

Advanced usages of Business Transaction Monitoring

SanjayNagchowdhury

Published on December 7, 2015 / Updated on December 8, 2015

Learn about different areas of administration for Business Transaction Monitoring (BTM); see how to configure BTM for aspects of MQ, your database, and IBM Integration Bus.

Business Transaction Monitoring is a new capability that has been provided in IBM Integration Bus 10.0.0.3. This new capability has been introduced in [Business Transaction Monitoring – Why, what, how](#) and the steps to set up and use BTM have been described in [Business Transaction Monitoring in IIB](#).



The following areas are covered in this article:

MQ administration for BTM: You will see what aspects of MQ are used by BTM and what further configuration or tuning you may need to do.

Database administration for BTM: You will see the database tables used by BTM and their inter-dependencies. You will see how to configure BTM for your database.

IIB administration for BTM: You will see how to configure the recording and viewing capability provided by BTM. You will also see how to transfer your configuration for BTM from a test system to a production system.

Planning your use of BTM: You will see how you should consider configuring monitoring events in your message flows and using multiple Business Transaction Definitions across a set of message flows that are used by your business.

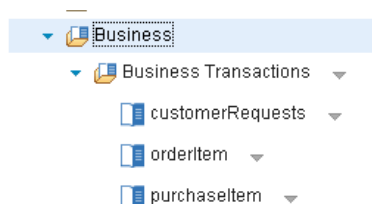
MQ administration for BTM

Monitoring events are configured on message flow nodes using the IBM Integration Toolkit or by creating a monitoring profile. When a message is processed by a node which has an event configured on it, an MQ message is published to an MQ topic. The MQ topic name has this structure:

```
$SYS/Broker/integrationNodeName/Monitoring/integrationServerName/flow_name
```









When a Business Transaction Definition is created, a selection of monitoring events are flagged as business events. They are deemed crucial to understanding the state of a business transaction. An MQ subscription is created for each topic that is used by these monitoring events that have been flagged.

For example, I have created three separate business transaction definitions (BTDs), as shown below:



After creating, each BTD, an MQ subscription is created for the relevant topics. In MQ Explorer, I have these subscriptions that were created automatically:

Subscriptions

Filter: Standard for Subscriptions
Subscription name
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/main/Request_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/main/Response_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/order/Order_Item_Despatch_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/order/Order_Item_Request_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/order/Order_Item_Response_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/purchase/Process_Payment_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/purchase/Purchase_Item_Request_Flow
 btm_node:/apiv1/policy/DataCapture/default#default:\$SYS/Broker/btm_node/Monitoring/purchase/Purchase_Item_Response_Flow

When a transaction is processed, the events are emitted to the MQ topic. The relevant subscription picks up the event and puts it to a destination queue. The destination queue is defined in the Data Capture policy supplied with IIB v10.0.0.3. It is defined in this policy attribute:

```
<queueName>SYSTEM.BROKER.DC.RECORD</queueName>
```

Messages are read from this queue and recorded into the database. If a database error occurs while recording the date, then the messages are put to a backout queue. The backout queue is also defined in the Data Capture policy and is defined in this policy attribute:

```
<backoutQueue>SYSTEM.BROKER.DC.BACKOUT</backoutQueue>
```

These queues are defined when you run the script `iib_createqueues.[cmd|sh]`. If you wish to use alternative queues, then you can modify the Data Capture policy provided by IIB. To, do this run this command:

```
mqsireportpolicy integrationNodeName -t DataCapture -l default -f default.xml
```

Edit the file `default.xml` that has been created and replace the queue names with the alternative ones that you want to use. Now, update the Data Capture policy, by running this command:

```
mqsichange policy integrationNodeName -t DataCapture -l default -f default.xml
```

The Data Capture policy will now be using the queues that you have defined. The depth of the RECORD queue should be monitored so that it stays at a constant depth. Performance of business transaction monitoring can be impacted if business events are written to the queue faster than they can be written to the database. For this reason, you should be careful to flag only those events that you deem critical for monitoring your business transaction.

