

## Contents

|  |    |
|--|----|
| Foreword.....  | 2  |
| Introduction .....   | 2  |
| How Tibco helps? .....   | 2  |
| Palates in Tibco .....   | 3  |
| Tutorial Examples.....   | 3  |
| XML/XPATH/Mapper .....   | 3  |
| FTP.....   | 4  |
| HTTP .....   | 7  |
| Simple HTTP Client .....   | 7  |
| Simple HTTP Server .....   | 9  |
| HTTP with Attachments .....  | 10 |
| JMS/EMS .....  | 14 |
| JMS Request Reply .....  | 15 |
| JMS Sender Wait .....  | 17 |
| JMS Message Selector.....  | 19 |
| JMS Topic .....  | 20 |
| JMS Local/XA Transaction .....   | 22 |
| JDBC .....   | 26 |
| JDBC Connection .....  | 27 |
| JDBC Query.....  | 27 |
| JDBC Procedure Call .....  | 29 |
| Batch Processing .....   | 30 |
| JDBC Update.....   | 32 |
| SOAP/SERVICE .....   | 34 |
| Building an abstract WSDL .....  | 35 |
| Creating a web service from the abstract WSDL and Building a concrete WSDL ..... | 36 |
| Using a SOAP Request Reply for calling the service.....                          | 39 |
| SOAP over JMS .....  | 40 |
| SOAP Event Source/ Reply and Fault .....   | 41 |
| SOAP with Attachments (SwA).....   | 43 |

|                 |    |
|-----------------|----|
| Conclusion..... | 47 |
|-----------------|----|

### Foreword

This document is written keeping in mind to provide a basic understanding for the developers on TIBCO tool using TIBCO Designer for the first time. It mainly focuses on the simple usage of the palates available in TIBCO.

Some basic examples are provided to understand the tool and its features.

### Introduction

The history of TIBCO, its features and usage is available in its official websites.

### How Tibco helps?

Understanding the utility of TIBCO demands knowhow for certain technical terms.

- SOA (Service Oriented Architecture)
- Middleware
- Java Messaging Service
- XML/XPath
- Web Services
- Adapters
- Routing

Before starting with the practical implementation of Tibco, it's compulsory to have a basic understanding of the above terms.

I will give you a scenario based example where TIBCO comes into play.

Consider you have a simple application A which is developed on Windows platform and there is another application B developed on Unix Platform which is basically a cross platform scenario. There are multiple ways to achieve it. Like you can convert the output of A to a readable format for B and then send. The problem to this approach is its high manual intervention. Every time there is a change in the required output, the code for compatibility of B and also for other platforms needs to be changed. Moreover changing the application code all the time is a bad practice and it should be avoided.

The TIBCO solution is create your application connect with TIBCO as the middleware.

TIBCO provides all the features to which makes implementation of business logics easy. Like based upon a condition, your application will go to different execution routes as TIBCO gives you a flow based approach to achieve your business requirements.

It follows SOA architecture and provides you with interoperability.

How does it do all this? That's what we will see when we go on more into this tutorial.

## Palates in Tibco

Tibco website has a detailed documentation on the palates that are available. My advice will be to go through the documentation. Here our prime focus will be the practical implementation of those palates.

## Tutorial Examples

- File
- FTP
- HTTP
- JMS/EMS
- RV
- Service
- SOAP
- TCP
- XML/XPATH/Mapper

We will start with the basic understanding of XML handling on TIBCO. XML is default data exchange format in TIBCO. Moreover, XML is understandable on any platform. It is a widely accepted common data format over the network. It comes with a lot of advantage like it's light weight and can be easily traversed and parsed. XML Data manipulation is easy as compared to other data formats.

To traverse to a particular node in an XML Document we use XPath.

Mapper is nothing but mapping data from an xml file node output value to another xml file node input value. Mapping will give you a better understanding.

## XML/XPATH/Mapper

In this example we will take the help of a demo xml document and map the node values into a flat file.

Suppose we have a books xml document with multiple book nodes containing title, price and author's list containing authors. This example file will be our input and as an output we want a flat text file which will contain only the book title and its first author for example "The Night by Stephen James". So for every book in books the output will be as shown above and will be written to a text flat file.

Once you are clear with the mapper concept in TIBCO, any program logic implementation will become easy.

Palates to be used:

- File Reader

- Parse XML
- Mapper
- Write File

Steps to follow:

- In the File Reader properties, set read as text under configurations tab. Go to input tab give your xml file path within quotes in the filename input field.
- For Parse XML node, choose input style under configuration tab as text. Now go to input tab and map the file reader output `textContent` to the parse xml input `xmlString`. Next go to output editor tab add any xml element reference and choose the xml schema for books xml file. If everything is fine then in output tab it will show your xml schema order accurately.
- Put the mapper node and write file node into a group with group action as iterate with any index name and choose the variable as the repeating book node in books. Now an iterator will be available in process data which will give the index value of the book node it is currently looping on.
- For the mapper node, go to input editor tab add a variable of content attribute of type string. In the input tab map the input variable added as shown in the figure below. Your input variable will hold a string value. So we have to use a concat function to join the name of the book with “by” and with the first author’s name.

For example, concat (books/book [\$i]/name,” by”, books/book [\$i]/authors/author [1])

Where \$i is the iterator index name.

- In the write file node, choose write as text and check the append option. Now in the input tab, map the filename input field with the desired txt file path in quotes and map the `textContent` as the output of the mapper node output variable. Give the `addLineSeparator` a Boolean value of 1 which will give a new line separator every time something is written on it. Checking the append option on will prevent the file from getting overwritten. Any new content will get written keeping the old content intact.

These are the simple steps to follow to manipulate our xml file, loop through it and change our output to a text file.

## FTP

Now we will see a basic example of ftp palate in Tibco.

You can use any FTP server for getting a FTP connection. As here I have used windows default FTP server (IIS7).

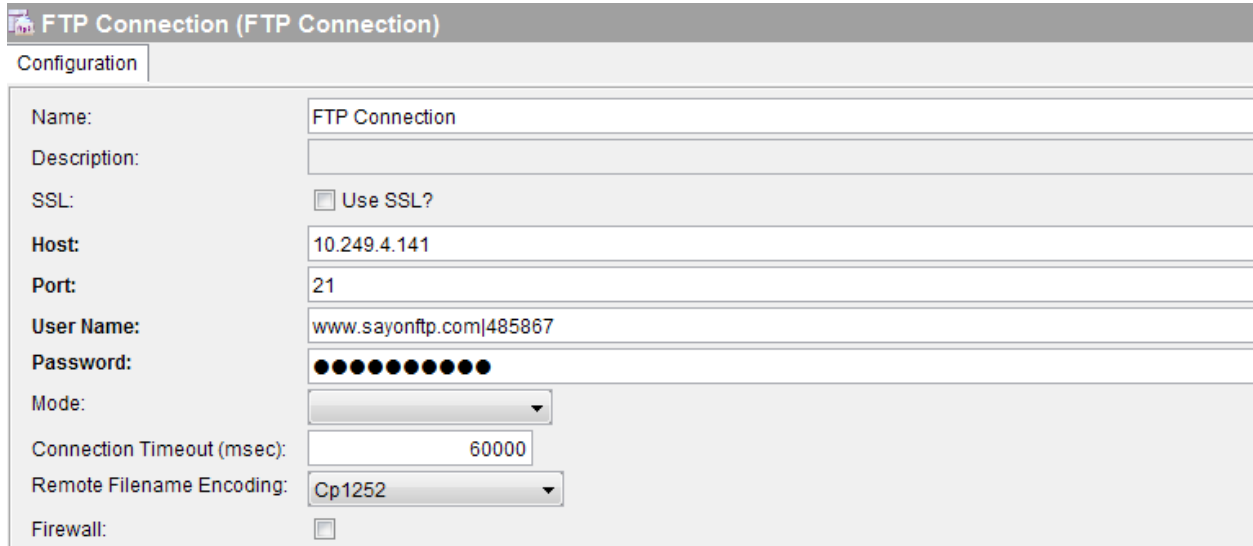
Our requirement is like we need to read a text file copy its text content to another file in the FTPServer then retrieve the newly created file and write it to another file in some local directory.

Prerequisites:

- FTP Connection

### Creating a FTP Connection

- Select a FTP Connection node from list of resources.
- Give your host name, port number, user name and password.
- Click Test Connection to check the connection.



**FTP Connection (FTP Connection)**

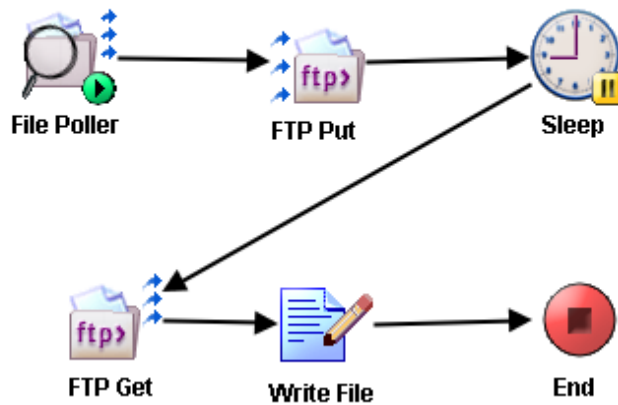
Configuration

|                            |   |
|----------------------------|---|
| Name:                      | FTP Connection                          |
| Description:               |   |
| SSL:                       | <input type="checkbox"/> Use SSL?       |
| Host:                      | 10.249.4.141                            |
| Port:                      | 21                                      |
| User Name:                 | www.sayonftp.com 485867                 |
| Password:                  | ●●●●●●●●●●                              |
| Mode:                      | <input type="button" value="v"/>        |
| Connection Timeout (msec): | 60000                                   |
| Remote Filename Encoding:  | Cp1252 <input type="button" value="v"/> |
| Firewall:                  | <input type="checkbox"/>                |

Palates required:

- FilePoller
- FTPPut
- FTPGet
- WriteFile

### /FTP/Main Process/FTP Put and Get















Steps to follow:

- In the file poller node give the file path of the file to be transferred text file and set the content as text.
- Next, at the configuration tab of FTPPut node select the FTPConnection you created earlier. Check the Process Data checkbox to put data in a file in the FTP server. Go to input tab and set the remotefilename with a file name of your choice and map the output textContent of the file poller to the data input field.

**FTP Put (FTP Put)**

Configuration | Input | Output | Error Output

**Process Data:** ☒ 

**Activity Input:**           

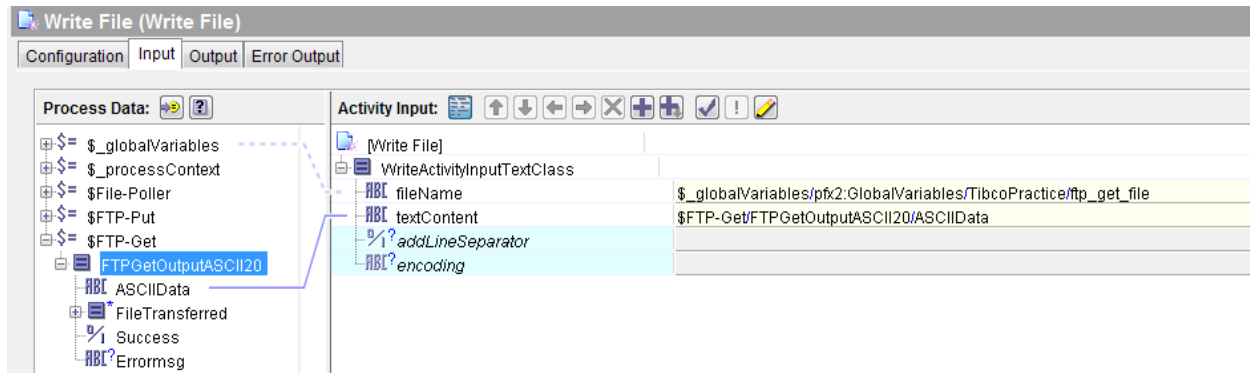
**Process Data:**

- \$\_globalVariables
- \$\_processContext
- \$File-Poller
  - EventSourceOutputTextClass...
    - action
    - timeOccurred
    - fileInfo
      - fullName
      - fileName
      - location
      - configuredFileName
      - type
      - readProtected
      - writeProtected
      - size
      - lastModified
      - fileContent
        - textContent

**Activity Input:**

| Field          | Value  |
|----------------|--|
| RemoteFileName | \$File-Poller/ptc:EventSourceOutputTextClass/fileInfo/fileName       |
| Data           | \$File-Poller/ptc:EventSourceOutputTextClass/fileContent/textContent |
| Host           |  |
| Port           |  |
| UserName       |  |
| Password       |  |
| Encoding       |  |
| Timeout        |  |

- For FTPGet, at the configuration tab select the FTPConnection and check the process data option. In the input tab, give the remotefilename field with the name of the file you want to get.
- For Write File node, select write as text under configuration tab then under input tab give the file name of the local text file (output) and map the textContent to the output textContent of the FTPGet node.



