

S H A R E

Technology • Connections • Results

Message Broker Monitoring, Auditing and Statistics

David Gorman (gormand@uk.ibm.com)
IBM Hursley

1st March 2011



Agenda



- Monitoring and Auditing
- Accounting and Statistics
- Resource Statistics

What is Business Activity Monitoring?



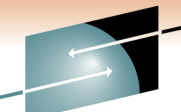
- Ability to monitor your business performance
 - Giving a real-time view of what is happening in your business
 - Identifying problems in your business processes
 - Opportunity to improve business processes and business competitiveness
- A way of making the business more transparent
 - Allows evidence-based decision making
 - “X-Ray for business processes”

Business Activity Monitoring

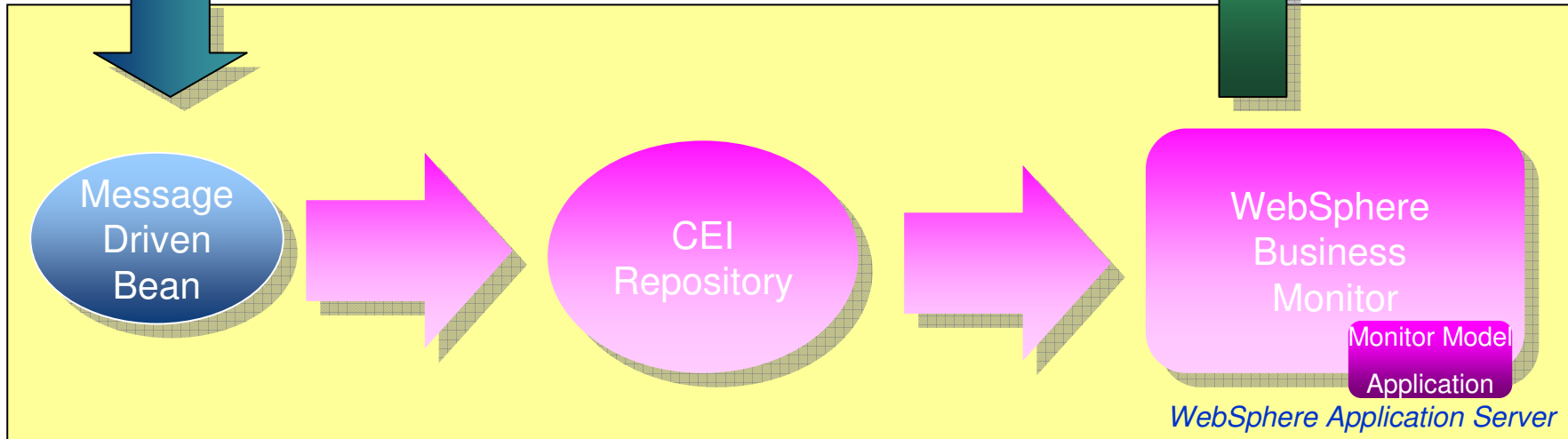
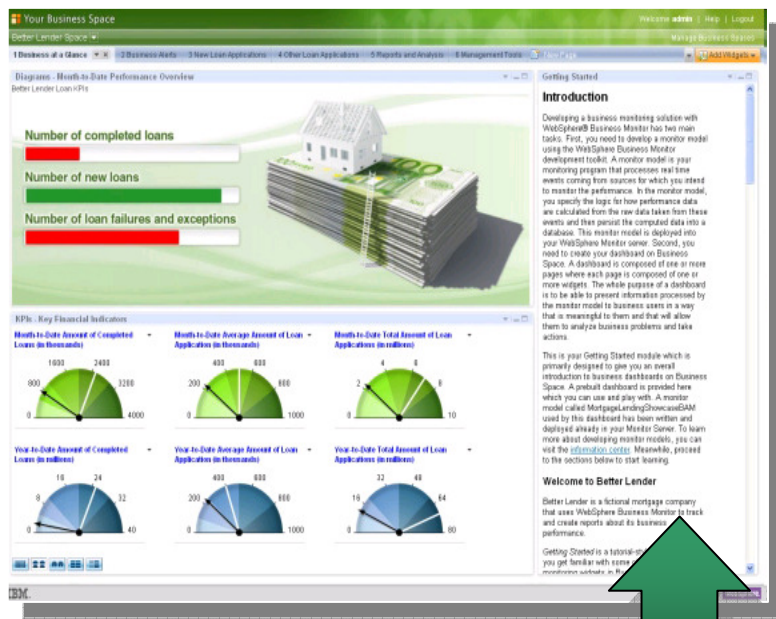
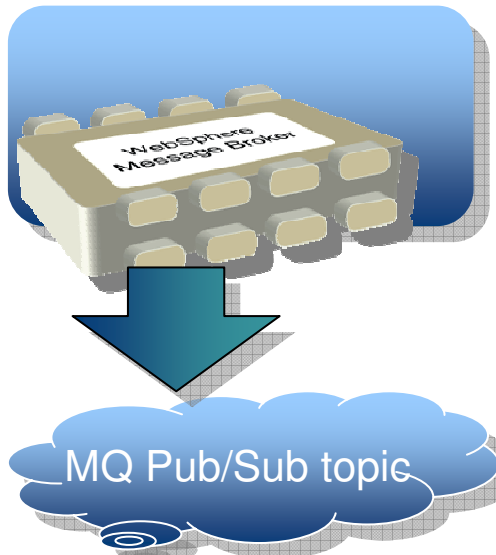


- Why?
 - Decision-makers need Key Performance Indicators (KPIs)
- Where?
 - Best source of KPIs are the applications which run the business
 - The ESB has visibility of all of those applications
- How?
 - One way is to configure the ESB to produce monitoring events
 - Send the monitoring events to a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

Typical BAM scenario with Message Broker



SHARE
Technology • Connections • Results



in Anaheim
2011

Notes



N
O
T
E
S

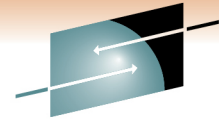
- The message-driven bean is hosted within WAS.
 - So the WAS instance must have an MQ queue manager
- WebSphere Business Monitor can only receive events from the CEI component.
 - it would be difficult for WebSphere Message Broker to submit events directly to CEI
 - Better to publish in a flexible XML 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



SHARE
Technology • Connections • Results

N
O
T
E
S

- 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



N
O
T
E
S

- 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



N
O
T
E
S

- 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.

N
O
T
E
S

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

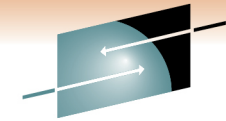
Notes



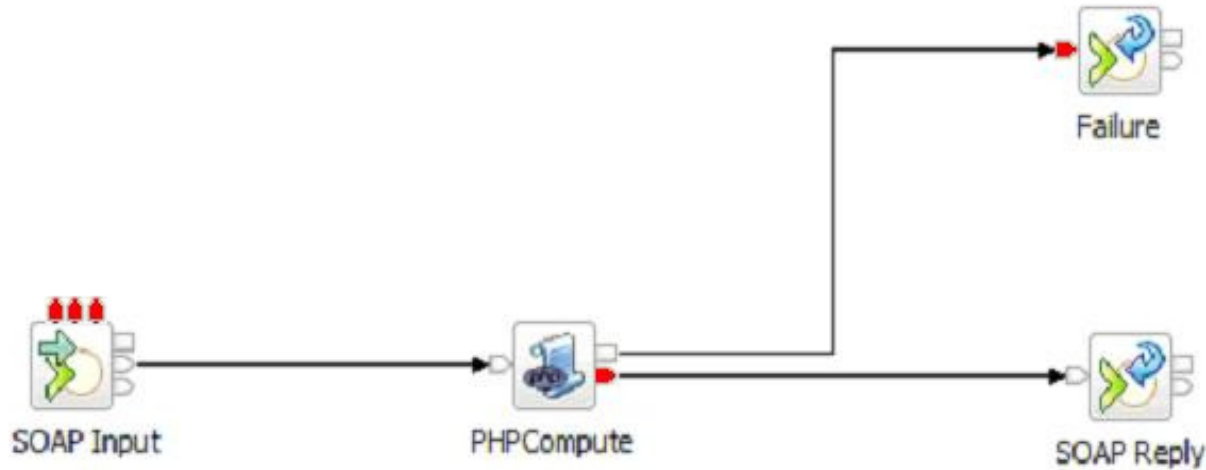
N
O
T
E
S

- 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



SHARE
Technology • Connections • Results



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
<input checked="" type="checkbox"/>	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()
<input checked="" type="checkbox"/>	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction start	SOAP Input.transaction.Start	SOAP Input.TransactionStart	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction end	SOAP Input.transaction.End	SOAP Input.TransactionEnd	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction rollback	SOAP Input.transaction.Rollback	SOAP Input.TransactionRollback	true()

Enable All
Disable All

Note: terminals are coloured 'red' on this slide to highlight those terminals with monitoring events defined.

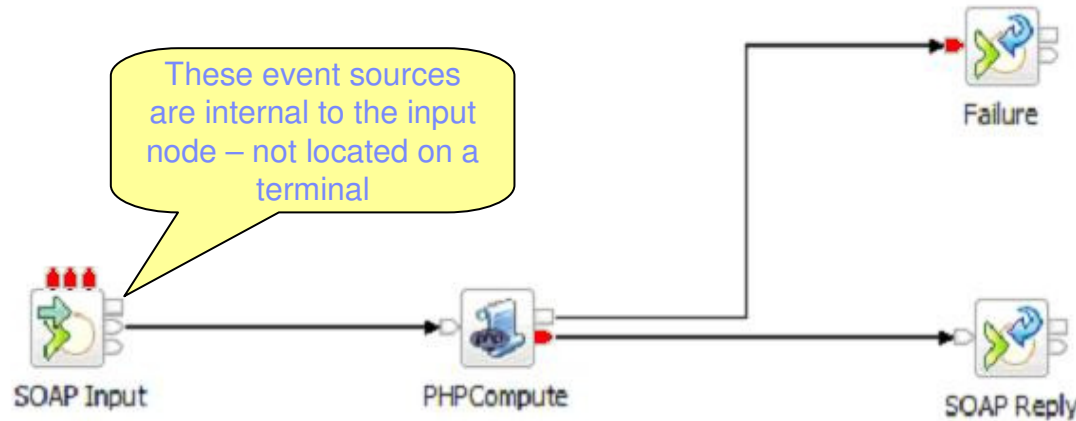
Notes



- 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!

N
O
T
E
S

Monitoring support 2



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.

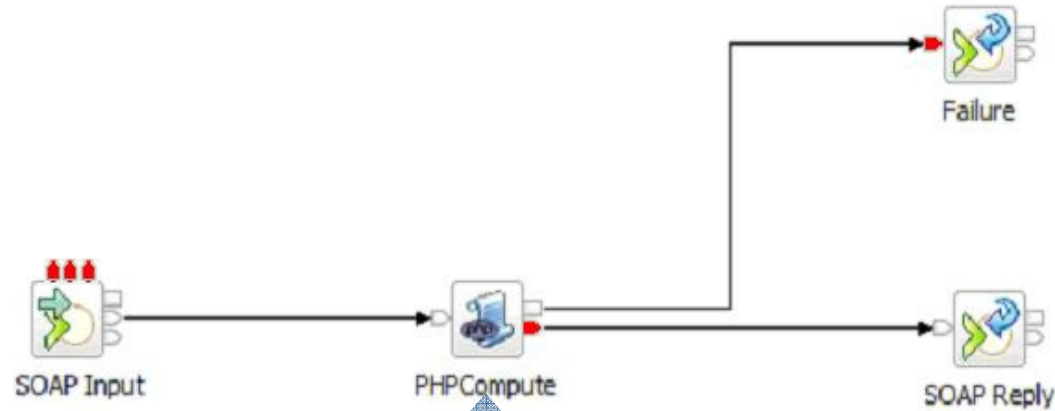
Enabled	Node	Event Source	Event Source Address	Event Name	Event Filter
<input checked="" type="checkbox"/>	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()
<input checked="" type="checkbox"/>	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction start	SOAP Input.transactionStart	SOAP Input.TransactionStart	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction end	SOAP Input.transactionEnd	SOAP Input.TransactionEnd	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction rollback	SOAP Input.transactionRollback	SOAP Input.TransactionRollback	true()

Enable All

Disable All

Note: terminals are coloured 'red' on this slide to highlight those terminals with monitoring events defined.

