**Simple PUB/SUB Example Using WebSphere Message Broker v7 and MQ v 7**

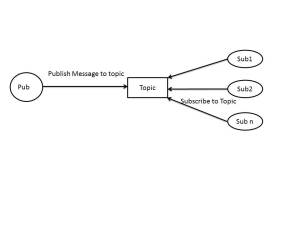
by [achourelabbes](https://transformatech.wordpress.com/author/achourelabbes/) on September 13, 2013



**1.  OVERVIEW PUB/SUB**

The Pub/Sub design pattern is based on a scenario where clients (publishers) “publish” messages to topics.  Then one or more clients (subscribers) that are interested in that topic can “subscribe” to that topic.   Publishers and Subscribers are loosely coupled and are not aware of each other.   This is different than the point-to-point implementation where sender and receivers are tightly coupled at minimum from a connectivity standpoint.

Following is a very high-level illustration of Pub/Sub:

**[](https://transformatech.files.wordpress.com/2013/09/sub_pub.jpg)**

**2.  OVERVIEW OF OUR WMB/MQ PUB/SUB SCENARIO**

There are many ways to implement a Pub/Sub design pattern depending on your platform.  To illustrate a simple Pub/Sub example, this blog is going to focus on one simple technique using WMB v7 and MQ v7.   There are also other ways to implement this depending on what version of WMB you are using.   These other techniques will not be covered in this blog, but detailed information can be found in the MQ and WMB infocenters.

Our Simple Example is going to leverage a WMB message flow with an MQ Input Node to define what topic to put to and WMB Publication Node to publish to an MQ topic.   The example is then going to leverage MQ Subscriptions to subscribe to that topic and route the topic to specific MQ queues that subscribers can read from.

Section 3 will provide step by step implementation for setting up this example.

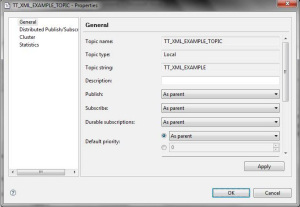
**3.  STEP BY STEP FOR SETTING UP WMB/MQ PUB/SUB EXAMPLE**

**0.   Pre-reqs:** Our scenario depends on the following MQ queues being defined.   Please create these  queues In MQ explorer or using runmqsc from command line before proceeding with the steps.

* TT.XML.EXAMPLE.PUB – Input queue for publication message flow
* TT.XML.EXAMPLE.INPUT – Input queue for one of our subscriber message flow
* TT.XML.EXAMPLE.OUT – Output queue for one of our subscriber message flow
* TT.XML.EXAMPLE.COPY – Queue for a second subscriber

**1.   Create MQ topic**

Go to MQ Explorer Topic folder and create a topic called TT\_XML\_EXAMPLE\_TOPIC.  In this example, the topic string we will use to publish messages to is “TT\_XML\_EXAMPLE” You can also give a topic a hierarchical structure using “/”.   More information on this can be found in MQ infocenter document.

**[](https://transformatech.files.wordpress.com/2013/09/tt_xml_example_topic.jpg)**

**2.   Create WMB Publication Flow**

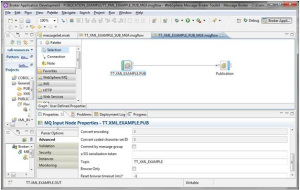
Next, we need to build a WMB Message Flow to publish message to the MQ Topic we setup in step one.

To build your Publication Flow, you need 2 Nodes:

**(1)   MQ Input Node** that uses TT.XML.EXAMPLE.PUB queue and also has the “topic” property set to our Topic string(“TT\_XML\_EXAMPLE”) in the advance property tab on the MQ Input Node (see image below).

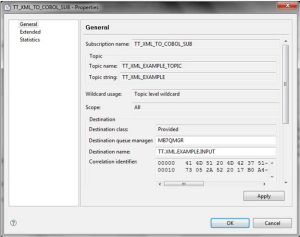
**(2)   Publication Node** – No properties should be set on this.   It will get the topic to publish to from MQ Input Node Topic property.

