# Business Transaction Monitoring Performance Considerations

[Dave Gorman](https://developer.ibm.com/integration/blog/author/dave_gorman/)  
Published on March 4, 2016*/ Updated on March 4, 2016*

[0](https://developer.ibm.com/integration/blog/2016/03/04/business-transaction-monitoring-performance-considerations/#comments)

## ****Introduction****

The Business Transaction Monitoring (BTM) capability provided in IBM Integration Bus V10.0.0.3 allows a business user to track the life-cycle of a business transaction that has been processed by multiple message flows. To do this, BTM exploits and builds on the existing message flow monitoring functionality to capture and correlate events that are published by the message flows involved in a business transaction and record this information to a database.

This article explores the performance considerations when using the BTM feature of IIB. It is suggested you also review these other articles in this series:

1. [Business Transaction Monitoring – Why, what, how](https://developer.ibm.com/integration/blog/2015/11/30/business-transaction-monitoring-why-what-how/)
2. [Business Transaction Monitoring in IIB](https://developer.ibm.com/integration/blog/2015/12/04/business-transaction-monitoring-in-iib/)
3. [Advanced usages of Business Transaction Monitoring](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/)
4. [Archiving Business Transaction Monitoring Data](https://developer.ibm.com/integration/blog/2016/01/13/archiving-business-transaction-monitoring-data/)
5. [Business Transaction Monitoring vs Record and Replay](https://developer.ibm.com/integration/blog/2016/01/14/business-transaction-monitoring-vs-record-and-replay/)

## ****Overview****

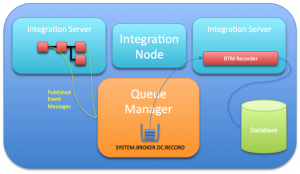
BTM builds on the existing Event Monitoring and Record/Replay features of IIB. Monitoring event messages are published from the message flow threads using IBM MQ PubSub. By configuring a subscription on the monitoring topic, these messages are put to a queue, from which they are read by a nominated Integration Server and written to a database.

The Integration Node also maintains an in-memory cache, which is used to correlate the events of a Business Transaction Definition (BTD). This cache is optimised to ensure viewing of the business transactions and their state performs well. There are no configuration options for the in-memory cache, and therefore it is not a focus of this article.

Instead, this article focuses on the performance considerations of recording the monitoring events.

The following diagram shows the different components required to record business transactions, which are:

1. IBM Integration Bus
2. IBM MQ
3. A database (IBM DB2 or Oracle)

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/overview.png)

As a message flow processes transactions, it publishes (using IBM MQ PubSub) monitoring event messages to a topic. These publications are written to the SYSTEM.BROKER.DC.RECORD queue via a subscription. The ‘BTM Recorder’ gets these messages from the queue and records them to the configured database, as well as maintaining the in-memory cache.

For simplicity, the diagram shows the ‘BTM Recorder’ in a separate Integration Server, but it can be a part of any Integration Server. Also, one or many message flows, deployed to one or many Integration Servers within the same Integration Node can publish monitoring event messages as part of a business transaction.

A single subscription queue (SYSTEM.BROKER.DC.RECORD) and BTM Recorder exist per Integration Node to record and manage all monitoring event messages.

## ****Performance Considerations****

Use the following guidance to ensure the Integration Node and BTM feature performs optimally.

### ****Monitoring events****

Each event message constructed by the message flow, and published, increases the amount of CPU work within the Integration Server. Therefore it is recommended that consideration is given to the number of event messages needed to record a business transaction, and only critical events of the business transaction are published and recorded.

Monitoring events from common message flows may also be included in more than one business transaction definition. If this is the case, the events will be recorded multiple times in the database.

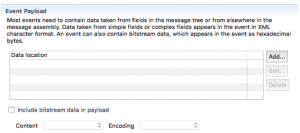
In addition to the number of events, there are several configuration options available when defining a monitoring event on a node within the IBM Integration Bus Toolkit, which can also have an impact on performance.

The first of these is the event filter:

[event filter default](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/event-filter-default.png)

Specifying an event filter increases the amount of work needed to publish a monitoring event. This may be especially true if the expression refers to data not yet parsed. However, if by using a filter the overall number of emitted event messages are reduced then there could be a net benefit. It is also important to remember, events that are part of a business transaction definition cannot be optional.

The second option, is the inclusion of a payload in the event:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/event-bitstream-default.png)

Including payload data may require the Integration Server to parse input data, add the extra data to the monitoring event message and subsequently serialise a larger message.

It is therefore recommended that care is taken when selecting which monitoring event messages to emit, and whether a filter should be applied, as well as whether payload data is really needed.

### ****SYSTEM.BROKER.DC.RECORD queue****

All monitoring events emitted by any Integration Servers for a single Integration Node are subscribed and put to the single MQ queue SYSTEM.BROKER.DC.RECORD.

Depending on the expected load on this queue, consideration should be given to ensure this queue performs optimally. In particular, if the queue depth is likely to increase during busy periods, the queue buffer sizes should be tuned to avoid I/O.

### ****BTM Recorder****

The BTM Recorder gets messages from the SYSTEM.BROKER.DC.RECORD queue, and correlates them as part of a business transaction, whilst maintaining the in-memory cache and writing the event messages to the database.

By default, the Integration Node dynamically determines on start-up which Integration Server will host the BTM Recorder. It is possible to specify an Integration Server for the BTM Recorder, which is discussed in one of the other articles in this BTM series.

### ****Database****

IIB provides the DDL to create the necessary tables and indexes to support the record/replay and BTM features. To ensure the database works optimally, it is recommended that any network latency between the BTM Recorder and the database be kept to a minimum. Also ensure the database has sufficient CPU resources, and is located on the fastest disks possible.

## ****Performance Testing****

As with all performance testing, it is recommended to start small and gradually increase the workload, whilst observing key metrics such as: transaction rate, CPU utilisation, network bandwidth and I/O.

When performance testing BTM, it may be preferable to test different types of business transactions separately, before combining them in a mixed workload test. This is to establish the cost for each of the different types.

The overall performance of publishing a message to the MQ queue and the BTM Recorder getting the messages and inserting them into the database, will determine the maximum number of monitoring events that can be handled by a single Integration Node.

If the rate at which the monitoring events are emitted exceeds that of the BTM Recorders capability, then it is likely that the queue depth of SYSTEM.BROKER.DC.RECORD will start to grow. Whilst this is OK for short periods of time, and assuming that there is an equivalent reduced period of workload to allow the BTM Recorder to catch-up, it would not be sustainable permanently. Therefore, performance of the BTM recorder will depend on the design of the application, the number of monitoring events that are part of a business transaction definition, and the throughput.

As each system on which the BTM solution is deployed is likely to perform differently, it is therefore essential to monitor the queue during the performance test phase, as it will be the indicator for when the BTM Recorder reaches it maximum throughput.

## ****Performance Results****

### ****Configuration****

To demonstrate the guidance above, the following 3 tests were run:

1. Message flow emitting 2 monitoring events (Start and End).
2. Message flow (same as for 1) including an XPath filter.
3. Message flow (same as for 1) including the message payload bitstream as binary.

In all 3 cases the message flow was:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/message-flow.png)

Monitoring events were added to both nodes. For the MQ Input node the monitoring event source was ‘Transaction Start’, and for the MQ Output node it was the ‘In terminal’.

The filter test included the following XPath filter expression for both events:

[event filter](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/event-filter.png)

The event payload bitstream test included the following options:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/event-bitstream.png)

The business transaction definition of the events (BTD) was:

[btd](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/btd.png)

### ****Hardware/Software****

All tests were conducted on the following hardware and software:

The hardware consisted of:

* IBM xSeries x3850 X6 with 1 x Intel(R) Xeon(R) CPU E7-4820 v2
* 2.00GHz processors with HyperThreading turned off
* ServeRAID M5210  SAS/SATA Controller with 4GB Flash/RAID 5 Upgrade option (47C8668)
* 136GB 15K 6.0Gbps SFF Serial SCSI / SAS Hard Drive – ST9146853SS x2 (mounted directly)
* IBM 120GB 2.5in G3HS SATA MLC Enterprise Value SSD – 00AJ395 – x2 (Configured in RAID0)
* IBM 200GB SAS 2.5in MLC SS Enterprise SSD – 49Y6144 – x2 (Configured in RAID0)
* 32 GB RAM
* Emulex Dual Port 10GbE SFP+ VFA IIIr

The software consisted of:

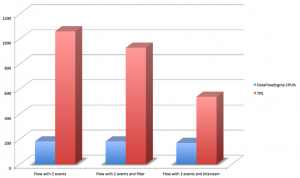
* Red Hat Enterprise Linux Server release 7.2
* WebSphere MQ V7.5.0.5
* IBM Integration Bus V10.0.0.3
* DB2 v10.5.0.7

## ****Results****

The results show the throughput as transactions per second (TPS) of the main flow (MQInput -> MQOutput), and the amount of CPU utilisation of the DataFlowEngine process (Integration Server).

Each transaction through the flow emits 2 monitoring event messages.

The CPU is approximately the same across all 3 tests at 170-180% of 1 CPU. This is only for the DataFlowEngine process, which includes the main flow and the BTM Recorder. IBM MQ and DB2 CPU utilisation is not included.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/03/results.png)

What can be observed from the above graph is the effect on throughput by including either a filter or an event payload bitstream.

Whilst running these tests, the depth of the SYSTEM.BROKER.DC.RECORD queue was continuously monitored.

It was found that at approximately **1000 TPS** on the main flow (**2000 monitoring event messages per second**) the queue depth would start to increase as the BTM Recorder was at its limit.

# Advanced usages of Business Transaction Monitoring

[SanjayNagchowdhury](https://developer.ibm.com/integration/blog/author/sanjay_nagchowdhury/)  
Published on December 7, 2015*/ Updated on December 8, 2015*

[3](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#comments)

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/bowandholly.jpg)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](https://developer.ibm.com/integration/blog/2015/11/30/business-transaction-monitoring-why-what-how/) and the steps to set up and use BTM have been described in [Business Transaction Monitoring in IIB](https://developer.ibm.com/integration/blog/2015/12/04/business-transaction-monitoring-in-iib/).