## HTTP

We will now discuss about using HTTP Palate in TIBCO.

Here we will discuss a complete demo of a HTTP request and HTTP response with attachments (MIME).

For any complete HTTP Process understanding the Http request response flow is important.

Our web browser use HTTP protocol so when we write a url and click go it is actually making a HTTP request to a server for which the sends a HTTP response to the browser (who is the requestor/client) back. Of course a HTTP client can be any application or any browser. Being in TIBCO a HTTP client is who makes a HTTP request and accepts a HTTP response from the server. A HTTP server is a HTTP request receiver which listens for any request made to it and once it receives a request it sends a HTTP response to the HTTP Client back. This is what the whole process of communications happens through HTTP as the transport.

Here we will create a simple HTTP request response flow i.e., a client and a server.

### Simple HTTP Client

Perquisites:

- HTTP Connection

#### Creating a HTTP Connection

- Select a HTTP Connection from Add Resource HTTP palate.
- Give it a host name as localhost or some ip address
- Give a specified port no which is free
- Select server type as Tomcat

**HTTP Connection (HTTP Connection)**

Configuration **Advanced**

Name: HTTP Connection

Description:

Host: localhost

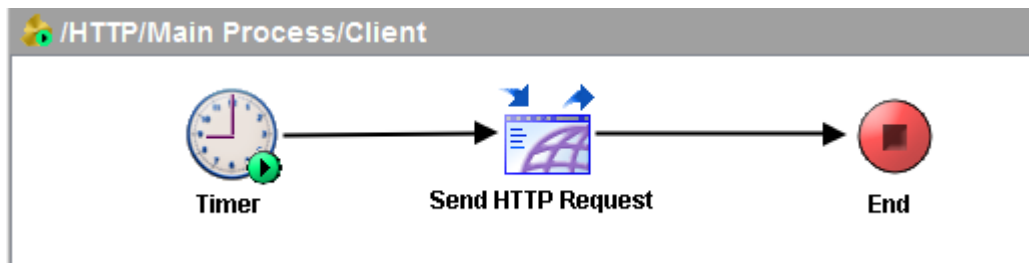
Port: 8065

SSL: ☐ Use SSL?

Server Type: Tomcat

Palates required:

- Send HTTP Request



Steps to follow:

- In the send HTTP request palate give the similar host name and port number as used for the HTTP Connection
- In the parameters box add a parameter like name of type string and give cardinality as required (optional).

**Send HTTP Request (Send HTTP Request)**

Configuration **Advanced** Input Headers Output Headers Input Output Error Output

Name: Send HTTP Request

Description:

Host: localhost

Port: 8060

Use Proxy Setting:

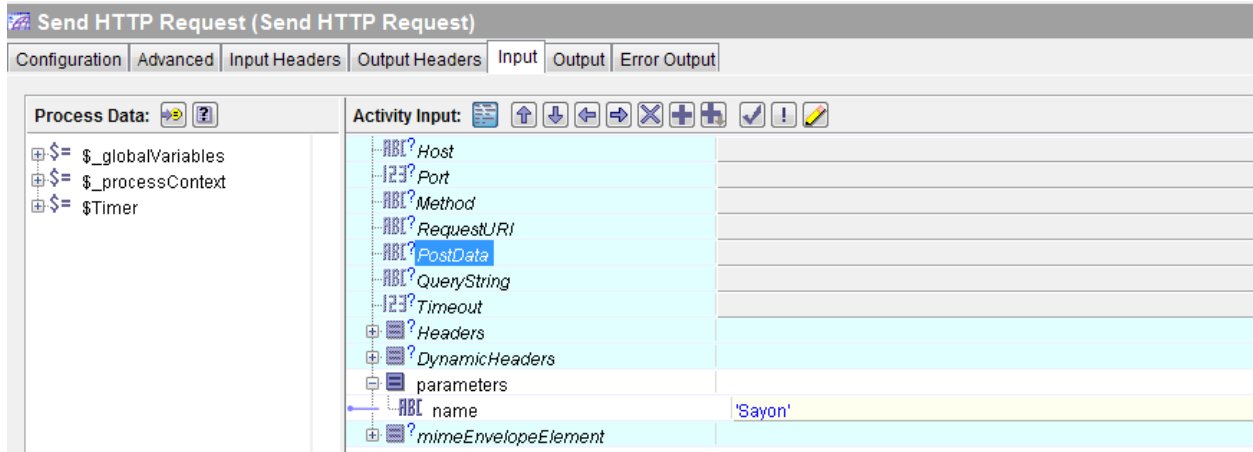
Accept Redirects: ☐

Parameters:

| Parameter Name | Parameter Type | Parameter Cardinality |
|----------------|----------------|-----------------------|
| name           | string         | Required              |

- Next go to input tab expand right tree you will name under parameter, give it a value in string.

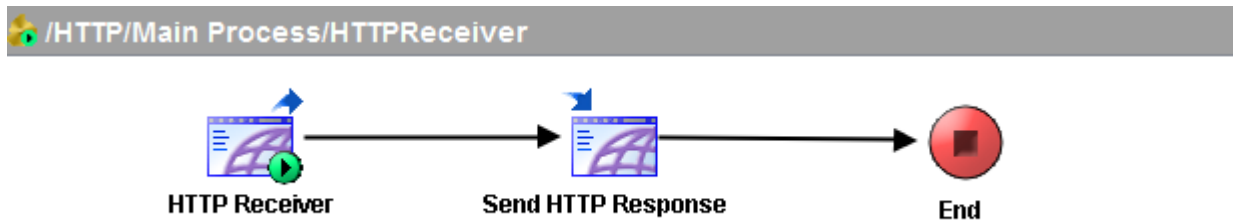




### Simple HTTP Server

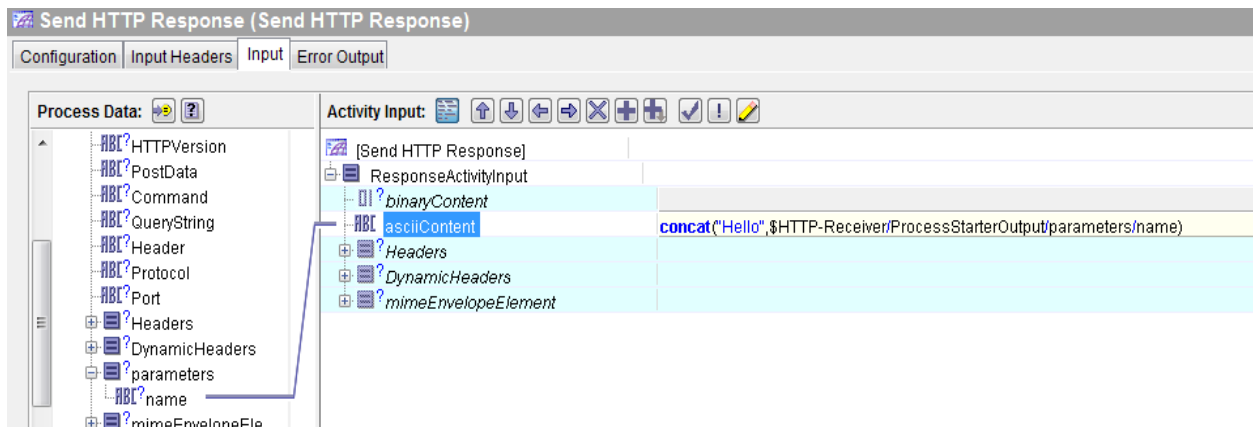
Palates required:

- HTTP Receiver
- Send HTTP Response



Steps to follow:

- In HTTP receiver, select the HTTP Connection created earlier.
- Add a parameter with same name and type as we did while sending request.
- For Send HTTP Response select reply for as the HTTP Receiver.
- In the Input tab, map the parameter name from the left tree to the right pane ascii content input field.



## HTTP with Attachments

It's now time to delve into a little deep and try to send files as attachments with http request and response.

This tutorial will cover two examples of this which will help you understand the way to achieve the requirement.

### *Sending a XML file over HTTP and getting a Doc file as response.*

For the first step we have to decide about the palates required.

There will be two process one Sender/Client and another one Receiver/Server.

#### Client

Perquisites:

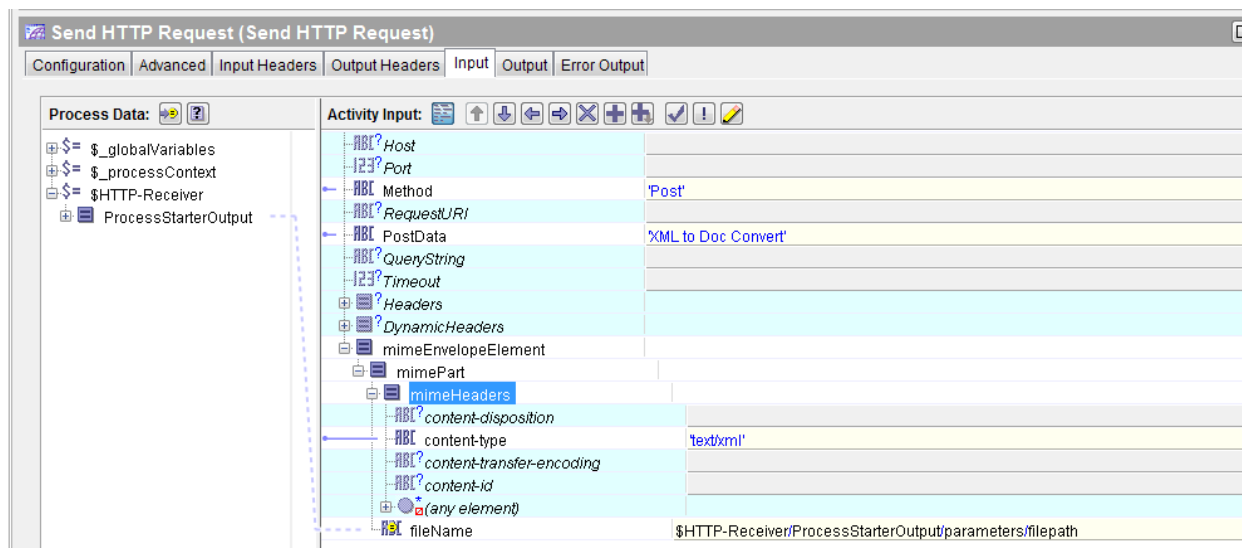
- HTTP Connection

Palates required:

- Send HTTP Request

Steps to follow:

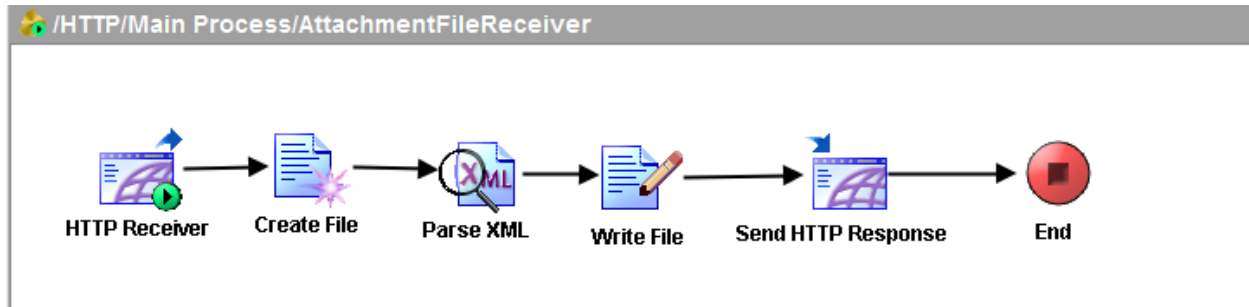
- In the configuration tab of the send HTTP Request give the hostname and port as of the HTTP Connection.
- Now go the input tab, give method as post and give some postdata.
- Then in the mimeEnvelopeElement part give content type as text/xml in the mimeHeader part
- In the mimePart give the file path of your xml file to be sent in the textContent|filename|binaryContent field.