Once you deploy this message flow and it receives a message, it will start publishing messages to the topic.

**[](https://transformatech.files.wordpress.com/2013/09/publication_flow_image.jpg)**

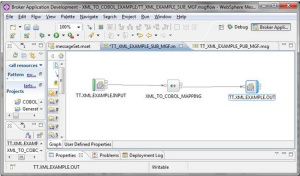
**3.   Create Subscription to Topic**

Now that we are publishing messages to a topic, we can have subscribers to subscribe to this topic.  To do this, we need to first create an MQ subscription in MQ Explorer.  In this example, we will call it “TT\_XML\_TO\_COBOL\_SUB” and setup the topic to use(TT\_XML\_EXAMPLE\_TOPIC) and the destination to be an MQ queue called “TT.XML.EXAMPLE.INPUT”.   This will be the input to our subscriber message flow that we will create in step 4.

**[](https://transformatech.files.wordpress.com/2013/09/tt_xml_to_cobol_sub.jpg)**

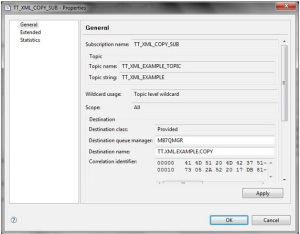
**4.   Create Subscription Message Flow**

For this blog, we will use our previous message flow (XML TO COBOL) where an XML message is mapped to a COBOL Output.   The implementation can be anything you want this subscriber to do with the message including any transformation, enrichment, and/or routing.

**[](https://transformatech.files.wordpress.com/2013/09/subscription_message_flow_image.jpg)**

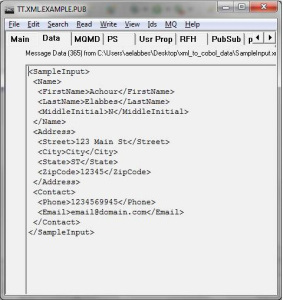
**5.   Create 2nd subscription to topic**

For simple illustration purposes, we are going to create a second subscription to just get a copy of the message published to the topic.   To do this, go to MQ Explorer again and create an MQ Subscription called “TT\_XML\_COPY\_SUB” using the same topic(TT\_EXAMLE\_XML\_TOPIC) and the destination queue to be “TT.XML.EXAMPLE.COPY”

**[](https://transformatech.files.wordpress.com/2013/09/tt_xml_copy_sub.jpg)**

**6.   Testing Simple Pub/Sub Example Scenario**

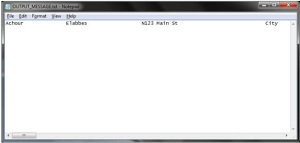
For our scenario, we will use RFHUtil to put a test XML message to the input queue (TT.XML.EXAMPLE.PUB) of the publication message flow.

**[](https://transformatech.files.wordpress.com/2013/09/original_inputmessage_image.jpg)**

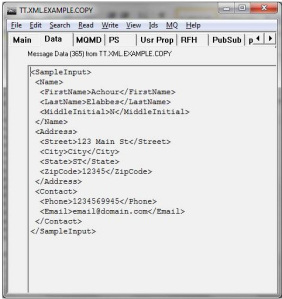
This XML will then get published to the topic (TT\_EXAMPLE\_XML\_TOPIC).   Our two subscriptions will then receive this XML message and we will get the following respective outputs from our two subscriptions:

**Output (Subscription 1):**

**1.**Subscription sends XML to destination queue TT.XML.EXAMPLE.INPUT. The subscription message flow will process the message and send COBOL formatted message to TT.XML.EXAMPLE.OUT queue.

**[](https://transformatech.files.wordpress.com/2013/09/cobol_output_message.jpg)**

**2.** The second subscription will send XML to destination queue TT.XML.EXAMPLE.COPY.

**[](https://transformatech.files.wordpress.com/2013/09/tt-xml-example-copy.jpg)**

This concludes our blog on using WMB v7 and MQ v7 to implement a simple Pub/Sub scenario.   We welcome any comments if you have any questions or want to provide feedback.

