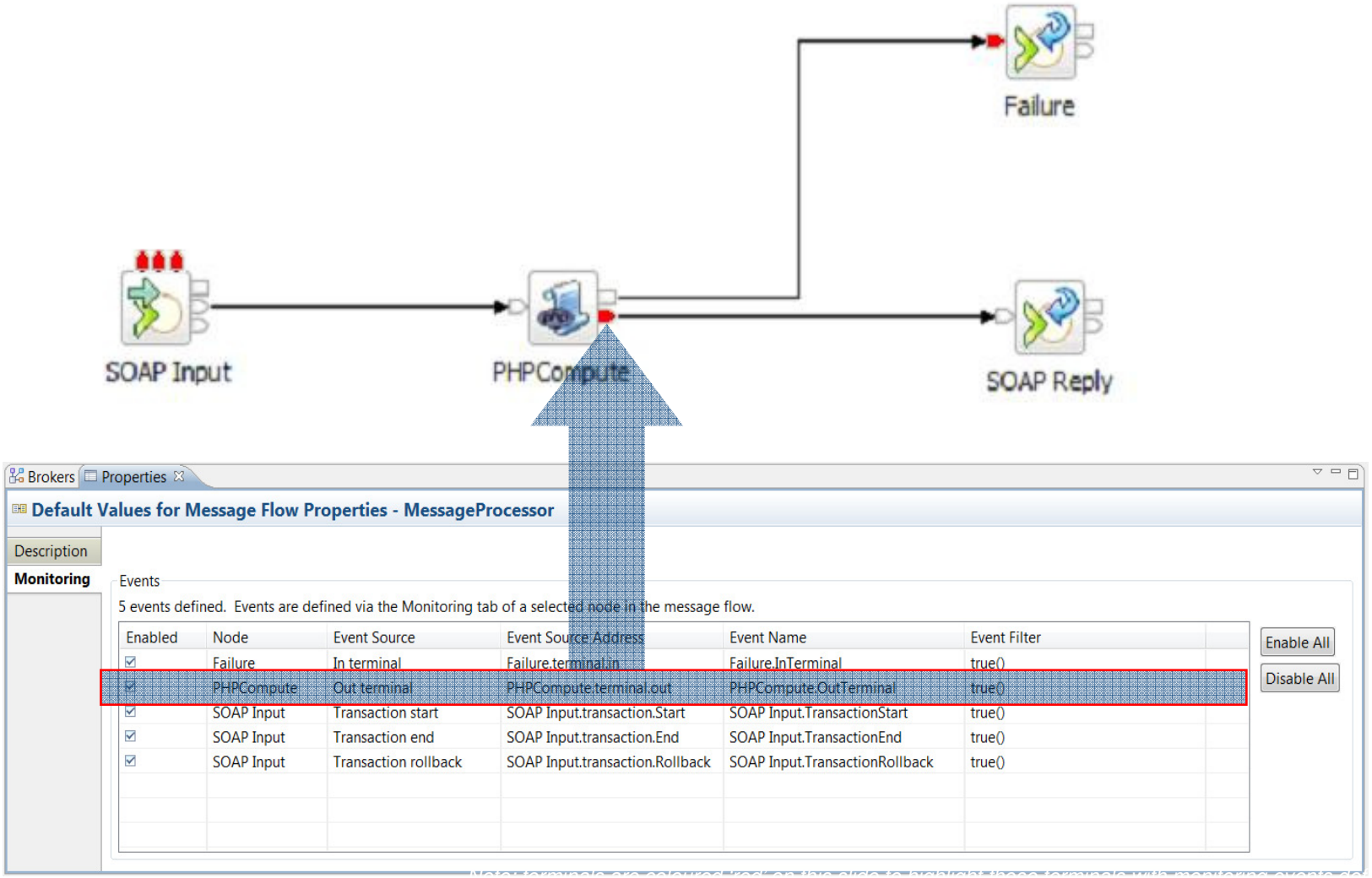
Notes

- Note the Monitoring page in the Properties view
  - The canvas of the message flow has been clicked, so it is displaying configured event sources for the message flow
    - not all event sources
  - Clicking an individual node would show event sources for that node
- How many potential event sources are there in this flow?
  - 15 (3 terminal events on each node + the 3 transaction events on the input node)
- Highlight the Event Source Address column
  - ESA is used to address an event source from the command line, or from a monitoring profile. It will be unique within a message flow, provided that the flow does not contain duplicate node names.
- Terminals highlighted in red have events
  - Not displayed like this in the editor!

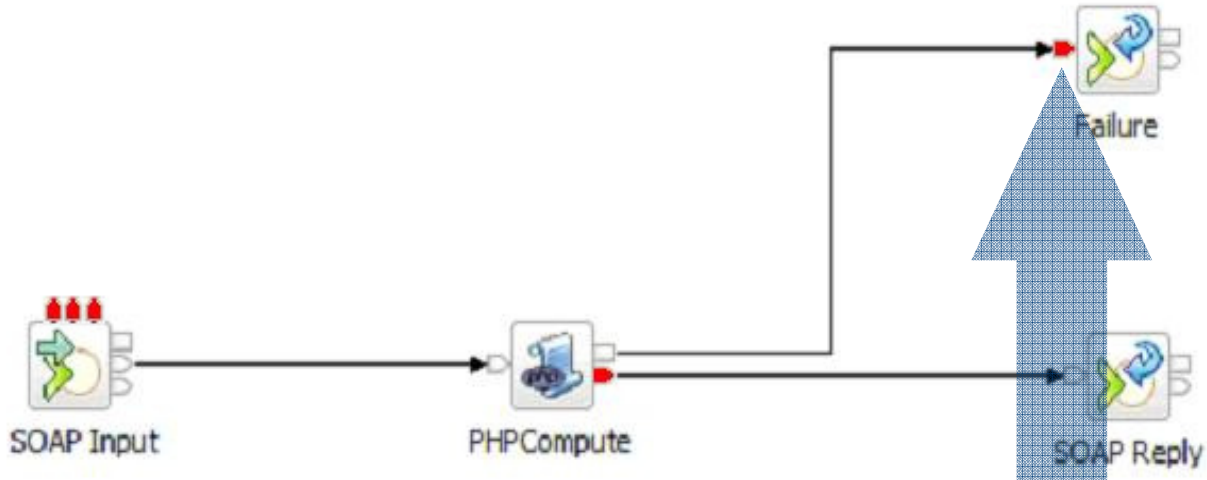
Monitoring support 2



Monitoring support 3



Monitoring support 4



Brokers | Properties

Default Values for Message Flow Properties - MessageProcessor

Description
Monitoring

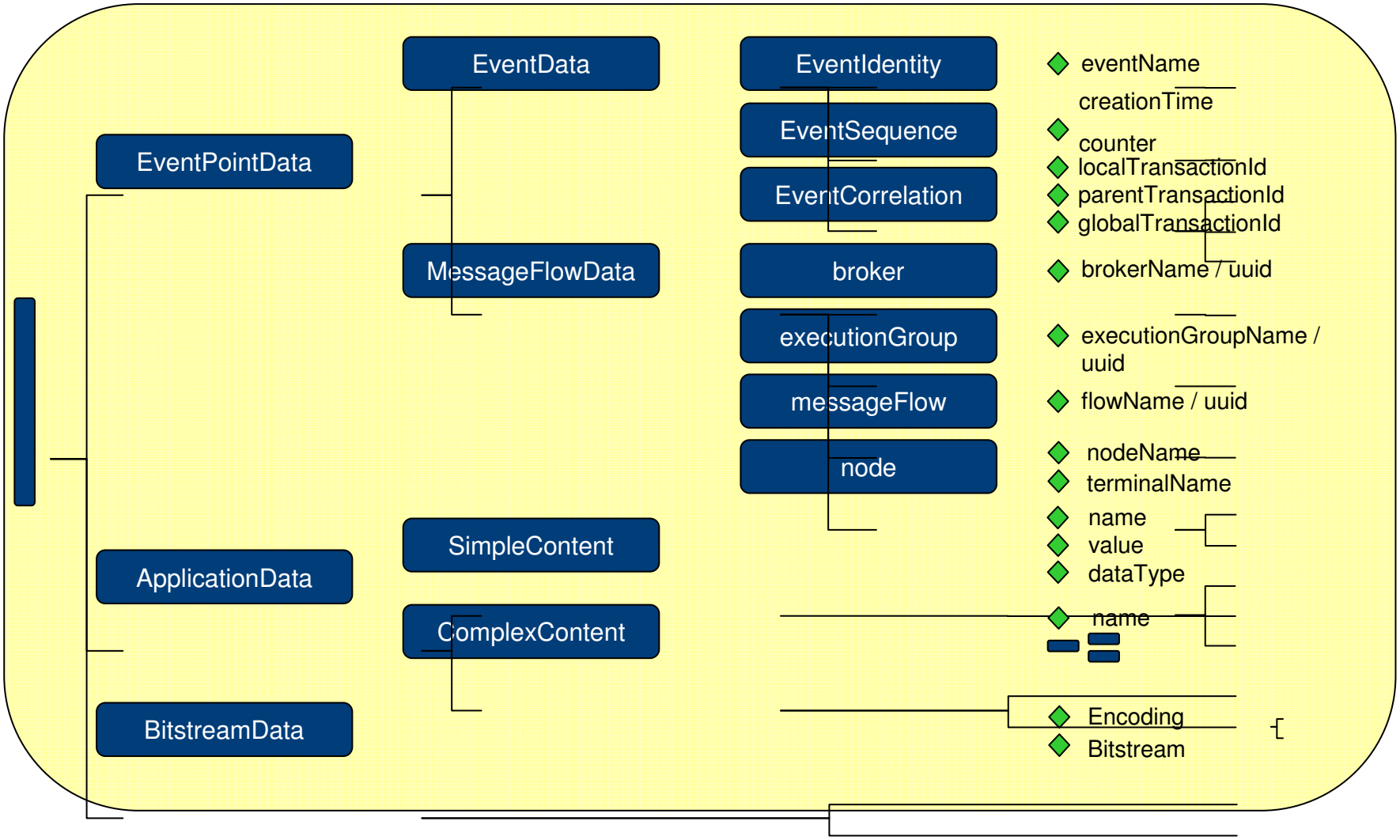
Events

5 events defined. Events are defined via the Monitoring tab of a selected node in the message flow.