Monitoring support 3



Default Values for Message Flow Properties - MessageProcessor

Monitoring

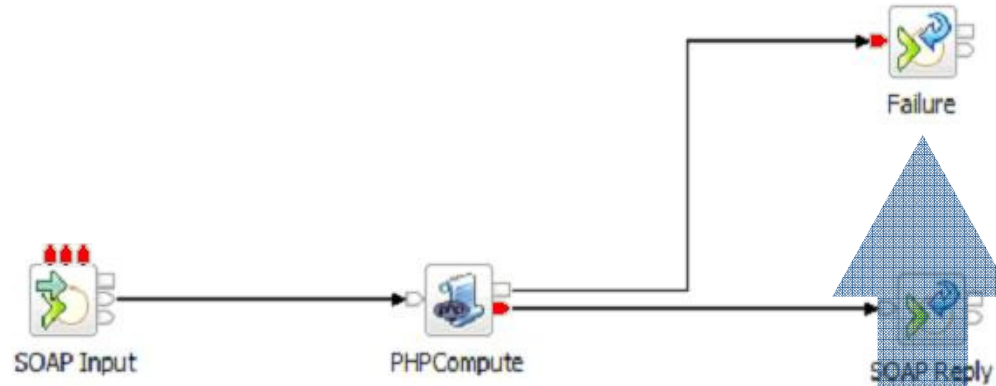
Events

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	
<input checked="" type="checkbox"/>	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()	Enable All
<input checked="" type="checkbox"/>	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()	Disable All
<input checked="" type="checkbox"/>	SOAP Input	Transaction start	SOAP Input.transaction.Start	SOAP Input.TransactionStart	true()	
<input checked="" type="checkbox"/>	SOAP Input	Transaction end	SOAP Input.transaction.End	SOAP Input.TransactionEnd	true()	
<input checked="" type="checkbox"/>	SOAP Input	Transaction rollback	SOAP Input.transaction.Rollback	SOAP Input.TransactionRollback	true()	

Note: terminals are coloured 'red' on this slide to highlight those terminals with monitoring events defined.

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.

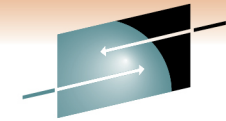
Enabled	Node	Event Source	Event Source Address	Event Name	Event Filter
<input checked="" type="checkbox"/>	Failure	In terminal	Failure.terminal.in	Failure.InTerminal	true()
<input checked="" type="checkbox"/>	PHPCompute	Out terminal	PHPCompute.terminal.out	PHPCompute.OutTerminal	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction start	SOAP Input.transaction.Start	SOAP Input.TransactionStart	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction end	SOAP Input.transaction.End	SOAP Input.TransactionEnd	true()
<input checked="" type="checkbox"/>	SOAP Input	Transaction rollback	SOAP Input.transaction.Rollback	SOAP Input.TransactionRollback	true()

Enable All

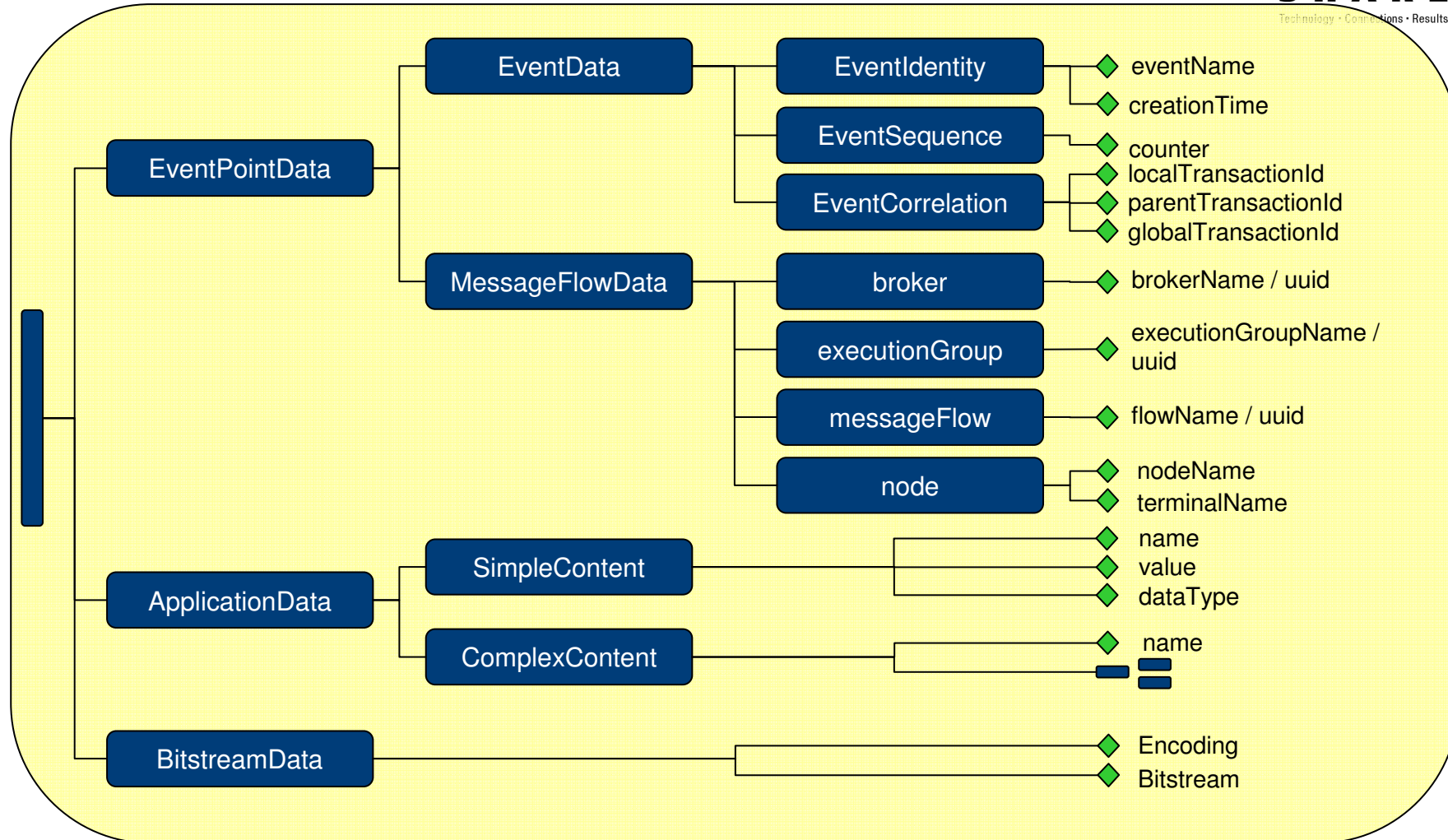
Disable All

Note: terminals are coloured 'red' on this slide to highlight those terminals with monitoring events defined.

Monitoring event format details



SHARE
Technology • Connections • Results



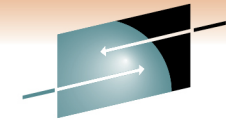
Notes



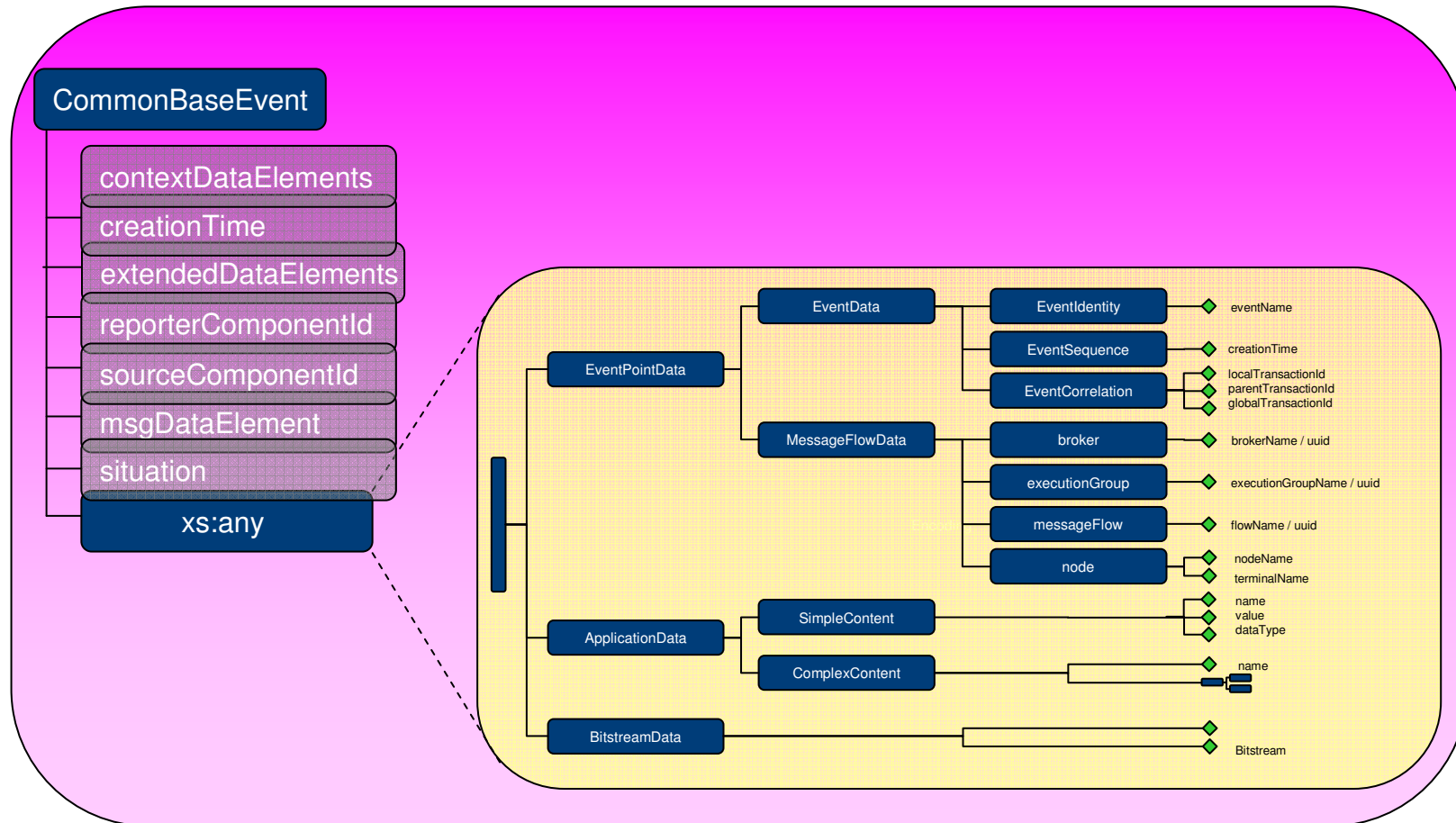
N
O
T
E
S

- 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



SHARE
Technology • Connections • Results



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

N
O
T
E
S

Monitoring events from message broker



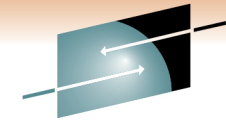
- Good interoperability with WebSphere Business Monitor
 - Identity, correlation and sequencing of events is explicitly built into the event format
- Flexibility
 - Events can be consumed by one or more clients subscribing to the appropriate topic. Topic can include wildcards, allowing the scope of the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



SHARE
Technology • Connections • Results

The screenshot shows the 'Add event' dialog box with three tabs: 'Basic', 'Correlation', and 'Transaction'. The 'Basic' tab is active. It contains three sections: 'Event Source', 'Event Name', and 'Event Filter'. A yellow callout bubble points to the 'Event Source' dropdown menu, which is open and showing options: 'Transaction start', 'Transaction end', 'Transaction rollback', 'Failure terminal', and 'Out terminal'. The 'Event Name' section has 'Literal' selected with the text 'SOAP Input1.TransactionStart'. The 'Event Filter' section has the text 'true()' entered.

Event Source
Select the source of the event.

Transaction start
Transaction end
Transaction rollback
Failure terminal
Out terminal

Select the event source. Input nodes include the special 'transaction' event sources

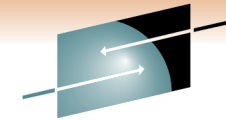
Event Name
Provide the name by which events emitted from this source are to be known. Specify either a literal name, or the location of a character field in the message tree or elsewhere in the message assembly.

Literal SOAP Input1.TransactionStart
 Data location Edit...

Event Filter
Provide an expression to control whether the event is emitted. The expression must evaluate to true or false, and can reference fields in the message tree or elsewhere in the message assembly. If you do not specify a value, the value true() is used.

true() Edit...

Configuring: Customizing an event



SHARE
Technology • Connections • Results

Add event

Basic | Correlation | Transaction

Event Source
Select the source of the event.
Transaction start

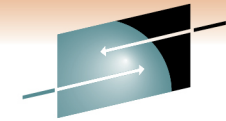
Event Source Address
The broker identifies an event source using an event source address. Use this value when you enable and disable event sources using runtime commands.
SOAP Input1.transaction.Start

Event Name
Provide the name by which events emitted from this source are to be known. Specify either a literal name, or the location of a character field in the message tree or elsewhere in the message assembly.
 Literal SOAP Input1.TransactionStart
 Data location Edit...

Event Filter
Provide an expression to control whether the event is emitted. The expression must evaluate to true or false, and can reference fields in the message tree or elsewhere in the message assembly. If you do not specify a value, the value true() is used.
true() Edit...

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



SHARE
Technology · Connections · Results

Add event

Basic | Correlation | Transaction

Event Source
Select the source of the event.
Transaction start

Event Source Address
The broker identifies an event source using an event source address. Use this value when you enable and disable event sources using runtime commands.
SOAP Input1.transaction.Start

Event Name
Provide the name by which events emitted from this source are to be known. Specify either a literal name, or the location of a character field in the message tree or elsewhere in the message assembly.
 Literal SOAP Input1.TransactionStart
 Data location Edit...