# Monitoring in IIB

#### by [Sujitha Sai Selvaraju](https://blogs.perficient.com/author/sselvara/) on April 24th, 2017 | ~ 7 minute read

### What is Monitoring

*Monitoring is a feature in IIB, which enables the developer to configure the flow to emit events when something interesting happens. These events can either be stored in a DB or consumed by monitoring applications such as Websphere Business Monitor for auditing and replay of messages.*

### Features

IIB flows can be configured to generate two types of events.

* Transactional Events – Events emitted by input node. To indicate the state of the transaction. When configured, transactional events are emitted under three circumstances

i. Transaction Start – To indicate the start of transaction

ii. Transaction End – To indicate the end of transaction

iii. Transaction Roll Back – To indicate failure

* Terminal Events – Events generated by the terminals in any node, including the input node.

The point at which an event is triggered, is termed as event source. In case of terminal events all the terminals that are active to emit events become terminal event sources. The input nodes have a transactional event source in addition to the terminal events. Input nodes can be configured to generate both terminal and transactional events.

### Monitoring Profile vs Monitoring Properties

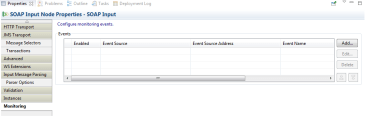
Monitoring can be Created/Configured, Enabled or disabled by using any one of the following methods:

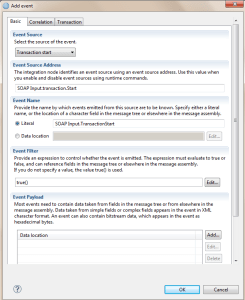
1. Monitoring properties in the node
2. Monitoring profile configurable service

### Monitoring Properties

*Monitoring Properties can be used when the events are customized during the design phase. The events can be configured using IBM Toolkit’s Message flow editor. Any node that supports monitoring, has a tab in properties called “Monitoring” for customization. The Monitoring properties are deployed in the bar file as part of the Message flow. If there are any changes in the event configuration, the properties are altered in the flow and re-deployed.*

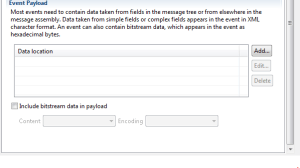
To Add Monitoring events, go to the Monitoring tab in Properties view, and click on the Add button.

[](https://i2.wp.com/blogs.perficient.com/files/2017/04/Pic1_Events-1.png?ssl=1)After clicking Add Event, the below window opens. It has three tabs for configuration.

[](https://i2.wp.com/blogs.perficient.com/files/2017/04/Pic2_Events-1.png?ssl=1)

The basic tab has details on

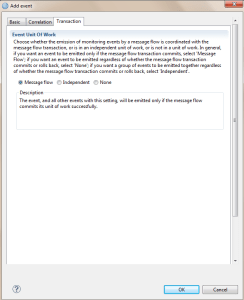
* the type of event (Event Source) – Terminal or Transactional Events
* Event Source Address – Depending on the type of Event source, the address is populated as read only property.
* Event Name – The name for the event, which could be a constant/Literal, or can be pulled from the payload.
* Event Filter – Events are generated only when the event filer evaluates to true(). The filter can be configured to evaluate an xpath, and depending on the value, the event can be published/suppressed.
* Data Payload – In addition to the general data, for auditing any business related data/or a particular field in the payload has to be captured, that can be achieved by adding the payload data location.
* Include Message Data – Use this option, if the entire message has to be captured for future reference.

[](https://i2.wp.com/blogs.perficient.com/files/2017/04/Pic5_Events-1.png?ssl=1)The Correlation tab has three types of Correlation:

Every monitoring event must have at least one type of correlation. Local, Parent or Global. The Values can be set Automatically or can be pulled from the specific location in the message tree using the Xpath Expression Builder.

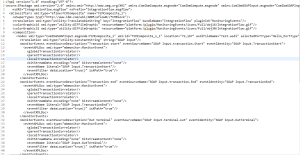
### [Monitoring in IIB](https://i2.wp.com/blogs.perficient.com/files/2017/04/Pic6_Events-1.png?ssl=1)

The last transaction tab is optional, and can be used to configure how the events from the message flow are coordinated. It could be a Message flow transaction (Message Flow) or an independent unit of work (Independent) or not in a unit of work (none).