The following areas are covered in this article:

[**MQ administration for BTM:**](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#mqadmin) 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:**](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#dbadmin) 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:**](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#iibadmin) 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:**](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#planbtm) 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

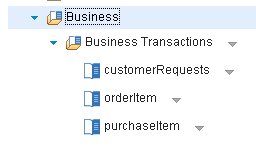
Monitoring events are configured on message flow nodes using the IBM Integration Toolkit or by creating a monitoring profile. When a message is processed by a node which has an event configured on it, an MQ message is published to an MQ topic. The MQ topic name has this structure:

$SYS/Broker/integrationNodeName/Monitoring/integrationServerName/flow\_name

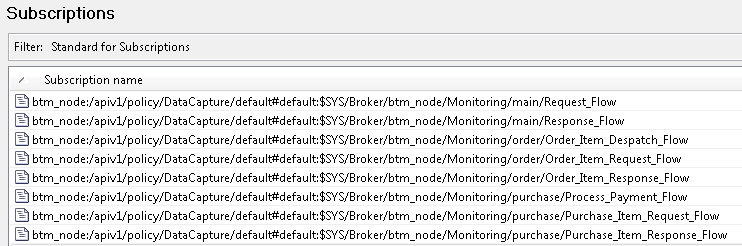
When a Business Transaction Definition is created, a selection of monitoring events are flagged as business events. They are deemed crucial to understanding the state of a business transaction. An MQ subscription is created for each topic that is used by these monitoring events that have been flagged.

[Read more...](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#pnext_collapsible_1)

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

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig01.png)

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

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig02.png)

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:

mqsichangepolicy 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](https://developer.ibm.com/integration/blog/2015/12/04/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.

[Read more...](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#pnext_collapsible_2)

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:

1. Only a Progress event has been received but no Start
2. Only a Progress and End event has been received but no Start
3. Only an End event has been received but no Start
4. 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.
5. 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 mqsireportpolicy to get a file containing the Data Capture policy, update the properties and then run mqsichangepolicy to set the Data Capture policy.

You can run mqsireportpolicy 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.

[Read more...](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#pnext_collapsible_3)

### 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 mqsireportpolicy 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.

[Read more...](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/#pnext_collapsible_4)

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.

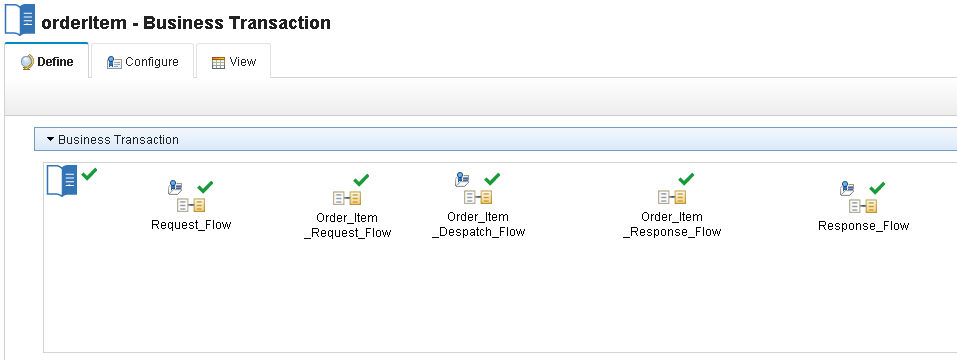
I have a simple Retail system which receives customer requests and forwards them to three different business areas: Stock Checking, Ordering and Purchasing. The responses are received from the difference business areas and sent back to the customer.

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

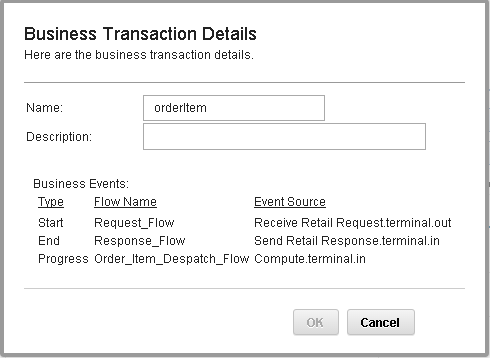
I have separate applications containing message flows for the different business areas.

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.

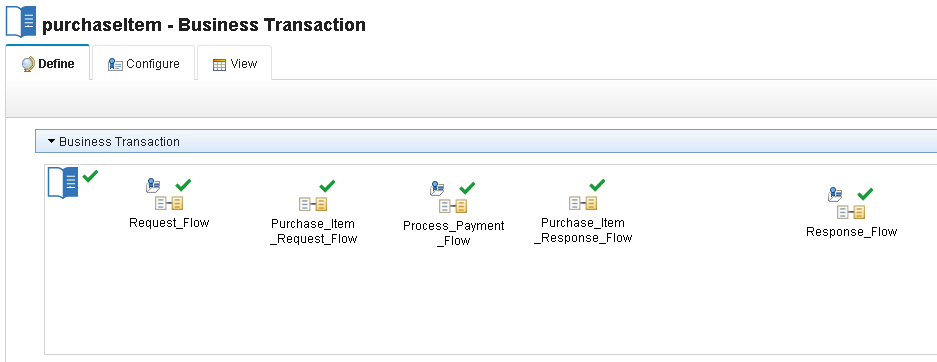
The orderItem BTD has these flows with events configured on 3 of the flows. You can see which flows has events configured on it as it has a[fig03](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig03.png) icon above the flow.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig04.png)

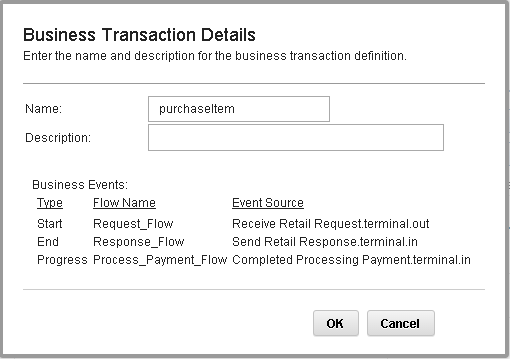
These events have been flagged as business events for the orderItem BTD:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig05.png)

The purchaseItem BTD 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.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig06.png)

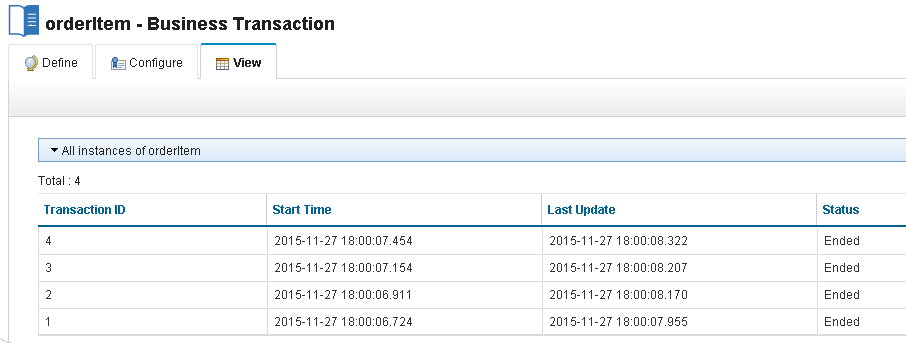
These events have been flagged as business events for the purchaseItem BTD:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig074.png)

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!!

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig08.png)

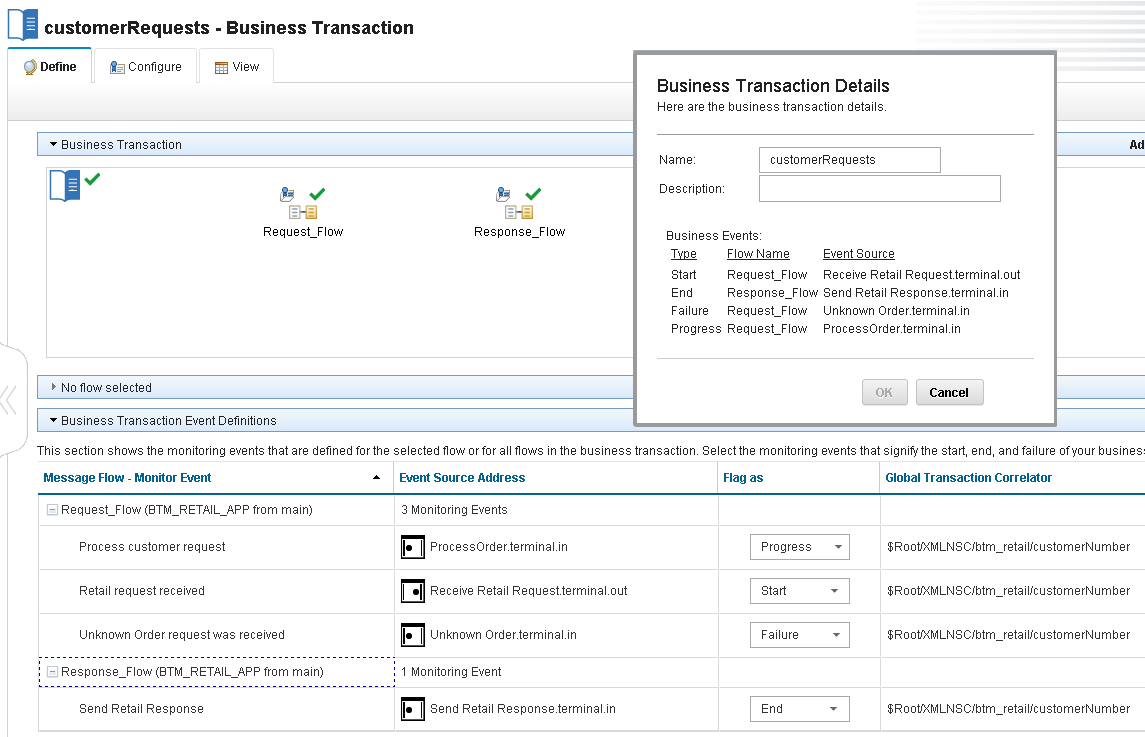
This has happened as the start and end events for both business transactions were defined in a common flow. The Start and End event were captured for all events by both BTDs. The business transaction definitions should have been split up in a better way so that we only see the transactions for orderItem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

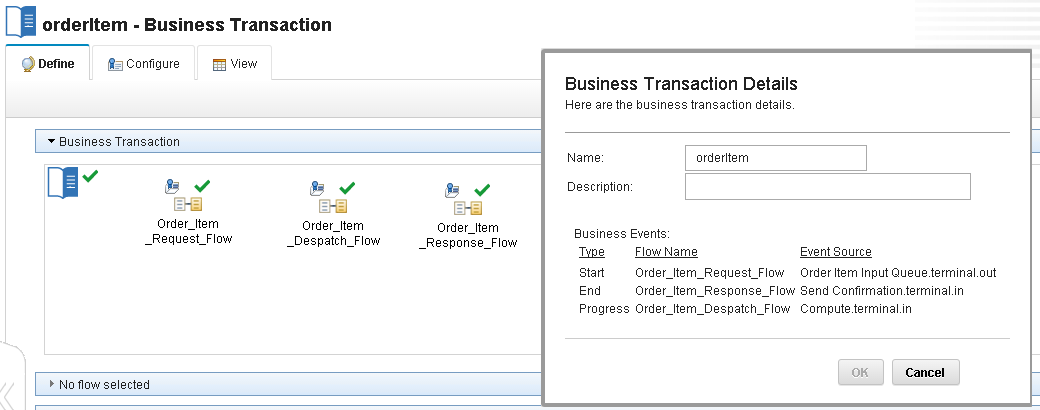
Here is a better structure for the BTDs.