Event Filter
Provide an expression to control whether the event is emitted. The expression must evaluate to true or false, and can reference fields in the message tree or elsewhere in the message assembly. If you do not specify a value, the value true() is used.
true() Edit...

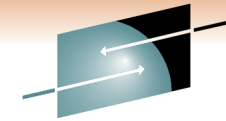
XPATH expression to indicate if this event should be emitted or not

Configuring: Adding a filter to an event (2)



N
O
T
E
S

- 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



S H A R E

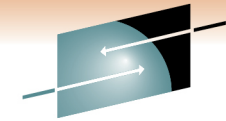
Technology • Connections • Results

Notes:

N
O
T
E
S

- 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



SHARE
Technology • Connections • Results

The screenshot shows the 'Add event' configuration window with the 'Correlation' tab selected. The window title is 'Add event'. The 'Correlation' tab is active, showing the 'Event Correlation' section. The text explains that event correlators match events from the same or related business transactions. It defines three types: local transaction correlator (links events from a single message flow invocation), parent transaction correlator (links events from a message flow to a parent message flow or external application), and global transaction correlator (links events from a message flow to one or more related message flows or external applications). Below this, the 'Local transaction correlator' section has two radio buttons: 'Automatic' (selected) and 'Specify location of correlator'. A 'Description' text area contains the text: 'The local correlator used by the most recent event for this message flow invocation will be used. If no local correlator exists yet, a new unique value will be generated.' Below this, the 'Parent transaction correlator' section also has two radio buttons: 'Automatic' (selected) and 'Specify location of correlator'. A yellow callout bubble points to the 'Specify location of correlator' radio button for the parent transaction correlator, containing the text: 'Use this option when your events must be correlated with events from an external process'.

Event Correlation

A monitoring application uses event correlators to match events emitted by the same, or related, business transactions. A local transaction correlator links the events emitted by a single invocation of a message flow. A parent transaction correlator links the events from a message flow to a parent message flow or an external application. A global transaction correlator links events from a message flow to one or more related message flows or external applications. An event must contain a local transaction correlator, but need not contain a parent transaction correlator or global transaction correlator.

Local transaction correlator:

Automatic Specify location of correlator

Description

The local correlator used by the most recent event for this message flow invocation will be used. If no local correlator exists yet, a new unique value will be generated.

Parent transaction correlator:

Automatic Specify location of correlator

Use this option when your events must be correlated with events from an external process

Correlators



N
O
T
E
S

- 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.

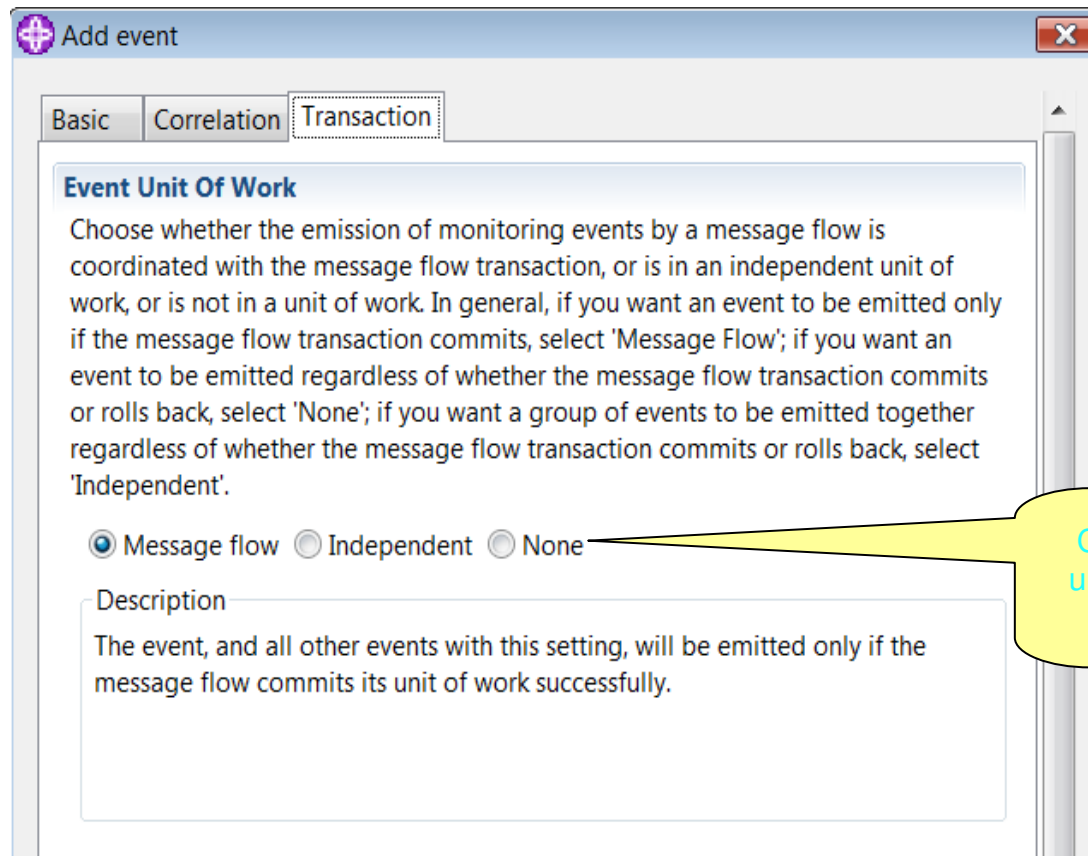
Notes



- 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.

N
O
T
E
S

Configuring: Customizing an event - Transaction



The screenshot shows a software window titled "Add event" with three tabs: "Basic", "Correlation", and "Transaction". The "Transaction" tab is active. It contains a section titled "Event Unit Of Work" with a text block explaining the options. Below the text are three radio buttons: "Message flow" (selected), "Independent", and "None". A yellow callout bubble points to the "Message flow" radio button.

Event Unit Of Work

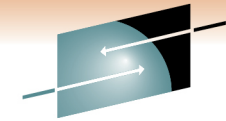
Choose whether the emission of monitoring events by a message flow is coordinated with the message flow transaction, or is in an independent unit of work, or is not in a unit of work. In general, if you want an event to be emitted only if the message flow transaction commits, select 'Message Flow'; if you want an event to be emitted regardless of whether the message flow transaction commits or rolls back, select 'None'; if you want a group of events to be emitted together regardless of whether the message flow transaction commits or rolls back, select 'Independent'.

Message flow Independent None

Description

The event, and all other events with this setting, will be emitted only if the message flow commits its unit of work successfully.

Choose the transaction under which the event is emitted



S H A R E
Technology • Connections • Results

Notes:

N
O
T
E
S

- 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

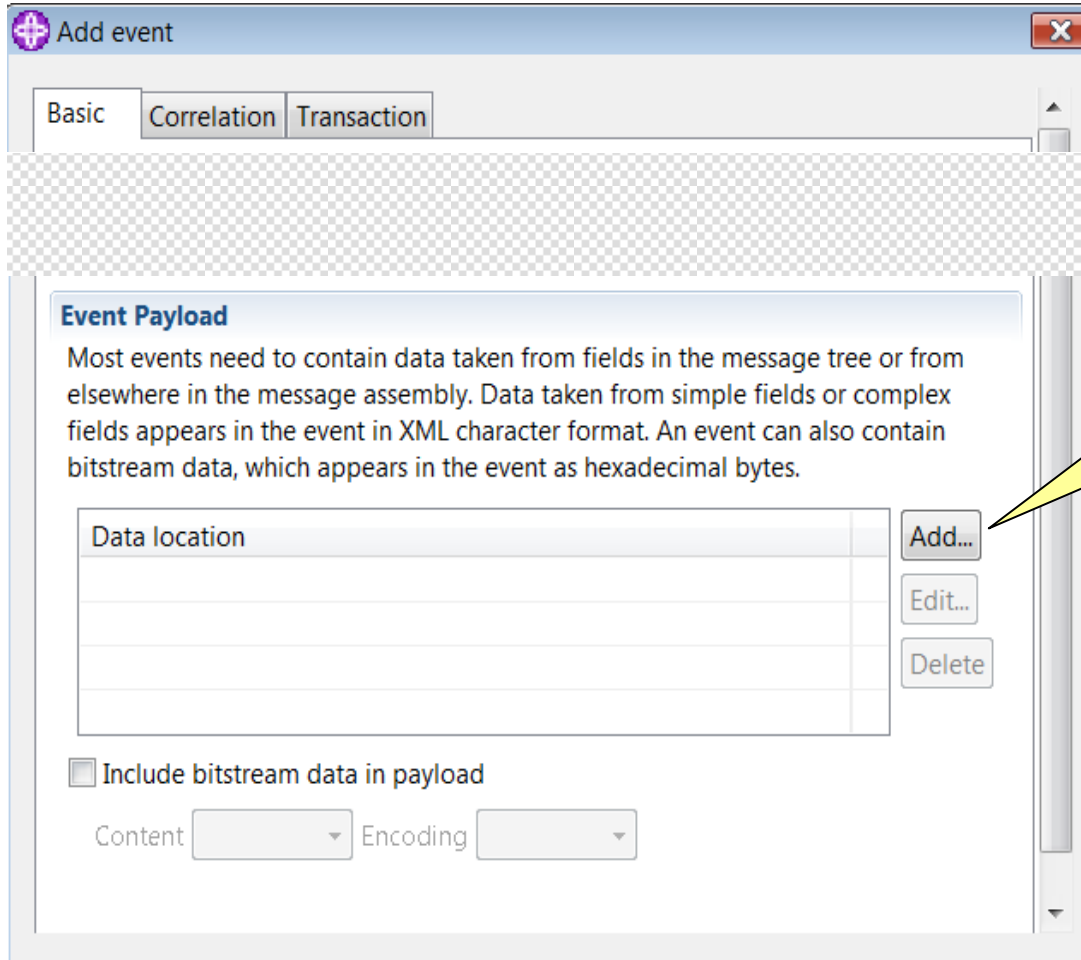
Notes 2:



N
O
T
E
S

- 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



Add event

Basic Correlation Transaction

Event Payload

Most events need to contain data taken from fields in the message tree or from elsewhere in the message assembly. Data taken from simple fields or complex fields appears in the event in XML character format. An event can also contain bitstream data, which appears in the event as hexadecimal bytes.

Data location

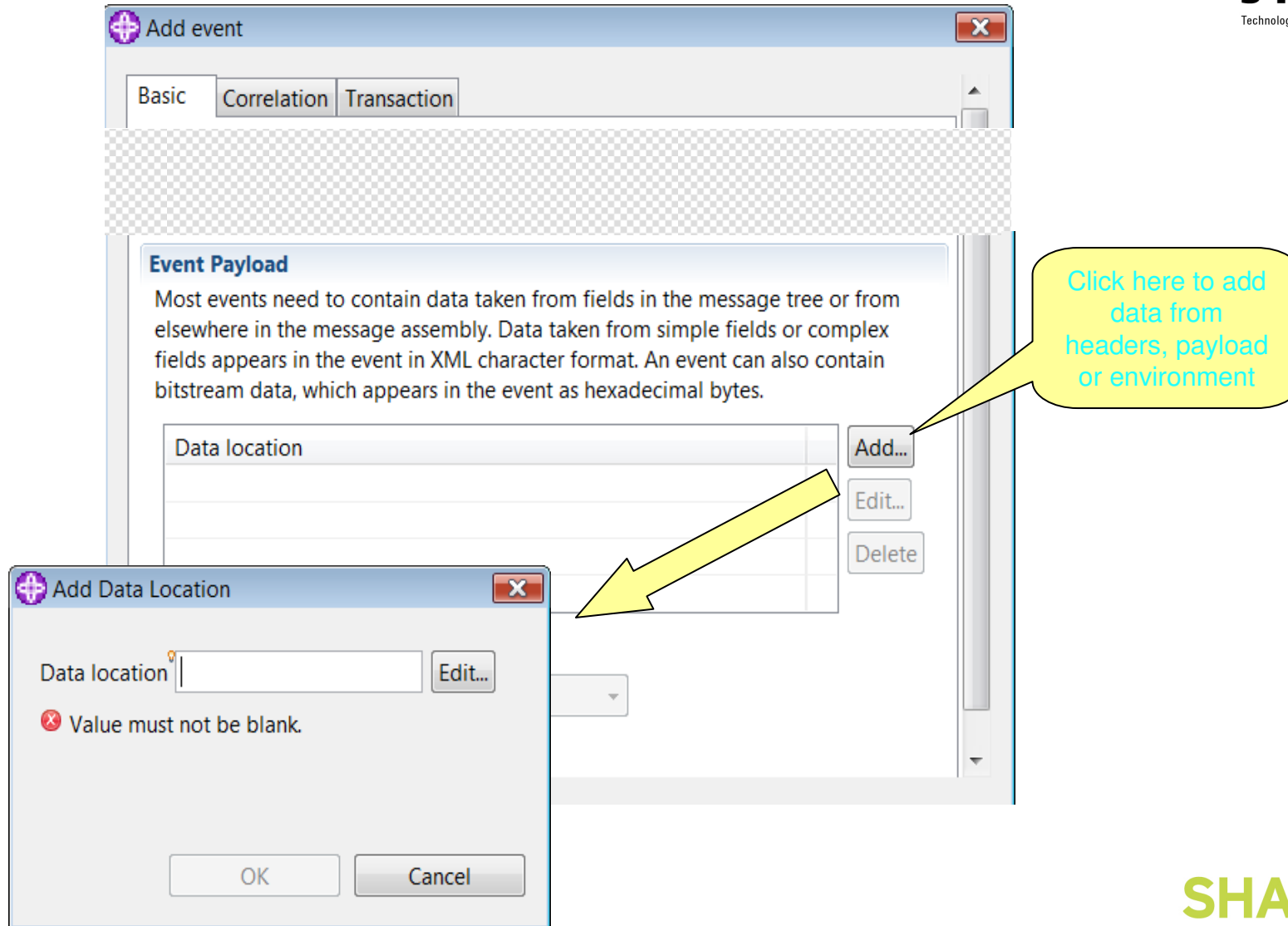
Add... Edit... Delete

Include bitstream data in payload

Content Encoding

Click here to add data from headers, payload or environment

Customizing an event – payload data



Add event

Basic Correlation Transaction

Event Payload

Most events need to contain data taken from fields in the message tree or from elsewhere in the message assembly. Data taken from simple fields or complex fields appears in the event in XML character format. An event can also contain bitstream data, which appears in the event as hexadecimal bytes.