Database administration for BTM

A prerequisite for using BTM is to have a database created and tables defined in it. The steps to configure the database are described in [Business Transaction Monitoring in IIB](#).

The tables that are used by Business Transaction are:

WMB_MSGS

WMB_BINARY_DATA

WMB_BUSTRANS

After you have configured your message flow nodes to emit events and created a business transaction definition containing those flows and events, entries are written to these database tables when business transactions are processed by those flows.

All monitoring events that were defined on the message flow nodes are recorded in the WMB_MSGS table. If you have specified that you want the payload data to be included in the recorded event, then the payload data is recorded in the WMB_BINARY_DATA table.

The set of events that are correlated by BTM to ascertain the state of a business transaction are recorded in the WMB_BUSTRANS table.

This table contains all the events that have been flagged across all BTDs and have been emitted by the message flow nodes.

Entries that are in the WMB_BUSTRANS table for business transactions must have corresponding entries in the WMB_MSGS table.

The state of a business transaction is calculated depending on the events that have been received for that business transaction. There are 4 states for a business transaction:

In progress – indicates that the transaction is still running. The Start and any Progress events have been recorded, but the End or Failure event has not been recorded. If only a Start event has been recorded, the business transaction is still considered to be In Progress.

Ended – indicates that the transaction has completed successfully. The Start, Progress (if any) and End events have been recorded.

Failed – indicates that the transaction completed in a way that the user has designed as a failure path when defining the BTD. The Start, Progress (if any) and Failure events have been recorded.

Inconsistent – indicates that the transaction's state could not be resolved from the set of events that have been recorded.

A business transaction may be marked as inconsistent for different reasons. Here are some examples:

Only a Progress event has been received but no Start

Only a Progress and End event has been received but no Start

Only an End event has been received but no Start

Two or more Start events or End events were received for the same transaction. This could have happened when 2 or more transactions used the same global transaction ID. A global transaction ID must be unique per business transaction.

A transaction was received, an error thrown during the transaction causing a rollback and then the transaction was re-processed using the same global transaction ID.

The Data Capture profile supplied with IIB can be tailored for settings that affect the recording of data into the database. These fields are used in the Data Capture policy which can be configured:

```
<dataSourceName>MBRECORD</dataSourceName>
```

This is the name of a datasource defined for the database using mqsisetdbparms. This can be altered using the web user interface in the Business Transactions section.

```
<schema></schema>
```

This is the schema used for the database.

```
<commitCount>1</commitCount>
```

This indicates the number of transactions that are processed on each thread before they are committed. This value must be 1 if the value of threadPoolSize is greater than 1. Otherwise, this value is ignored.

```
<commitIntervalSecs>3</commitIntervalSecs>
```

This indicates the time interval at which a commit is taken when the commitCount value is greater than 1 but the number of transactions that have been processed has not reached the value of the commitCount property.

```
<threadPoolSize>10</threadPoolSize>
```

This indicates the number of threads that are used by the integration server used for recording the business data.

```
<useCoordinatedTransaction>>false</useCoordinatedTransaction>
```

This indicates whether transactions are globally coordinated across queue manager and database resources.

If you wish to modify any of these values, run mqsisreportpolicy to get a file containing the Data Capture policy, update the properties and then run mqsischange policy to set the Data Capture policy.

You can run mqsisreportpolicy again to check your new values are in action.

IIB administration for BTM

Read how to configure the recording and viewing capability provided by BTM, and how to transfer your configuration for BTM from a test system to a production system.

How to specify servers for recorder and viewing

To make it easier to get going with Business Transaction Monitoring, an Integration Server is chosen for recording and viewing automatically after you have created your business transaction definition.

You might want to change this to designate a specific server for recording and another for viewing to achieve better utilisation or distribution of work in your deployment scenario.

If you want to configure a specific Integration Server to use for recording and viewing, you can configure this in the Data Capture policy.