1. I have defined a general BTD called customerRequests which has Start, Progress, End and Failure events on the Request\_Flow and Response\_Flow flows.
2. I have removed the Request\_Flow and Response\_Flow flows from the orderItem and purchaseItem flows.
3. For orderItem, I have defined the Start event in the Order\_Item\_Request\_Flow and the End event in the Order\_Item\_Response\_Flow.
4. For purchaseItem, I have defined the Start event in the Purchase\_Item\_Request\_Flow and the End event in the Purchase\_Item\_Response\_Flow.

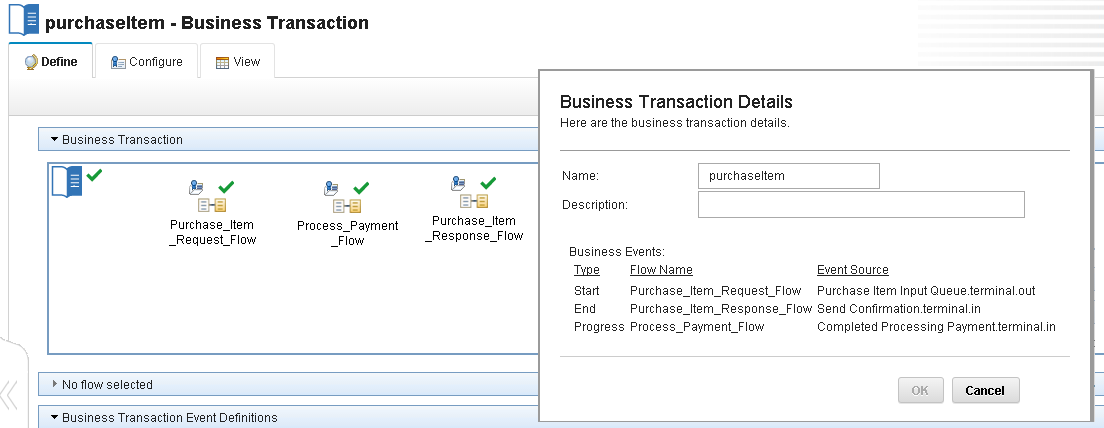
### customerRequests – define tab

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig09.png)

### orderItem – define tab

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig10.png)

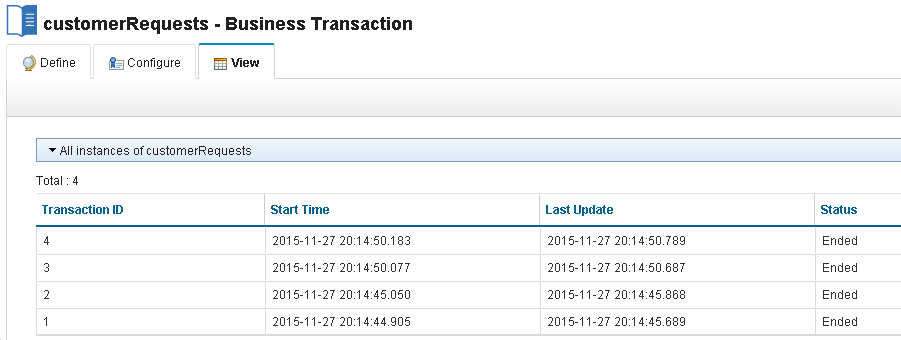
### purchaseItem – define tab

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig11.png)

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.

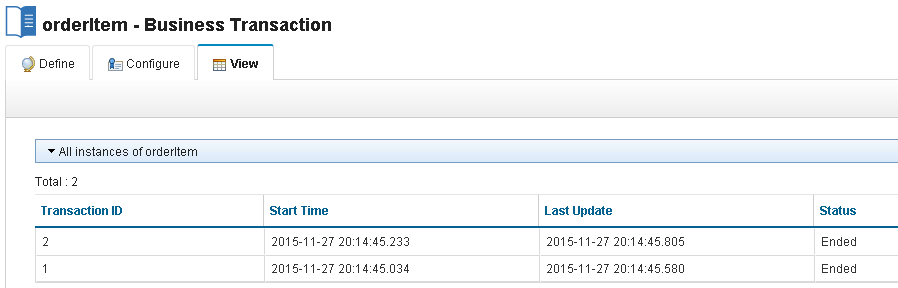
### customerRequests – view tab

4 events are shown as expected.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig12.png)

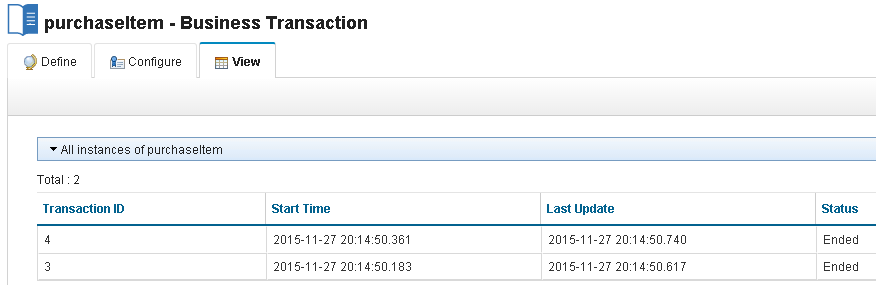
### orderItem – view tab

2 events are shown as expected.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig13.png)

### purchaseItem – view tab

2 events are shown as expected.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/fig14.png)

## 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.

PS: Do you like the festive IIB posts? You can easily find them all by having a look at our [festive calendar](https://developer.ibm.com/integration/festive-greetings-and-best-wishes/)or click on the [festive2015 tag](https://developer.ibm.com/integration/blog/tag/festive2015/).

# Business Transaction Monitoring vs Record and Replay

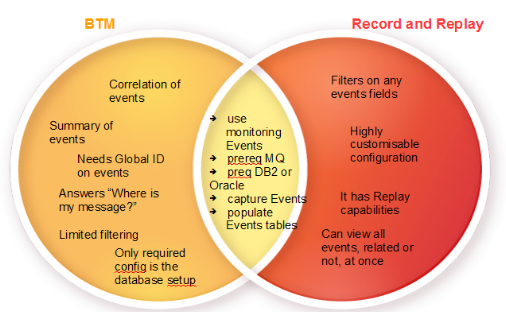
[DoinaKlinger](https://developer.ibm.com/integration/blog/author/dklinger/)  
Published on January 14, 2016

[3](https://developer.ibm.com/integration/blog/2016/01/14/business-transaction-monitoring-vs-record-and-replay/#comments)

The recently released Business Transaction Monitoring (BTM) allows a business user to track the life cycle of a business transaction that has been processed by multiple message flows. Events are captured, correlated and a summary, the business transaction state, is calculated. Previous articles have described how you [get started with BTM](https://developer.ibm.com/integration/blog/2015/12/04/business-transaction-monitoring-in-iib/) and [how customise it further](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/).

IBM Integration Bus advanced users might recognise some of these concepts in the existing Record and Replay (RR) feature that have been available in the product since version 8.0.  
This article compares and contrasts the two sets of capabilities with a view to guiding you when you might want to use one versus the other and explains a few considerations to take into account if you use them side by side.

**Features**

Here are the two features at a glance:  
Let’s explore the common area and the differentiators for each of these technologies.  
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/btmVsRR.png)

**BTM and RR common points**  
An event is a processing step that occurs during an IIB integration. It contains information about the source of the event such as which message flow and node and which action was performed e.g. transaction started/ended or a node terminal fired. The event may include the message payload and exception list. When a message flow is processed, the events configured on its nodes are emitted to the MQ topic.  
To add events to monitoring flows you will need to use either the toolkit or the mqsichageflowmonitoring command.

Both BTM and RR are consumers of these monitoring events. Other consumers might also be configured. Both BTM and RR capture events. They set up subscriptions to the corresponding topics for message flows like  
$SYS/Broker/integrationNodeName/Monitoring/integrationServerName/flow\_name  
that receive the events and put it to a destination queue. Messages are read from this queue and recorded into the database.  
As this capturing mechanism is shared by both BTM and RR, they both prereq MQ and can only be configured on integration nodes that have a queue manager specified.

Both BTM and RR use tables of the same structure to record events. However, as explained later, it is better to use different schemas when using both BTM and RR and not to have them write to the same table as it will make it more difficult for correlation capabilities in BTM.  
In both cases, you need to configure the database for use by IIB. This involves running the provided scripts to create the WMB\_MSGS table for recording monitoring events and the WMB\_BINARY\_DATA table for storing payload data and exception information. Additionally, you must configure the environment to allow IIB to make ODBC connections to the database and use mqsisetdbparms to set a user identifier and password for that connection.

**BTM only capabilities**

Below are some characteristics that are BTM specific and cannot be done by RR.

Correlation and Global ID  
BTM correlates events in order to report the state of a business transaction in a way RR cannot. BTM correlation works by analysing a received event to see if it is mentioned in any business transaction definition. If that is case, the global ID is used to determine the particular business transaction instance this event belongs to and its effect on the business transaction state.  
As described, the global ID is essential for BTM. The monitoring events need to have a global ID specified with an Xpath as opposed to selecting the ‘automatic’ option when specifying the event. RR does not depend on a global ID being specified.  
So both the correlation and the requirement to have a global ID specified are extras of BTM over RR.

BTM correlation does not come for free and has a cost. You can choose what which monitoring events to include in the business transaction and depending on your requirements, so you might want to flag only those events that you deem critical for monitoring your business transaction. As the correlation is a BTM only, the extra correlation costs are incurred only when using BTM.

Summarises events  
BTM summarises correlated events in the business transaction states. You can see at a glance the state of the business transaction. It is easier to understand what happened to a given request by seeing the events that make a business transaction.  
These are the main use cases for BTM, so clearly you cannot achieve this with RR and these are some keys differentiators.

Configuration  
Once the monitoring events have been defined with the globalIDs set up, the required part of the configuration is database related. The database needs to be configured as described above in the common parts. In addition, to the common database configuration described above a new table for business transactions, WMB\_BUSTRANS, also needs to be created by running an sql script server\ddl\db2\BusinessCaptureSchema.sql  
BTM uses a default data capture policy for its configuration. A default policy is available out of the box and only needs a database name set to be operational. Even taking into account the extra table that needs to be created, configuring BTM is easier and relies on a set of defaults and runtime decisions on servers to use for recording and viewing.

Grouping and filtering of results  
The results of business transactions are grouped by the type of BTD and you cannot mix and match, say PurchaseOrder business transactions with StockAlert business transactions. You can filter by the status of a business transaction, failed, in progress, which is a new BTM specific capability.

**Record and Replay only**

Here are the Record and Replay characteristics that cannot be done by BTM.