## Server

Palates required:

- HTTP Receiver
- Write File
- Send HTTP Response



Steps to follow:

- In the HTTP Receiver select the connection as the HTTP Connection created previously. Select output style as string.
- Now in the write file palate select write as text in the configuration tab. In the input tab, give a the file path of your blank document file in the filename field. For the textContent map the textContent of the mimeType of the HTTP Receiver to it.
- In the send HTTP Response, select reply for the HTTP receiver. For input, write application/msword in the content type in the mimeType and map the filename with the fullname of the write file node.

**Send HTTP Response (Send HTTP Response)**

Configuration | Input Headers | Input | Error Output

**Process Data:**

- \$HTTP-Receiver
- ProcessStarterOutput
  - Method
  - RequestURI
  - HTTPVersion
  - PostData
  - Command
  - QueryString
  - Header
  - Protocol
  - Port
  - Headers
  - DynamicHeaders
  - mimeTypeEnvelopeElement
  - mimeTypePart
  - Context
- \$Create-File
- \$Parse-XML
- \$Write-File

**Activity Input:**

- ResponseActivityInput
  - binaryContent
  - asciiContent
- Headers
  - StatusLine
  - Content-Type: application/msword
  - Set-Cookie
  - Pragma
  - Location
- DynamicHeaders
- mimeTypeEnvelopeElement
  - mimeTypePart
    - mimeTypeHeaders
      - content-disposition
      - content-type: application/msword
      - content-transfer-encoding
      - content-id
      - (any element)
- fileName: \$Write-File/ns2:WriteActivityOutputClass/fileInfo/fullName

By doing all these you will be able send a file and receive a file over HTTP as attachments using the MIME protocol.

In the above example we have used `textContent` and `filename`. For sending images we will use binary content.

### *Sending an image over HTTP then displaying it on browser.*

#### Client

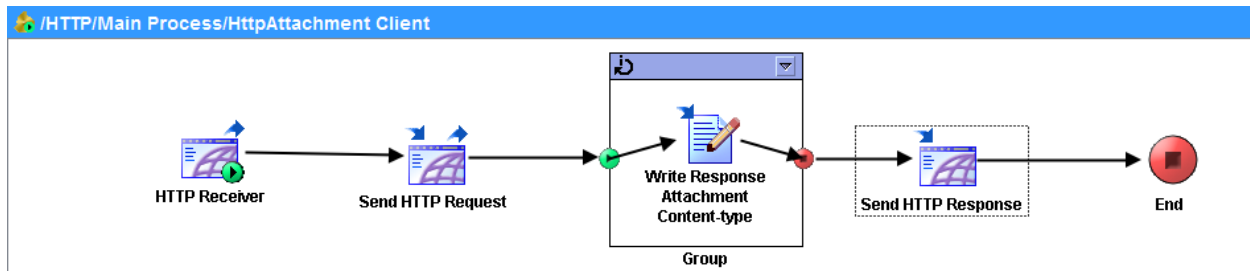
Here we send a HTTP request from the browser and receive it by a HTTP receiver and that in turn make a HTTP request to a server process which will send a HTTP response to the requestor now this response is used to send a HTTP response to the browser which is our image.

Prerequisites:

- HTTP Connection (Server)
- HTTP Connection (Client)

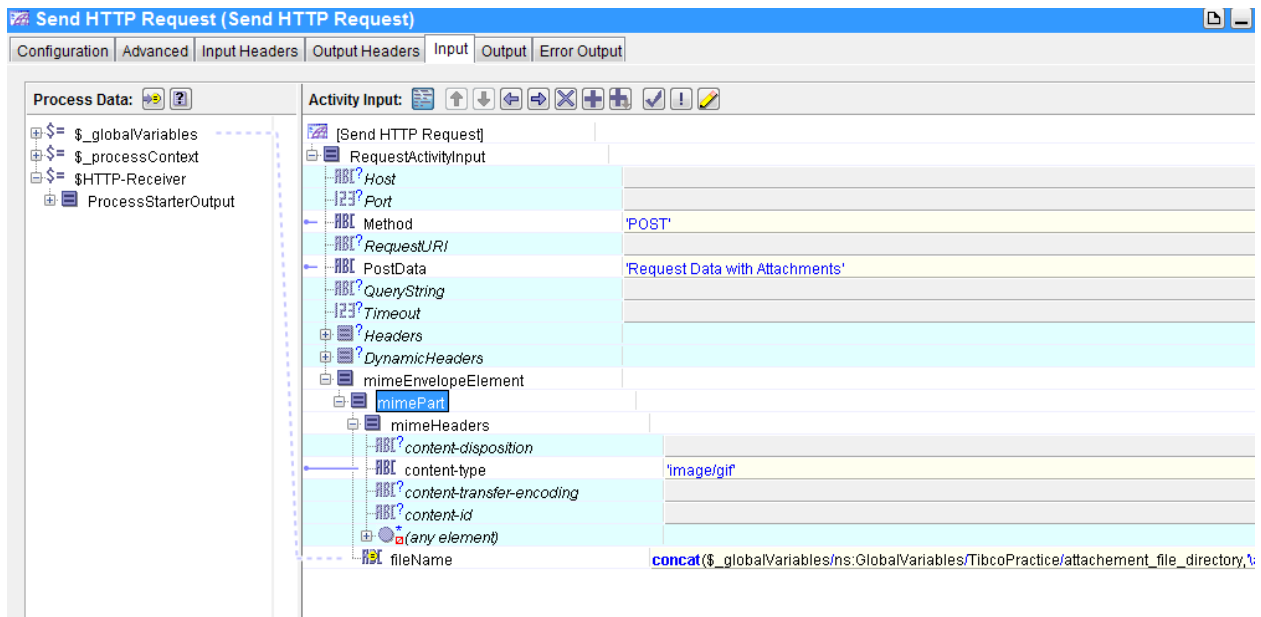
Palates required:

- HTTP Receiver
- Send HTTP Request
- Send HTTP Response

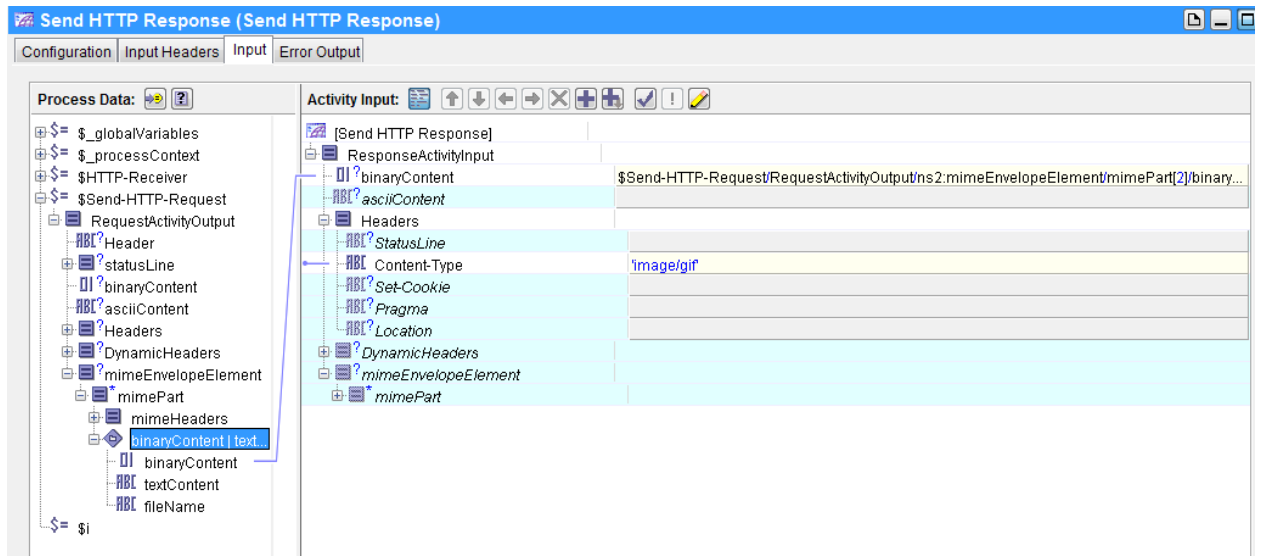


Steps to follow:

- For the HTTP Receiver set the connection as the HTTP Connection (client) and output style as string.
- For this Send HTTP Request set the hostname and port as that of the HTTP Connection (Server) then in the input set the `mimePart` filename with the file path of the image and give the content type in the `mimeHeader`.



- For the Send HTTP Response set the reply for as the HTTP Receiver and then in the input map the binaryContent with the the binaryContent of the mimepart of the Send HTTP Request output and give appropriate contentType like 'image/gif'.

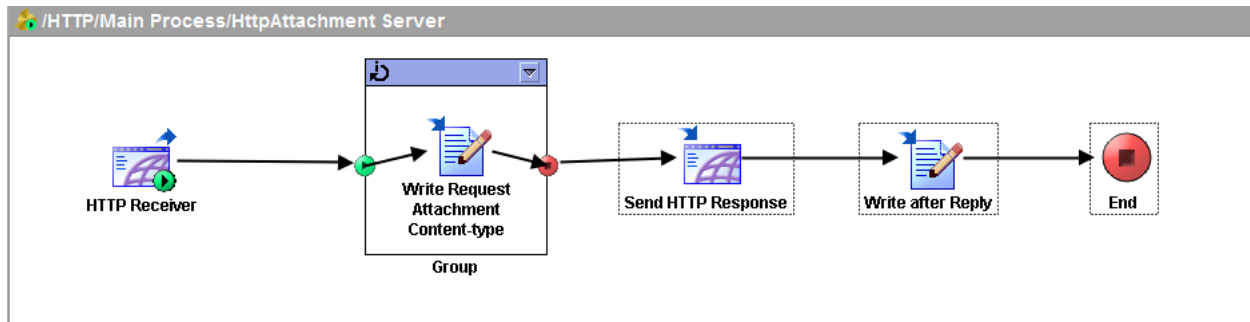


### Server

Here we receive the HTTP Request by the client by a HTTP Receiver and then send a HTTP Response back to the client.