Data location

Add... Edit... Delete

Click here to add data from headers, payload or environment

Add Data Location

Data location Edit...

Value must not be blank.

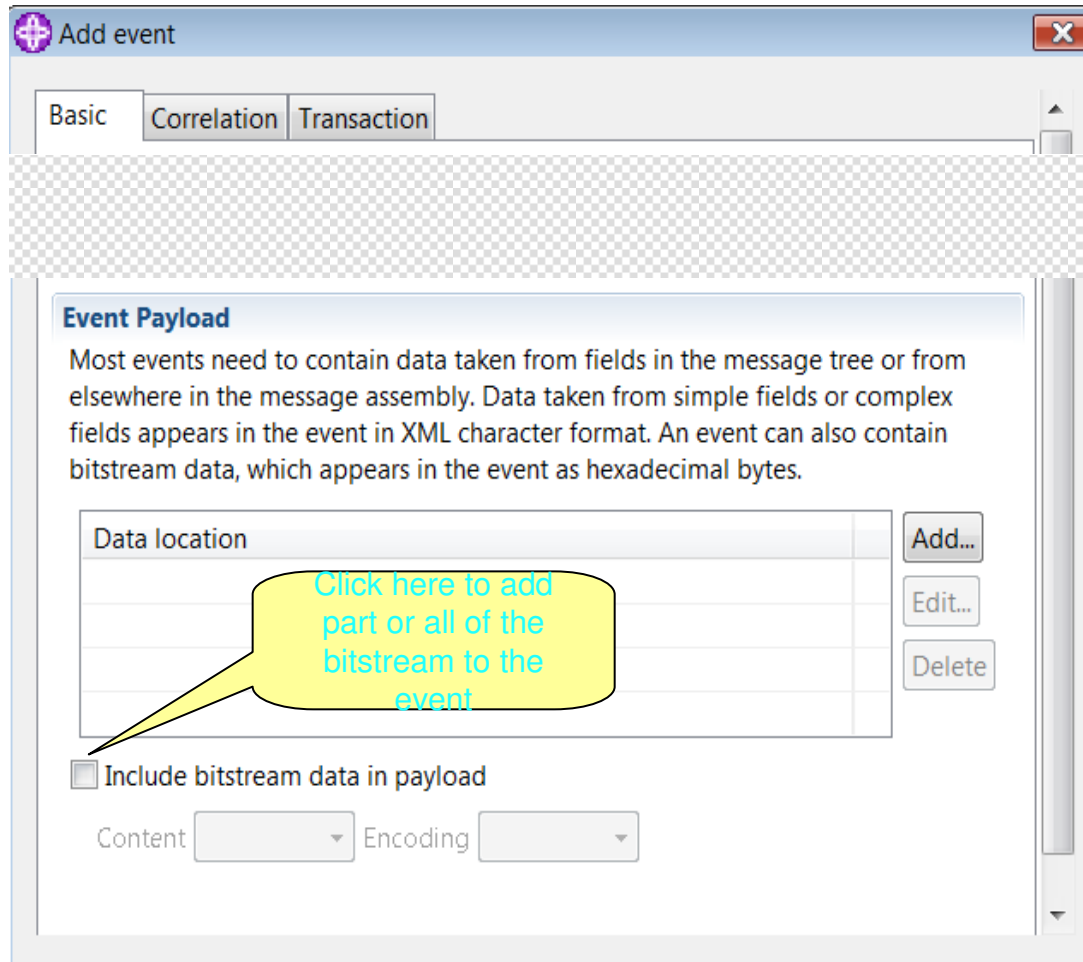
OK Cancel

Payload data

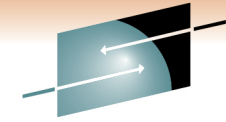


- Multiple XPath queries can be specified
 - Or ESQL paths; the support for both is generic
 - XPath builder provides assistance with constructing the path
- Simple fields automatically go into applicationData/simpleContent
 - If monitoring profile is used, the @dataType attribute can be set for each 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



SHARE
Technology • Connections • Results

Add event

Basic Correlation Transaction

Event Payload

Most events need to contain data taken from fields in the message tree or from elsewhere in the message assembly. Data taken from simple fields or complex fields appears in the event in XML character format. An event can also contain bitstream data, which appears in the event as hexadecimal bytes.

Data location	

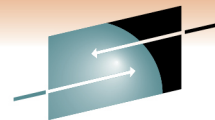
Add...
Edit...
Delete

Include bitstream data in payload

Content: All (dropdown menu open showing: Headers, Body, All)
Encoding: base64Bir (dropdown menu)

Include headers, body or entire bitstream

Customizing an event – bitstream data



SHARE
Technology • Connections • Results

Add event

Basic Correlation Transaction

Event Payload

Most events need to contain data taken from fields in the message tree or from elsewhere in the message assembly. Data taken from simple fields or complex fields appears in the event in XML character format. An event can also contain bitstream data, which appears in the event as hexadecimal bytes.

Data location

Add...
Edit...
Delete

Include bitstream data in payload

Content All Encoding base64Bir

- CDATA
- hexBinary
- base64Binary

Click here to add part or all of the bitstream to the event

Encode bitstream as hexBinary, base64 or CDATA

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'

N
O
T
E
S

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

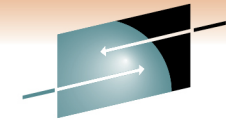
N
O
T
E
S

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

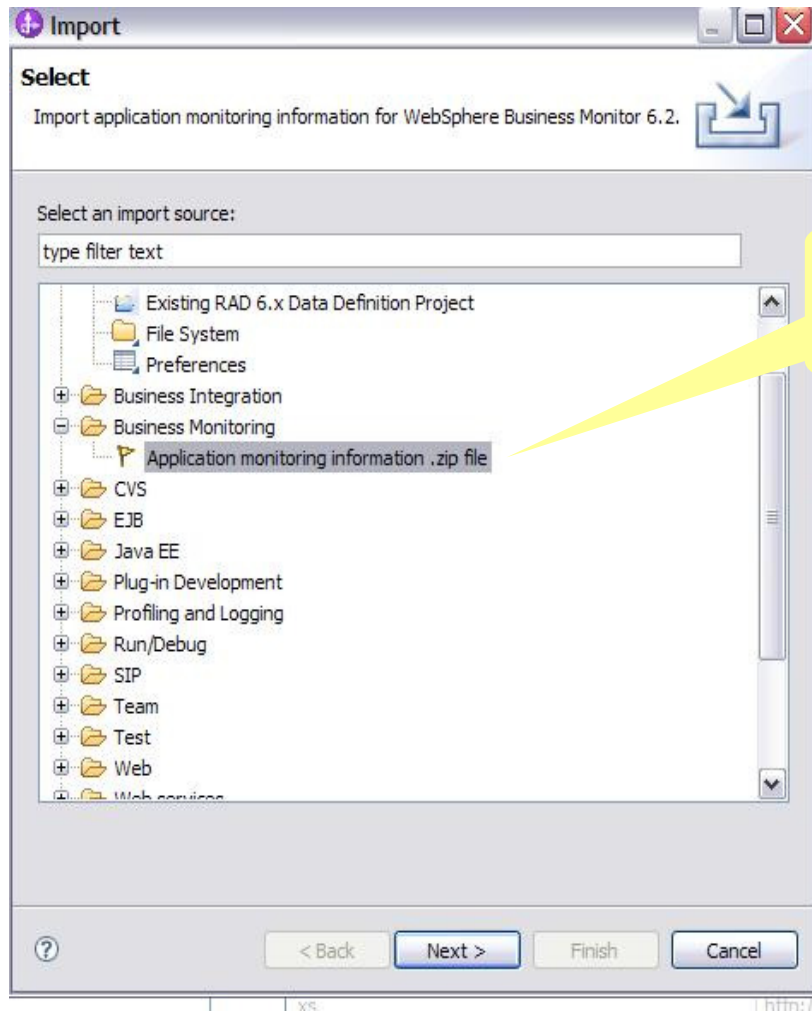
Exporting Message Flow monitoring information



SHARE
Technology • Connections • Results

The screenshot displays the IBM Business Monitoring interface. On the left, the 'Resources' tree shows a project named 'WBMonitorEventsProject' containing several artifacts, including 'TotalPurchaseOrderFlow.msgflow'. On the right, the 'Export' dialog is open, showing a list of export destinations. The 'Business Monitoring' folder is expanded, and 'Application Monitoring Information' is selected. A yellow callout bubble points to this selection with the text: "Select Application Monitoring Information from the Business Monitoring folder".

Import Monitoring Information into WebSphere Business Monitor Toolkit

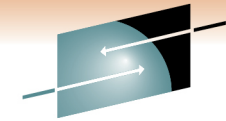


Generate Monitor Model (multi-step process)



The screenshot shows the 'Generate Monitor Model' wizard in a multi-step process. The current step is 'Monitoring Templates'. On the left, under 'Event Source', a tree view shows 'TotalPurchaseOrderFlow Application' expanded to 'TotalPurchaseOrderFlow', which includes 'GoldOrderTotal' and 'RegularOrderTotal'. A yellow callout bubble points to 'TotalPurchaseOrderFlow' with the text 'Select message flow to choose Templates from Monitoring Templates page'. On the right, the 'Monitoring Templates' tab is active, showing a list of templates: 'Average Transaction Duration', 'Number of Failed Transactions', and 'Message Flow Correlation'. Each has a checked checkbox. Below the list are 'Select All' and 'Clear' buttons. A 'Description of selected template' field contains the text 'Select a template to see the description.'. At the bottom, there is a checkbox for 'Limit my selection of events and templates based on the events that have been turned on in the application' and navigation buttons: '< Back', 'Next >', 'Finish', and 'Cancel'.

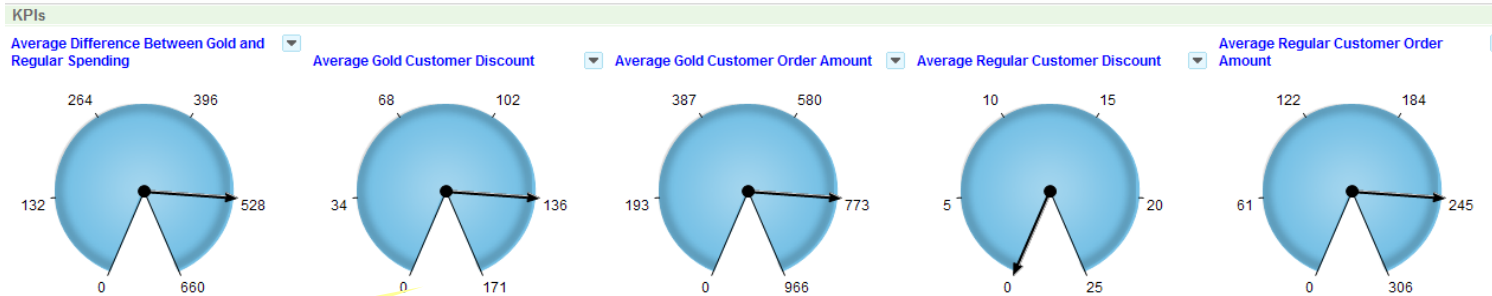
Business Space Manager



SHARE

Technology • Connections • Results

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	11112222	1
	2 September 2009 11:51:12	111111	GOLD	163.44	WBMonitorEventsExecGroup 0		12.156 s	d4c40c0d-ccd1-4676-9c0a-f63cceb3cb36-1	TotalPurchaseOrderFlow 11112222	11112222	1
	2 September 2009 11:58:14	222222	GOLD	58.86	WBMonitorEventsExecGroup 0		0.453 s	d4c40c0d-ccd1-4676-9c0a-f63cceb3cb36-2	TotalPurchaseOrderFlow 22223333	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	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	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	22223333	1