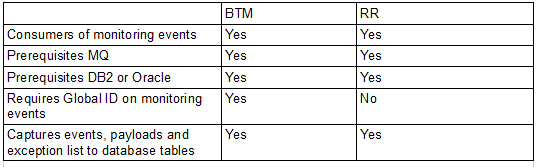
Event filtering  
With RR, it is possible to filter on any event fields, including time based queries based on timestamps, and to combine any number of conditions. With BTM, you see the events for a particular business transaction instance, but there is no further event filtering. The only filtering is at the business transactions level when you can choose a specific transaction ID and a transaction state.

Configuration  
There is more flexibility in distributing the recording and viewing work on the servers but at the cost of more complicated configuration via Configurable Services, Data Capture Store and data capture source and Data Destination.

Event View per store  
You can see all the events captured to a store, regardless of their source or whether they are related or not. This can be an advantage or not.

Replaying events  
Another difference has a clue in the name: with RR you can select events to replay. Not only does BTM not support replay, but replaying certain messages can interfere with the summarising capabilities and reporting of the state of the business transaction. Say a start message of a completed business transaction is replayed. This would cause the business transaction to be reported as inconsistent.

Finer grained access rights  
RR can be configured for a datacapture type of user. Combining these with broker access right, it is possible to configure set up a user that can only do RR operations. This is not supported for BTM.

To summarise:  
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/btmVsRRTable.png)

**Using both BTM and RR**

With these characteristics, you can hopefully choose better what technology is right for you. What if you have deployed IIB applications with different monitoring requirements and need to run both BTM and RR?  
Then you need to take these points into account.

Separation at data level  
It is recommended that you do not share the event tables that RR and BTM use even if they have the same structure. The reason for this is that if you have flows monitored by both RR and BTM you might have the same event recorded twice which might give you unexpected results for your RR queries.  
One way to achieve the data separation is to have different schemas, so you would have two sets of tables, say BTM.WMB\_MSGS, BTM.WMB\_BINARY\_DATA and RR.WMB\_MSGS and RR.WMB\_BINARY\_DATA  
Alternatively, you can use different databases, one for BTM, the other for RR.

Separation at server level  
It is recommended that you distribute your BTM recorder and your RR recorder to different integration servers. The main reason is performance.  
The simplest way to achieve this is to have different integration nodes dedicated to the different types of monitoring. If you need to mix and match the technologies on the same node, you will need to specify different servers. For RR in the dataCaptureStore configurable service you will need to specify an egForRecord and egForView. This is described here https://www-01.ibm.com/support/knowledgecenter/SSMKHH\_10.0.0/com.ibm.etools.mft.doc/bn28727\_.htm

BTM has a default data capture policy that means the runtime decides which server to use for recording and viewing. To overwrite these, you will need to modify the default data capture policy to assign the recording and viewing to different servers than those used by RR. How to do this is describe [here](https://www-01.ibm.com/support/knowledgecenter/SSMKHH_10.0.0/com.ibm.etools.mft.doc/bn28820_.htm?lang=en)

Separation at queue level  
Again for performance reasons, you might want to configure BTM and RR to use different queues for recording and backout. When a monitoring event is emitted it is published to a MQ Topic on the associated Queue Manager. The relevant subscription picks up the event and puts it to a destination queue. For BTM, the destination queue is defined in the default Data Capture policy to be SYSTEM.BROKER.DC.RECORD. A further queue is also needed to handle events that cannot be at process time be written to the database, and thus are backed out onto this queue. The default is SYSTEM.BROKER.DC.BACKOUT. The same concepts exist for RR. They are specified in the queueName and backoutQueue properties of the Data Capture Store Configurable Service. For improved performance it is recommended that you overwrite one of these defaults, say in the Data Capture Store Configurable Service to have BTM and RR use different recording and backout queues.

Note that other configurations are also possible.  
The article has described the similarities and differences between BTM and RR, the means to choose or to combine them.

Business Transaction Monitoring in IIB

[MattClarke](https://developer.ibm.com/integration/blog/author/mattclar/)  
Published on December 4, 2015*/ Updated on March 22, 2016*

[19](https://developer.ibm.com/integration/blog/2015/12/04/business-transaction-monitoring-in-iib/#comments)

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/hero_530_star.png)With the availability of IBM Integration Bus v10 Fixpack 3, we introduced the concepts on Business Transaction Monitoring (BTM) in [Business Transaction Monitoring – Why, what, how](https://developer.ibm.com/integration/blog/2015/11/30/business-transaction-monitoring-why-what-how/).

**In this blog post** we will detail an end-to-end example scenario involving BTM from definition to viewing of transaction results.

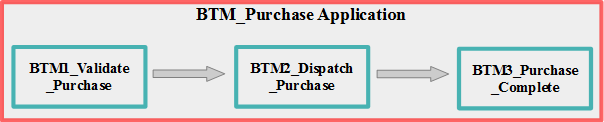
We will create a new Business Transaction Definition (BTD), perform the configuration steps to leverage recording of BTM events in preparation for sending messages that will be monitored by our definition. Then view the results of the recorded business transactions.

**What is required**

* An IIB node associated with a queue manager. (The monitoring feature in IIB requires WebSphere MQ to function)
* DDL scripts to generate required table
* The IIB resource BTM\_Purchase.bar that contains the application that will be used
* create\_queues.mqsc script
* Sample messages BTM\_NewPurchase.xml and BTM\_InvalidPurchase.xml files
* IBM DB2 (or other supported Database – Oracle)

[IIB Getting Started with BTM Artefacts .zip](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/IIB_Getting_Started_with_BTM_Artefacts.zip)

**Scenario**  
In our application there are 3 message flows that form a path for purchase order messages to be processed across.  
BTM1\_Validate\_Purchase message flow performs initial validation checks on the message.  
BTM2\_Dispatch\_Purchase message flow receives a valid message and sends it out for dispatch.  
BTM3\_Purchase\_Complete message flow completes the purchase order.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/Scenario_Overview.png)BTM\_Purchase application overview

The purpose of these message flows is to facilitate the ordering of purchases made by a customer by validating the purchase message, and then forwarding onto dispatch and purchase.  
The end to end execution of this can be considered to be one single transaction, and as a business level entity (making a purchase) it is known as a business transaction.  
Suppose you would like to know about when such transactions occur, how they are handled, how many there are, when they occurred and so on?  
This is where the BTM feature offers a representation of a defined business process, in a multi-flow, event-driven environment that can be defined and viewed as a set of transaction results.

**MQ Setup**  
The messaging technology used for this scenario is MQ, and several queues need to be created in order to facilitate the sending and receiving of messages between flows.

MQ is also required for using BTM, so the integration node must have an associated queue manager using the -q parameter when the node was created, or when running the mqsichangebroker command. MQ is used for publishing the monitoring events to MQ topics. BTM sets up subscriptions to these MQ topics to process and correlate the business events which contribute to a particular business transaction.

Several queues need to be created under the queue manager. Running the script against the relevant queue manager create\_queues.mqsc will create the correct queues. The queues that are created are bespoke for the application, but there are 2 further queues needed for BTM to function.

When a monitoring event is emitted it is published to a MQ Topic on the associated Queue Manager. These subscriptions purpose is to handle event emission and write the event to a MQ queue, the SYSTEM.BROKER.DC.RECORD queue, which queues up event messages to be processed and written to a database. A further queue SYSTEM.BROKER.DC.BACKOUT is also needed to handle events that cannot be at process time be written to the database, and thus are backed out onto this queue. These queues need to be created prior to using BTM, and are generated by running the script located at *{install path}/sample/wmq/iib\_createqueues.cmd*.

**Database Setup**  
To facilitate the recording of transactions, a supported database must be configured before using BTM. For BTM this tutorial uses IBM DB2.

If Record & Replay is currently used, then only a single database table needs creating for BTM. If it isn’t then a new database and tables need to be created first. The IIB installation includes an SQL script, located at *{install path}/ddl/db2/DataCaptureSchema.sql* to generate the necessary database tables. You will need to customize the script to suit your recording needs and database naming standards. At the top of the file, you will find the database name MBRECORD, and you can optionally use a database schema.  
Run this script using the command:  
**db2 -tvf DataCaptureSchema.sql**  
from a db2 command shell. Check that the statements executed correctly before continuing.

To define the BTM table, the IIB installation includes an SQL script , located at {install path}/ddl/db2/BusinessCaptureSchema.sql.  
Run the script using the command:  
db2 -tvf BusinessCaptureSchema.sql

The next task is to create an ODBC datasource for this database so that the integration node can connect to it. This task is platform-dependent. For more information, see [Enabling ODBC connections to databases](http://www-01.ibm.com/support/knowledgecenter/SSMKHH_10.0.0/com.ibm.etools.mft.doc/ah14440_.htm) in the IBM Integration Bus v10 Knowledge Center.

In this example, we use the name MBRECORD for the ODBC definition to match the database name.  
The database should now be configured and ready to use, and you can now connect the integration node to the database. For example you would use the following mqsisetdbparms command to store the database credentials:

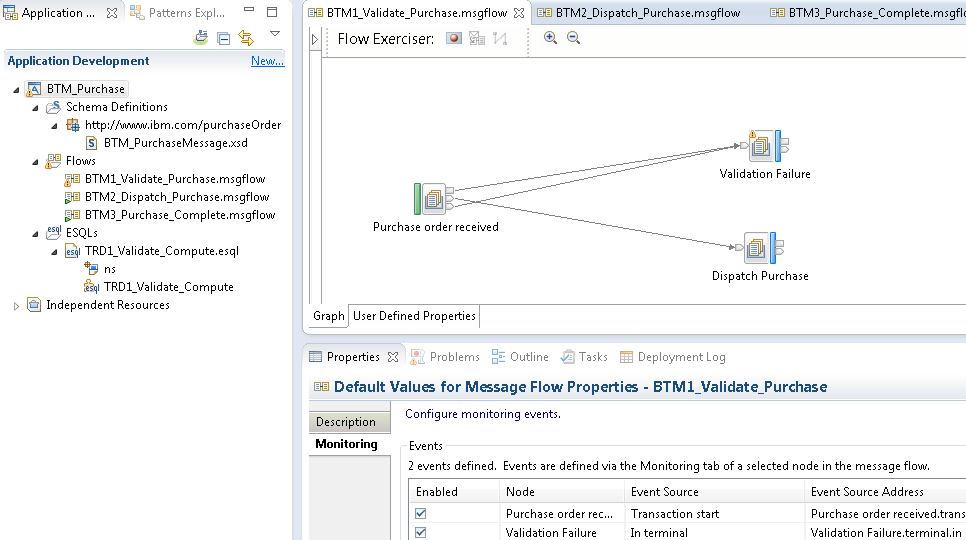
mqsisetdbparms IB10NODE -n odbc::MBRECORD -u {DBUSER} -p {DBPASSWORD}

After this command completes successfully, restart the integration node to pick up the new credentials. Then use the mqsicvp command to verify that the database connection is correct:

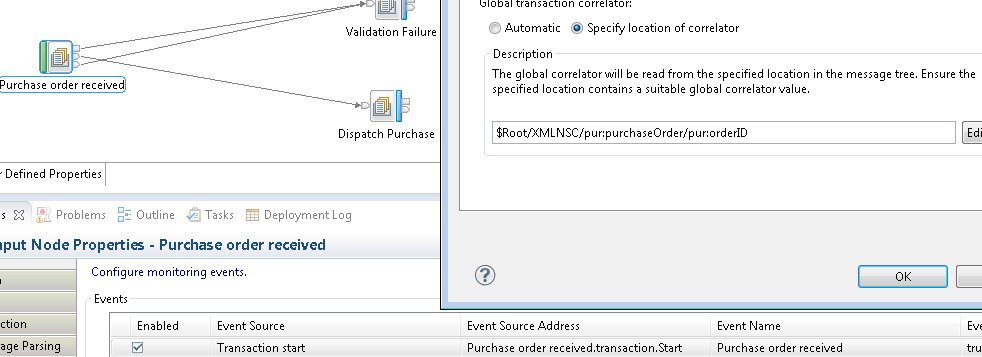
mqsicvp IB10NODE -n MBRECORD

If successful, this command returns a list of supported operations against the database.