[](https://i2.wp.com/blogs.perficient.com/files/2017/04/Pic7_Events-1.png?ssl=1)

The events created can be enabled or disabled by checking/unchecking the enable box in the Monitoring tab of the Properties view.

[Monitoring in IIB](https://i0.wp.com/blogs.perficient.com/files/2017/04/Pic8_Events-1.png?ssl=1)On opening the message flow in the text editor, the configured monitoring events can be viewed in xml format.

[](https://i1.wp.com/blogs.perficient.com/files/2017/04/Pic9_Events-1.png?ssl=1)

### Monitoring Profile

*Monitoring profile is the xml representation of the monitoring events configured in the nodes. One main advantage of using monitoring profile is that, since it is a configurable service, any changes made to the profile don’t require any deployment, whereas changes made at the node level require a re-deployment. The monitoring profile takes precedence over the monitoring events configured in the message flow. Another advantage of Monitoring profile is that, a single monitoring profile can be used for multiple flows. The profiles aren’t deployed in a BAR file, it is created directly on the Integration Node/Broker.*

The **MonitoringProfile.xsd**defines the structure of the monitoring profile xml. The schema and a sample profile can be found in,

*Install\_root/server/sample/RecordReplay/MonitoringProfile.xsd*

*Install\_root/server/sample/RecordReplay/basicMonitoringProfile.xml*

The monitoring profile can be created from the IIB web console, as it is a configurable service. Once created, use the below commands to link the monitoring profile to the flows.

##### LINKING MONITORING PROFILE TO APPLICATION/MESSAGEFLOW

mqsicreateconfigurableservice <Broker> -c MonitoringProfiles -o myMonitoringProfile

mqsichangeproperties <Broker> -c MonitoringProfiles -o myMonitoringProfile -n profileProperties -p <MonitoringProfile.xml>

mqsichangeproperties <Broker> -c MonitoringProfiles -o myMonitoringProfile -n useParserNameInMonitoringPayload -v TRUE

mqsichangeflowmonitoring <Broker> -e <EG> -k <AppName> -f <FlowName> -m <MonitoringProfileName>

-g => to implement for all Execution Groups

-j => to implement for all applications in an execution group.

mqsichangeflowmonitoring <Broker> -g -j -m <MonitoringProfileName>

### Activating monitoring

*Deploying message flows with monitoring properties enabled or associating monitoring profile doesn’t help in generating events. Use the below command to activate monitoring. Irrespective of using monitoring profile or using monitoring in the nodes, the below command should be executed. Also, please note that the below command must be executed for every new deploy.*

mqsichangeflowmonitoring <Broker> -e <EG> -k <AppName> -f <FlowName> -c active

mqsichangeflowmonitoring <Broker> -g -j -c active

mqsichangeflowmonitoring <Broker> -e <EG> -f <FlowName> -s "Node1\_Label.terminal.out,Node2\_Label.terminal.in" -i enable

### Verify monitoring

Execute the below command to check on the monitoring status

mqsireportflowmonitoring <Broker> -e <EG> -k <AppName> -f <FlowName>

mqsireportflowmonitoring <Broker> -g -j

mqsireportflowmonitoring <Broker> -e <EG> -f <FlowName> -k <AppName> -s "Node1\_Label.terminal.out,Node2\_Label.terminal.in"

### Subscribing to Monitoring

*The events generated by the flows must be consumed by the applications. The monitoring event feature leverages the publish/subscribe (pub/sub) mechanism of Message Broker. Event messages are published to the specific topics, which can be registered by subscribing applications. The form of the topic name is:*

$SYS/Broker/<Broker>/Monitoring/<EG>/<flowName>

The topic hierarchical structure enables subscribing applications to filter the events. For example, one application can receive events from all message flows deployed on all execution groups in a broker, while another can receive only the events from all message flows in a single execution group.