Run `mqsiportpolicy` as described above to get a file containing the Data Capture policy. Update these properties for the Integration Server to be used for recording or viewing:

```
<recordingServer></recordingServer>
```

```
<viewingServer></viewingServer>
```

After you have configured the Integration Server, run `mqsichangepolicy` as described above for the changes to take effect.

You will see the following message in the syslog to indicate which Integration Server is being used for recording:

```
BIP2159I: ( btm_node.main ) Integration server 'main' is now recording data for the following data capture stores: '/apiv1/policy/DataCapture/default#default'.
```

The Data Capture policy provided by IIB has no specific Integration Server configured for recording and viewing. It uses the first Integration Server that has been started for recording and viewing.

Moving BTDs between different Integration Nodes

If you want to move your BTDs between systems, you can use a REST client using an HTTP GET to get your BTD from one system and then use a HTTP PUT put it to another system.

Specify this URL to get your BTD with an HTTP GET:

<http://hostname1:4414/apiv1/business/businesstransactions/BTDname>

using these HTTP Headers:

```
Content-Type: application/x-www-form-urlencoded
```

```
Accept: application/javascript, application/json
```

This will return a JSON response.

You can now put it to the target system using your REST client, specifying an HTTP PUT.

Specify this URL to put your BTD with an HTTP PUT:

<http://hostname2:4414/apiv1/business/businesstransactions/BTDname>

You must specify these HTTP Headers when issuing the request:

```
Content-Type:application/json
```

```
Accept:application/json
```

You must copy the JSON output from the HTTP GET request that you just did and place it in the Request Body for the HTTP PUT request.

You may need to create the same Integration Servers on your target machine as you did on your source machine, or you may need to tailor what you put in the Request Body according to the Integration Servers that you are using on your target machine.

Planning your use of BTM

In order to get the maximum benefit from using Business Transaction Monitoring, you should take the following approach:

Be very clear on what events are most important to you for understanding the state of your business transaction.

Consider what message rates you are expecting your system to be handling and based on that consider how many events you want recorded per transaction.

Think about what aspects of your business that you want to monitor using Business Transaction Monitoring. If you have a very heavily loaded system, then you may want to consider just tracking transactions that have failed, rather than tracking all transactions.

You are likely to be using multiple applications with many message flows across multiple Integration Servers. Business Transaction Monitoring is quite capable of supporting that topology, but in order to understand the business transaction results, it is encouraged that you separate your BTDs rather than them overlapping across flows.


Here is a simple example which illustrates what can happen if you do not plan your BTDs and events.

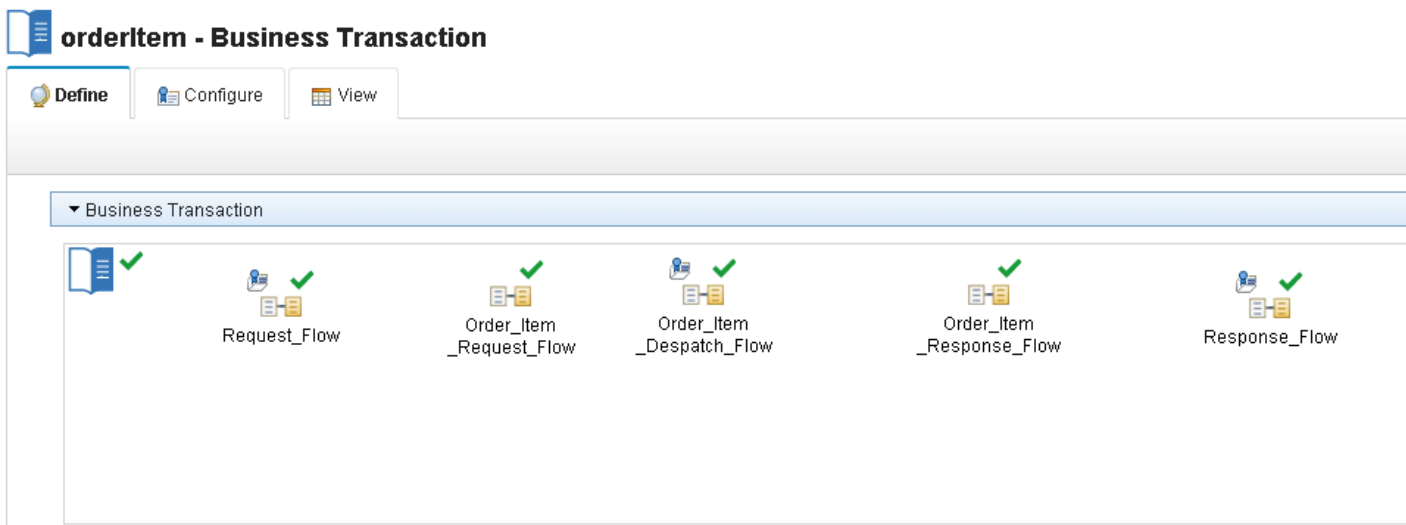
I have a simple Retail system which receives customer requests and forwards them to three different business areas: Stock Checking, Ordering and Purchasing. The responses are received from the different business areas and sent back to the customer.