Enabled	Node	Event Source	Event Source Address	Event Name	Event Filter
☑	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()
☑	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()
☑	SOAP Input	Transaction start	SOAP Input.transaction.Start	SOAP Input.TransactionStart	true()
☑	SOAP Input	Transaction end	SOAP Input.transaction.End	SOAP Input.TransactionEnd	true()
☑	SOAP Input	Transaction rollback	SOAP Input.transaction.Rollback	SOAP Input.TransactionRollback	true()

Enable All
Disable All

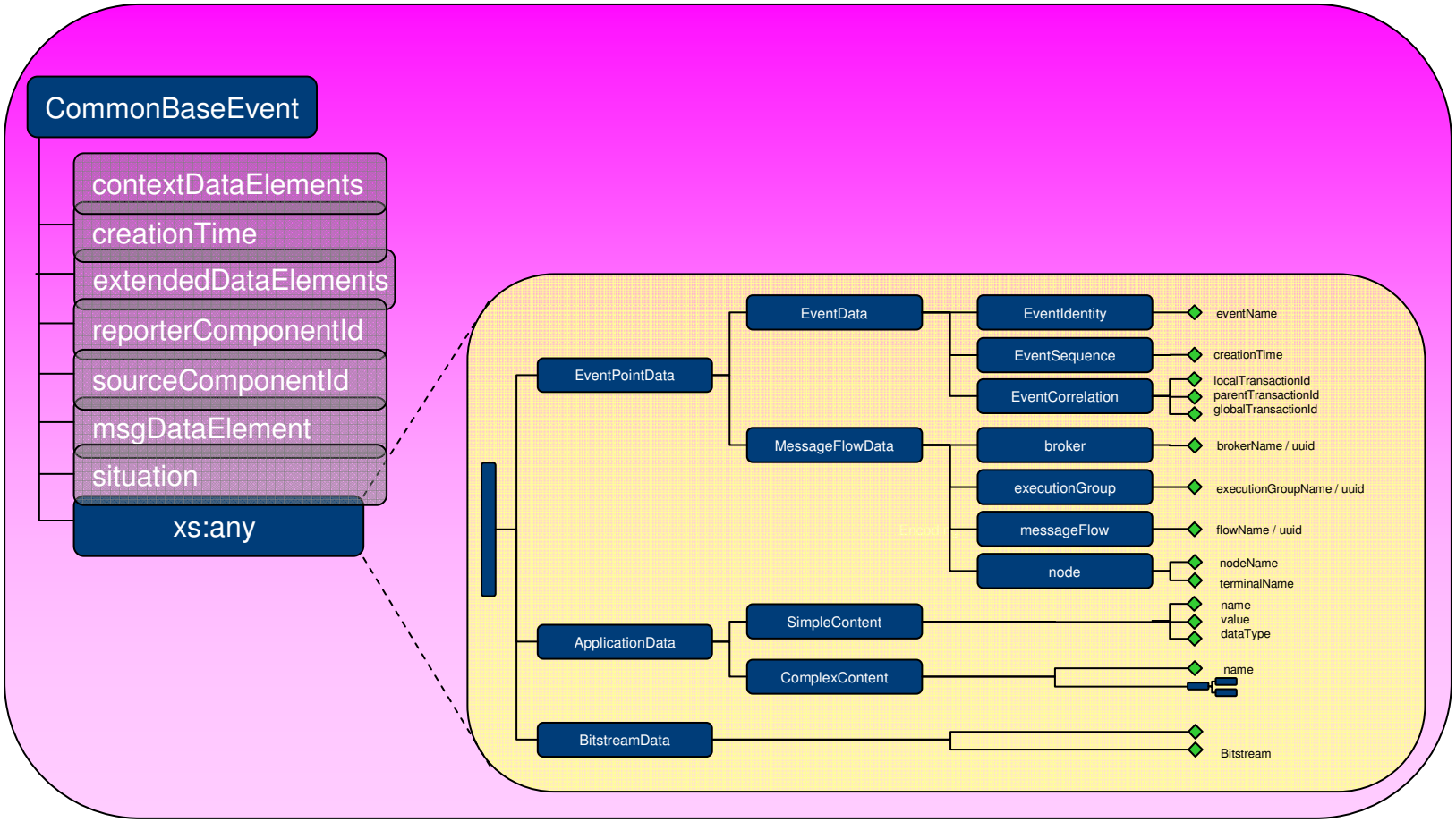
Monitoring event format details



Notes

- EventPointData/EventData
  - Provides the 3 main items of data for a monitoring application
    - Identity, Correlation, Sequence
- EventPointData/MessageFlowData
  - Should be self-explanatory – no surprises here
- ApplicationData
  - Can come from message headers or body
  - Automatically included as XML, even if source message was non-XML
  - If the XPath/ESQL returned a simple element, it is placed in simpleContent, else it goes into complexContent
    - simpleContent was the only available option in v6.1.0.2
- BitstreamData
  - More details in later slides

Monitoring event format : after Message Driven Bean



Notes

- MDB is wrapping the WMB monitoring event in a CBE envelope
  - Fields in the CBE should be familiar to some in the audience
- Note that the WMB data goes into the xs:any slot in the CBE
  - Most fields in the CBE wrapper are simply left at their default values
    - Including `@cbe:severity/@cbe:priority`
  - This is now the recommended way to construct a CBE which contains complex application data
    - Because extendedDataElements is poor at carrying complex subtrees

Monitoring events from message broker

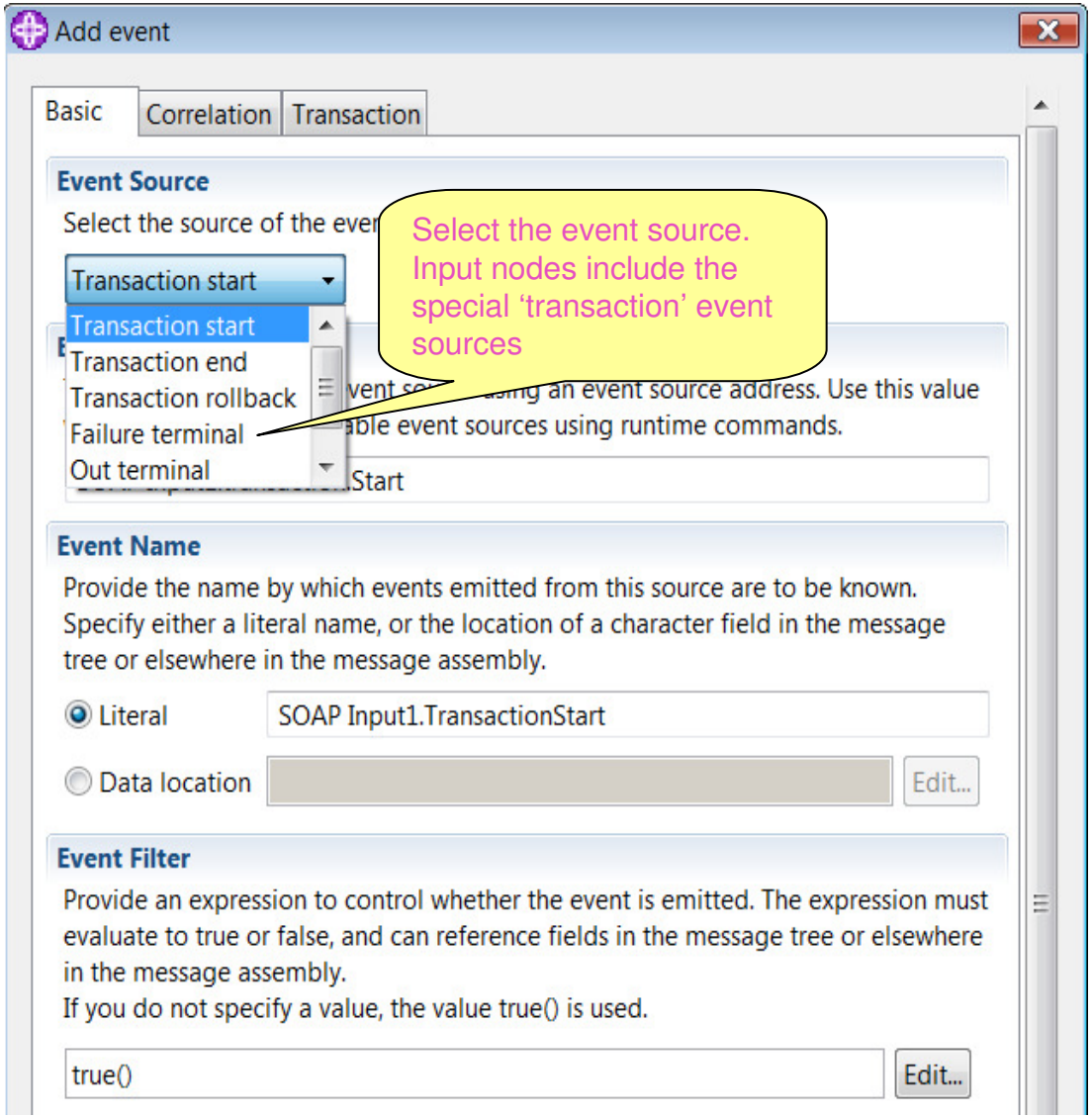
- Good interoperability with WebSphere Business Monitor
  - Identity, correlation and sequencing of events is explicitly built into the event format
- Flexibility
  - Events can be consumed by one or more clients subscribing to the appropriate topic. Topic can include wildcards, allowing the scope of the subscription to vary