Palates required:

- HTTP Receiver
- Send HTTP Response



Steps to follow:

- For the HTTP Receiver, set connection as HTTP Connection (server) and output style as string in the configuration tab.
- For the Send HTTP Response, set reply for as the HTTP Receiver in the configuration tab. Now go to the input tab, and map the mime part binary content with the mime part binary content of the HTTP Receiver and write the content type in the mimeTypeHeader as 'image/?' where ? can be any image file format like gif, jpeg, bmp etc.

**Send HTTP Response (Send HTTP Response)**

Configuration | Input Headers | Input | Error Output

**Process Data:**

- \$globalVariables
- \$processContext
- \$HTTP-Receiver
  - ProcessStarterOutput
    - Method
    - RequestURI
    - HTTPVersion
    - PostData
    - Command
    - QueryString
    - Header
    - Protocol
    - Port
    - Headers
      - DynamicHeaders
      - mimeEnvelopeElement
        - mimePart
          - mimeHeaders
            - content-disposition
            - content-type
            - content-transfer-encoding
            - content-id
            - (any element)
          - binaryContent | textContent...
          - binaryContent

**Activity Input:**

- [Send HTTP Response]
  - ResponseActivityInput
    - binaryContent
    - asciiContent
      - concat("Response\_asciiContent\_Field-", \$HTTP-Receiver/ProcessStarterOutput/Mel)
    - Headers
      - StatusLine: "HTTP/1.1 200 OK"
      - Content-Type
      - Set-Cookie
      - Pragma
      - Location
      - Server
      - DynamicHeaders
      - mimeEnvelopeElement
        - mimePart
          - mimeHeaders
            - content-disposition
            - content-type: "image/gif"
            - content-transfer-encoding
            - content-id
            - (any element)
          - binaryContent: \$HTTP-Receiver/ProcessStarterOutput/pfc:mimeEnvelopeElement/mimePart[1...

## JMS/EMS

JMS is Java Messaging Service. It's a service used to send messages between applications using SOAP or HTTP as the transport protocol. Typically a JMS Message has a specific standard with a message id for each message and a correlation id for the continuations or replies w.r.t. to that message.

There are different palates available for JMS. We will see their usage by discussing about some examples.

## JMS Request Reply

In this example, we will just use a JMS Queue Requestor where we put a message and send to the preferred queue. Now there will be queue receiver for that queue which will receive the message sent and then Reply to JMS Message will reply the JMS Queue Requestor with a response message with the correlation id as the sent message id. Here the response message will by default be send to the requesting queue.

### Sender

Prerequisites:

- JMS Connection

#### Steps to Create a JMS Connection

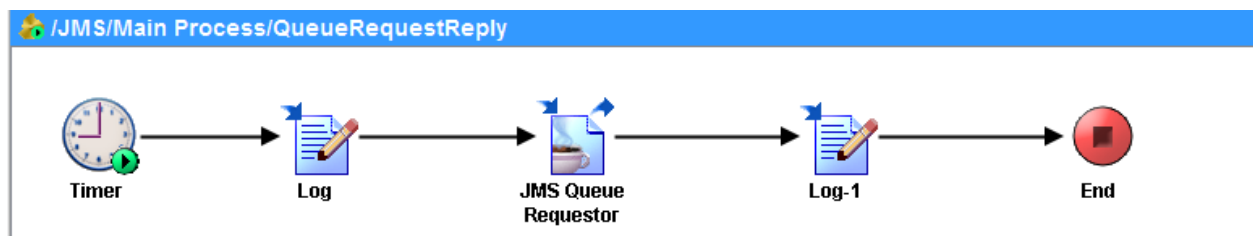
- Select the JMS Connection Palate from the list
- Check on use JNDI Connection Factory

The screenshot shows the 'EMS Connection (JMS Connection)' configuration window with the 'Advanced' tab selected. The configuration fields are as follows:

- Name: EMS Connection
- Description: (empty)
- User Name: (empty)
- Password: (empty)
- Auto-generate Client ID: ☒
- Client ID: (empty)
- SSL: ☐ Use SSL? (Configure SSL... button)
- Use JNDI for Connection Factory: ☒
- Use Shared JNDI Configuration: ☐
- JNDI Context Factory: com.tibco.tibjms.naming.TibjmsInitialContextFactory
- JNDI Context URL: tibjmsnaming://localhost:7222
- JNDI User Name: (empty)
- JNDI Password: (empty)

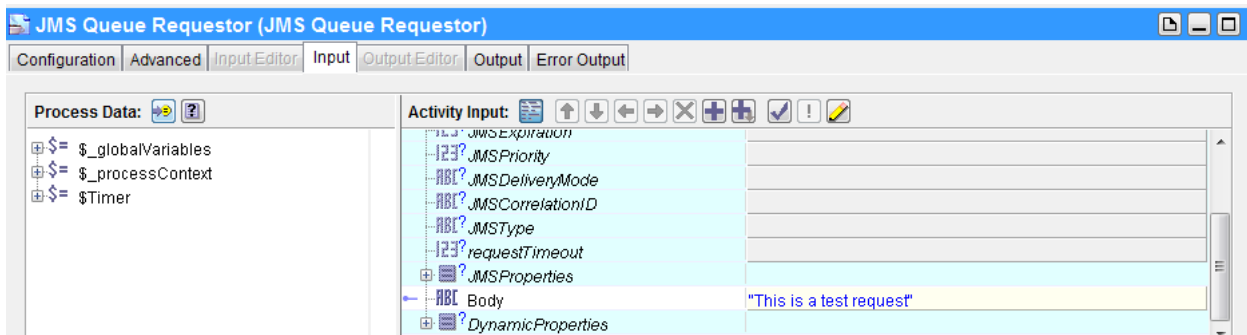
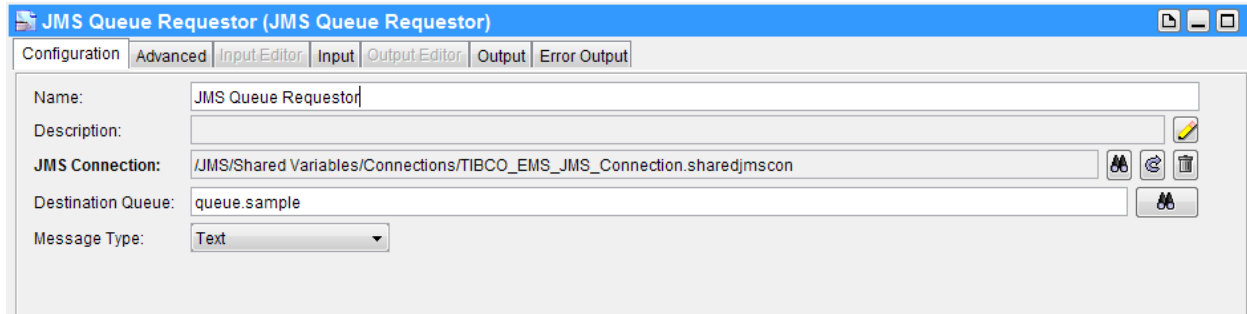
Palates required:

- JMS Queue Requestor



Steps to follow:

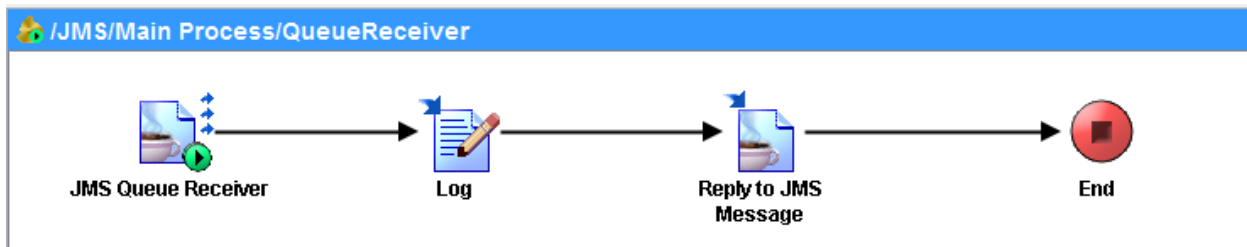
- In the configuration tab of the JMS Queue Requestor, give the JMS Connection and the destination queue name and message type as text. Go to the input tab, give a sample message in the body field.



## Receiver

Palates required:

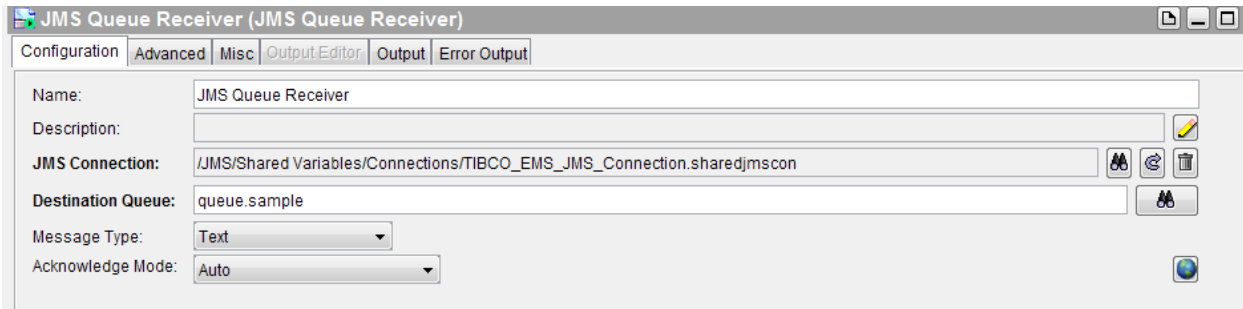
- JMS Queue Receiver
- Reply to JMS Message



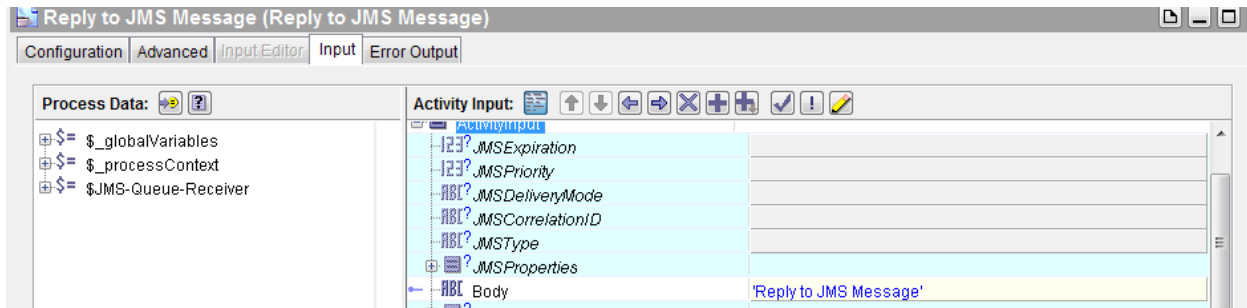
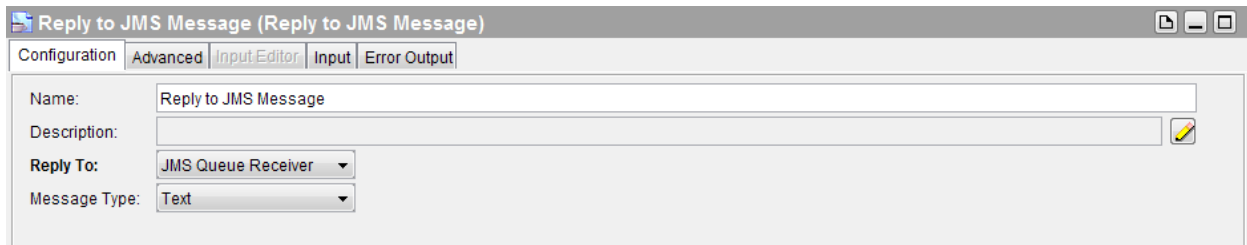
Steps to follow:

- For the JMS Queue Receiver, in the configuration select the JMS Connection and the give destination queue name as given for the sender. Set the Message type as text and acknowledgement as auto.