Dashboard populated with data from Message Broker events

Command Line: mqsichange flow monitoring



- 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

N
O
T
E
S

Command Line: mqsireportflowmonitoring



N
O
T
E
S

- 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.TransactionStart', Configured?: yes, State?: enabled
BIP8912I: Event: 'InputOrder.transaction.End', Event name: 'InputOrder.TransactionEnd', Configured?: yes, State?: enabled
BIP8912I: Event: 'InputOrder.transaction.Rollback', Event name: 'InputOrder.TransactionRollback', Configured?: yes, State?: enabled
BIP8912I: Event: 'GoldOrderTotal.terminal.in', Event name: 'GoldOrderTotal.InTerminal', Configured?: yes, State?: enabled
BIP8912I: Event: 'RegularOrderTotal.terminal.in', Event name: 'RegularOrderTotal.InTerminal', Configured?: yes, State?: enabled

BIP8071I: Successful command completion.
```

N
O
T
E
S

Notes



N
O
T
E
S

- 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



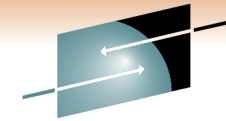
- Basic diagnosis
 - Check that monitoring is active for the message flow itself
 - Issue `mqsiereportflowmonitoring` with `-n` option to see the list of active event sources.
 - Take a user trace, and look for BIP3912 which is logged every time a message flow emits an event.

N
O
T
E
S

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.



S H A R E

Technology • Connections • Results

Message Flow accounting and statistics

“Message flow accounting and statistics data is the information that can be collected by a broker to record performance and operating details of message flow execution.”

SHARE
in Anaheim
2011

Areas of Interest

Broker

Execution group (process)

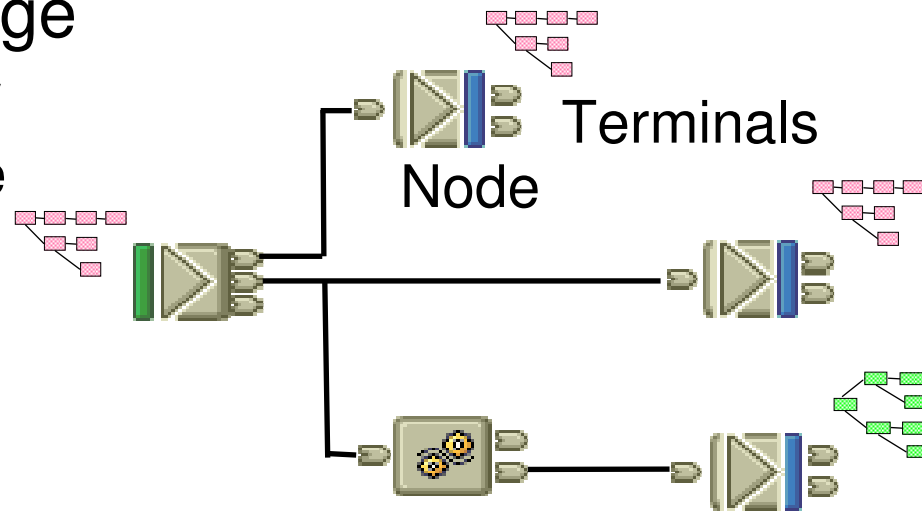
Thread

Message

Flow

Message

Model



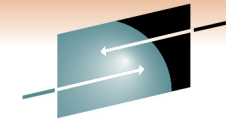
Notes



N
O
T
E
S

- Before we understand Accounting and Statistics, let's just recall the three most important concepts in the broker.
 - Message Flows: Flows represents the message routing and transformations required to join together applications through a broker. Each flow takes an input message and (usually) generates one or more output messages, potentially in different physical formats for connected applications to process. In the example shown, a message flow might take an MQ message containing XML, and make a copy that's unchanged for an XML literate server, and another copy that's transformed to a record based format to drive an existing CICS application. Note that the message flow can also deal with error conditions, and drive appropriate processing in these cases. Message flows can be made multithreaded using the "additionalInstances" flow attribute.
 - Message Nodes: Nodes represents the individual operations of which a flow is comprised. The broker supplies nodes to get and put messages to queues, perform database queries and updates and so on. Each node takes an input message and usually generates one or more output messages. The result of the processing occurring in each node determines which set of outputs ("terminals") are driven and therefore which connected nodes subsequently receive control.
 - Message Modelling: As messages pass through a broker, either being read from an input, or written to an output, they are transformed between their physical or "wire" format (XML, Record, Tagged) and a tree representation. Within broker nodes, ESQL is used as a single, standard language to perform message and database manipulation. As messages and parsing are at the core of the implementation it is important to gather information on their processing.

What Data is Collected?



S H A R E

Technology • Connections • Results

Message Flow

- ExecutionGroup name
- Broker name
- Flow name
- Sample time, start, end
- "Archive" or "Snapshot"
- Msg: Size(Total,Min,Max), number processed, bytes processed
- Times(Total,Min,Max): Elapsed, CPU
- Time: Waiting for input
- Threads: Total, high water count
- Commit/backout count
- Error counts

Threads

- Arbitrary thread number
- Msg: Total processed
- Total Times: Elapsed, CPU, Wait, Input Wait
- Msg: Min, Max input size

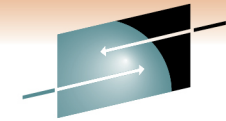
Nodes

- Label name
- Type
- Times(Total,Min,Max): Elapsed, CPU,
- Invocation count
- Msg: Size(Total,Min,Max), number input messages processed
- Bytes processed

+ Terminals

- Labels and Counts

Notes



S H A R E

Technology • Connections • Results

● Four Distinct Types of Data

- These correspond to Flows, Nodes, Terminals and Threads. We now give a brief description of the data collected. If you're familiar with the subject, there are few surprises here, but if this area is new, you should be pleasantly surprised with this information!

● Message Flow Statistics

- ExecutionGroup name, Broker name and Flow name are self explanatory. You also receive the flow labels and UUIDs.
- Sample time, start, end: The time range for the sample collected.
- "Archive" or "Snapshot": Whether the interval was an "Archive" or "Snapshot" interval. Note the difference - you don't want to double count these records, if you're collecting both records, e.g. diagnosing a problem via snapshot when archive is active.
- Msg: Size(Total,Minimum,Maximum), number processed, bytes processed: The number of messages and their total, minimum, maximum size. Also note the total number of messages and bytes processed in the interval by the flow.
- Times(Total,Minimum,Maximum): Elapsed, CPU, Wait: These show how much CPU and elapsed time the flow used to process the messages. For cases where I/O is an important factor the wait times may prove more interesting.
- Time: Waiting for input: How long the flow was not processing messages in the interval. Gives you an idea of the level of activity.
- Threads: Total, high water count: How many threads the flow has assigned to it, and the maximum in use at any given time. You might like to use the thread statistics to determine whether "additionalInstances" is set appropriately.
- Commit/backout count: How many messages were backed out and committed. Useful check messages processed as expected.

● Thread Statistics

- Arbitrary thread number: This has no significance other than identification. One record is produced per thread.
- Msg: Total processed: The number of messages processed by the thread in the interval.
- Total Times: Elapsed, CPU, Wait, Input Wait:: Processing times similar to those for flows, but we can see how effective the threads are "supporting" the flow's processing.
- Msg: Min, Max input size: Various statistical sizes of messages processed by the thread.

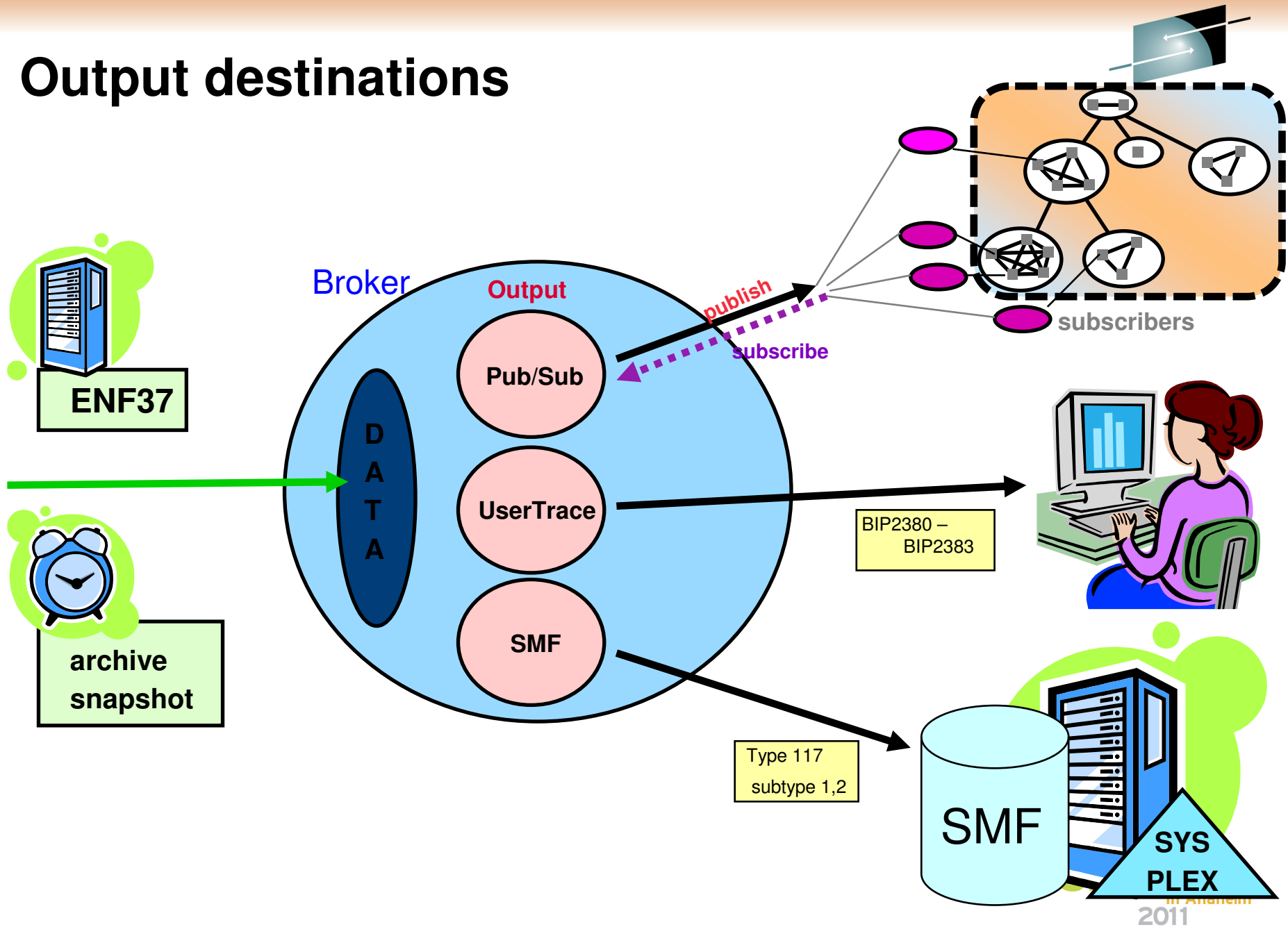
● Node Statistics

- Label name and Type: to identify the node. One record is produced per node.
 - Times(Total,Minimum,Maximum): Elapsed, CPU, Wait: Standard statistical times to give node level granularity of processing. Allows isolation of hot spots, or charging according to specific processing.
 - Invocation count: How many times the node has been traversed. This is useful to understand the critical path in a flow.
 - Msg: Size(Total,Minimum,Maximum), number processed, bytes processed: Various statistics on of msgs processed by the thread.
- +Terminals. One record is produced per terminal.

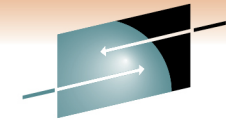
- Labels of each terminal within a node and count of the number of times traversed. Identify critical path and exceptions!

SHARE
in Anaheim
2011

Output destinations



Notes



S H A R E

Technology • Connections • Results

N
O
T
E
S

- Destinations and Collection Scope

- This foil identifies the main areas for consideration when examining accounting and statistics data
 - ◆When are the data produced?
 - ◆In what format are the data?
 - ◆How are the data accessed?
 - ◆What is the granularity and scope of the information that can be seen?

- Accounting and Statistics Timers

- Information is gathered at regular intervals according to timers. There are two classes of timers, internal and external.
- Archive and Snapshot timers are internal timers set by broker parameters which govern when these data are written to their destinations.
- An external timer is available on z/OS, namely ENF37. This can be used to drive SMF, UserTrace and PubSub intervals. ENF is also important to allow consolidated reporting of SMF information across major subsystems, e.g. you might coordinate queue manager and broker activity to best understand how to tune your queue manager for particular flows.

- A Variety of Output Destinations and Formats

- It's possible to gather this information in different formats according to technology used to access it.
- Publish Subscribe: You can use the built-in WMQI publish/subscribe technology to report and retrieve A&S information. This is a very flexible option. Data provided over publish subscribe is provided in an XML format.
- UserTrace: This is the default mechanism which allows you to gather A&S information on the broker's file system. Data provided in UserTrace is provided as architected BIP messages, BIP2380,1,2,3.
- z/OS SMF: For z/OS, this option generates SMF type 117 records having subtypes 1 and 2 depending on the granularity of information requested by the user for a particular flow.
- You may request different output destinations for Snapshot and Archive Stats by Message Flow.

- Options for Reporting Scope and Granularity

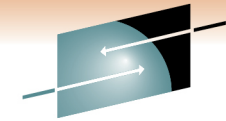
- UserTrace is collected for the broker. If several brokers are operating on a single machine, then several datasets may be accessed with ease. This data is human readable and relatively useful if you're prototyping.
- PubSub can be collected for any broker throughout the domain. This is incredibly powerful; it means you can sit anywhere in a domain and request information about particular nodes in a flow on a particular execution group on a broker on a different machine! OR you could put in wild card subscriptions to gather information from several different sources! XML is very structured, which makes it ideal for multi platform reporting.
- z/OS SMF can be collected for any broker within a SYSPLEX. It can be integrated with all other SMF enabled z/OS products and subsystems - literally hundreds!
- Moreover, the fact that each report has the details of the broker and execution group in it means that you can aggregate information as well.

- An extensible architecture into the future.

- If we require additional writer in the future (for example JMX), it is very easy to "plug in" this destination type, as the writing and collection mechanisms are separated from each other.

SHARE
in Anaheim
2011

XML Publish Subscribe Output



S H A R E

Connections • Results