    `$SYS/Broker/<brokerName>/Monitoring/<executionGroupName>/<flowName>`
  - Events are emitted in a published (documented) XML format (schema provided), MDB supplied with the product submits events to CEI

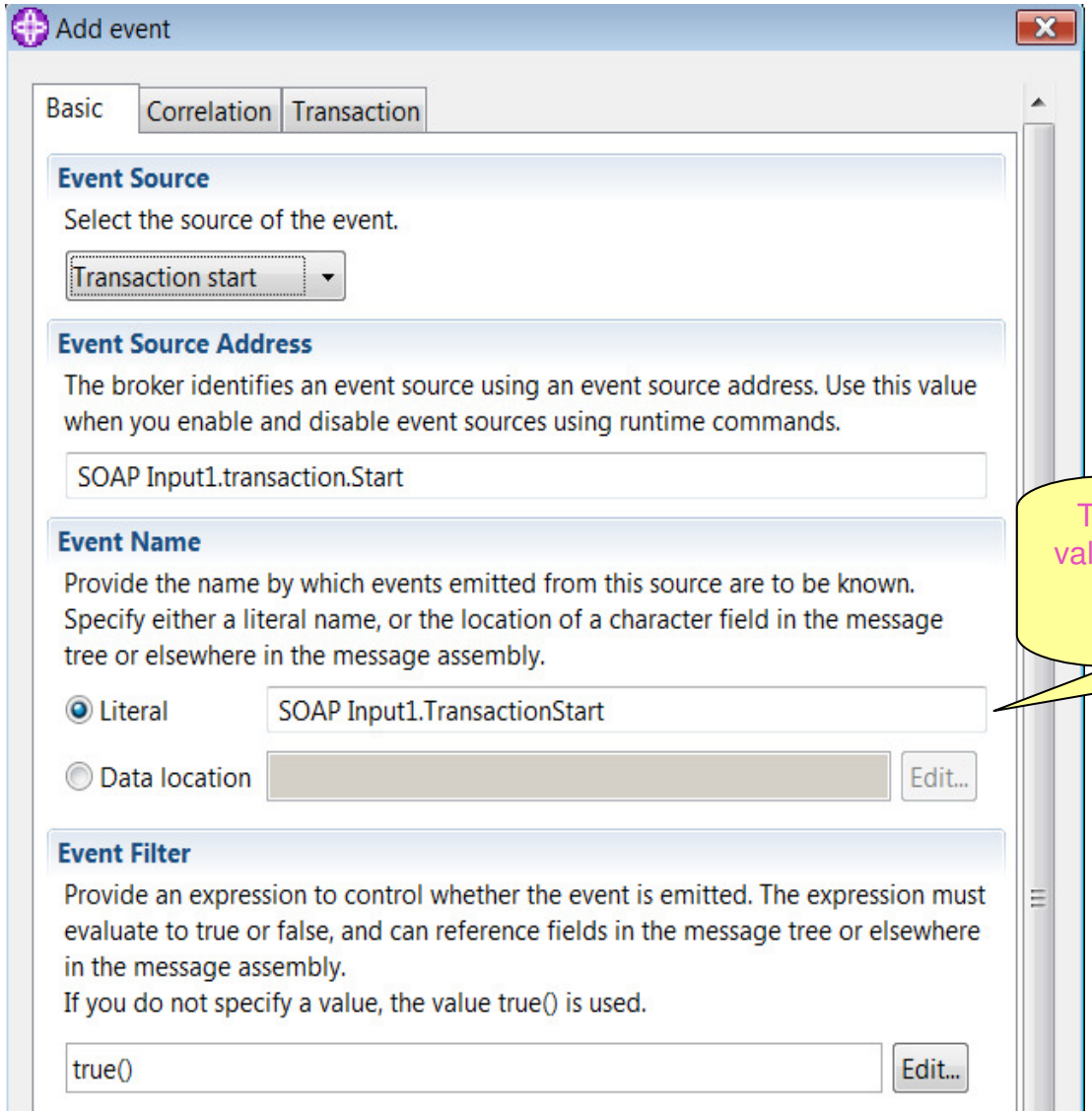
Monitoring events : features

- Unique default event name is automatically assigned
  - Can be overridden with a fixed value
  - Can be read from message payload
- Sequence field is automatically populated
  - Creation time of event
  - Auto incrementing counter
    - Start at 1 for the first event issued
    - Increment by 1 on each subsequent event emitted
    - Reset to 1 at the start of the next message
- Correlator field in event is automatically populated
  - Same for all events from one invocation of a flow
  - Many more options (details later)

Configuring: Adding an event to a node

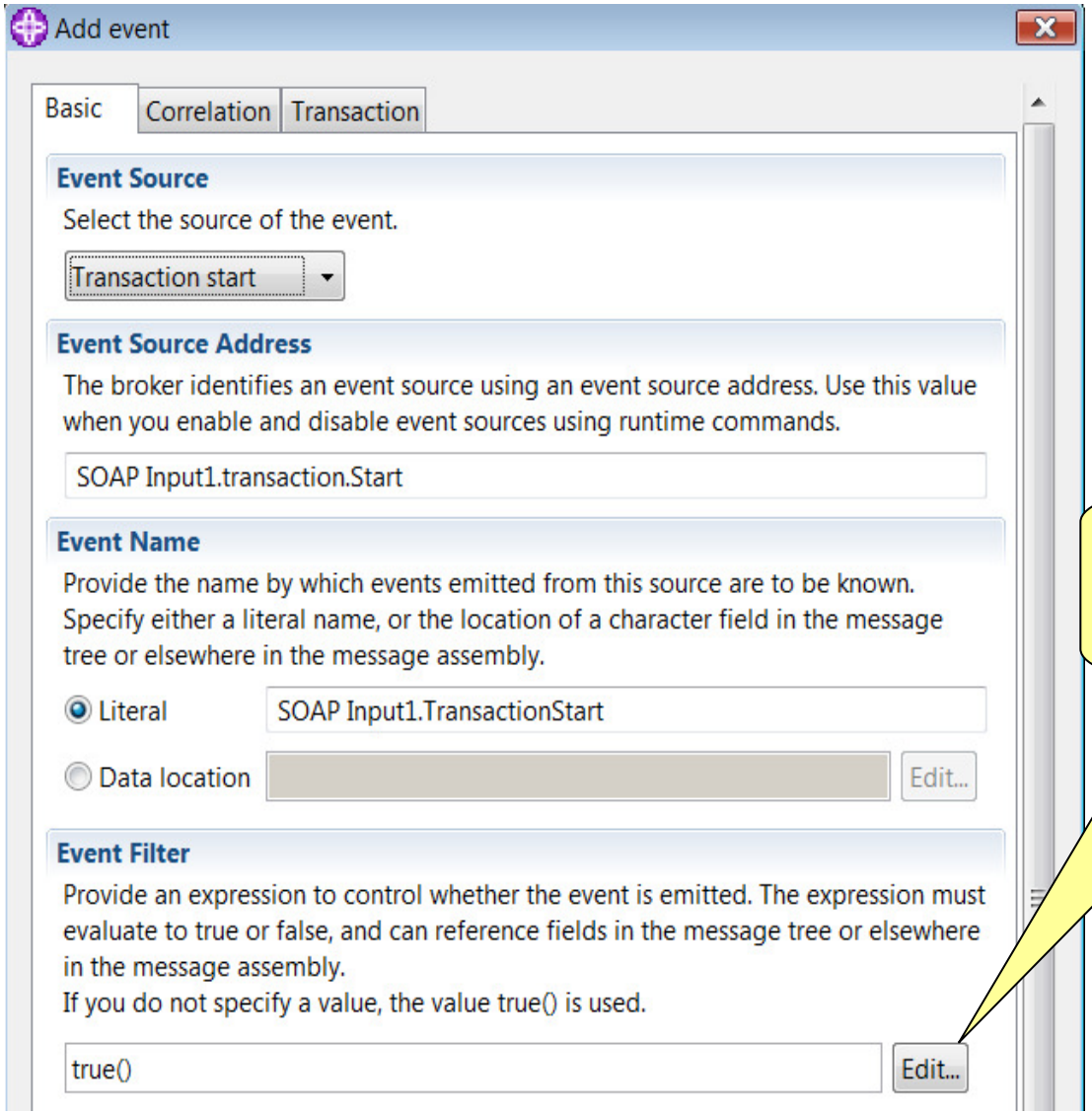


Configuring: Customizing an event



The event name can be a literal value, or can be extracted from the message payload using an expression