- For the Reply to JMS Message, select reply for as the JMS Queue Receiver. Go to the input tab, give a sample reply message in the body field.



After this run both the process and in the output tab of the JMS Queue Requestor you can see your given reply message and that confirms the flow.

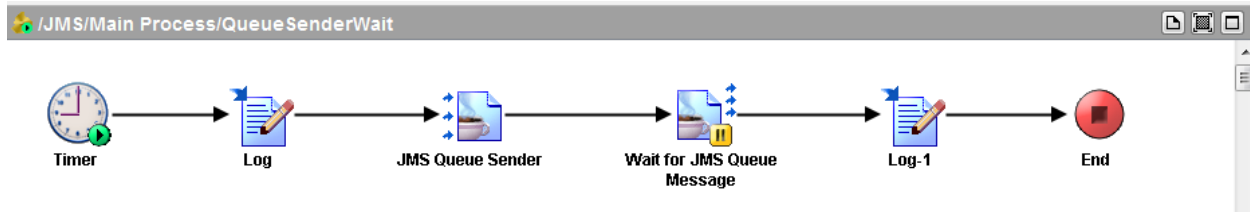
## JMS Sender Wait

In this example, we have a JMS Queue Sender then a Wait for JMS Message palate. JMS Queue Sender will send a message to a destination queue then in another process there will be a JMS Receiver which will receive the message and then the Reply to JMS Message will reply to this message in the queue name given as the destination queue in Wait for JMS Message Palate. Here we can set the reply queue by giving the desired queue name in the destination queue in the Wait for JMS Message palate.

### Sender

Palates required:

- JMS Queue Sender
- Wait for JMS Queue Message



Steps to follow:

- For the JMS Queue Sender, give the JMS Connection and name of the destination queue. In the input tab give a sample message in the body field.
- For the Wait for JMS Queue Message give the JMS Connection and name of the destination queue where you want the reply message from the receiver process to be delivered. Set the message type as text and acknowledgement as auto. Now go to Message Event tab and set the Candidate Event Key as the correlation id of the reply message. Then go to the input tab and map the key field with the MessageId of the JMS Queue Sender message.

**Wait for JMS Queue Message (Wait for JMS Queue Message)**

Configuration | Advanced | **Message Event** | Input | Output Editor | Output | Error Output

Name: Wait for JMS Queue Message

Description:

JMS Connection: /JMS/Shared Variables/Connections/TIBCO\_EMS\_JMS\_Connection.sharedjmscon

Destination Queue: queue.reply

Message Type: Text

Acknowledge Mode: Auto

**Wait for JMS Queue Message (Wait for JMS Queue Message)**

Configuration | Advanced | **Message Event** | **Input** | Output Editor | Output | Error Output

Candidate Event Key: ns1.ActivityOutput.JMSHeaders.ns1.JMSCorrelationID

The candidate event key can be used to filter events. Only events where the candidate event key matches the key provided as input will trigger this activity.

Event Timeout (msec): 60

The amount of time an event will wait (in milliseconds) if it is received before this task is reached within the process. If the event timeout expires then the event is discarded.

**Wait for JMS Queue Message (Wait for JMS Queue Message)**

Configuration | Advanced | **Message Event** | **Input** | Output Editor | Output | Error Output

**Process Data:**

- \$\_globalVariables
- \$\_processContext
- \$Timer
- \$JMS-Queue-Sender
  - aEmptyOutputClass
    - MessageID

**Activity Input:**

| Field          | Value  |
|----------------|--|
| key            | \$JMS-Queue-Sender/ns1:aEmptyOutputClass/ns1:MessageID |
| processTimeout |  |

## Receiver

Same process as for JMS Request Reply

## JMS Message Selector

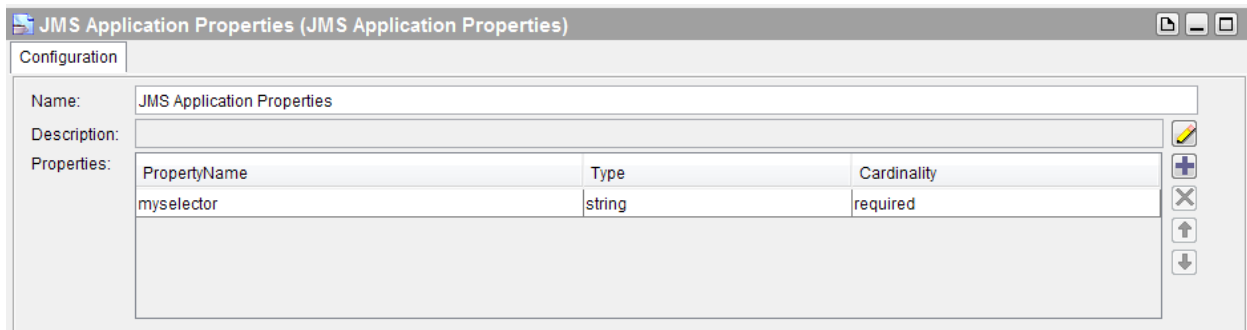
Here we will send five messages in a queue by looping on JMS Queue Sender from there we will only fetch the third message using Get JMS Message palate. We have to create a customized property for the messages we send so that based on the property value we can do the desired selection and for that we need to create JMS Application Property.

Prerequisites:

- JMS Connection
- JMS Application Properties

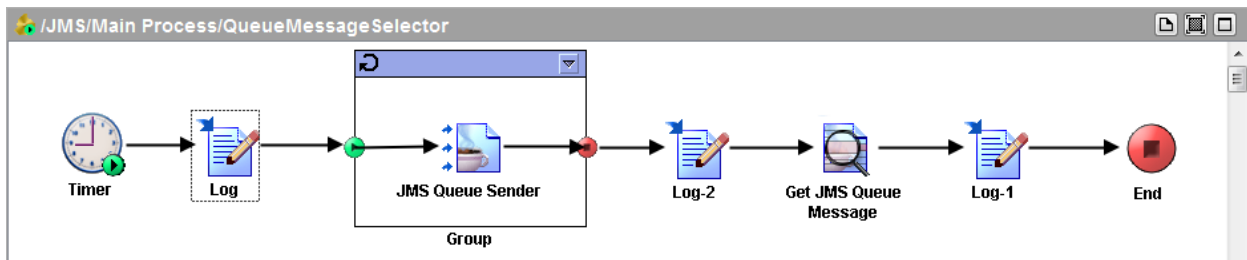
### Creating a JMS Application Property

- Select JMS Application Properties palate then a property with name 'myselector' of type string and cardinality required.



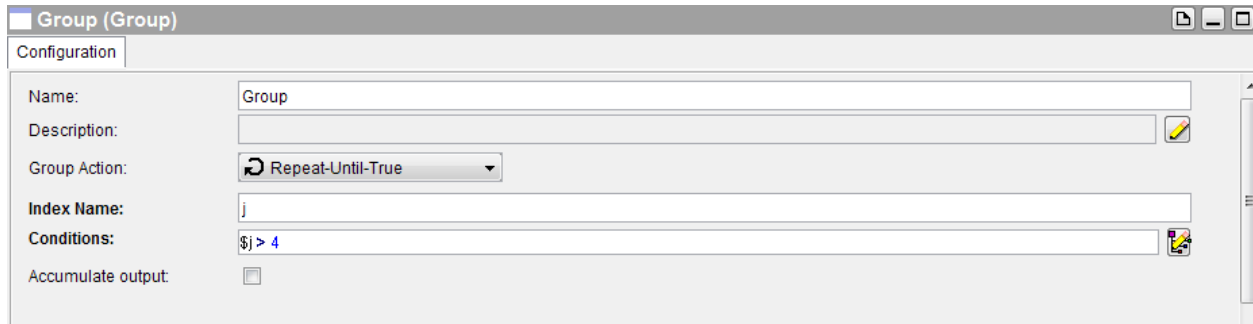
Palates required:

- JMS Queue Sender in a group
- Get JMS Queue Message

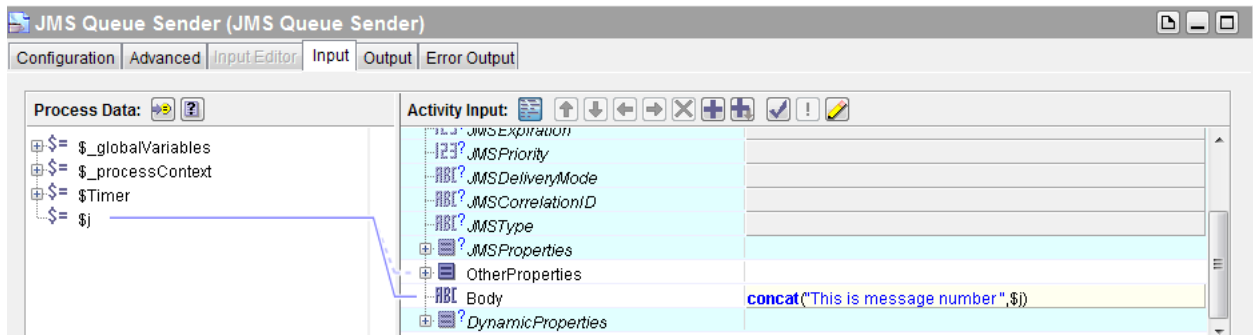


Steps to follow:

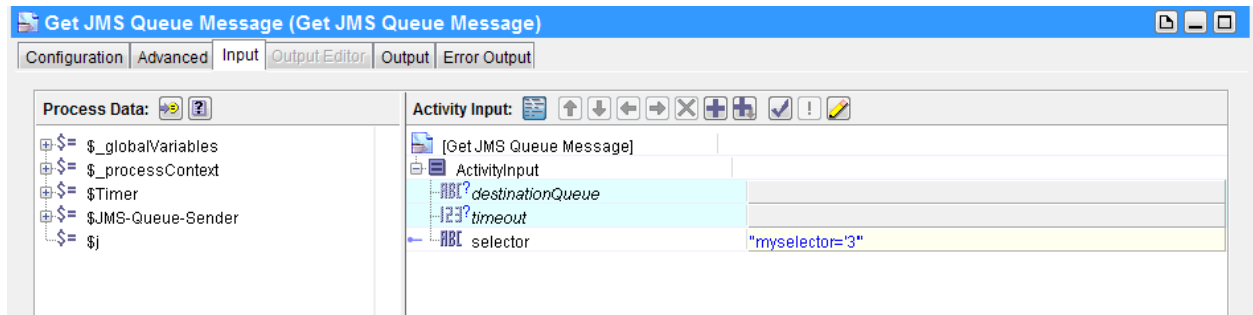
- For the group activity, set group action as repeat until true and give a index variable name like i or j and give the conditions as  $j > 4$ . So it will loop until the value of j goes to 5 starting from 1.



- For every loop, in the JMS Queue Sender set a JMS Connection and a destination queue name then in the advanced tab select the JMS Application Property and in the input tab map 'myselector' with index variable and also give a message in the body like "concat ('This is message ', \$j)" so as to understand the message number by its content only.



- Now in the Get JMS Queue Message give the JMS Connection, queue name, message type and in advanced tab select the JMS Application Property and then in input tab give the selection string for the message like "myselector='3'".



Now after running the flow we can see the third message in the output body of the Get JMS Queue Message.