```
<psc>
  <Command>Publish</Command>
  <PubOpt>RetainPub</PubOpt>
  <Topic>$$SYS/Broker/MQ02BRK/StatisticsAccounting/Archive/default/XMLflow</Topic>
</psc>

<WMQIStatisticsAccounting RecordType="Archive" RecordCode="StatsSettingsModified">

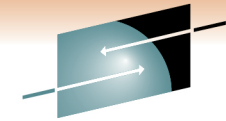
  <MessageFlow BrokerLabel="MQ02BRK" BrokerUUID="7d951e31-f200-0000-0080-efelb9d849dc"
  MessageFlowName="XMLflow" StartDate="2003-01-17" StartTime="14:44:14.550824"
  TotalNumberOfBackouts="0", AccountingOrigin="DEPT A"/>

  <Threads Number="1">
    <ThreadStatistics Number="0" TotalNumberOfInputMessages="2" TotalElapsedTime="10005200" ...
    ElapsedTimeWaitingForInputMessage="10001425" MinimumSizeOfInputMessages="0" />
  </Threads>

  <Nodes Number="3">
    <NodeStatistics Label="FAILQueue" Type="MQOutput" TotalElapsedTime="0"
    MaximumElapsedTime="" NumberOfInputTerminals="1" NumberOfOutputTerminals="2">
      <TerminalStatistics Label="failure" Type="Output" CountOfInvocations="0" />
      <TerminalStatistics Label="in" Type="Input" CountOfInvocations="0" />
      <TerminalStatistics Label="out" Type="Output" CountOfInvocations="0" />
    </NodeStatistics>...
  </Nodes>

</WMQIStatisticsAccounting>
```

Notes



S H A R E

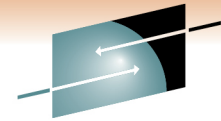
Technology • Connections • Results

N
O
T
E
S

- We have shown a very simple XML Publish Subscribe message. Note the 5 sections
 - Publish Subscribe folder in the RFH2
 - Message flow folder
 - Threads folder
 - Nodes folder: This folder may contain several terminals folders if this granularity of reporting has been requested.
 - Published messages will always include these folders, even if the appropriate data isn't currently being collected.
- SnapShot and Archive publications are available to measure the broker.
 - Both short term "SnapShot", and longer term "Archive" data messages are published. Since the SnapShot data is intended to be used for performance analysis (being sent to a real time monitor, where it is graphically displayed, for example), it is published as retained and non-persistent. Archive data is used for accounting, so an audit trail is more important - therefore it is published persistently and retained, so that late subscribers can get the most recent update. Using MQ as a publication delivery mechanism is recommended as you don't want to lose these messages if you're using them for chargeback purposes!
 - All publications are global which means they can be collected anywhere in the broker domain. And don't forget that multiple subscribers means that the data can be collected by more than one subscriber.
- Publication topics allow subscribers to retrieve a broad range of information
 - The publication tree contains the broker, execution group label and flow label, which allows subscribers anywhere in the broker domain to request a highly flexible *range* of information. This could vary from information on a specific flow running in a specific execution group (and more, see later), to all information about every flow running within the domain!
 - Snapshot publications are made according to the topic tree:
 - ♦ `$$$SYS/Broker/<brokerName>/StatisticsAccounting/SnapShot/<executionGroupLabel>/<messageFlowLabel>`
 - Archive publications are made according to the topic tree:
 - ♦ `$$$SYS/Broker/<brokerName>/StatisticsAccounting/Archive/<executionGroupLabel>/<messageFlowLabel>`
 - Hopefully you can see how easy it would be to request only publications for a particular flow. For example, what would a subscription for `$$$SYS/Broker/myBroker/StatisticsAccounting/SnapShot/+popularFlow` return? (Ans: SnapShot information on "importantFlow" in *all* execution groups on "myBroker").
 - Moreover, using ESQL filters you can further refine your subscriptions to examine the *contents* of A&S information. Subscribe for information with certain values - maybe to generate alerts when certain terminals are being driven more than you expect!
 - Note that you should be careful not to "double count" SnapShot and Archive information!
- Publish Subscribe Flexibility
 - This mechanism for collection is very granular and allows you to gather data from anywhere and everywhere inside the broker domain.

UserTrace Output

.../.../agent.userTrace.bin.0



SHARE

```
BIP2380I: WMQI message flow statistics. ProcessID='196767', Key='3',  
Type='SnapShot', Reason='Snapshot', BrokerLabel='MQ01BRK',  
BrokerUUID='a0a1a981-f000-0000-0080-9f945b3d6b5b',  
ExecutionGroupName='PubSubGrp', MaximumElapsedTime='20457211',  
MinimumElapsedTime='20457211', TotalNumberOfBackouts='0'.
```

...

```
Statistical information for message flow 'PubSubTest' in broker  
'MQ01BRK'.
```

```
This is an information message produced by WMQI statistics.
```

```
BIP2381I: WMQI thread statistics. ProcessID...  
Key='3', Number='0', TotalNumberOfInputMessages='1',  
TotalElapsedTime='20457211', TotalCPUTime='395405',  
CPUTimeWaitingForInputMessage='10425',  
ElapsedTimeWaitingForInputMessage='3302147',  
TotalSizeOfInputMessages='690', MaximumSizeOfInputMessages='690',
```