**Working with the BTM\_Purchase Application**  
BTM uses the existing message flow monitoring capability to record and correlate events across a business transaction. To enable this for BTM, there is a requirement to specify such a correlator throughout the intended flow of a transaction to be monitored. For BTM this is the global transaction ID. This must be specified when defining monitoring events within a message flow to be used for recording business transactions. You must define the value of a global transaction ID in each monitoring event. The global transaction ID is used to correlate the individual events which contribute to the business transaction. The global transaction ID is typically some part of the message that is being processed, for example, a customer number or an order ID.

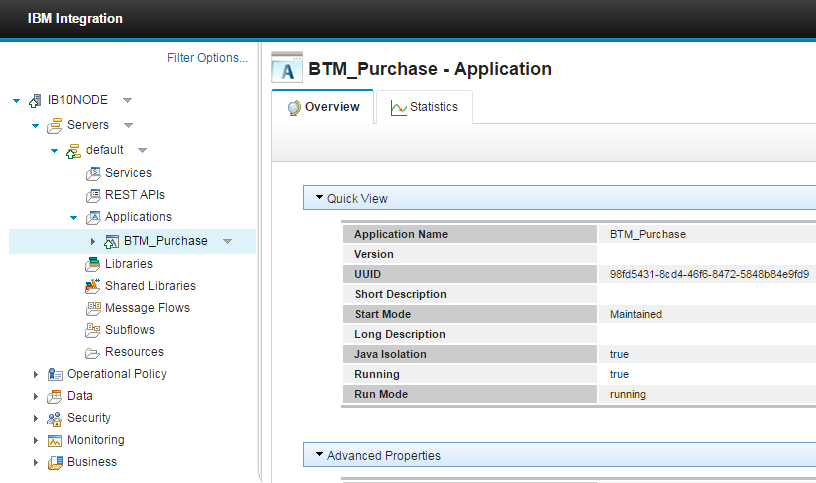
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_Purchase_App_Toolkit.png)Overview of BTM\_Purchase in IBM Integration Toolkit

The application’s message flows already have the required monitoring events defined with a global transaction correlator value set. In this scenario we use the purchase order ID in the message.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_GlobalID_Toolkit.png)Monitoring event global transaction correlator value

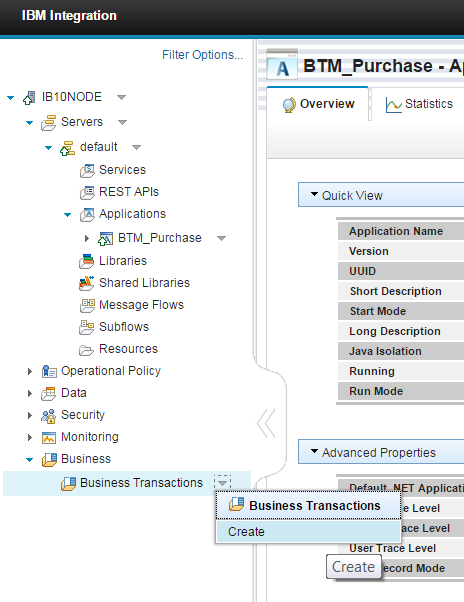
All other monitoring events defined on the application’s message flows also use the same global transaction correlator value. This means that in order to be able to distinguish between recorded business transactions, a single message’s purchase order ID must be unique from other’s that have previously been processed.

**Creating the Business Transaction Definition**  
After deploying the BTM\_Purchase application to your integration node, all remaining configuration can be performed using the IBM Integration Bus WebUI for your Integration Node. Once deployed you may typically have a WebUI tree populated in the following manner:

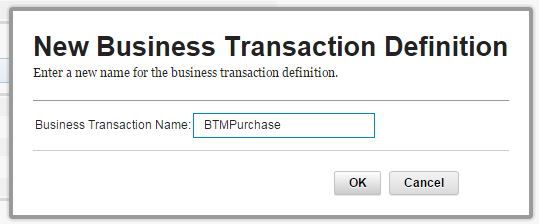
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Tree.png)IIB WebUI tree with BTM\_Purchase application deployed

Notice that for IBM Integration v10 Bus Fix pack 3 there is now a new category in the tree named “Business”. This is where all the BTM capability is accessed in the WebUI.

BTM uses the concept of a Business Transaction Definition (BTD) that contains the configuration for the transaction. In order to be able to start monitoring transactions a new definition must be created. Go to the business section of the tree and expand the Business node. Then to create a new definition click on the down-arrow to bring up the business transactions options, and select Create.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Create_New_BTD.png)Create new BTD in Business category

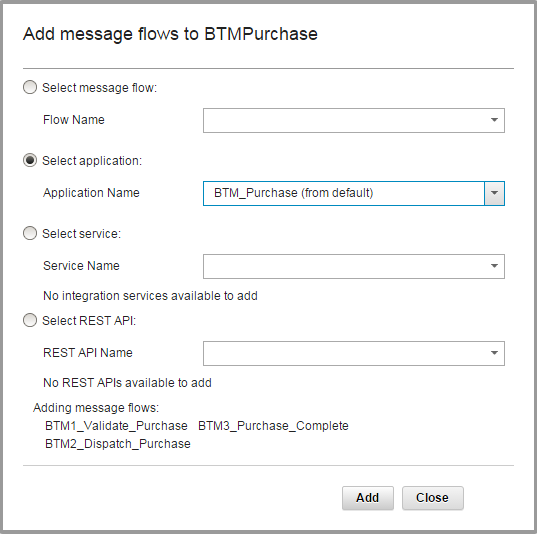
When prompted, specify a name for the new BTD and confirm by clicking OK.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_New_BTD_Dialog.png)Specify new BTD name

This creates a new BTD instance on your integration node, ready to be defined. There are 3 tabular sections for a BTD:  
Define – Where the BTD structure is defined and visualised.  
Configure – Where configuration information is given and potentially guided tasks to resolve any problems.  
View – Where results can be viewed for recorded business transactions for this definition.

**Define the definition**  
A BTD is able to work with artefacts that are currently deployed to the same integration node. In this instance we want to work with our recently deployed BTM\_Purchase application and the message flows that it contains.

You’ll notice that there is a blank canvas in the top section. The canvas is a visual tool to illustrate the currently included message flows in the definition, and provides additional information related to the definition that is relevant to BTM. To populate the definition with message flows the mechanism provided is to use the Add Flow button. This will display the Add flow dialog. Here all deployed artefacts are placed into their respective categories as a series of selectable drop-down lists.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Add_Flows.png)Add BTM\_Purchase application message flows to BTD

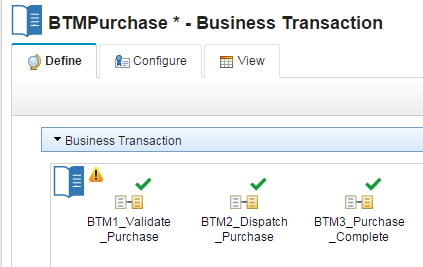
To add a single message flow at a time, the message flow list has all available message flows deployed on the integration node that are not currently included in the definition.

To add all message flows within a single application, the application list has all available applications deployed on the integration node. Any message flows currently included in the definition that are a part of the selected application will not be added.

To add a single Integration Service object, the services list has all available Integration services deployed on the integration node.

To add a single REST API object, the REST APIs list has all available REST APIs deployed on the integration node.

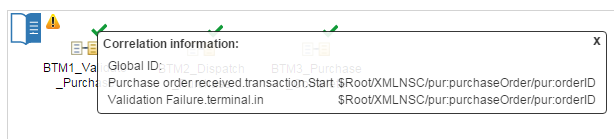
As we want to include monitor events that span across multiple message flows in our business transaction, then it is easier to add all message flows in the BTM\_Purchase application. When an application is selected, a summary is presented of the message flows that will be added upon confirmation. Clicking Add will add the message flows to the definition and each flow is rendered as a message flow artefact on the canvas.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_BTD_Flows.png)New BTD with added message flows

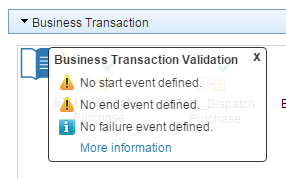
An \* (asterisk) next to the BTD name now denotes that modifications have taken placed since the last save of the definition. This is the mechanism used throughout the define task to denote that there are currently unsaved changes against the definition.

If a message flow is not required to be a part of the BTD, then they can be removed using the remove flow button to bring up a similar dialog as is used when adding message flows. A message flow can also be deleted by selecting it on the canvas and pressing the delete key.

**Available Monitoring Information**  
On the define tab there are several methods for inspecting and discovering information about the message flow that you have just added.  
Validation – Clicking on the green tick brings up information on the validation assessment of the message flow with respect to BTM and the definition. In this instance everything is resolved to ok for the message flow because it contains some monitoring events on its nodes. We can also see the global transaction ID correlator information for all monitoring events that are defined on message flow nodes for the message flow.

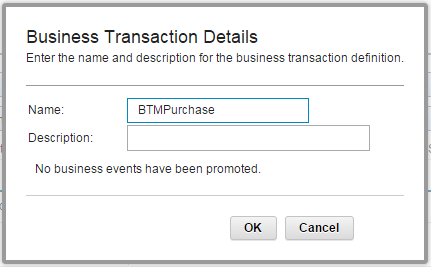
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Flow_Info.png)Monitoring event correlation information for a message flow

If the message flow is not valid a warning icon will be shown, and clicking on that will give further information as to what is not currently valid. Such a warning icon can be seen next to the BTD icon on the canvas. This object is a representation of the current BTD entity. Clicking on the warning icon will give details as to why the BTD might not currently be defined correctly:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_BTD_Info.png)BTD validation overview

No business transaction events have yet to be defined, so the BTD is not complete and not ready to start recording business transactions.