## JMS Topic

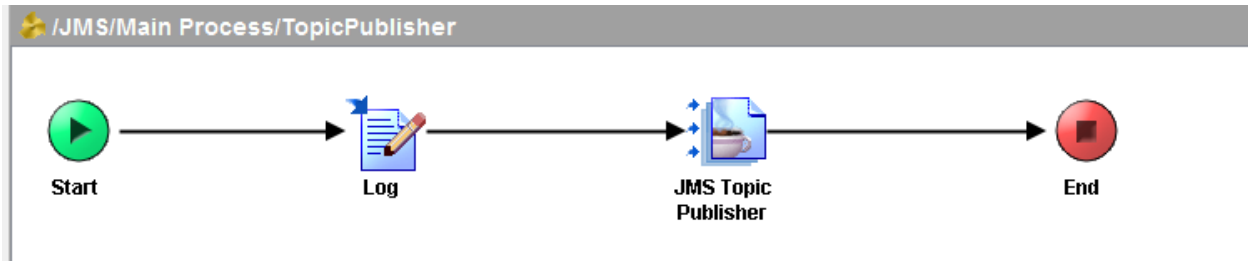
This is simple where we will publish a message in a topic using JMS Topic Publisher then in another process there will be a JMS Topic Subscriber which will be listening on the same topic and on publication of the message will receive the message. If the subscriber is made durable over a topic then any

message intended to it will remain in the topic if it's also not listening at the time the message is published then when it comes online it will receive the intended message.

### Publisher

Palates required:

- JMS Topic Publisher



Steps to follow:

- For the JMS Topic Publisher, set a JMS Connection and a destination topic name and under the input tab give a sample text message in the body field.

The screenshot shows the "JMS Topic Publisher (JMS Topic Publisher)" configuration window. The "Input Editor" tab is selected. The fields are as follows:

- Name: JMS Topic Publisher
- Description: (empty)
- JMS Connection: /JMS/Shared Variables/Connections/TIBCO\_EMS\_JMS\_Connection.sharedjmscon
- Destination Topic: topic.sample
- Message Type: Text

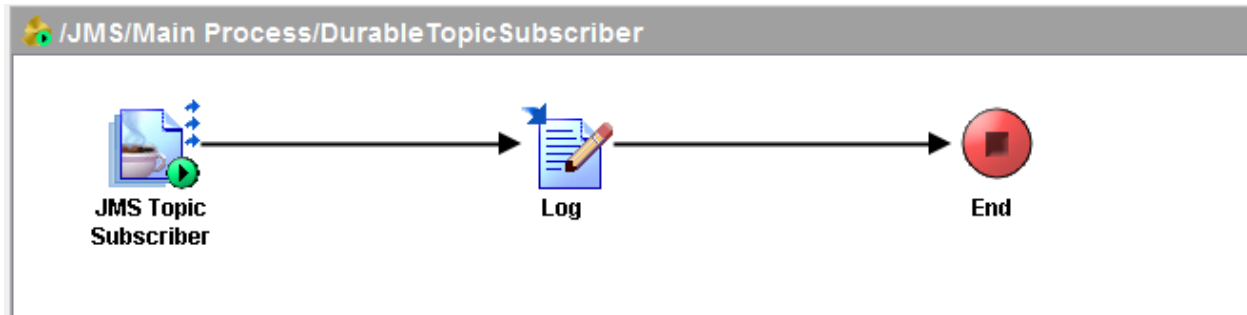
The screenshot shows the "JMS Topic Publisher (JMS Topic Publisher)" configuration window, specifically the "Input" tab. The "Activity Input" section is visible, showing a list of properties for the message:

- replyToTopic
- JMSExpiration
- JMSPriority
- JMSDeliveryMode
- JMSCorrelationID
- JMSType
- JMSProperties
- Body: "This is a test Topic Message"
- DynamicProperties

### Subscriber

Palates required:

- JMS Topic Subscriber



Steps to follow:

- For the JMS Topic Subscriber, set a JMS Connection, same destination topic name as like the publisher. Check on durable subscription and give a subscription name. Set acknowledgement mode as auto.

|                          |   |
|--------------------------|---|
| Name:                    | JMS Topic Subscriber  |
| Description:             |   |
| JMS Connection:          | /JMS/Shared Variables/Connections/TIBCO_EMS_JMS_Connection.sharedjmscon |
| Destination Topic:       | topic.sample  |
| Message Type:            | Text  |
| Durable Subscription:    | <input checked="" type="checkbox"/>                                     |
| Subscription Name:       | MyDurableSubscriber   |
| Suppress Local Messages: | <input type="checkbox"/>  |
| Acknowledge Mode:        | Auto  |

Now after running the two processes we can see the published message in the output body of the subscriber.

Here we have made durable subscription that mean suppose if the publisher process is run first and then after a time gap the subscriber process is run then also our subscriber will receive the message published in the topic. If it was not set durable and the same scenario was upheld the message won't be received. Then to get the message we needed to run both the process together or in other terms the non-durable subscriber must be listening to the topic when the publisher publishes a message to get that message where areas a durable subscriber won't have to.

### JMS Local/XA Transaction

Transactions means ACID property is maintained that is when there multiple jobs are dependent on each other and had to be completed in one shot or else the entire process fails is called transactions.

So for transactions we need a stable connection to the JMS connection factory. If the transaction is done locally then we will use our JNDI Connection Factory but it's an distributed transactions multiple network connected queues or topics will be updated then we use XAConnection factory as it provide more secure transport and the protocols and standards are more robust.

### JMS XA Connection

To create a JMS XA Connection

- Select JMS Connection as we did before but not check the use JNDI Connection Factory instead just use XA Connection Factory.

**EMS XA Connection (JMS Connection)**

Configuration **Advanced**

Name: EMS XA Connection

Description:

User Name:

Password:

Auto-generate Client ID: ☒

Client ID:

SSL: ☐ Use SSL? Configure SSL...

Use JNDI for Connection Factory: ☐

Provider URL: tcp://localhost:7222

Use XA Connection Factory: ☒

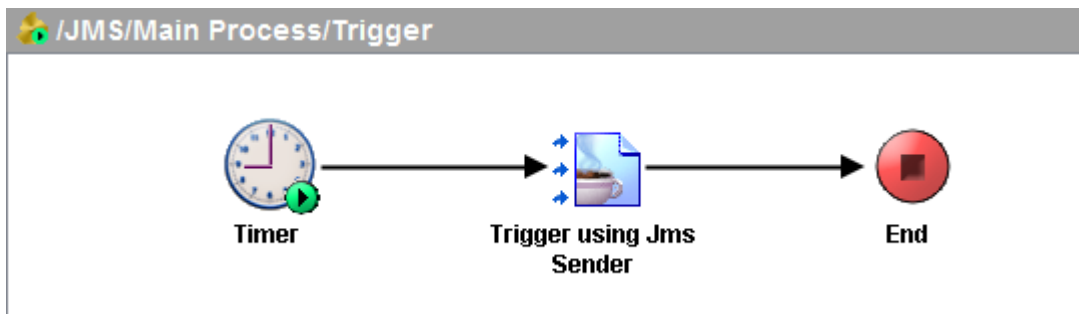
Use UFO Connection Factory: ☐

### Trigger

It's just a message which will be used as a trigger for the transaction process.

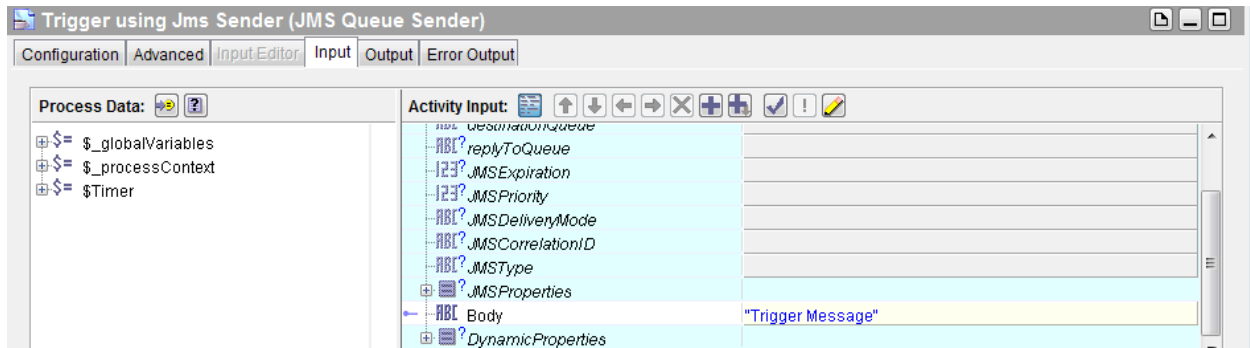
Palates required:

- JMS Queue Sender



Steps to follow:

- In the configuration tab for JMS Queue Sender, set the JMS\_XA Connection, destination queue and a sample message.

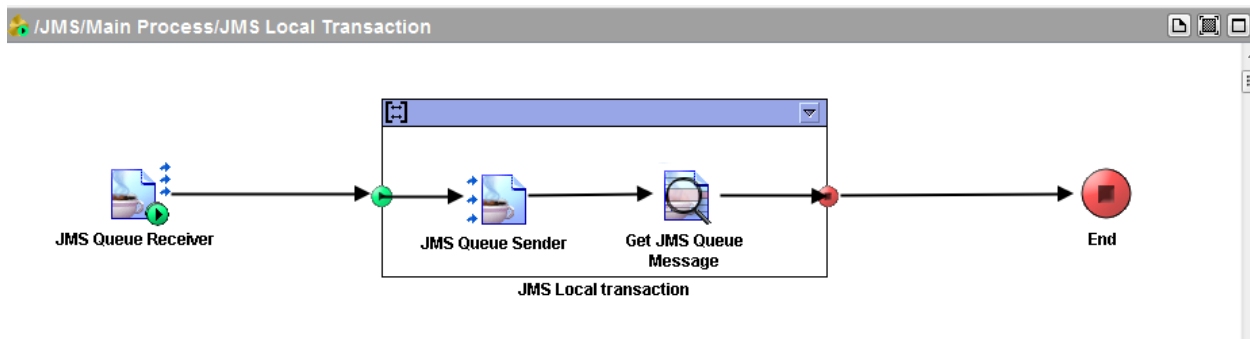


### JMS Local Transaction

Local Transaction demands a transaction to happen. When we say transaction we mean multiple activities grouped together if one fails the whole thing fails. Only if all the activities are successful commit happens.