```
BIP2382I: WMQI node statistics. ProcessID=..., Key='3', Label='', Type='  
'', TotalElapsedTime='0', MaximumElapsedTime='0', MinimumElapsedTime='0',  
TotalCPUTime='0', MaximumCPUTime='0', MinimumCPUTime='0',  
NumberOfOutputTerminals='1'.
```

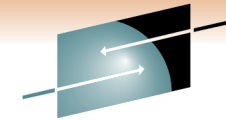

Notes



N
O
T
E
S

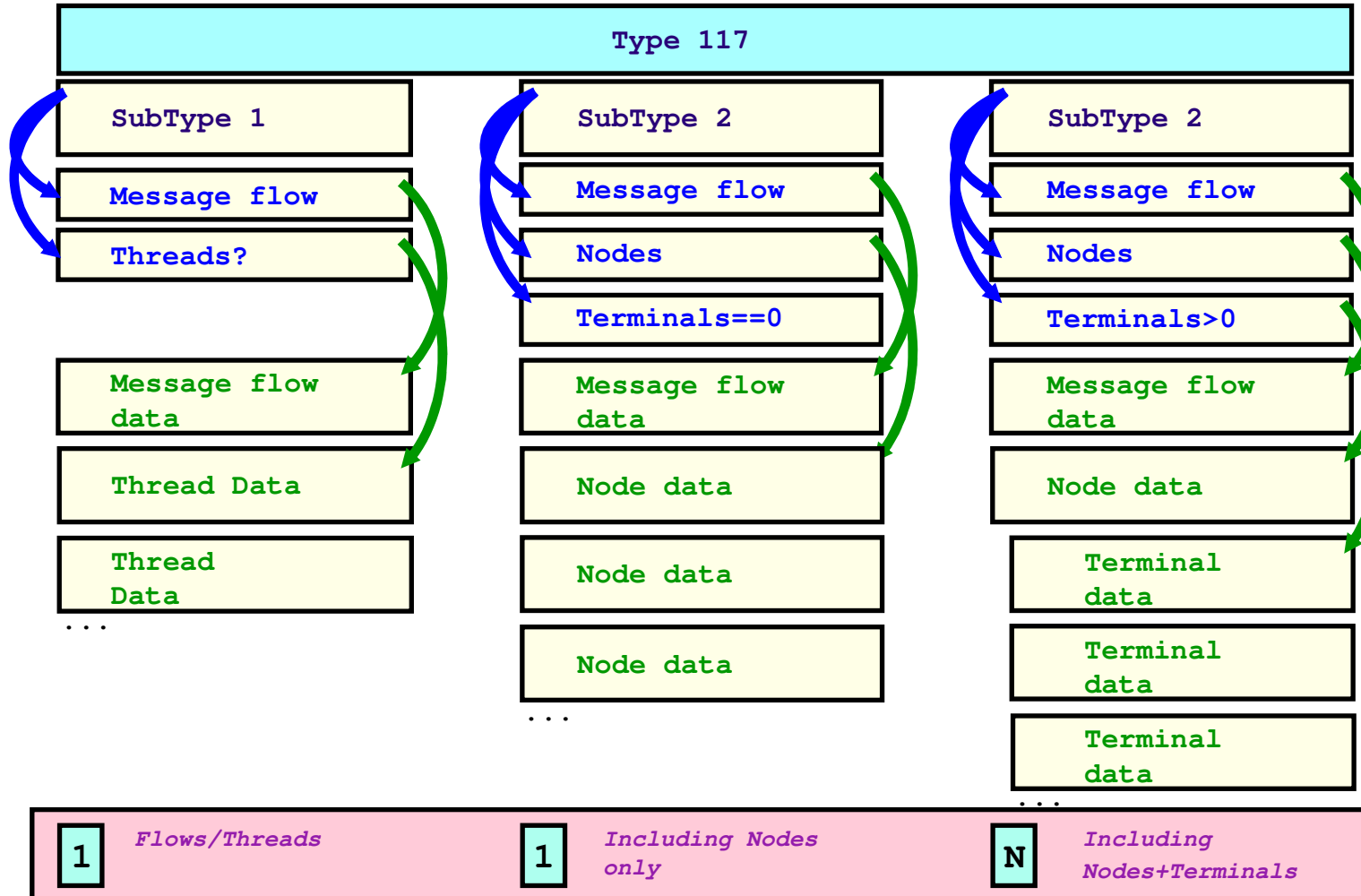
- UserTrace uses architected BIP messages to report data
 - The following BIP messages are written to UserTrace by the broker.
 - BIP2380 includes message flow data.
 - BIP2381 includes thread data. One message per thread data collected.
 - BIP2382 includes node data. One message per node data collected.
 - BIP2383 includes terminal data. One message per terminal data collected, per Node.
 - If output is to UserTrace, BIP2380 will always be written. BIP2381, BIP2382 and BIP2383 will only be written if the appropriate data is currently being collected.
- Collation of BIP messages
 - As several threads groups may be writing to the trace at the same time, a mechanism is provided to associated userTrace records with each other.
 - Each BIP message starts with a ProcessID and KEY value, which can be used to group BIP messages issued for a particular Archive or Snapshot instance.
 - KEY is a numeric value, which is incremented for each call to the UserTrace writer. It starts with value 0, and is reset each time the DFE address space is started.
 - PROCESSID is the ID of the process in which the UserTrace writer is running.
- UserTrace Collection
 - When writing to UserTrace, the broker will always write to the UserTrace files, regardless of current UserTrace settings. This makes sense because it would be annoying to have to turn on UserTrace to get accounting and statistics information, as well as it being full of other less interesting (possibly) UserTrace not related to accounting and statistics.
 - UserTrace files are formatted in the normal way using mqsireadlog and mqsiformatlog.
- User Trace Benefits
 - User Trace is a quick way to measure the performance characteristics of a message flow on the machine you running on. Typically individual developers doing a high level performance overview might use user trace to get a feel for the behaviour of their message flow.

z/OS SMF Record Structure

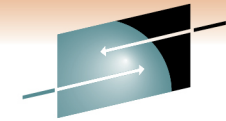


SHARE
Technology • Connections • Results

BipSmf.h



Notes

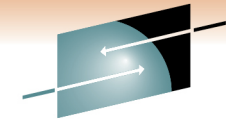


SHARE
Technology • Connections • Results

N
O
T
E
S

- SMF 117 records describe the A&S information
 - The subtype of SMF 117 records written depends on the data currently being collected.
 - A type1 record is produced when a flow is only collecting message flow data or Threads data. A single type1 record is produced with all threads included.
 - Type2 records are produced when a flow is collecting nodes data. When *only* nodes data is collected, a single type 2 record is written to SMF, whereas when nodes *and* terminals are being collected, multiple type 2 records are written to SMF.
- A sample formatter is available to get you started.
 - We provide a sample formatter which produces output very similar to user trace.
 - The sample formatter also outputs information in a format ready for DFSORT so that you can do highly flexible post processing on the collected data.
 - See SupportPac IPxx for more details on this.
- BipSMF.h describes the SMF records
 - You can use the sample formatter and the BipSMF.h header file to produce your own reports.
- The broker userid must be permitted to the BPX.SMF facility class profile to write SMF records.

View accounting/statistics in MBX



SHARE

Technology • Connections • Results

Statistics

- Start Message Flow Statistics
- Stop Message Flow Statistics
- Open Statistics Views
- Start Resource Statistics
- Stop Resource Statistics
- Open Resource Views

User Trace All Flows

Trace Nodes All Flows

Service Trace

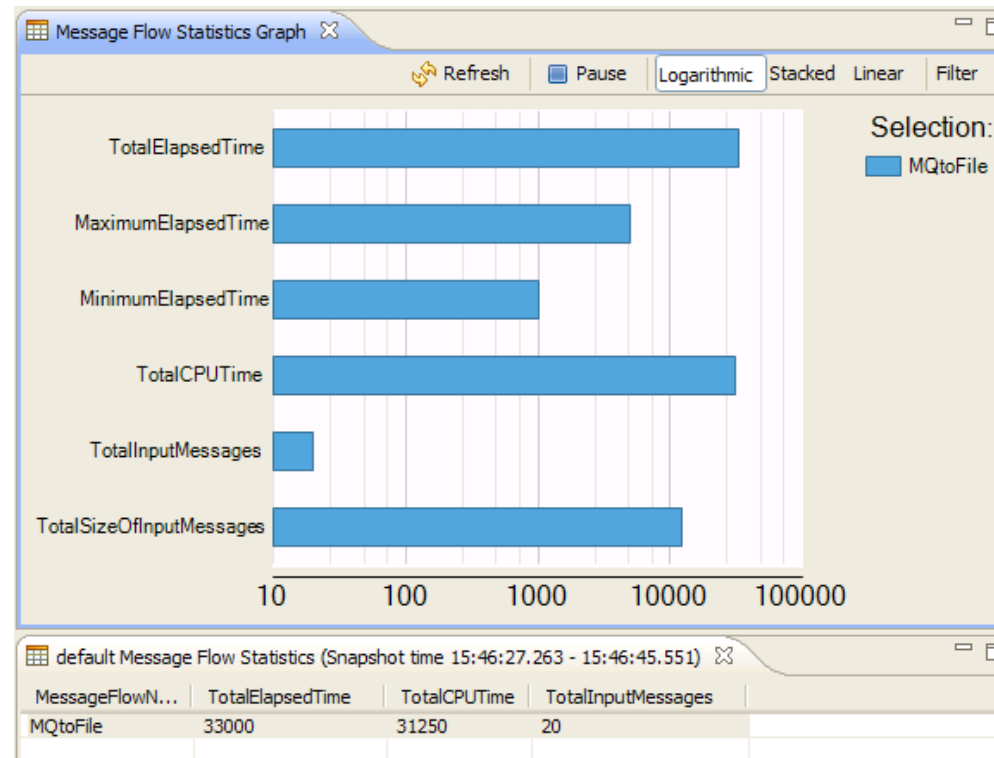
Flow Debug Port

MQ Explorer - Content testbrk Administration Log

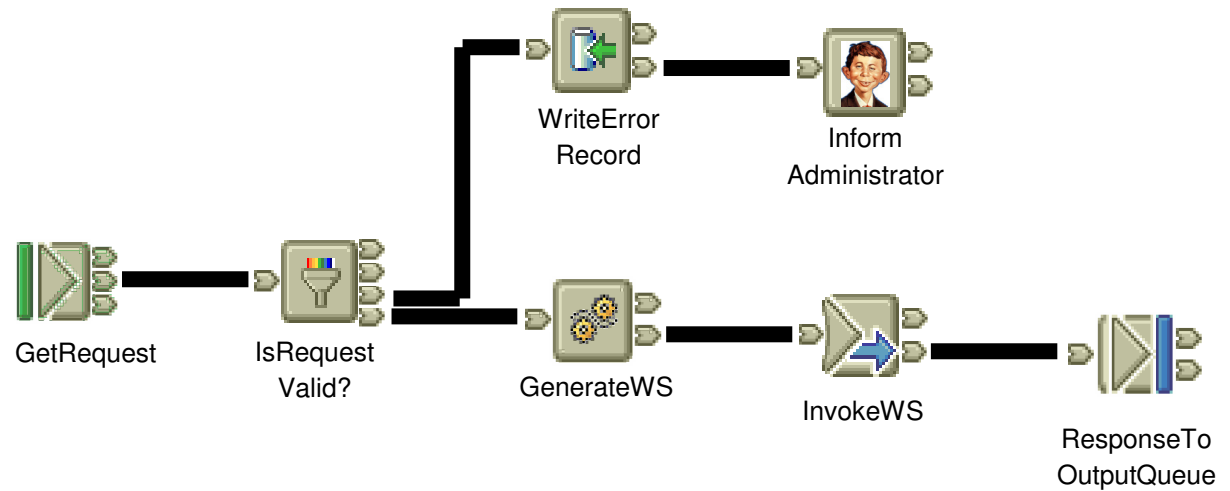
Execution Group default

Warning: Performance Impact - Statistics Reporting active

- The Message Broker Explorer enables you to start/stop message flow statistics on the broker, and view the output.
- New in V7 (although supportpac IS02 available for V6.1)
- Warnings are displayed advising there may be a performance impact (typically ~3%)

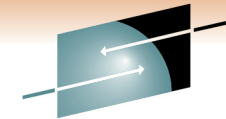


Chargeback Scenario



Question: Who pays for this flow?

Accounting Origin



SHARE
Technology • Connections • Results



Dept A



Dept B

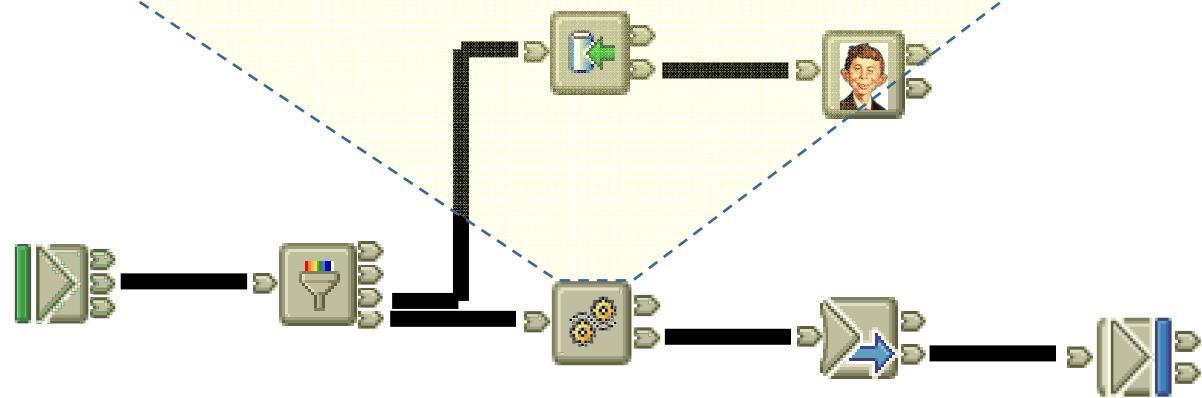


Dept C

(Anonymous)

...

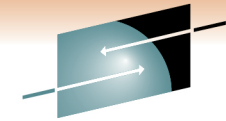
```
SET Environment.Broker.Accounting.Origin =
    InputRoot.XML.DepartmentName; -- Dept A,B,C
```



```
<WMQIStatisticsAccounting RecordType="Archive"
  <MessageFlow BrokerLabel="TestBroker3" BrokerUII
    ExecutionGroupName="default" Execu
    MessageFlowName="ParentFlow" Start
    AccountingOrigin="Dept A"/>
```

...

Notes



S H A R E

Technology • Connections • Results

● A Chargeback Scenario

- This is the second scenario to examine. It's a bit less frantic than the Performance Analysis Scenario, thank goodness.
- Let's imagine that you've got your flow up and running and it's performing well (it was the bad messages being sent after all, not the "Inform Administrator" user-defined node. (I knew it!) You've got a whole raft of people that want to use the capability provided by your flow. In fact, they don't even know they are using a message flow - they just send a message to a queue and get a response. Excellent - the world couldn't be better, you've got happy users.
- Now, you do some calculations and realize that to cope with all these users of your flow, you're going to be using quite a lot of CPU and IO resources. You're running an infrastructure group which charges back the cost of these resources ("CPU's don't grow on trees you know!") to the folk who generate the work in the first place.
- Therefore, you know how much the computer resource is going to cost you and you want to be able to get this cost, or some appropriate portion of it, from these users. This isn't fair or unfair - it's how many infrastructure groups recover their costs.

N

● Different Users make Work for your Flows

- Now the originators of the messages can be many and varied. For example, some messages might correspond to input from real people (they still exist apparently), however, work may come from other organizations, geographical areas (countries even), or other computer systems.
- You might identify and separate different channels of input, using WebSphere MQ queues for example, which naturally partition the users. Different message flows might be running against these different input streams.
- In more complex cases, many different users (e.g. many clients), or many classes of users (partner organizations, other machines) might actually **share** a message flow. A message flow might therefore be processing work for many different users of different classes.

O

T

E

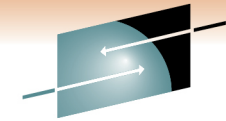
● Accounting for this Usage

- However, the basic problem is that you want to see how much resource is being used, so that you can charge your users accordingly for the you are incurring to maintain this service.
- The cost is exactly the same as the metrics we discussed in the performance analysis scenario. More on this soon.
- But how do we identify the users. We'll return to this problem in some detail later on, but suffice it to say that there needs to be some way to separate the users, but flow, input queue, execution group, broker, whatever. We'll see that the users will need to be mapped to the same collections that we discussed earlier.
- These two scenarios highlight the difference between accounting and statistics. In the performance scenario, you were treating the numbers as **statistics** - they gave you an understanding of the behaviour of your message flows. When you use these same numbers with a view to charging for them, then your focus has become one of **accounting** for usage. And that's the difference between **accounting** and **statistics** - the way in which one set of numbers can be used in two different contexts.
- Let's now have a look at the Chargeback analysis we might like to perform.

S

SHARE
in Anaheim
2011

Notes



SHARE

Technology • Connections • Results

N
O
T
E
S

●AccountingOrigin Allows you to Classify Reports

- In a consolidated flow, i.e. one being used by several different users or classes of users, it's important to be able to identify the costs associated with a particular user or class of user. From Version 5, you can request Accounting and Statistics reports to be generated to identify the originator of the input messages.
- This allows brokers to have a minimum number of flows for accounting purposes and still chargeback, benefitting from economy of scale, and administrative burden reduction.
- The foil shows messages originating from three different departments - A, B and C. These messages are all processed by the same message flow, but the gathered reports identify the cost breakdown by originating department.
- If you examine the report snippet, you can see that the AccountingOrigin tag identifies the report as belonging to "Dept. A".
- The Accounting origin is set inside the flow when it has determined the origin of the message. Any technique can be used to do determine this; for example a flow might use a field in the MQMD, or a field in the body of the message. The AccountingOrigin is completely virtual, for example it might periods of the day to allow you to profile usage with time! (You could also use archive statistics for this.)

●Messages are Classified by the Flow

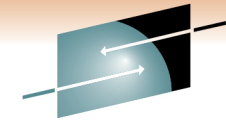
- As a message passes through a message flow, if the flow at some points decides that the message needs to be classified, it may do so.
- It sets the **Environment.Broker.Accounting.Origin** tree element to identify the origin. When the message has been finished with, the data related to its processing is collected separately to messages with different AccountingOrigin.
- Classification is usually done using User defined ESQL in a Compute node (or using a plug-in node, since the Environment tree is available to all nodes) which sets a field thus:
 - ♦**SET Environment.Broker.Accounting.Origin = '...'**
- When the message has been processed by the flow the information identifying the origin is stored with all the other information for this origin and separate to information from other origins. Different origins can therefore be collected and reported separately.

●AccountingOrigin needs to be Enabled

- This function is enabled using a new option on the mqsichange flowstats command (see later). It is not always enabled because there is extra processing associated with this processing and although not excessive, it would have an impact on performance.
- You should be aware that enabling this function could generate a large volume of reports, that's because you will get a different report for each AccountingOrigin identified by your flow in a given time period.
- If the function is enabled, and **Environment.Broker.Accounting.Origin** is not populated then the Statistics and Accounting data collected while processing that input message will be accumulated to the default Accounting Origin.
 - ♦**The default value for the AccountingOrigin is "Anonymous".**
- If the Accounting Origin function is disabled then the default action is to accumulate all Accounting and Statistics data to this default Origin.

SHARE
in Anaheim
2011

Notes: Command Syntax and Usage



SHARE
Technology • Connections • Results

N
O
T
E
S

```
mqsichangeflowstats <broker name>
```

```
-c active | inactive      activate/deactivate  
-s                        collect snapshot data  
-a                        collect archive data  
-t none | basic          thread collection  
-n none | basic | advanced node collection  
-e <execution group label>  
-g                        all execution groups  
-f <message flow label>  
-j                        all message flows  
-o smf | xml | usertrace output type  
-r                        reset statistics  
-b none | basic          accounting Origin
```