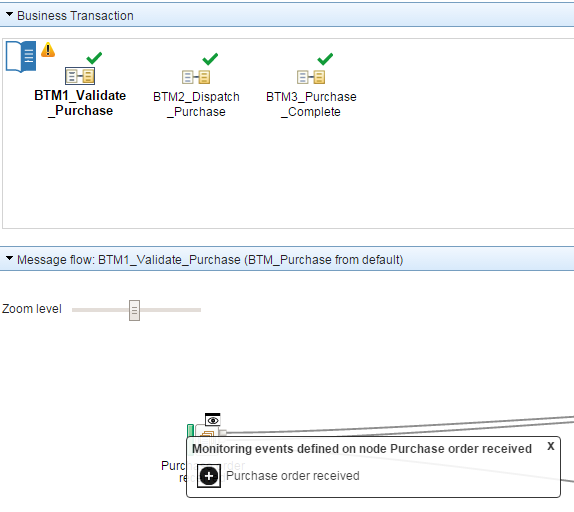
You can also inspect a summary of the BTD by clicking on the BTD icon itself. A new dialog will appear and show information on the BTD:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_BTD_Overview.png)BTD definition overview

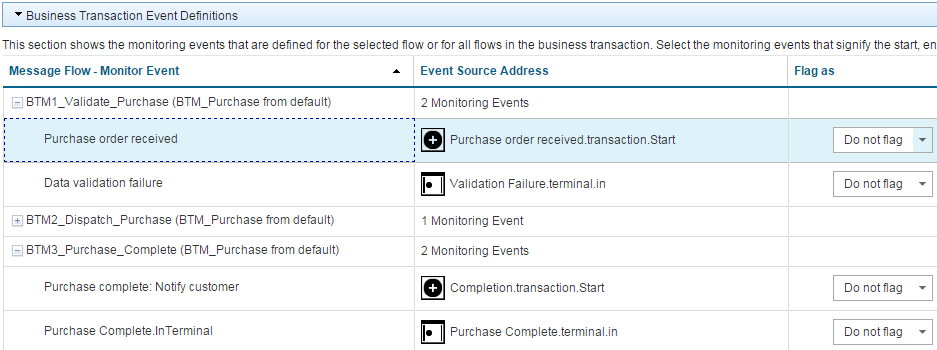
As is confirmed by the validation warning, no business events have been defined yet.

**Flagging Business Events**  
Currently there are added message flows that have valid monitoring events associated with them but by default they are not included in the business transaction. The definition needs to know which events it needs to include in its monitoring of a transaction. This is achieved through the “Flagging“ of monitoring events to become business events.

To assist in making such a decision, the middle section provides a flow profile view of a message flow when it is selected on the canvas. Inspection of event information (denoted by the event icon) on individual nodes provides a visual message flow perspective of what monitoring events are available to be flagged to business events.

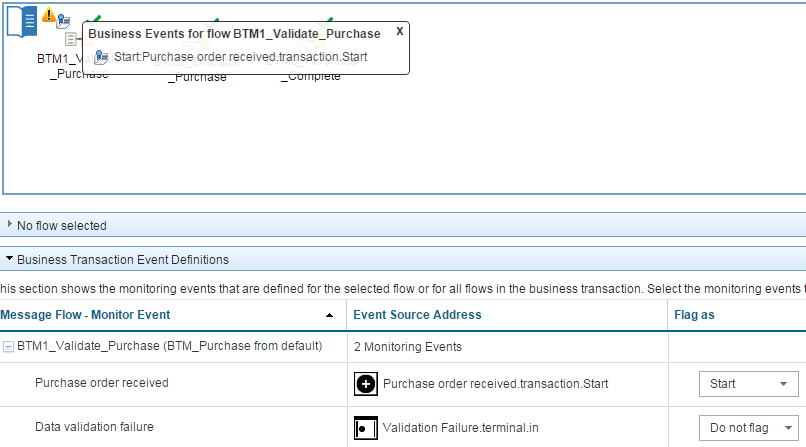
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Inspect_Flow.png)Inspection of monitoring events on flow profile node

The bottom section on the define tab provides a grid view of the current definition. When no message flow is selected it provides a tree grid of all included flows, and their respective monitor events details. When a message flow is selected on the canvas, the scope of the grid narrows to that message flow monitoring event information.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Flow_Event_Table.png)BTD Message Flow Event Grid

It is within this grid that the flagging of monitoring events to business events is performed. Changing the value of the drop-down in the Flag as column will change the role of that monitoring event with respective to the current BTD. By default every monitoring event is set to Do not flag, so it will not participate in the business transaction. Any other value will mean it has some meaning to the business transaction and will be recorded when an event is emitted.

For example changing the Purchase order received monitor event in BTM1\_Validate\_Purchase message flow to have a Flag as value of Start will cause a new business transaction to be recorded each time an event is emitted at this location. The inclusion of a business event within a message flow now decorates the flow object with a business event icon on the canvas that when clicked details the current business event(s) for that message flow.

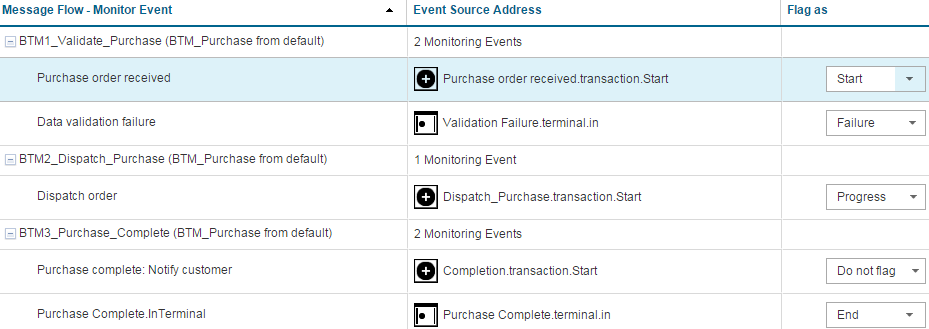
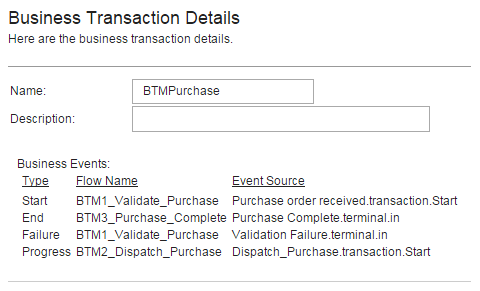
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Flagged_Event.png)Monitoring event flagged as Business event for BTD

To end the transaction we want to ensure that the purchase has complete. This means setting a business event on a monitoring event in a different message flow, the Purchase Complete.InTerminal event in the BTM3\_Purchase\_Complete message flow.

We now have the minimum required number of business events set to be able to record a business transaction, a start and an end.

For this scenario we want to also know when a received message has failed validation, and complete the business transaction as failed. Setting the monitoring event on that validation failure node to be a Failure business event will ensure that a business transaction can be distinguished between failed and completed.

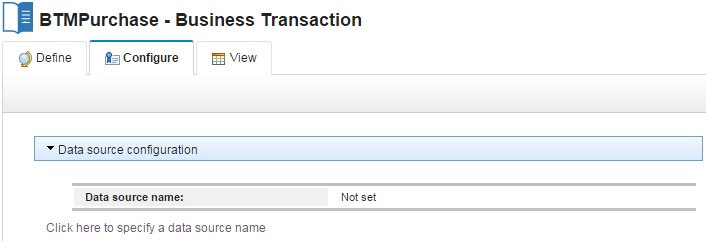
A value of Progress can also be set on a monitoring event as the business event type. These events are recorded as part of the business transaction, but don’t pertain a particular business meaning. They may still be important for example in order to track the path of a message through the message flow, or they have some valuable message payload associated at this point that might want to be recorded. After setting a further monitoring event to Progress the final definition has the following structure:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Final_Definition.png)Final BTD flagged events[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Final_Definition_Overview.png)Final BTD overview

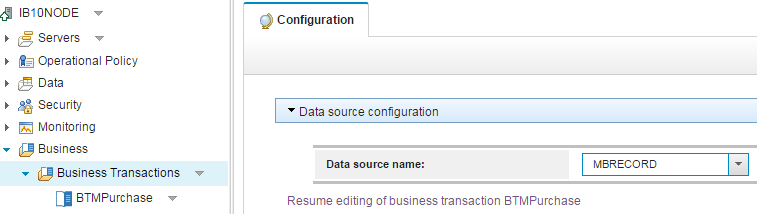
Upon the completing of any changes to a BTD these must be saved by clicking the save button in the toolbar in order to be persisted. Any changes that do not want to be changed can be undone by clicking the cancel button and BTD will revert to the last saved state.

**Completing BTM Configuration**  
Now that the definition of the BTD is complete, there is one or potentially two more setup steps that need to be complete in order to prepare the integration node for the recording of business transactions. This is performed on the Configure tab for the BTD.

The first is that the data source to which business transactions will be recorded to needs to be defined. The default value for first time use is set to empty (or Not set), so this needs to be set for BTM to function. Note that the data source name that is chosen will apply for all BTDs.

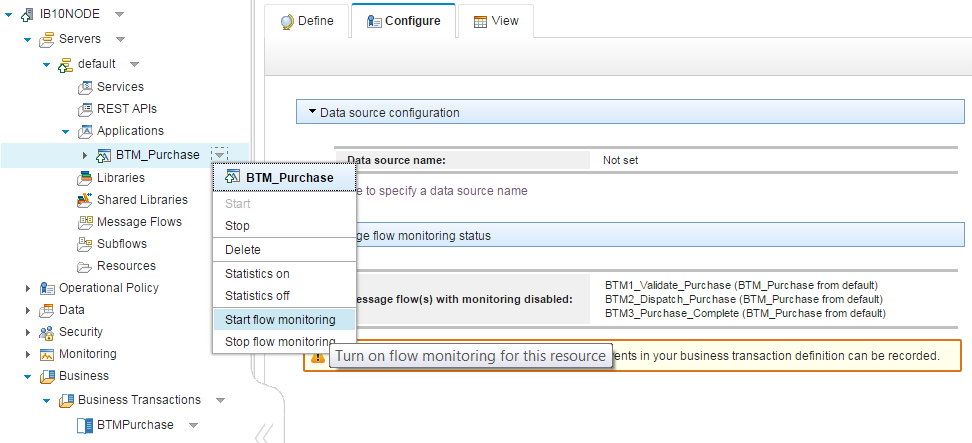
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Configure_Initial_DSN.png)Default configuration data source name

There is click-able link “Click here to specify a data source name” that takes you to the configuration page for BTM. This is also the Business landing page in the tree location. We will set this to the data source name we specified in the mqsisetdbparms command, or MBRECORD as this should be available from the drop-down list.