There is a Request flow which handles the customer requests and a Response flow which sends the responses back to the customer:

I have separate applications containing message flows for the different business areas.

Lets create an orderItem BTD and a purchaseItem BTD. I will specify the start event in the request flow, a progress event in each business area and the end event in the response flow.

The orderItem BTD has these flows with events configured on 3 of the flows. You can see which flows has events configured on it as it has a  icon above the flow.



orderItem - Business Transaction

Define | Configure | View

Business Transaction

- Request_Flow
- Order_Item_Request_Flow
- Order_Item_Despatch_Flow
- Order_Item_Response_Flow
- Response_Flow

These events have been flagged as business events for the orderItem BTD:

Business Transaction Details

Here are the business transaction details.

Name:

Description:

Business Events:


Type	Flow Name	Event Source
Start	Request_Flow	Receive Retail Request.terminal.out
End	Response_Flow	Send Retail Response.terminal.in
Progress	Order_Item_Despatch_Flow	Compute.terminal.in

The purchaseItem BTM has these flows with events configured on 3 flows as well. Although the Progress event for orderItem is from the Order_Item_Despatch_flow, while the Progress event for purchaseItem is from the Process_Payment_Flow.


purchaseltem - Business Transaction

Define


Business Transaction




Request_Flow




Purchase_Item
_Request_Flow



Process_Payment
_Flow



Purchase_Item
_Response_Flow



Response_Flow

These events have been flagged as business events for the purchaseItem BTM:

Business Transaction Details

Enter the name and description for the business transaction definition.

Name:

Description:


Business Events:

Type	Flow Name	Event Source
Start	Request_Flow	Receive Retail Request.terminal.out
End	Response_Flow	Send Retail Response.terminal.in
Progress	Process_Payment_Flow	Completed Processing Payment.terminal.in

You can see that both orderItem and purchaseItem are using the same flows: Request_Flow and Response_Flow for their Start and End events.

Let's send in 2 orderItem and 2 purchaseItem transactions.

When I view the transactions for orderItem, I would want to see that 2 order item transactions were processed. However, I see 4 transactions!!

 **orderItem - Business Transaction**

Define | Configure | **View**

▼ All instances of orderItem

Total : 4

Transaction ID	Start Time	Last Update	Status
4	2015-11-27 18:00:07.454	2015-11-27 18:00:08.322	Ended
3	2015-11-27 18:00:07.154	2015-11-27 18:00:08.207	Ended
2	2015-11-27 18:00:06.911	2015-11-27 18:00:08.170	Ended
1	2015-11-27 18:00:06.724	2015-11-27 18:00:07.955	Ended

This has happened as the start and end events for both business transactions were defined in a common flow. The Start and End event were captured for all events by both BTDs. The business transaction definitions should have been split up in a better way so that we only see the transactions for orderItem in its BTD and the same for purchaseItem.