```
mqsireportflowstats <broker name>
```

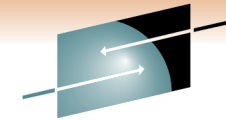
```
-s      report snapshot settings  
-a      report archive settings  
-e <execution group label>  
-g      all execution groups  
-f <message flow label>  
-j      all message flows
```

```
mqsicreatebroker, mqsichangebroker <broker name>
```

```
-v      set archive interval
```

SHARE
in Anaheim
2011

Notes

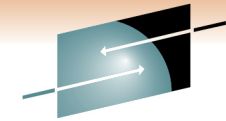


SHARE

Technology • Connections • Results

N
O
T
E
S

- Controlling and Reporting
 - The mqsichangemsgflowstats command is used to start, stop and modify accounting and statistics collection.
 - The mqsireportmsgflowstats command is used to determine the current settings for accounting and statistics collection.
 - UserTrace is the default output on all platforms.
- z/OS specific SDSF commands
 - mqsichangeflowstats
 - ◆ /F <broker name>,CS
 - mqsireportflowstats
 - ◆ /F <broker name>,RS
 - mqsichangebroker
 - ◆ /F <broker name>,CB
 - The command options are for example 'c=' instead of '-c'.
- Option -v (Statistics Major Interval) range is 10 - 144000 (10 mins to 10 days).
 - On z/OS only, setting this to zero seconds indicates ENF will be used as the timer.



S H A R E

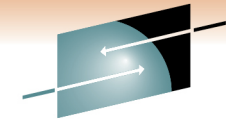
Technology • Connections • Results

Message Broker Resource Statistics

“*Resource statistics* are collected by a broker to record performance and operating details of resources that are used by execution groups.”



View resource statistics in MBX



SHARE

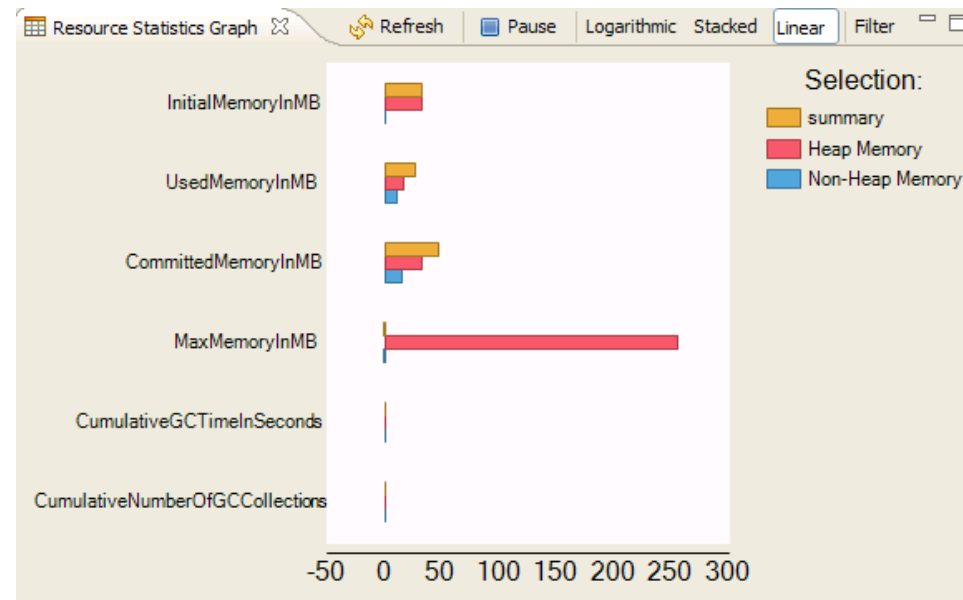
Technology • Connections • Results

- Statistics
 - Start Message Flow Statistics
 - Stop Message Flow Statistics
 - Open Statistics Views
 - Start Resource Statistics**
 - Stop Resource Statistics
 - Open Resource Views
- User Trace All Flows
- Trace Nodes All Flows
- Service Trace
- Flow Debug Port

MQ Explorer - Content testbrk Administration Log

Execution Group default

Warning: Performance Impact - Statistics Reporting active



- The Message Broker Explorer enables you to start/stop resource statistics on the broker, and view the output.
- New in V7
- Warnings are displayed advising there may be a performance impact (typically ~1%)

default Resources Statistics (Snapshot time 15:46:29 - 15:46:49)

	CICS	CORBA	FTEAgent	JDBCConnectionPools	JVM	ODBC	SOAPInput	Security	Sockets
name									
summary					32	27	47	-1	0
Heap Memory					32	16	32	256	0
Non-Heap Mem...					0	11	15	-1	0
Garbage Collect...									0

Enabling Resource Statistics on command line



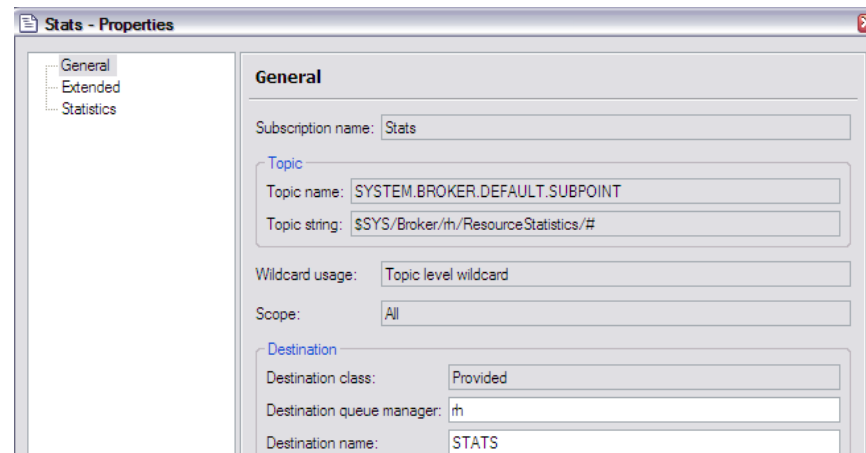
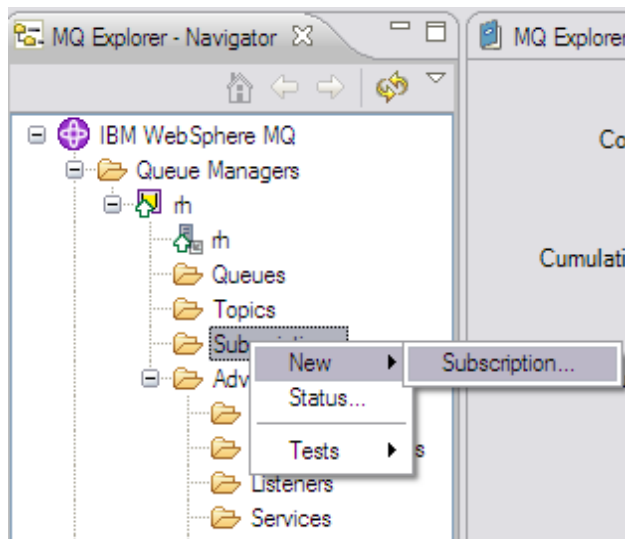
- Enable resource statistics
 - `mqsichangeresourcestats brokerName -c active -e executionGroup`
- Disable resource statistics
 - `mqsichangeresourcestats brokerName -c inactive`
- Display current status
 - `mqsireportresourcestats brokerName -e executionGroup`
BIP8940I: Statistics settings for resource type JVM in execution group default - On?: inactive, measurements: [CommittedMemoryInMB, etc
BIP8940I: Statistics settings for resource type Sockets in execution group default - On?: inactive, measurements: [AverageBytesReceivedPerMessage, etc
- Each command applies to all execution groups if none is specified.
- See Information Center for command syntax, or simply type the command with no parameters.

N
O
T
E
S

Subscribing to receive statistics



- Resource statistics are published to a well-defined topic
- A user subscribes to a topic string to view the statistics they want, e.g.
 - `$SYS/Broker/MB7BROKER/ResourceStatistics/default`
for statistics related to execution group 'default' on broker 'MB7BROKER'
 - `$SYS/Broker/+/ResourceStatistics/+`
for all execution groups on all brokers
- One option is to set up a subscription to a queue in MBX



Example publication



```
<ResourceType name="Sockets">

  <resourceIdentifier name="summary" TotalMessages="5"
  TotalSocketsOpened="2" AverageSocketsOpenedPerMinute="6"
  TotalBytesSent="24484" AverageBytesSentPerSecond="1223"/>

  <resourceIdentifier name="localhost.7080" TotalMessages="3"
  TotalSocketsOpened="1" AverageSocketsOpenedPerMinute="3"
  TotalBytesSent="12891" AverageBytesSentPerSecond="644"/>

  ...
</ResourceType>
```

- This is an example of the type of data published for a specific resource – in this case sockets (shows a subset of the data we actually publish).
- Each resource manager will always publish a ‘summary’ record which collates the information for the whole execution group
- Some resource managers will also publish finer grained information for individual resources that they manage. For instance in the sockets case we publish information about sockets managed for each endpoint that the broker is sending messages to, and then roll all of these up into the summary record.

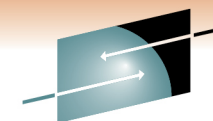
Example: JVM Stats Data (1)



A JVM has two memory areas, Heap and Non Heap, the following properties are reported for each:

- InitialMemoryInMB
 - The initial amount of memory (in megabytes) that the Java virtual machine requests from the operating system for memory management during startup.
- UsedMemoryInMB
 - The amount of memory currently used (in megabytes).
- CommittedMemoryInMB
 - The amount of memory (in megabytes) that is guaranteed to be available for use by the Java virtual machine.
- MaxMemoryInMB
 - The maximum amount of memory (in megabytes) that can be used for memory management.
- The value -1 is returned for measurements that are undefined or have not been set.

JVM Stats Data (2) Garbage Collection



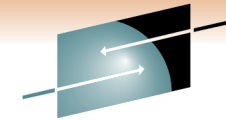
S H A R E

Technology • Connections • Results

A JVM may have one or more garbage collectors to report on. For example one collector may do minor collections which are frequent but take little time and another collector may do major collections which are not as frequent but take considerably more time. The following properties will be reported for each

- resourceIdentifier name
 - The name of the garbage collector that is being reported on.
- CumulativeNumberOfGCCollections
 - The total number of garbage collections that have occurred for this instance of the JVM. Its value may be undefined which is indicated by a value of -1.
- CumulativeGCTimeInSeconds
 - The accumulated garbage collection elapsed time in seconds for this instance of the JVM.
 - Its value may be undefined which is indicated by a value of -1.

JVM Stats – Interpretation



SHARE
Technology • Connections • Results

- How much memory is the JVM using?
 - OS tools can give you the total memory used by the dataflowengine process, but do not show you how that memory is divided between Java and non-Java processing in the execution group. **CommittedMemoryInMB** tells you how much memory is currently allocated to the JVM and **MaxMemoryInMB** tells you how big this may grow.
- How often am I doing garbage collection?
 - If **CumulativeNumberOfGCCollections** is increasing frequently then the garbage collection (GC) may be excessive. A certain level of GC is expected and required, but excessive GC can impact performance. If **CumulativeGCTimeInSeconds** is increasing at more than 2 seconds per 20 second stats interval then you should consider increasing the JVM Max Heap size for your execution group. Also inspect any Java Plugin nodes or Java Compute nodes in your message flows to ensure that you are not creating and deleting many objects that could be reused.
- Should I change my min or max heap size?
 - If your **CumulativeGCTimeInSeconds** is increasing by more than 2 seconds per stats interval then look to increase the Max Heap Size to reduce this. If your **UsedMemoryInMB** is never close to your **InitialMemoryInMB** then this indicates that you are preallocating more memory for the heap than necessary and could reduce the JVM Min Heap Size value for you execution group to a value closer to the **UsedMemoryInMB** value. Changes to these values this should be done gradually to find the optimum settings.

SHARE
in Anaheim
2011

Comparison with Existing Accounting and Stats



- The existing (pre-7.0) accounting and stats
 - provides metrics related to message flows (e.g. number of messages processed)
 - is useful for charge-back (accounting) as well as monitoring performance
 - allows users to specify snapshot or archive reporting periods, and how data should be collected - by thread, node or specific node terminals
 - see Information Center topic "Monitoring message flow performance"
- The new resource statistics
 - is not tied to individual message flows (e.g. JVM memory usage)
 - is of limited use for accounting purposes
 - complements the existing accounting and stats metrics and is enabled/disabled separately
 - supports the Snapshot reporting period

Summary



- Message Broker 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
- Message Broker has built-in accounting/statistics reporting
 - Performance of flows, nodes, threads
 - Accounting origin (useful for chargeback)
 - Visualised in MBX
- Message Broker has built-in resource statistics reporting
 - Performance of resource managers (such as JVM)
 - Visualised in MBX

Copyright and Trademarks



© IBM Corporation 2011. All rights reserved. IBM, the IBM logo, ibm.com and the globe design are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml. Other company, product, or service names may be trademarks or service marks of others.