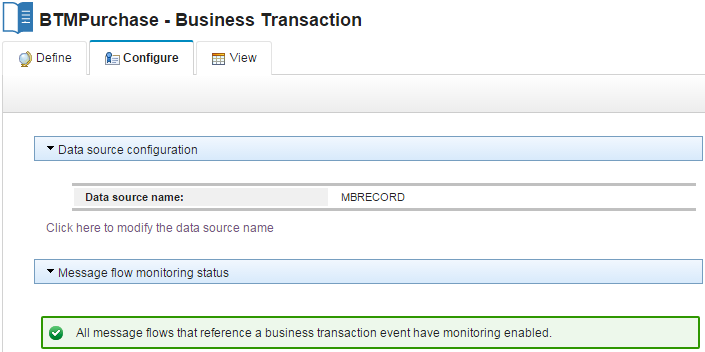
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Configure_Set_DSN.png)Set the data source name for BTM configuration

Once set, clicking save will store this value, and begin using the associated database for BTM. To resume editing of the BTD that we came from, there is a click-able link that will take a user back to the configure tab for that BTD.

There is just one more configuration step that needs to be performed if not done so already. That is to turn on flow monitoring for those flows that will be part of the business transaction. Under the flow monitoring status section, the current state of the BTD message flows have been assessed and any warnings shown. Turning flow monitoring on and off can is a new feature available in IBM Integration Bus v10 Fix Pack 3. Either at the message flow level, or the parent container, selecting the options panel on that node in the tree will present an option to change the flow monitoring state. So turning it on for the BTM\_Purchase application will mean all message flows in the BTD now have monitoring enabled:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Configure_Enable_Monitoring.png)Enable flow monitoring on BTM\_Purchase application

Once turned on subsequent visits (switching back and forth between tabs will perform a reassessment of message flow monitoring status) to the configure tab for the BTD will now show a good message for flow monitoring.

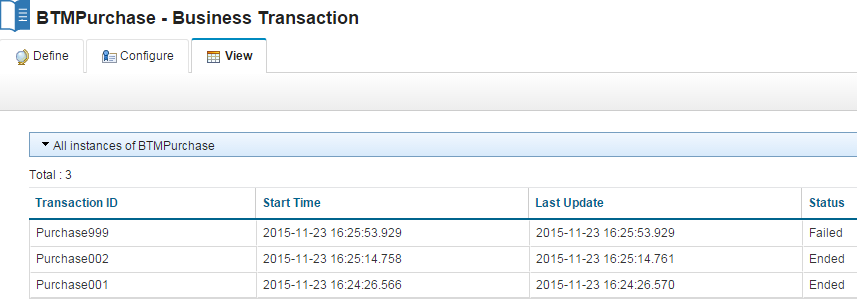
[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_Configure_Complete.png)Completed configuration for BTD

Now everything is set to start recording transactions. To test that we can record, and then view we must send a test message. Placing a message on the earlier created BTM.VALIDATE.IN queue will then begin processing of the message across the message flows until it reaches the final out queue. Whilst being processed specified monitoring events should emit and be recorded in the database, for then later viewing as a single correlated business transaction.

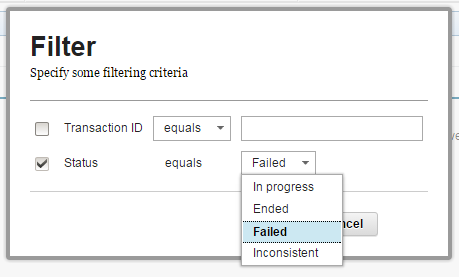
Using rfhUtil (or other queue writing method) place the BTM\_NewPurchase.xml message on the BTM.VALIDATE.IN queue. If you want to send further test messages, you can use the same test message file, but ensure the orderId value is changed from Purchase001 to another unique purchase order ID. We also want to test the ability to record failed business transactions, to do this we need to emit the monitoring event defined to be failed business event. The BTM\_InvalidPurchase.xml is not a fully valid message, and should cause a transaction to be failed.

**Viewing BTM Results**  
Now that some transactions should have occurred, we can view the results for that BTD by going to the View tab. There are two grid showing results, the top is the view for business transaction results. The bottom is event results for a single business transaction when one is selected from the top grid.

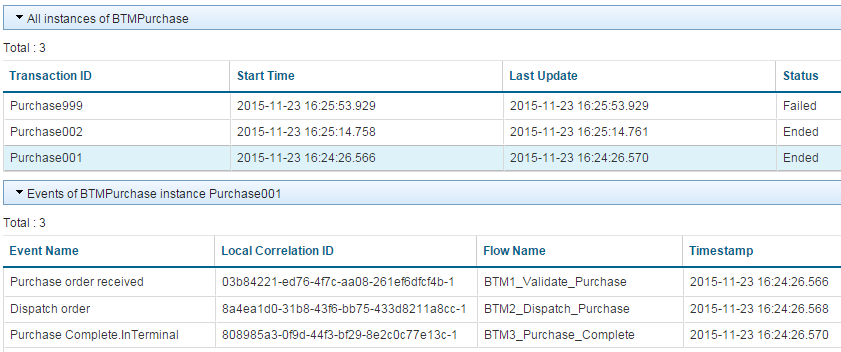
The querying of business transaction results requires either requesting all results, or a subset based on filtering upon some of the columns. As prompted, clicking the refresh button will query the database for viewing results:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_View_Results.png)Results grid for BTMPurchase

To apply filtering conditions on results, use the Filter button to launch the filter dialog and set filtering parameters to be applied to the BTD results query.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_View_Filter.png)Filter dialog for BTD results

Clicking on a single business transaction result in the grid will populate the events grid with the events that were emitted and contribute to that business transaction instance.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2015/12/BTM_WebUI_View_Event_Results.png)Event results for a single business transaction

Where applicable, bitstream and exception data can be downloaded and viewed for a single monitoring event that was emitted.

**Summary**  
In this tutorial, we have taken you through the new Business Transaction Monitoring capability that has been added in IBM Integration Bus v10.0.0.3. Using this capability, you are able to track the life cycle of your business events across your lines of business. You can view which business transactions end successfully or which have failed. You can track specific business transactions using a global transaction identifier which can help you find what happened to a lost order or why a particular business transaction has taken longer to complete than normal.

Further to this tutorial there is also an informative video of the using BTM at [Business Transaction Monitoring Demo](https://www.youtube.com/watch?v=LwurjJ3j368&cm_mc_uid=11322837804414470650368&cm_mc_sid_50200000=1448975894)

PS: Do you like the festive IIB posts? You can easily find them all by having a look at our [festive calendar](https://developer.ibm.com/integration/festive-greetings-and-best-wishes/)or click on the [festive2015 tag](https://developer.ibm.com/integration/blog/tag/festive2015/).

# Archiving Business Transaction Monitoring Data

[Simon Stone](https://developer.ibm.com/integration/blog/author/sstone1/), [DoinaKlinger](https://developer.ibm.com/integration/blog/author/dklinger/), [SanjayNagchowdhury](https://developer.ibm.com/integration/blog/author/sanjay_nagchowdhury/) and [GREGH](https://developer.ibm.com/integration/blog/author/greghatt/)  
Published on January 13, 2016*/ Updated on January 13, 2016*

[0](https://developer.ibm.com/integration/blog/2016/01/13/archiving-business-transaction-monitoring-data/#comments)

## Overview

The Business Transaction Monitoring (BTM) capability provided in IBM Integration Bus V10.0.0.3 allows a business user to track the life-cycle of a business transaction that has been processed by multiple message flows. To do this, BTM exploits and builds on the existing message flow monitoring functionality to capture and correlate events that are published by the message flows involved in a business transaction and record this information to a database.

Business Transaction Monitoring introduces the WMB\_BUSTRANS table which is described in the [Advanced usages of Business Transaction Monitoring](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/) article. This table is used to associate monitoring events that have been published by message flows and stored in the existing WMB\_MSGS and WMB\_BINARY\_DATA tables with a particular instance of a business transaction. Over time this data will build up, potentially resulting in detrimental impacts to the performance of the BTM capability. It is therefore important to perform housekeeping of this data in order to keep the data current and maintain acceptable performance.

This article provides information on the relationships between the database tables used to store Business Transaction data and provides an example of one possible approach where the tools provided by the database are used to archive data that is no longer required. The BTM capability is currently supported with DB2 and Oracle and the example provides a technique that is applicable to both these Relational Database Managements Systems (RDBMS). In some cases there are small differences between DB2 and Oracle in the queries that are used and these are highlighted where appropriate.

**Tip : Before running queries to manipulate and delete data from the database tables used by IBM Integration Bus it is strongly recommended that you ensure there is an up-to-date backup from which deleted data can be restored if necessary.**

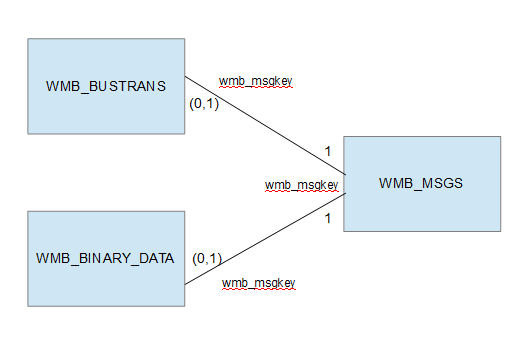
**It is important not to delete data for a transaction when further activity is expected on that transaction. If this is done then the transaction state will become inconsistent.**

## Database Tables Used By BTM

As described in [Advanced usages of Business Transaction Monitoring](https://developer.ibm.com/integration/blog/2015/12/07/advanced-usages-of-business-transaction-monitoring/) the database tables used by BTM are:

* WMB\_MSGS – All monitoring events that were defined on the message flow nodes are recorded into this table.
* WMB\_BINARY\_DATA – If you have specified that you want the payload data to be included in the recorded event, then the payload data is recorded in this table.
* WMB\_BUSTRANS – The set of events that are correlated by BTM to ascertain the state of a business transaction are recorded into this table

The following diagram shows the relationships between these tables:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/table_relationships.png)Figure 1.0 – Relationships between tables used by BTM

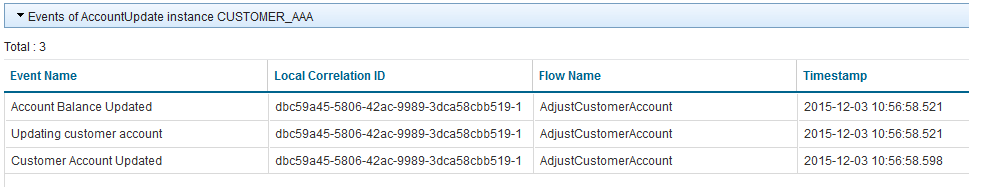
From a Business Transaction perspective this relationship means that an event in the WMB\_BUSTRANS table must have a corresponding event, matched on ‘wmb\_msgkey’, in the WMB\_MSGS table. However, not all events in the WMB\_MSGS table are required to have a corresponding event in the WMB\_BUSTRANS table. Similarly, a row in the WMB\_BINARY\_DATA table must have a corresponding row, matched on ‘wmb\_msgkey’, in the WMB\_MSGS table but not all rows in the WMB\_MSGS table are required to have a matching row in the WMB\_BINARY\_DATA table.