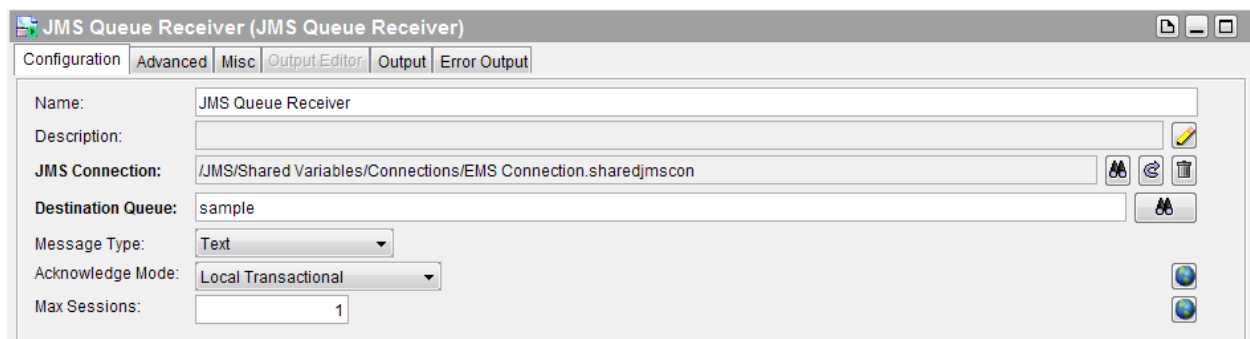
Palates required:

- JMS Queue Receiver
- JMS Queue Sender and Get JMS Message in a group



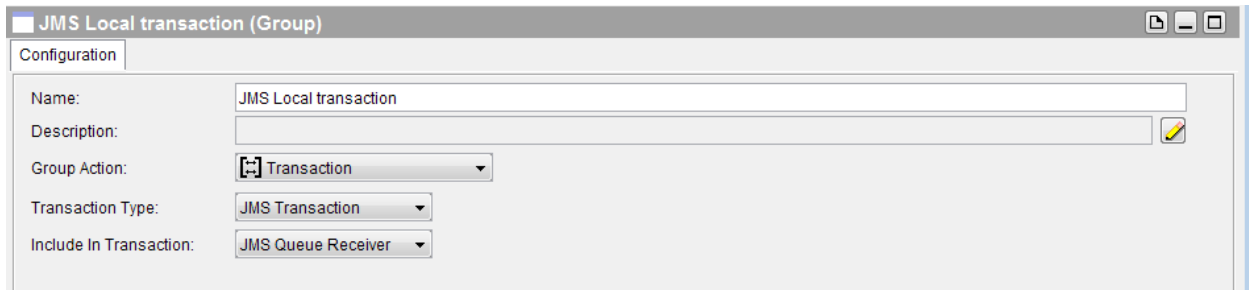
Steps to follow:

- In the JMS Queue Receiver, select the destination queue as used in the trigger, give a JMS Connection. Select Acknowledgement mode as local transactional and message type as text.

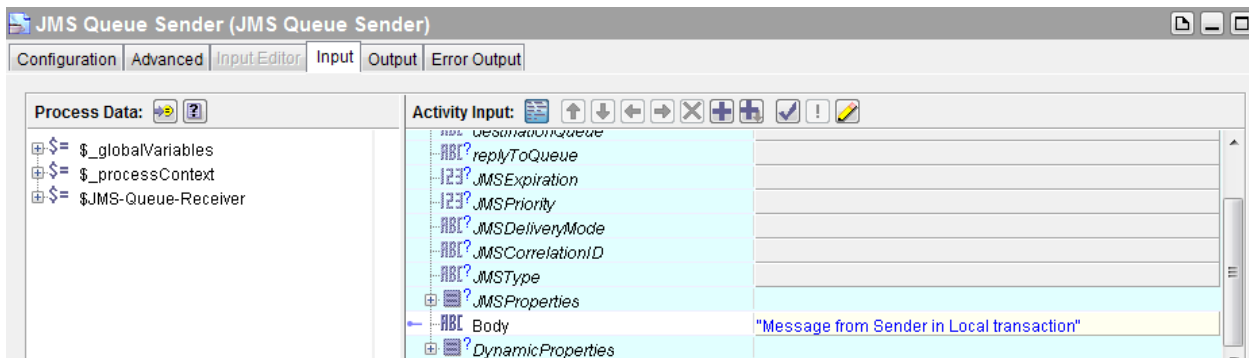




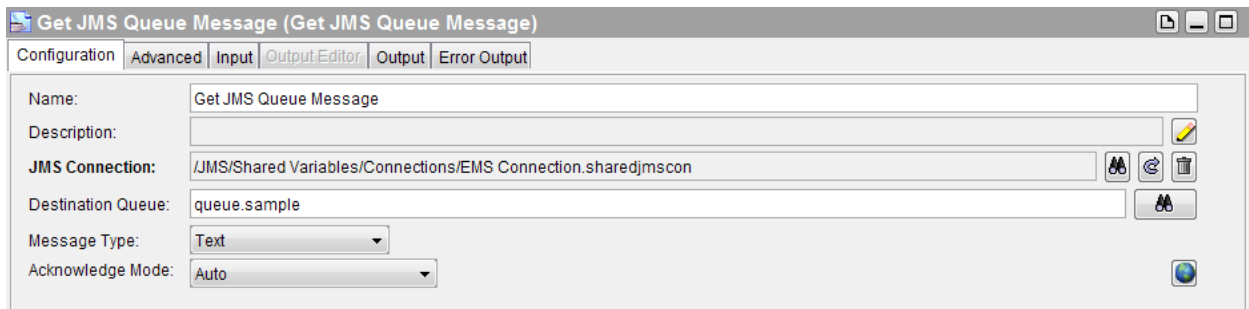
- For the group, give group action as transactional. Select Transaction Type as JMS Transaction and include JMS Queue Receiver in the transaction.



- For JMS Queue Sender, set a input message to any queue.



- For Get JMS Message, select a destination queue same used in the JMS receiver.



After running the Trigger and the Local Transaction Flow we can see our Trigger Message in the output of Get JMS Message.

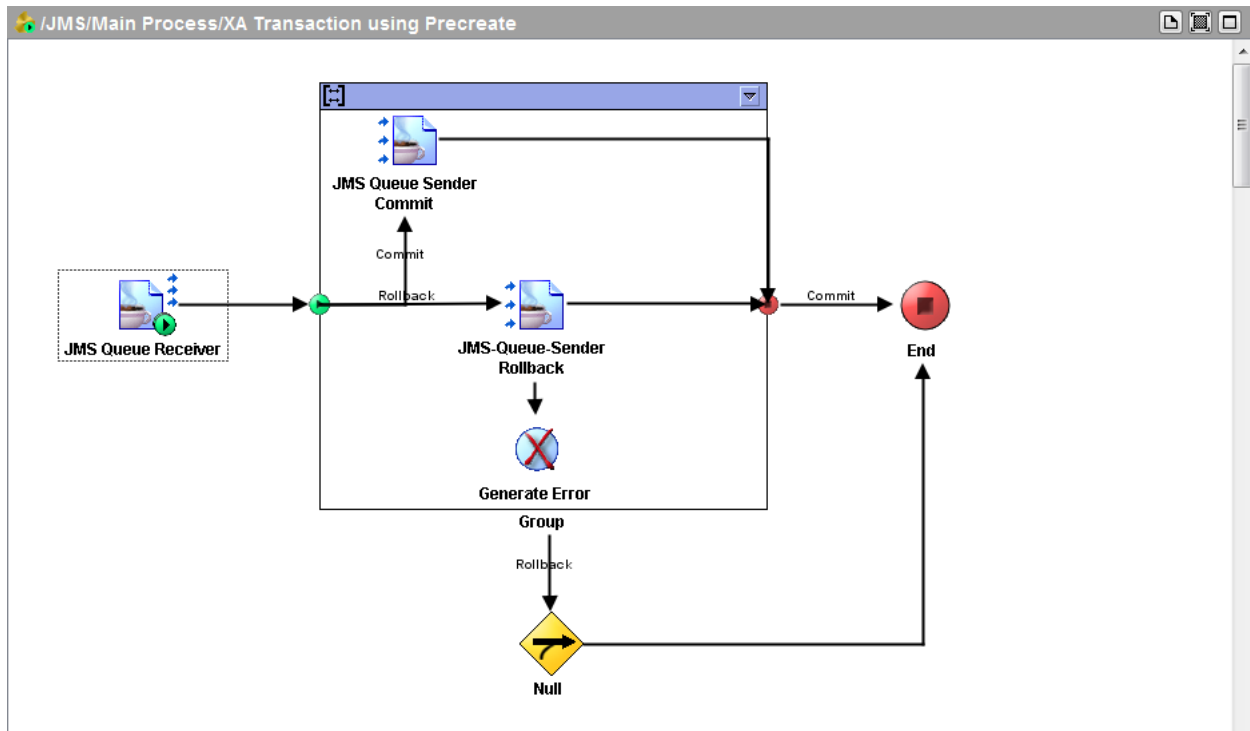
### *XA Transaction using Precreate*

XA transaction has the fundamentals of commit and rollback. We will use the same Trigger we used for Local Transaction.

For this flow,

Palates required:

- JMS Queue Receiver
- JMS Queue Sender-Commit
- JMS Queue Sender-Rollback
- Generate Error
- Null



Working:

- All Queue Sender and Generate Error will be in the group with the action as transaction.
- A trigger message will be received by the JMS Queue Receiver and then it will flow to the transactional group where it will flow to the commit sender or rollback sender depending upon the success condition. If its commit then it will flow to end. Else if rollback it will generate an error null message and outside the group it will flow through the error connector to check for null then to end.

For the details please explore the flow. It's like a collection of everything we learnt so far.

## JDBC

Well JDBC palates are there to establish a link to the database and perform operations on the database. TIBCO by default provides support connectivity for multiple database vendors like oracle, mysql, microsoft sql server, weblogic, skybase and others. So to connect to these database we just need to have the vendor specific driver jar file downloaded and put into the jdbc folder under tpcl directory in the TIBCO installation directory.

Here we will use oracle thin driver for Oracle 11g express edition for our examples.

### JDBC Connection

To create any link to a database we need to use the JDBC Connection node and set its configuration variables with right values for successful connection.

#### Creating a JDBC Connection:

- Select the connection as JDBC in the configuration of the JDBC Connection palate.
- Set the JDBC driver from the list and give the driver url.
- Set the Database url.
- Give the authentication username and password.

The screenshot shows the 'JDBC\_ORACLE (JDBC Connection)' configuration window. The 'Configuration' tab is active. The fields are as follows:

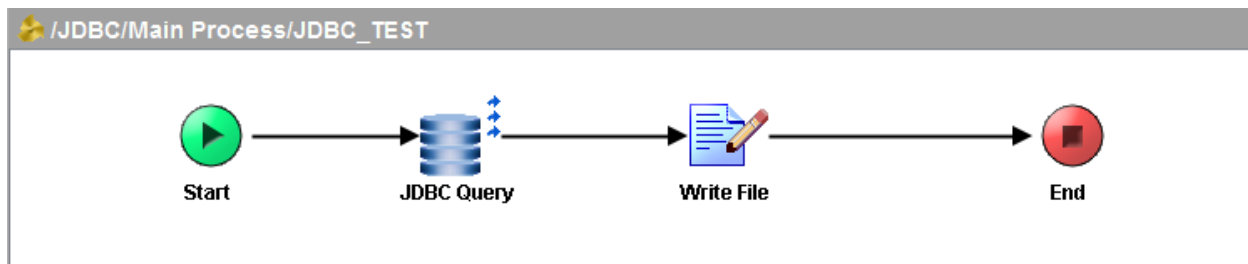
| Field                | Value                                  |
|----------------------|--|
| Name:                | JDBC_ORACLE                            |
| Description:         |  |
| Connection Type:     | JDBC                                   |
| JDBC Driver:         | oracle.jdbc.driver.OracleDriver (thin) |
| Database URL:        | jdbc:oracle:thin:@localhost:1521:XE    |
| Maximum Connections: | 10                                     |
| User Name:           | system                                 |
| Password:            | ••••••••                               |
| Login Timeout(sec):  | 0                                      |

### JDBC Query

Let's now see how to execute a sql statement from TIBCO and get the results and then write that results to a file. To execute a sql query we have to use the JDBC Query palate.