Configuring: Adding a filter to an event



XPATH expression to indicate if this event should be emitted or not

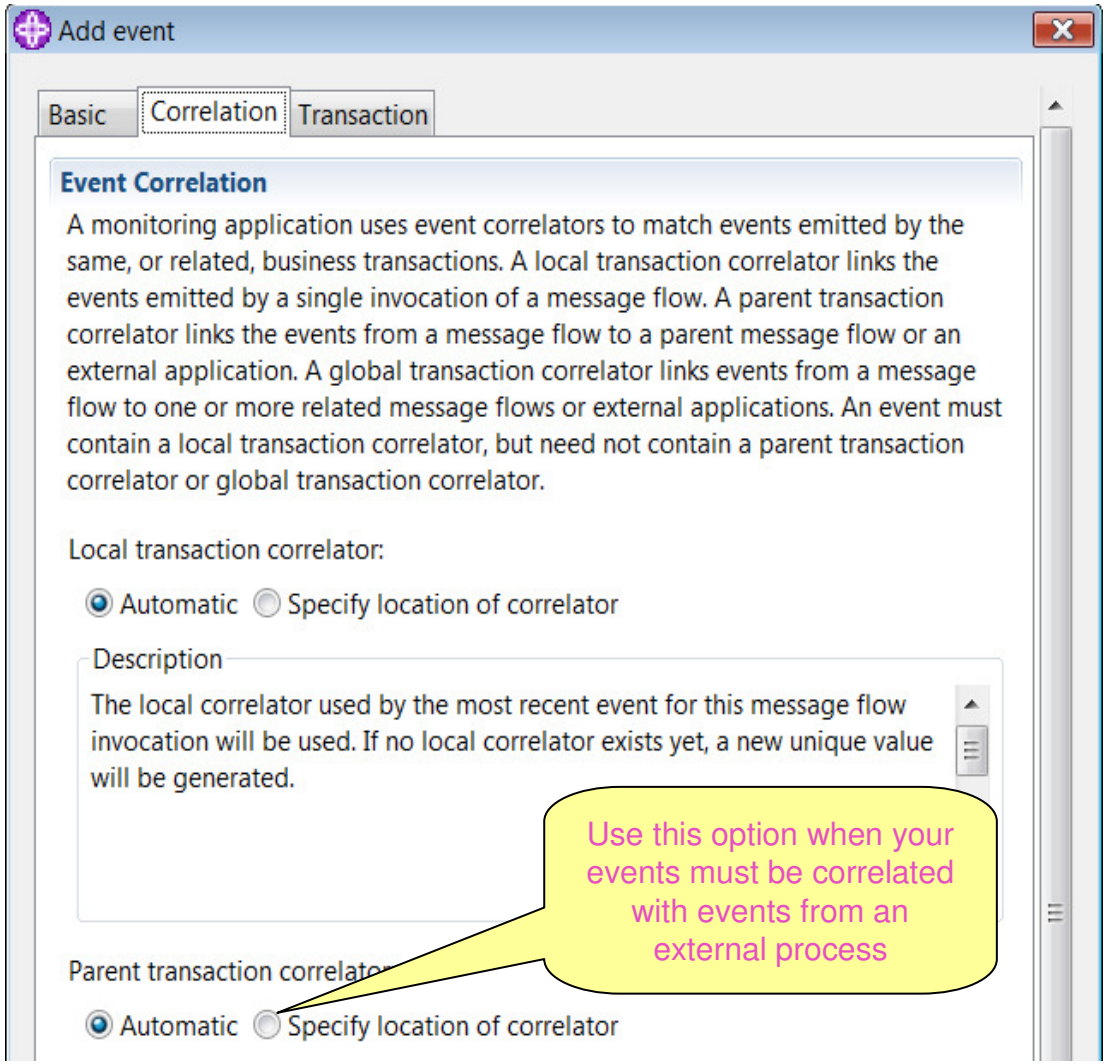
Configuring: Adding a filter to an event (2)

- Expression evaluates to
  - True – event emitted
  - False – event not emitted
- Evaluated at runtime
- Set on event source definition
- Expression can reference fields from anywhere in the message assembly
- XPath expression builder support available
- Event filter appears on Monitoring summary table

Notes:

- Event Filter section in the Monitoring tab (V7 onwards)
- Enables user to filter out events which do not match a business rule. So events can be filtered at Message Broker rather than emitting to Business Monitor and filtering there - which will help performance
- The event filter can be set to a numeric, boolean or string XPath expression which will evaluate to boolean true or false. Typically the result of an a = 'b' or x > 'y' type test
  - If the expression evaluates to true then events are emitted
  - If the expression evaluates to false then events are not emitted
- The default setting is true()

Configuring: Customizing an event - Correlation



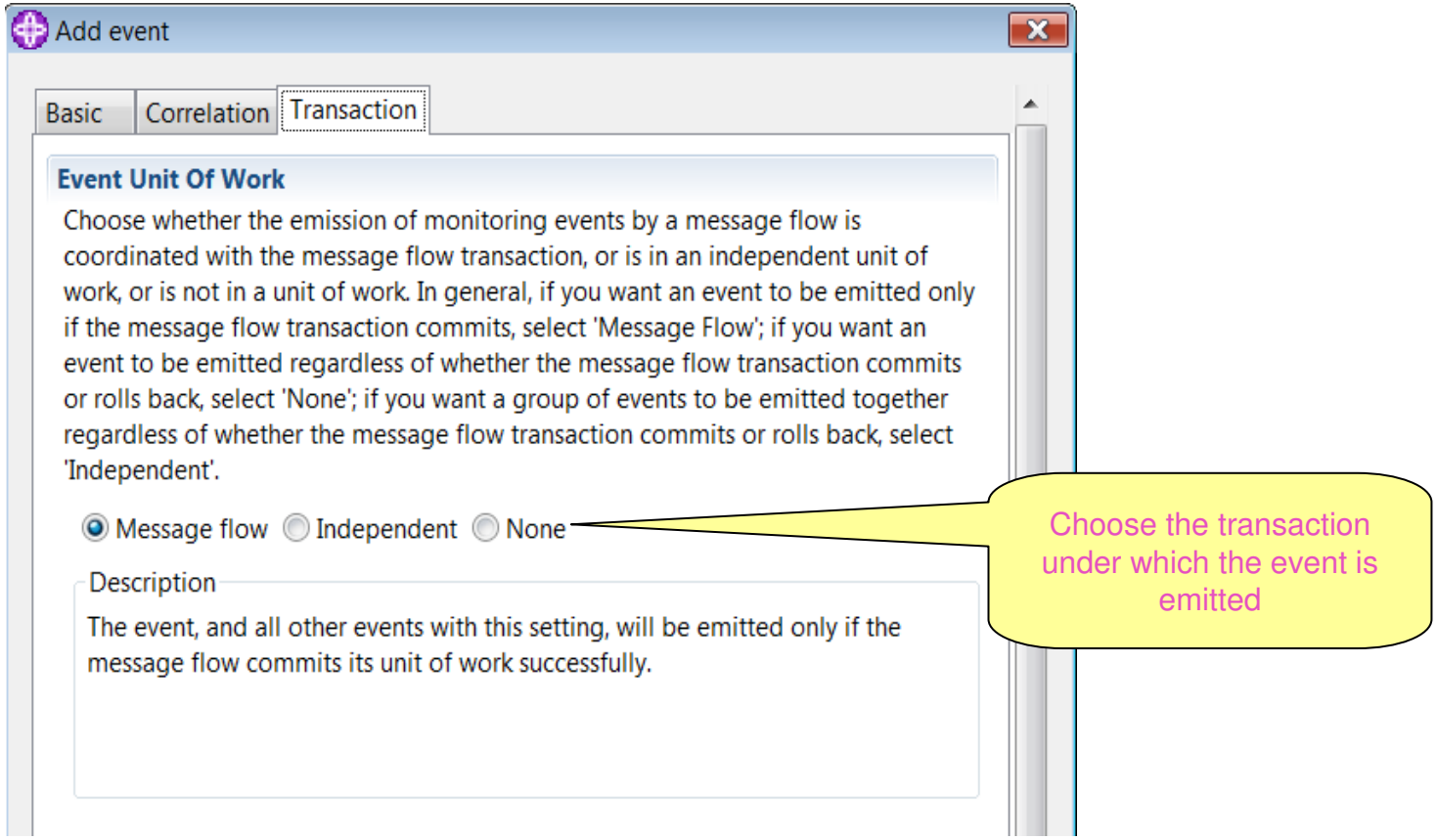
Correlators

- Each event contains up to three correlation fields
  - localTransactionId
    - Automatically populated with a unique identifier which will be the same for all events emitted during a single invocation of the message flow
    - Its value can be set from a field in the message (often from a header). **Once set, later events inherit the same value.**
  - parentTransactionId and globalTransactionId
    - Empty by default.
    - Value can be set from a field in the message (often from a header). Once set, later events inherit the same value.
- Simple scenarios are easy, complex scenarios are possible.

Notes

- If correlator is read from a message header
  - Only specify it once
    - Usually on the transactionStart event
  - Do not copy the XPath/ESQL expression to later event definitions
    - It would work, but would incur a needless performance penalty
  - Correlators are cached in the Environment tree, and later event sources automatically reuse them if they have been set.