## Example

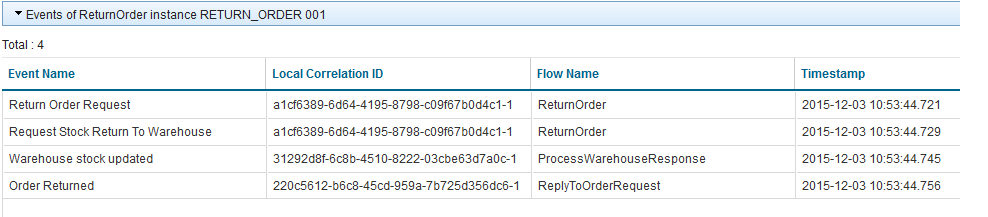
The example presented in this article shows an approach where CancelOrder transactions are archived from a database where the following three business transaction types have been recorded:

* AccountUpdate
* ReturnOrder
* CancelOrder

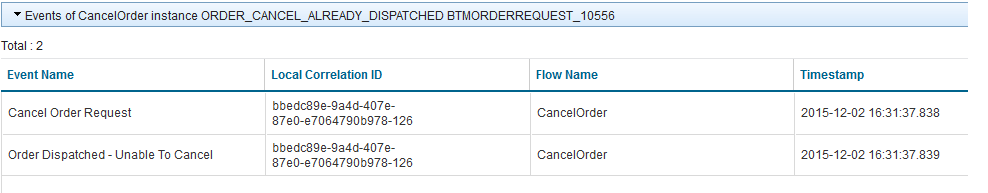
An AccountUpdate transaction consists of three events, as follows:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/AccountUpdateTransactions.png)Figure 2.0 – Example of events for an AccountUpdate transaction.

A ReturnOrder transaction consists of four events, as follows:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/ReturnOrderTransactions.png)Figure 3.0 – Example of events for a ReturnOrder transaction.

Finally, a CancelOrder transaction consists of two events, as follows:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/CancelOrderTransactions.png)Figure 4.0 – Example of events for a CancelOrder transaction.

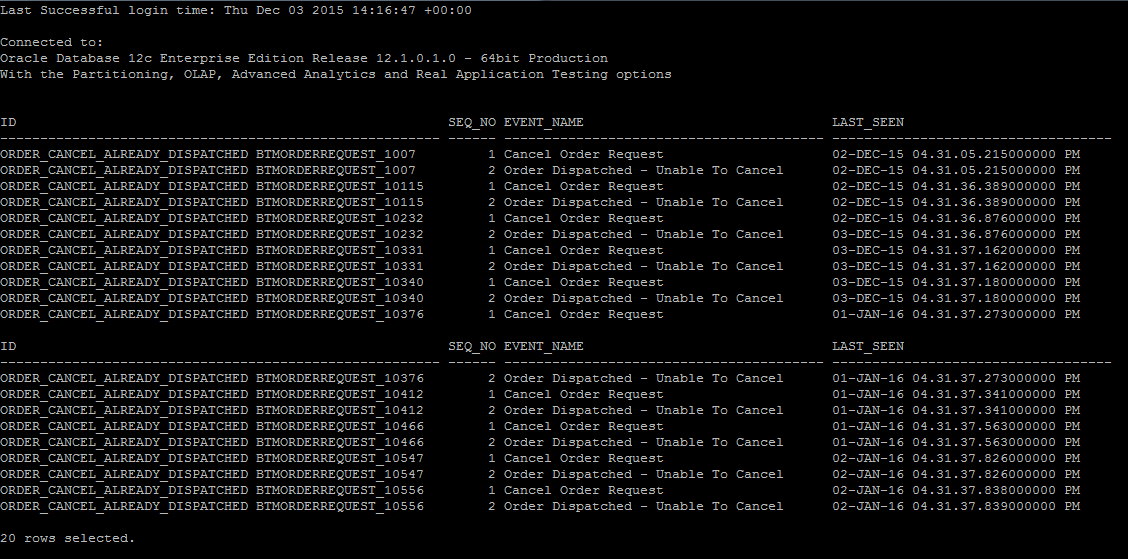
The database contains AccountUpdate and ReturnOrder transactions that were processed in December 2015. In addition to these, CancelOrder transactions exist that were recorded in December 2015 and January 2016. In this article we will archive the December 2015 CancelOrder transactions. To simplify the example, the events for the CancelOrder transactions are not configured to capture message payload data into the WMB\_BINARY\_DATA.

### The CancelOrder Transactions Data Set

The following query was used in Oracle’s ‘sqlplus’ tool to return information about CancelOrder transactions from the WMB\_BUSTRANS and WMB\_MSGS table. The same query can also be used in the DB2 Command Line Processor (CLP).

select id, seq\_no, event\_name, to\_timestamp(last\_seen\_timestamp,’YYYY-MM-DD HH24:MI:SS.FF’) last\_seen  
from wmb\_bustrans, wmb\_msgs  
where btd\_name like ‘CancelOrder’  
and wmb\_bustrans.wmb\_msgkey=wmb\_msgs.wmb\_msgkey  
order by last\_seen, seq\_no;

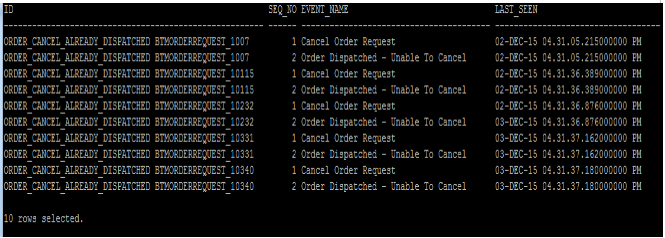
In this example, this query returns twenty rows for the ten CancelOrder business transactions that are stored in the database. Remember, each CancelOrder transaction has two events.

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/AllCancelOrderTransactions.png)Figure 5.0 – All CancelOrder transactions

As can be seen in Figure 5.0, the database contains CancelOrder transactions for the 2nd and 3rd December 2015 and the 1st and 2nd January 2016. By including an additional comparison in the WHERE clause of this query we can retrieve only the five CancelOrder transactions that occurred in December 2015. This is achieved by asking for transactions with a last\_seen\_timestamp before 2016-01-01.

select id, seq\_no, event\_name, to\_timestamp(last\_seen\_timestamp, ‘YYYY-MM-DD HH24:MI:SS.FF’) last\_seen  
from wmb\_bustrans, wmb\_msgs  
where to\_timestamp(last\_seen\_timestamp, ‘YYYY-MM-DD HH24:MI:SS.FF’) < to\_date(‘2016-01-01’, ‘YYYY-MM-DD’) and btd\_name = ‘CancelOrder’ and wmb\_bustrans.wmb\_msgkey=wmb\_msgs.msgkey order by last\_seen, seq\_no

This returns the following data:

[](https://developer.ibm.com/integration/wp-content/uploads/sites/25/2016/01/DecemberCancelOrderTransactions.png)Figure 6.0 – CancelOrder transactions recorded before 01-01-2016

For the purposes of this example, these are the CancelOrder transactions that we are not expecting any further activity on and are the ones we want to remove from the ‘live’ tables and archive. There are a number of approaches that could be employed to achieve this but, in this example, queries on the WMB\_BUSTRANS and WMB\_MSGS tables are used to create temporary ‘archive’ tables from the result sets that are returned. The contents of the archive tables can then be exported before deleting data from both the archive and live tables to free up space.

First we create the DEC\_CANCELORDER\_TXNS table from the result set of CancelOrder transactions that were taken prior to ‘2016-01-01’. For Oracle, this table can be created with a CREATE TABLE AS SELECT statement like this:

create table dec\_cancelorder\_txns as  
select \* from wmb\_bustrans  
where to\_timestamp(last\_seen\_timestamp,’YYYY-MM-DD HH24:MI:SS.FF’) < to\_date(‘2016-01-01’, ‘YYYY-MM-DD’) and btd\_name=’CancelOrder’

For DB2, the DEC\_CANCELORDER\_TXNS table is created first and then a separate INSERT statement is run to populate the table with the desired data, as follows:

create table dec\_cancelorder\_txns like wmb\_bustrans  
insert into dec\_cancelorder\_txns  
(select \* from wmb\_bustrans where to\_timestamp (LAST\_SEEN\_TIMESTAMP, ‘YYYY-MM-DD HH24:MI:SS.FF’) < to\_date(‘2016-01-01′,’YYYY-MM-DD’) and btd\_name=’CancelOrder’)

Having created a table containing the CancelOrder business transaction data that occurred before ‘2016-01-01’ we can select the event data that those transactions are comprised of from the WMB\_MSGS table and insert it into the DEC\_CANCEL\_ORDER\_EVENTS table.

For Oracle, the following SQL can be used:

create table dec\_cancelorder\_events as  
select \* from wmb\_msgs  
where wmb\_msgkey in  
(select wmb\_msgkey from dec\_cancelorder\_txns)

For DB2, the DEC\_CANCELORDER\_EVENTS temporary table is created first and then populated with the required data.

create table dec\_cancelorder\_events like wmb\_msgs  
insert into dec\_cancelorder\_events  
(select \* from wmb\_msgs where wmb\_msgkey in  
(select wmb\_msgkey from dec\_cancelorder\_txns))

The data from the ‘live’ tables can then be deleted. To delete the archived rows from the WMB\_BUSTRANS table the following SQL can be used.

delete from wmb\_bustrans  
where id in  
(select id from dec\_cancelorder\_txns)

The archived data from the WMB\_MSGS table can be removed using the following SQL.

delete from wmb\_msgs  
where wmb\_msgkey in  
(select wmb\_msgkey from dec\_cancelorder\_events)

The database export utilities such as ‘Oracle Data Pump’ or DB2’s EXPORT CLP command can then be used to unload the data from the temporary ‘archive’ tables, in this case DEC\_CANCELORDER\_TXNS and DEC\_CANCELORDER\_EVENTS, to operating system files.

## Summary

This article described an approach that allows you to maintain the data recorded into an Oracle or DB2 database when using the IBM Integration Bus – Business Transaction Monitoring (BTM) capability. It showed the database tables that are used to store this data and the relationship between them. There are several strategies that could be used to archive data from the live tables but this article focused on providing an example using temporary tables and archiving BTM data based on the time that last activity was seen for a business transaction.