The recommended practice for this scenario is to define a general BTD to capture the events by the main processing flows Request_Flow and Response_Flow. The orderItem BTD should have only events that are specific to that business area, so only flag the events the message flows for ordering an item. The same goes for purchaseItem.

Here is a better structure for the BTDs.

I have defined a general BTD called customerRequests which has Start, Progress, End and Failure events on the Request_Flow and Response_Flow flows.

I have removed the Request_Flow and Response_Flow flows from the orderItem and purchaseItem flows.

For orderItem, I have defined the Start event in the Order_Item_Request_Flow and the End event in the Order_Item_Response_Flow.

For purchaseItem, I have defined the Start event in the Purchase_Item_Request_Flow and the End event in the Purchase_Item_Response_Flow.

customerRequests – define tab

customerRequests - Business Transaction

Define | Configure | View

Business Transaction

Request_Flow | Response_Flow

No flow selected

Business Transaction Event Definitions

This section shows the monitoring events that are defined for the selected flow or for all flows in the business transaction. Select the monitoring events that signify the start, end, and failure of your business:

Message Flow - Monitor Event	Event Source Address	Flag as	Global Transaction Correlator
Request_Flow (BTM_RETAIL_APP from main)	3 Monitoring Events		
Process customer request	<input checked="" type="checkbox"/> ProcessOrder.terminal.in	Progress	\$Root/XMLNSC/btm_retail/customerNumber
Retail request received	<input checked="" type="checkbox"/> Receive Retail Request.terminal.out	Start	\$Root/XMLNSC/btm_retail/customerNumber
Unknown Order request was received	<input checked="" type="checkbox"/> Unknown Order.terminal.in	Failure	\$Root/XMLNSC/btm_retail/customerNumber
Response_Flow (BTM_RETAIL_APP from main)	1 Monitoring Event		
Send Retail Response	<input checked="" type="checkbox"/> Send Retail Response.terminal.in	End	\$Root/XMLNSC/btm_retail/customerNumber

Business Transaction Details

Here are the business transaction details.

Name:

Description:

Business Events:

Type	Flow Name	Event Source
Start	Request_Flow	Receive Retail Request.terminal.out
End	Response_Flow	Send Retail Response.terminal.in
Failure	Request_Flow	Unknown Order.terminal.in
Progress	Request_Flow	ProcessOrder.terminal.in

OK Cancel

orderItem - define tab

orderItem - Business Transaction

Define | Configure | View

Business Transaction

Order_Item_Request_Flow | Order_Item_Despatch_Flow | Order_Item_Response_Flow

No flow selected

Business Transaction Details

Here are the business transaction details.

Name:

Description:

Business Events:

Type	Flow Name	Event Source
Start	Order_Item_Request_Flow	Order Item Input Queue.terminal.out
End	Order_Item_Response_Flow	Send Confirmation.terminal.in
Progress	Order_Item_Despatch_Flow	Compute.terminal.in

OK Cancel

purchaseItem - define tab

purchaseItem - Business Transaction

Define | Configure | View

Business Transaction

Purchase_Item_Request_Flow | Process_Payment_Flow | Purchase_Item_Response_Flow

No flow selected

Business Transaction Event Definitions

Business Transaction Details

Here are the business transaction details.

Name:

Description:

Business Events:

Type	Flow Name	Event Source
Start	Purchase_Item_Request_Flow	Purchase Item Input Queue.terminal.out
End	Purchase_Item_Response_Flow	Send Confirmation.terminal.in
Progress	Process_Payment_Flow	Completed Processing Payment.terminal.in

OK Cancel

Now when we send in the events again, we see 4 events shown for customerRequests, 2 events for orderItem and 2 events for purchaseItem which is exactly as expected.

customerRequests – view tab

4 events are shown as expected.

customerRequests - Business Transaction

Define | Configure | View

All instances of customerRequests

Total : 4

Transaction ID	Start Time	Last Update	Status
4	2015-11-27 20:14:50.183	2015-11-27 20:14:50.789	Ended
3	2015-11-27 20:14:50.077	2015-11-27 20:14:50.687	Ended
2	2015-11-27 20:14:45.050	2015-11-27 20:14:45.868	Ended
1	2015-11-27 20:14:44.905	2015-11-27 20:14:45.689	Ended

orderItem – view tab

2 events are shown as expected.