Configuring: Customizing an event - Transaction



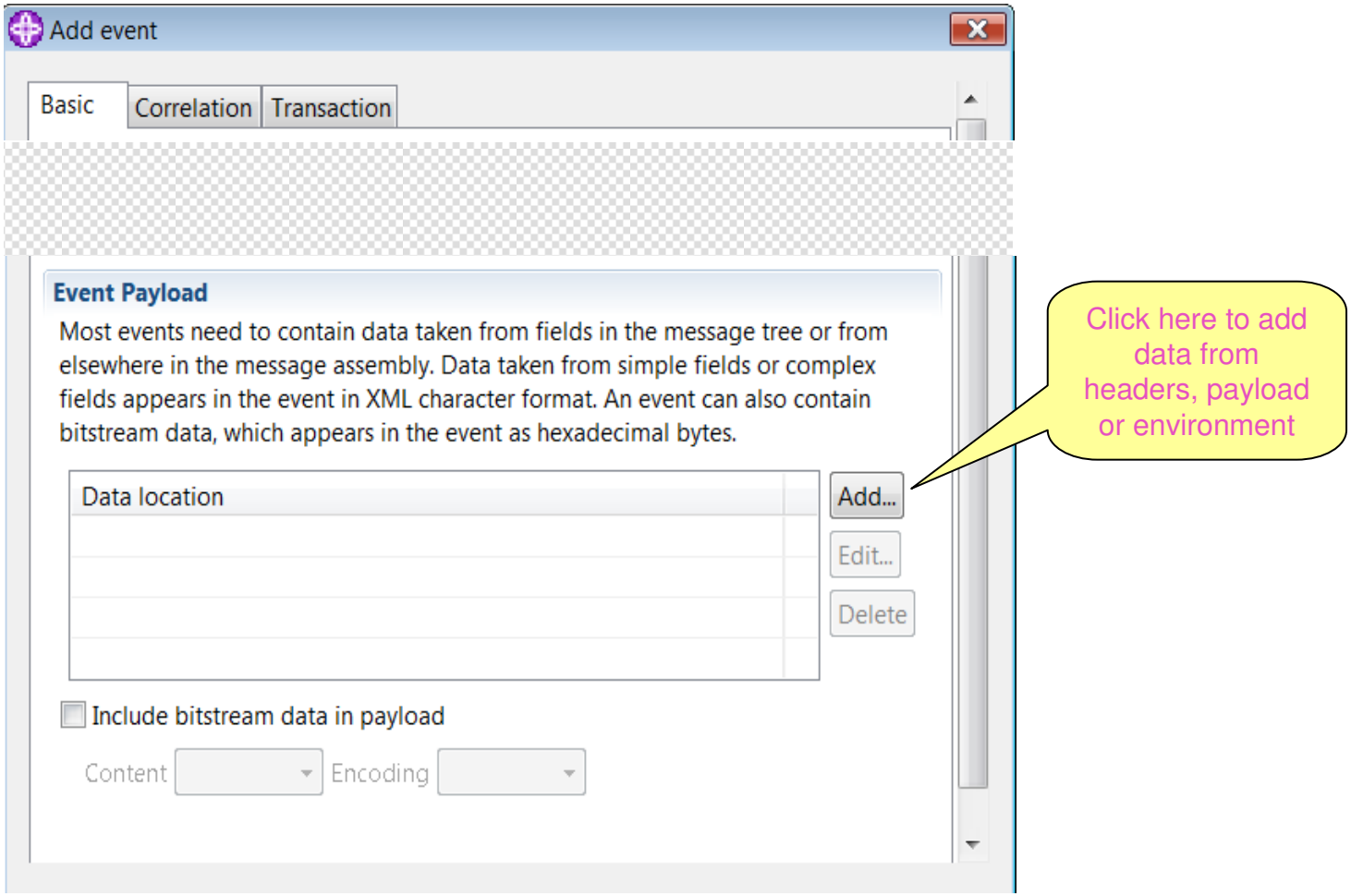
Notes:

- What are the different units of work?
  - When a message is processed, the MQ updates are included in a unit of work referred to as the "**Message Flow**" unit of work. It is committed if the message processing is successful and rolled back if it fails
  - The "**Independent**" unit of work is a separate unit of work which is created and committed regardless of whether the message is processed successfully or not. Use this for events, such as those related to error paths, that must be published even if the flow fails.
- If you don't want a monitoring event to be included in any unit of work, choose the "**None**" option

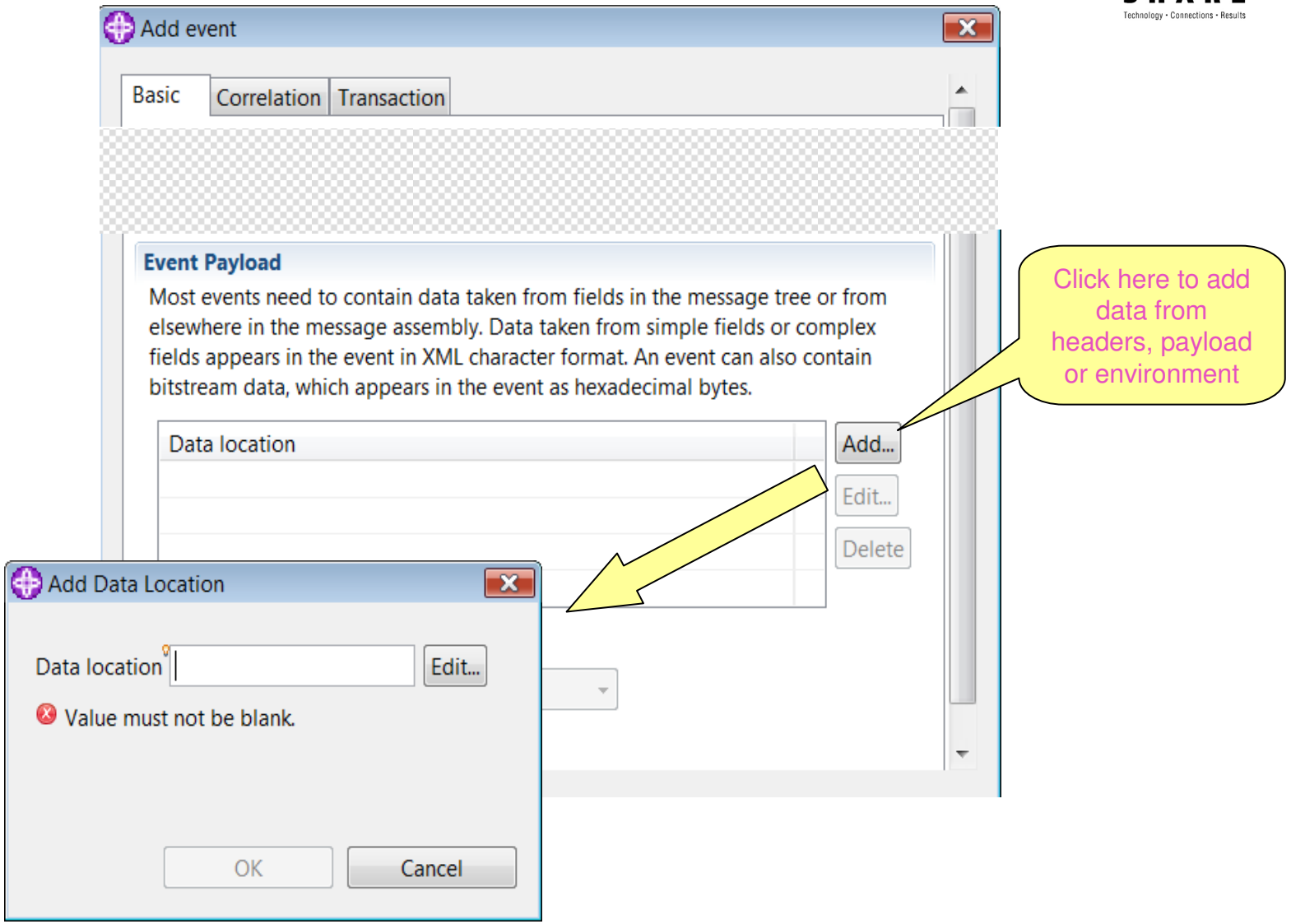
Notes 2:

- Some differences in behaviour depending on the event type:
  - Consider the following events generated from a message flow:
    - Seq no 1 a transaction start event
    - Seq no 2 an event in the message flow unit of work
    - Seq no 3 an event in the independent unit of work
    - Seq no 4 an event specified as not in a unit of work
    - Seq no 5 an event in the message flow unit of work
    - Seq no 6 a transaction end or rollback event (see below)
    - If the message is successful, all these events, plus seq no 6, a transaction end event, are published
    - If the message fails, events 2 and 5 are rolled back and only events 1, 3, 4 and seq no 6, a transaction rollback event, are published
    - Note that as event 4 is published as soon as it is generated, outside of a unit of work, it will be the first to appear external to Message Broker and is unaffected by any commit or rollback processing