orderItem - Business Transaction

Define | Configure | View

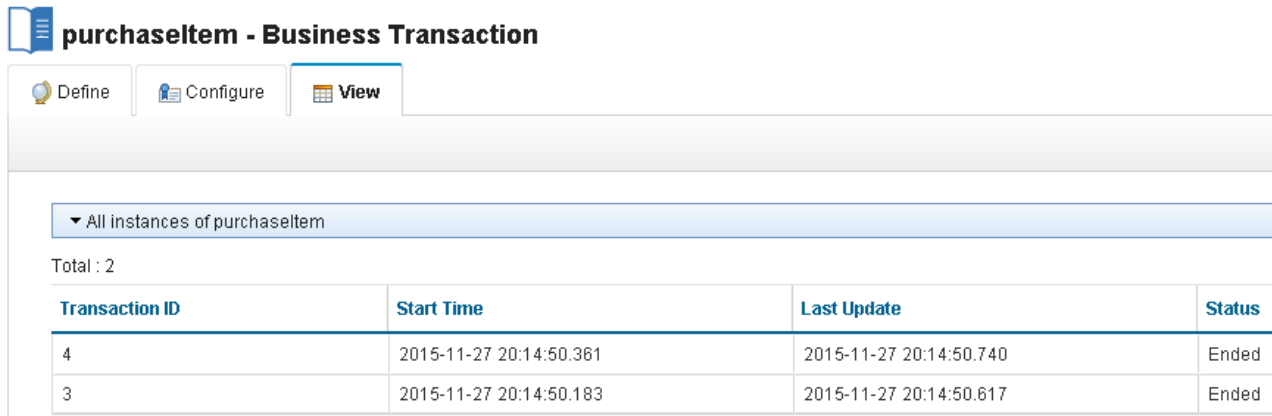
All instances of orderItem

Total : 2

Transaction ID	Start Time	Last Update	Status
2	2015-11-27 20:14:45.233	2015-11-27 20:14:45.805	Ended
1	2015-11-27 20:14:45.034	2015-11-27 20:14:45.580	Ended

purchaseItem – view tab

2 events are shown as expected.



purchaseItem - Business Transaction

Define Configure **View**

▼ All instances of purchaseItem

Total : 2

Transaction ID	Start Time	Last Update	Status
4	2015-11-27 20:14:50.361	2015-11-27 20:14:50.740	Ended
3	2015-11-27 20:14:50.183	2015-11-27 20:14:50.617	Ended

Summary

In this article, we have discussed different areas of administration for the new capability provided in 10.0.0.3, Business Transaction Monitoring. We have shown you how to configure BTM for aspects of MQ, your Database and IIB. You should consider where you define monitoring events and which business transaction definitions should flag those events as business events.

TAGS IBM-INTEGRATION-BUS, IIB, BTM, FESTIVE2015

SanjayNagchowdhury

3 comments on "Advanced usages of Business Transaction Monitoring"

Mark April 20, 2018

"If you wish to modify any of these values, run mqsiexportpolicy to get a file containing the Data Capture policy, update the properties and then run mqsiexportpolicy to set the Data Capture policy."

Simple when you know how...

Roy Ahoor January 25, 2018

Our client has an existing IIB 10 installation utilizing JMS .bindings files to connect to a remotely located queue manager for performing business as usual publish and subscribe operations. Does BTM require a local MQ installation with IIB or is there an alternative approach where the queues listed here are added to a .bindings file for them to use?

The preference is to not create a new MQ co-existence dependency if possible for this BPM implementation.

Sanjay Nagchowdhury January 30, 2018

Hi Roy,

BTM will only work using an Integration Node which has an associated Queue Manager. You cannot use JMS, nor can you use a remote Queue Manager.

Thanks

Sanjay