Customizing an event – payload data



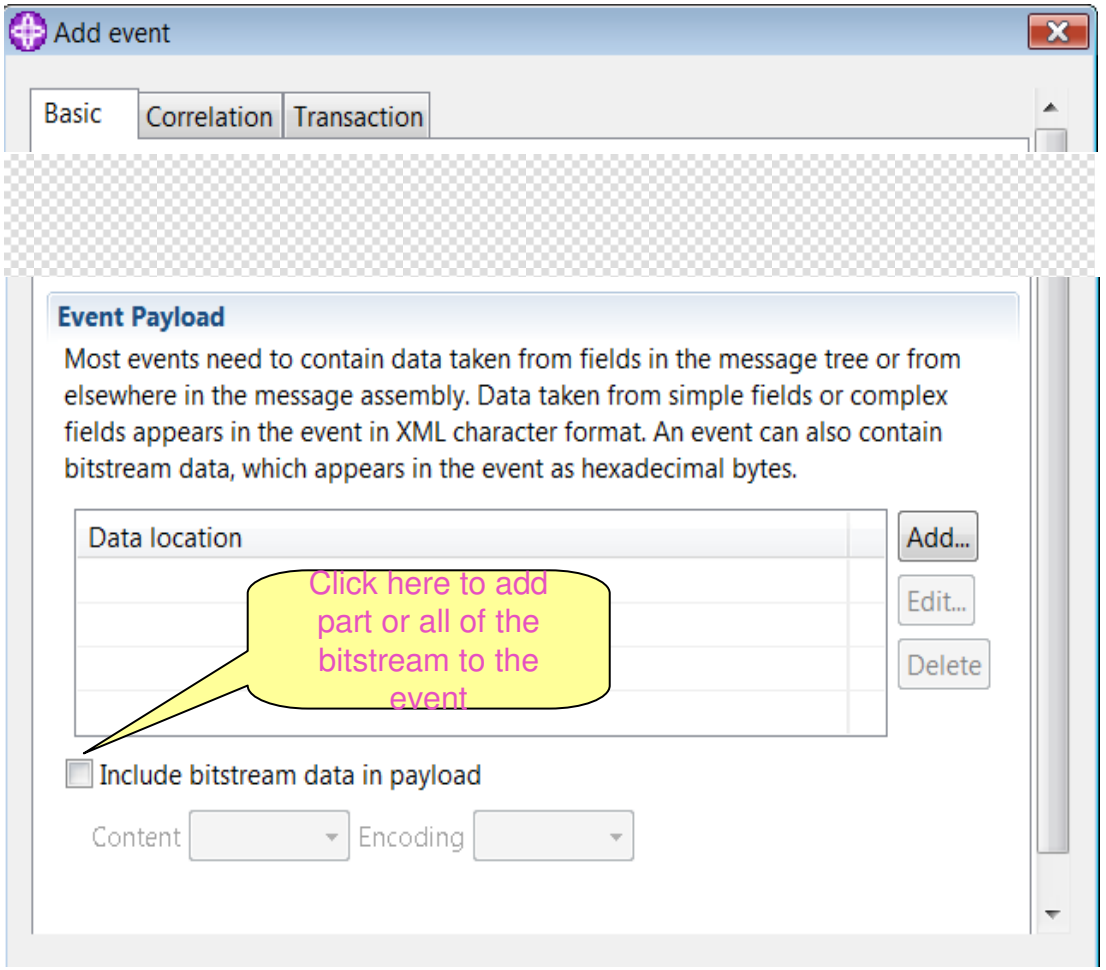
Customizing an event – payload data



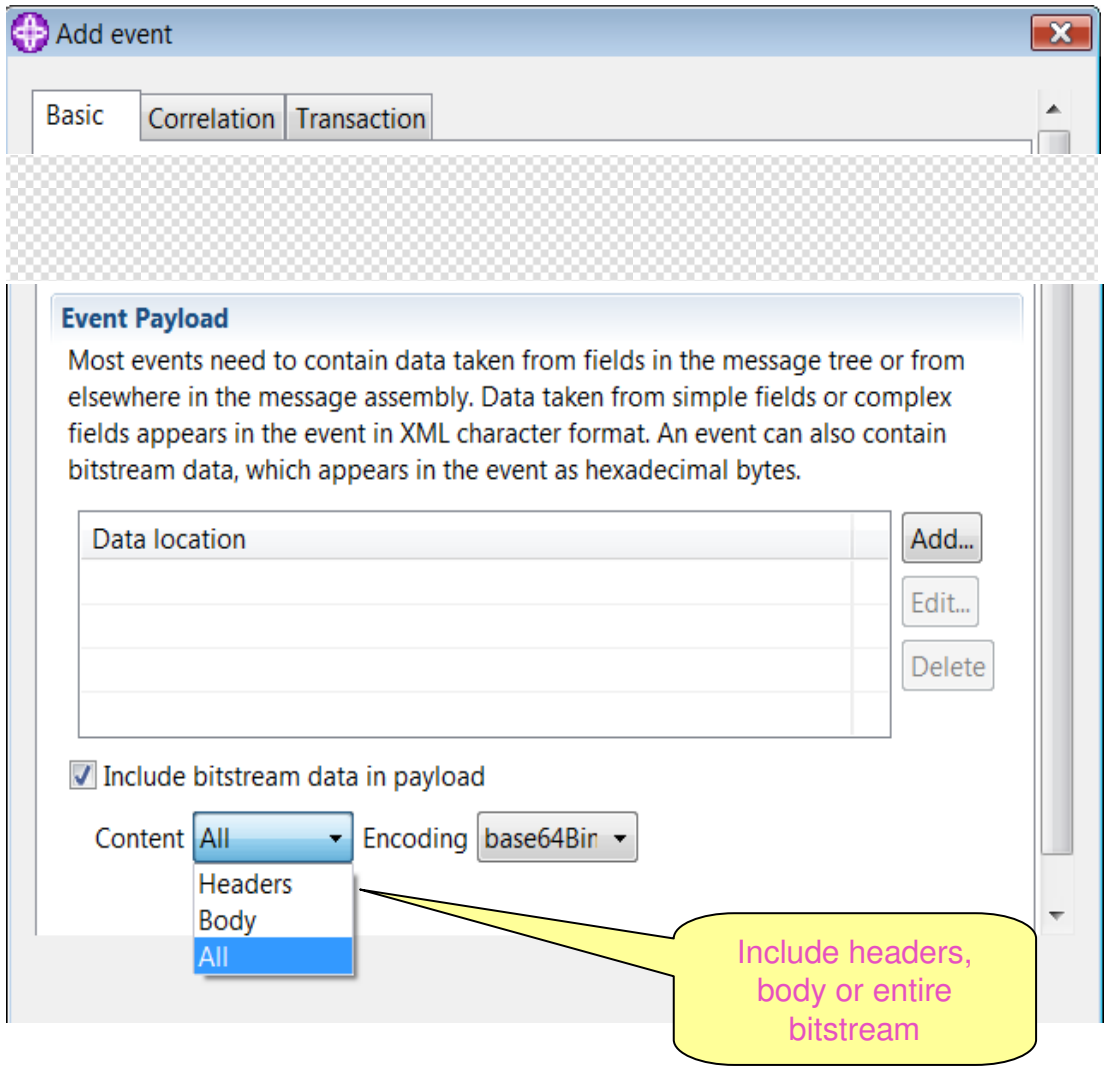
Payload data

- Multiple XPath queries can be specified
  - Or ESQL paths; the support for both is generic
  - XPath builder provides assistance with constructing the path
- Simple fields automatically go into applicationData/simpleContent
  - If monitoring profile is used, the @dataType attribute can be set for each item of simpleContent. Even if the message broker tree holds the data as characters, WBM can be instructed to treat it as integer / date etc
- Complex fields automatically go into applicationData/complexContent
  - Non-XML data from the MRM parser is automatically converted to XML when included in a monitoring event.

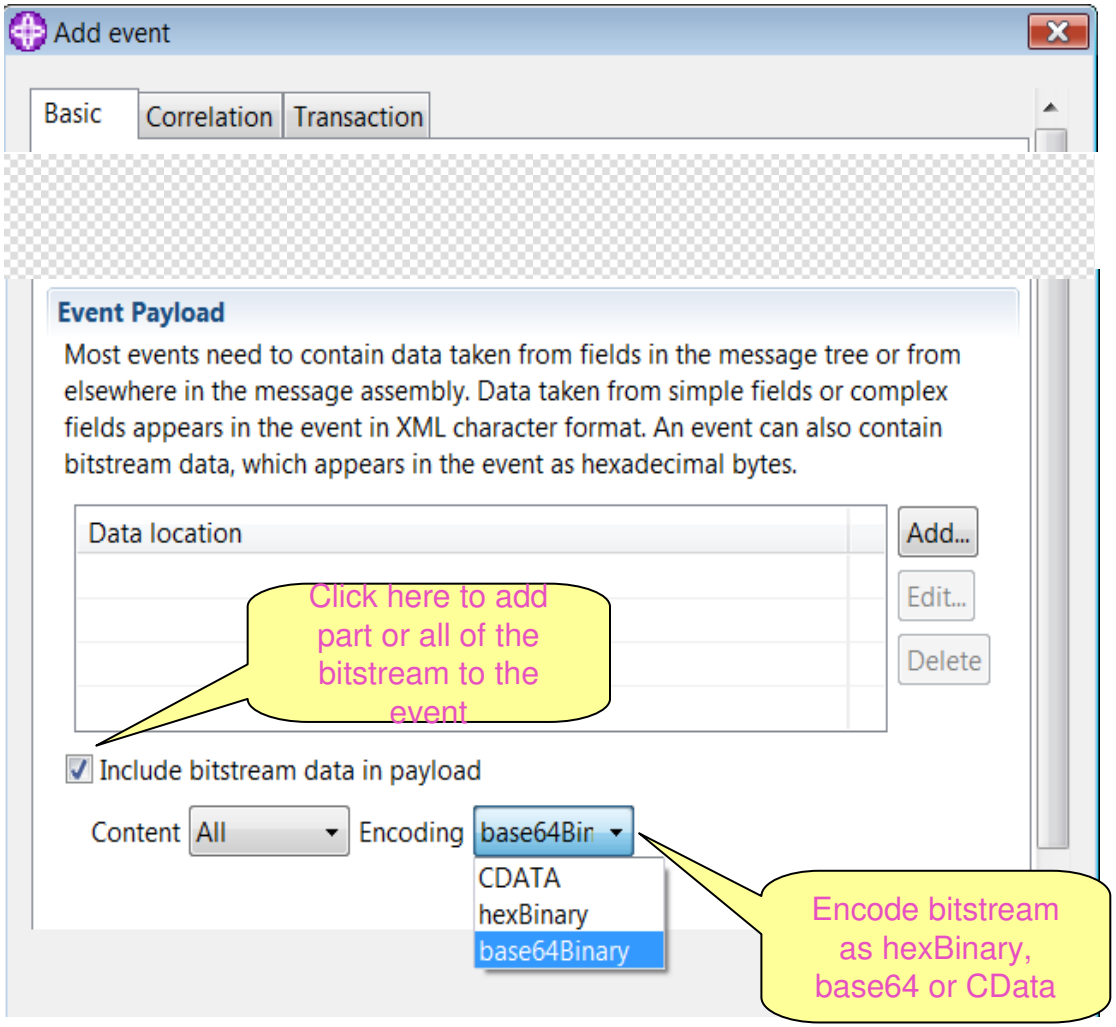
Customizing an event – bitstream data



Customizing an event – bitstream data



Customizing an event – bitstream data



Notes

- CData format for bitstream
  - Not safe unless you **know** that the XML is free from invalid characters
    - NB: CData does not protect you from invalid XML characters
  - So not usually safe for use with ‘All’
    - Because that will include headers which may contain binary data.
  - **Recommendation**: Use CData encoding with care, and only with content set to ‘Body’

Bitstream data

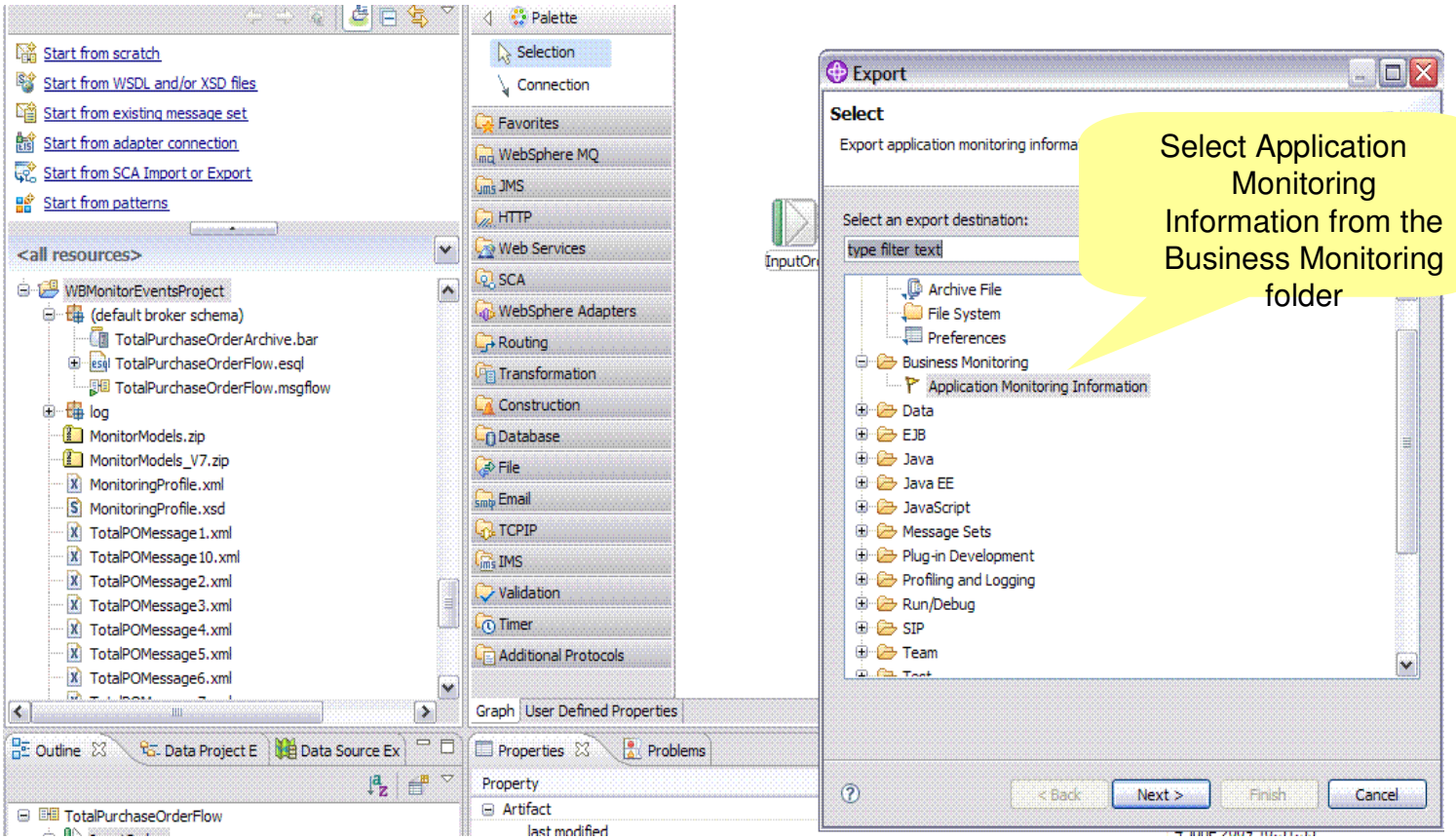
- Bitstream data for auditing
  - Not expected to be used in standard BAM scenarios.
  - Events can be captured and written to a database.

- Bitstream data for resubmission
  - WMBTM offering can provide capture/repair/resubmit based on the new monitoring events
  - Custom solutions are also possible

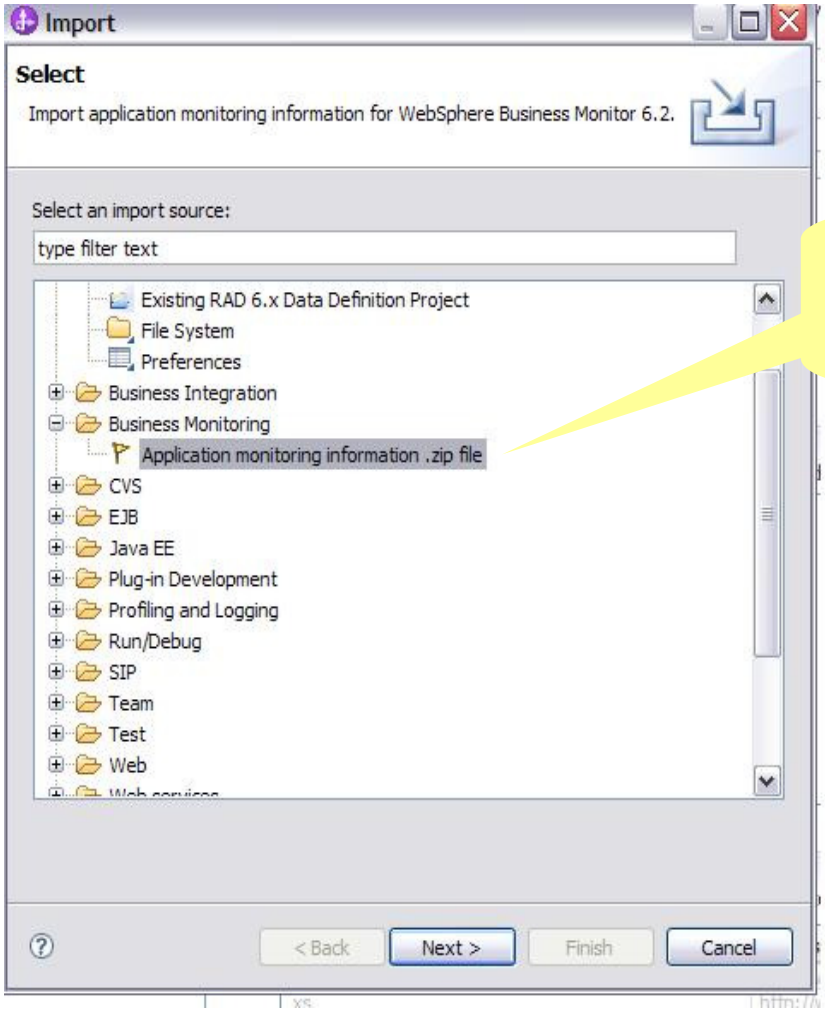
Generate Monitoring information for Websphere Business Monitor

- Using an export monitoring information option a user can use a Generate Monitor Model wizard to create a model which has automatically created:
  - Inbound events for each event source defined in the message flow
    - event parts describing event payload
    - filter condition
    - correlation expression
    - event sequence path expression
  - localTransactionId defined as a key
  - Additional metrics and KPIs are available by selecting templates during the Generate Monitor Model wizard
- Log file created in Message Broker message flow project to show output from generate process

Exporting Message Flow monitoring information

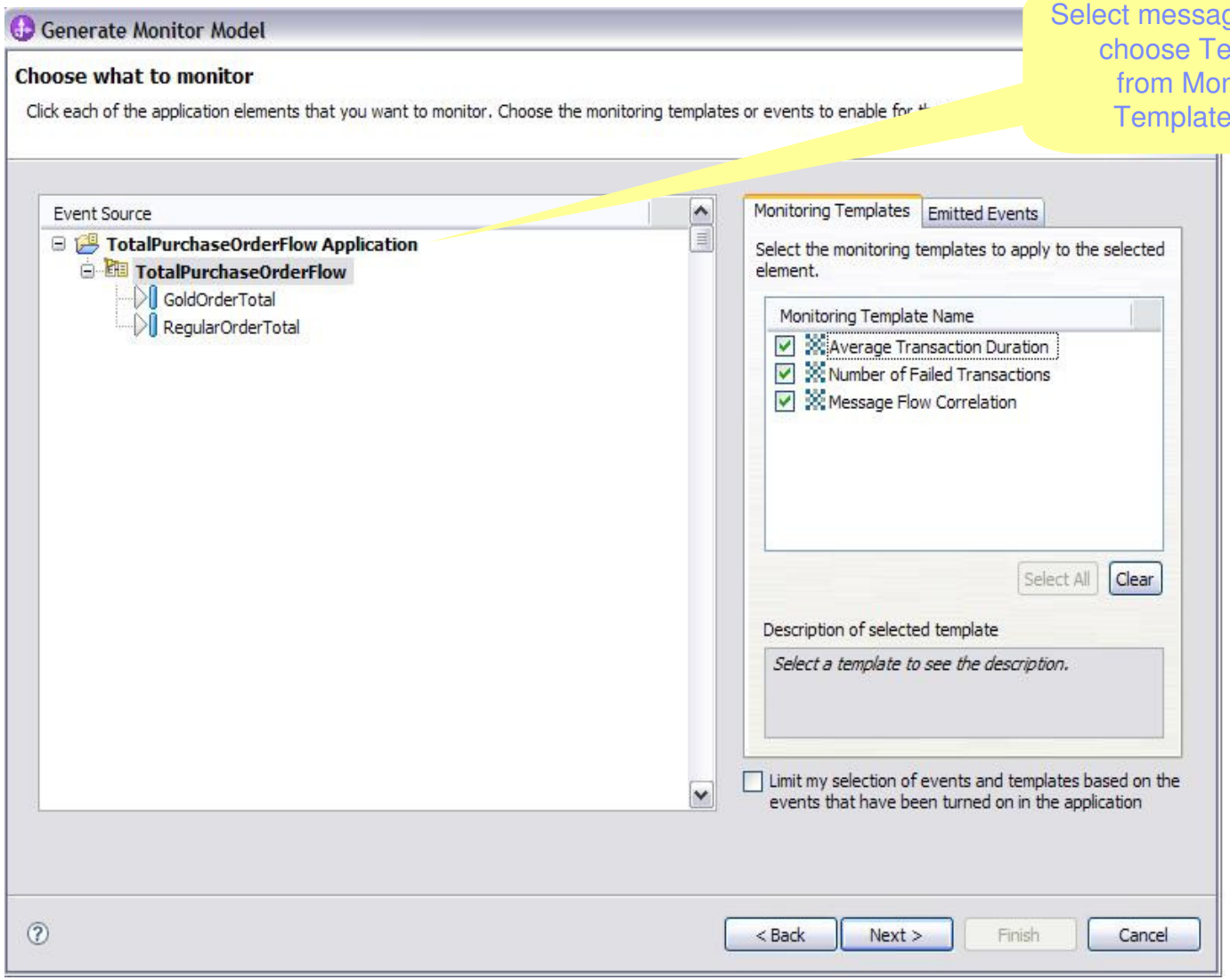


Import Monitoring Information into WebSphere Business Monitor Toolkit



Select Import then Application monitoring information .zip file from Business Monitoring folder

Generate Monitor Model (multi-step process)



Select message flow to choose Templates from Monitoring Templates page

Business Space Manager

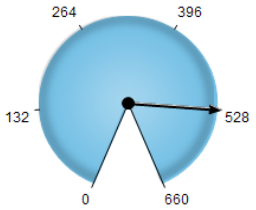
Instances

ED	CreationTime	CustomerID	CustomerType	Discount	ExecutionGroup	Failed	FlowProcessingTime	LocalTransactionID	Message Flow	PurchaseOrderID	Succeeded
	27 August 2009 10:20:49	111111	GOLD	163.44	WBMonitorEventsExecGroup	0	8.656 s	ff8298eb-af6b-4640-b6cc-655d00540e8b-4	TotalPurchaseOrderFlow	11112222	1
	2 September 2009 11:51:12	111111	GOLD	163.44	WBMonitorEventsExecGroup	0	12.156 s	d4c40c0d-ced1-4676-9c0a-f63cceb3cb36-1	TotalPurchaseOrderFlow	11112222	1
	2 September 2009 11:58:14	222222	GOLD	58.86	WBMonitorEventsExecGroup	0	0.453 s	d4c40c0d-ced1-4676-9c0a-f63cceb3cb36-2	TotalPurchaseOrderFlow	22223333	1
	17 September 2009 13:42:29	111111	GOLD	163.44	WBMonitorEventsExecGroup	0	9.281 s	0d6254f6-d4c2-4802-bfc9-3706f904dccc-1	TotalPurchaseOrderFlow	11112222	1
	17 September 2009 13:45:31	222222	GOLD	58.86	WBMonitorEventsExecGroup	0	0.61 s	0d6254f6-d4c2-4802-bfc9-3706f904dccc-2	TotalPurchaseOrderFlow	22223333	1
	17 September 2009 13:56:17	222222	GOLD	58.86	WBMonitorEventsExecGroup	0	0.75 s	0d6254f6-d4c2-4802-bfc9-3706f904dccc-3	TotalPurchaseOrderFlow	22223333	1

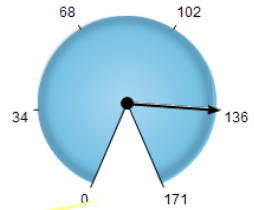
55 - 60 60

KPIs

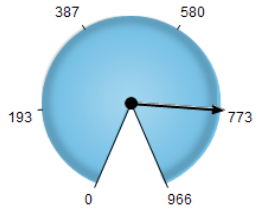
Average Difference Between Gold and Regular Spending



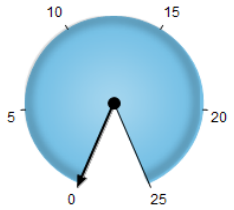
Average Gold Customer Discount



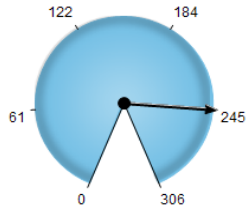
Average Gold Customer Order Amount



Average Regular Customer Discount



Average Regular Customer Order Amount



Dashboard populated with data from Message Broker events

Command Line: mqsichangeflowmonitoring

- v6.1.0.2 usage still supported.
  - -c to activate monitoring for the specified message flow(s)
  - -m to set name of monitoring profile to use for the message flow(s)
- Extra flags –s and -i
  - enable and disable individual event sources in a message flow
  - Multiple event sources can be modified in a single command invocation
  - No need to edit message flow and redeploy

Command Line: mqsireportflowmonitoring

- v6.1.0.2 usage still supported.
  - Reports whether monitoring is active, and name of monitoring profile
- Extra –n flag
  - report all configured event sources for a single message flow
- Extra –a flag
  - report all available event sources in a single message flow
- Extra –x –p <path> flags
  - Export the current monitoring properties as a monitoring profile.
  - If monitoring profile is in use, registry contents are written to file
  - If node properties are in use, XML is constructed from them
  - **Tip: Use this to easily construct a monitoring profile, rather than hand-crafting it in a schema editor.**

Example output from mqsireportflowmonitoring

Example output from mqsireportflowmonitoring command with –n option

```
BIP8911I: Monitoring settings for flow 'TotalPurchaseOrderFlow'
in execution group 'EventsEmitter.1' -  State?: active, ProfileName: ''.
BIP8912I: Event: 'InputOrder.transaction.Start',  Event name: 'InputOrder.Trans
ctionStart',  Configured?: yes,  State?: enabled
BIP8912I: Event: 'InputOrder.transaction.End',  Event name: 'InputOrder.Transac
ionEnd',  Configured?: yes,  State?: enabled
BIP8912I: Event: 'InputOrder.transaction.Rollback',  Event name: 'InputOrder.Tr
nsactionRollback',  Configured?: yes,  State?: enabled
BIP8912I: Event: 'GoldOrderTotal.terminal.in',  Event name: 'GoldOrderTotal.InT
rminal',  Configured?: yes,  State?: enabled
BIP8912I: Event: 'RegularOrderTotal.terminal.in',  Event name: 'RegularOrderTot
l.InTerminal',  Configured?: yes,  State?: enabled

BIP8071I: Successful command completion.
```

Notes

- Refer back to first screenshot
  - `mqsireportflowmonitoring` with –n option is equivalent to selecting the message flow canvas.
    - Both will list the configured event sources
  - `mqsireportflowmonitoring` with –a option is useful when you need to discover the *event source addresses* of the available event sources.
    - So that you can write a monitoring profile which configures them
    - …but there's a better way
  - `mqsireportflowmonitoring` with –x –p allows you to avoid hand-crafting the monitoring profile.

Diagnosing problems

- Basic diagnosis
  - Check that monitoring is active for the message flow itself
  - Issue mqsireportflowmonitoring with –n option to see the list of active event sources.
  - Take a user trace, and look for BIP3912 which is logged every time a message flow emits an event.

More information

- Product documentation
  - http://publib.boulder.ibm.com/infocenter/wmbhelp/v6r1m0/topic/com.ibm.etools.mft.doc/ac37850_.htm

- Sample supplied with Message Broker Toolkit
  - End to end scenario with KPIs generated in WBM.

Summary

- Message Broker now has built-in support for Business Activity Monitoring
  - Designed for WebSphere Business Monitor / CBE integration
  - Highly configurable
  - Can be administered without the toolkit
  - Also supports audit, capture/replay scenarios

- V7 support includes
  - Tighter integration with WBM
  - More features

Copyright and Trademarks