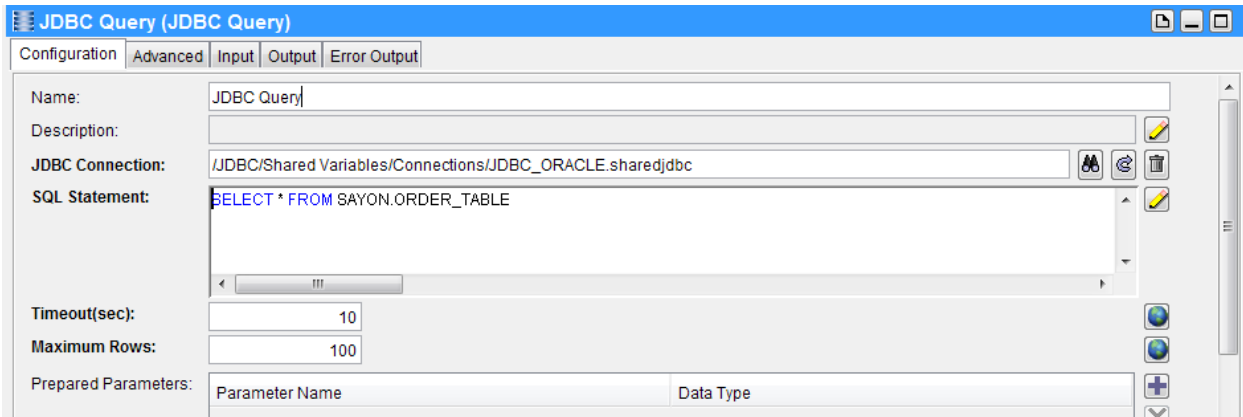
Palates required:

- JDBC Query
- Write to File

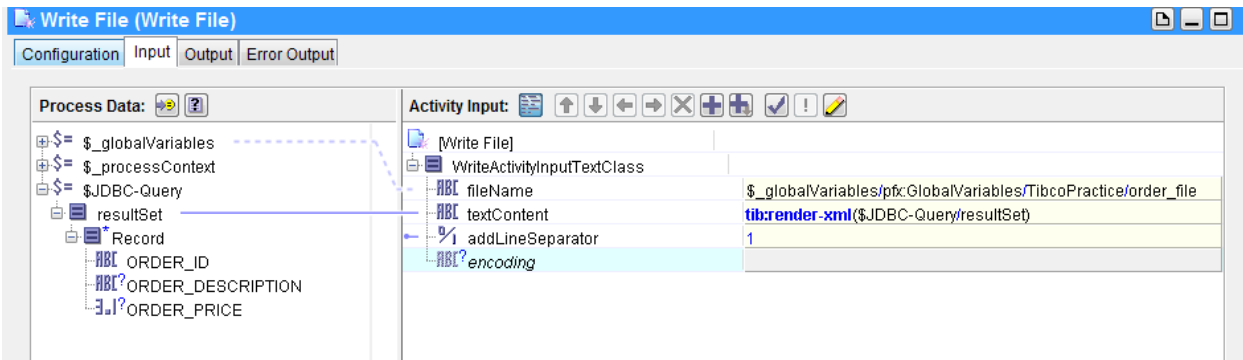


Steps to follow:

- For the JDBC Query, set the JDBC Connection and write your sql statement and test it.

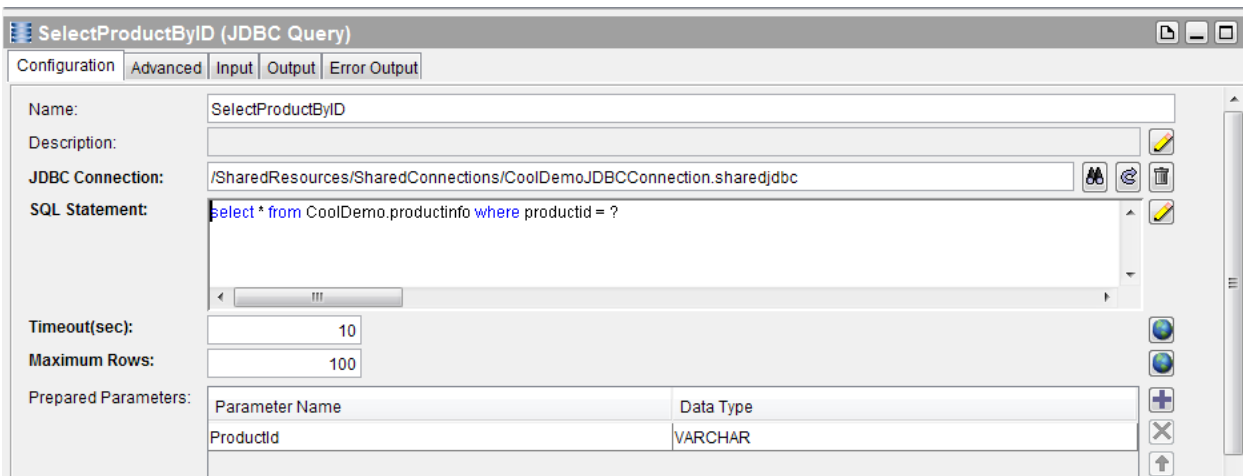


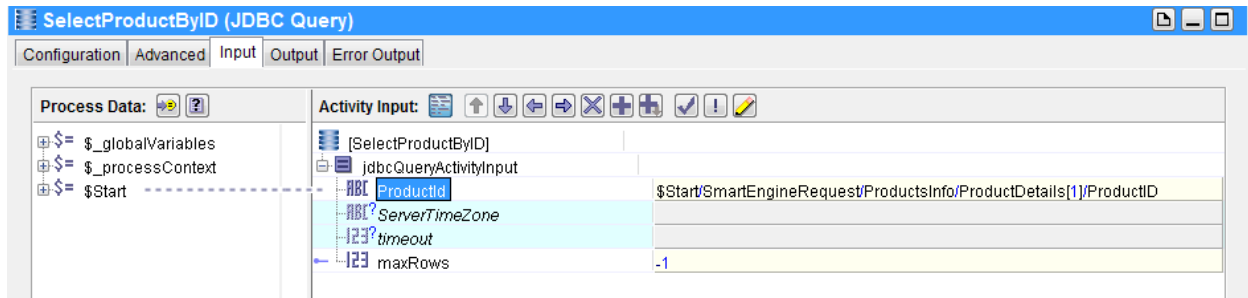
- Go to the input tab of write file and then set the filename and map the textContent as the output resultset of the JDBC Query node.



#### Notes on SQL Statement:

- We can execute a DDL or DML statements in the sql block. For specific insert query we have even set prepared parameters and map values to them having the feature to use them dynamically. The usage will depend upon the sql statement. For example (external reference just for understanding),





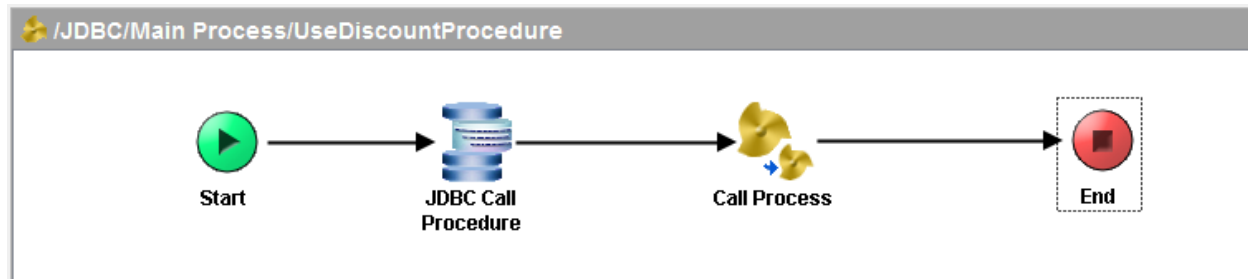
Now after running the process we can see the records are written in the file in text format.

### JDBC Procedure Call

We can create procedures and functions in the database which can do a multiple set of operations in one shot. In TIBCO we can all call such procedures using the JDBC Call Procedure palate.

Palates required:

- JDBC Call Procedure
- Call Process



Working:

- We will call a JDBC Procedure for example discount (number percent) where it will just update a data of column by the percent given as input. No return for this procedure.
- After the procedure is called the table will be updated and then we call our earlier JDBC Query process where we perform a simple select operation, get the changed records and write into a file.

Steps to follow:

- For JDBC Call Procedure, in its configuration, set the JDBC Connection, select the schema and the procedure. If the procedure has some in variables then we have to create a parameter variable. After we are done adding a parameter as input go to the input tab and expand the inputSet under which the parameter name field will be there. Map desired value to that field which will be the input for the procedure.

**JDBC Call Procedure (JDBC Call Procedure)**

Configuration | Advanced | Input | Output | Error Output

Name: JDBC Call Procedure

Description:

JDBC Connection: /JDBC/Shared Variables/Connections/JDBC\_ORACLE.sharedjdbc

Schema: SAYON

Catalog / Package:

Procedure/Function Name: DODISCOUNT

Timeout(sec): 10

Maximum Rows: 100

Parameter Types:

| Parameter Name | Parameter Types   |
|----------------|-------------------|
| DISC_PERCENT   | procedureColumnIn |

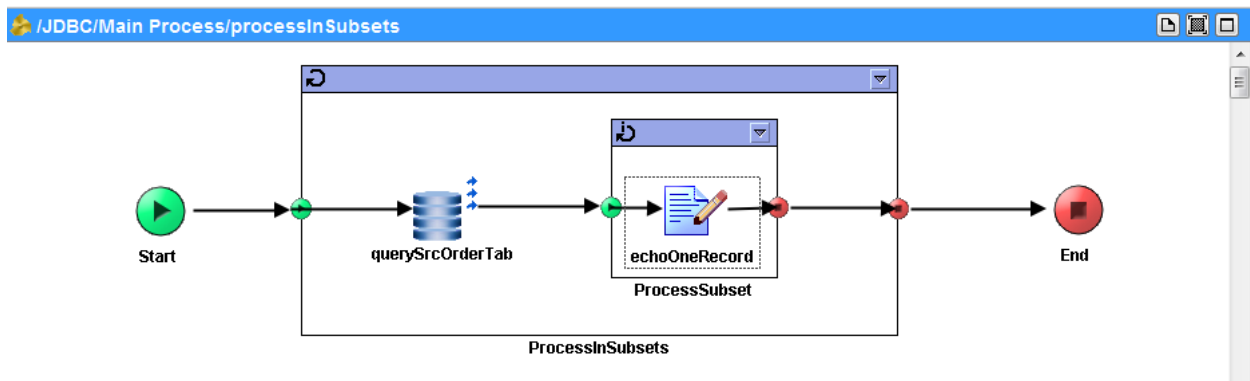
- For Call Process we will just select the above JDBC Query Process.

### Batch Processing

Supposedly imagine we are working on database table which have huge amount of rows. Therefore fetching all the rows in one shot and processing it will be resource eating. So there is a need of process data in batches or subsets like if we have 100 rows in a table and we did a fetching in subsets of 10. So it will be like fetch first 10 rows in first shot then process those data then again in the second pass take second 10 rows and so on. Doing will like this will increase efficiency and will be resourceful.

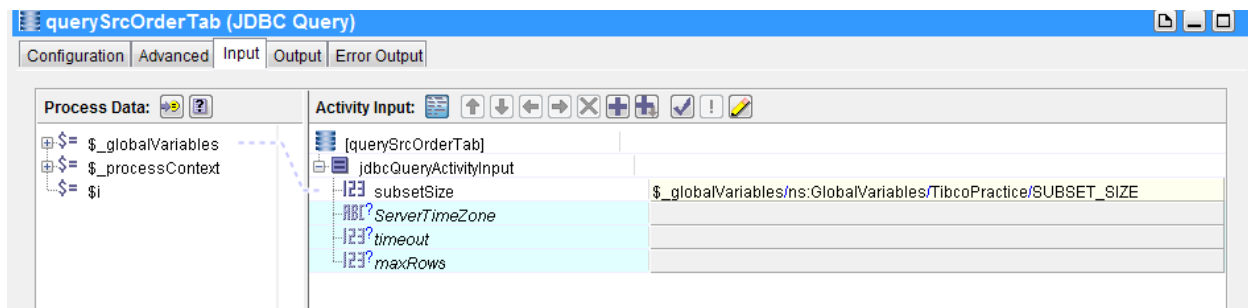
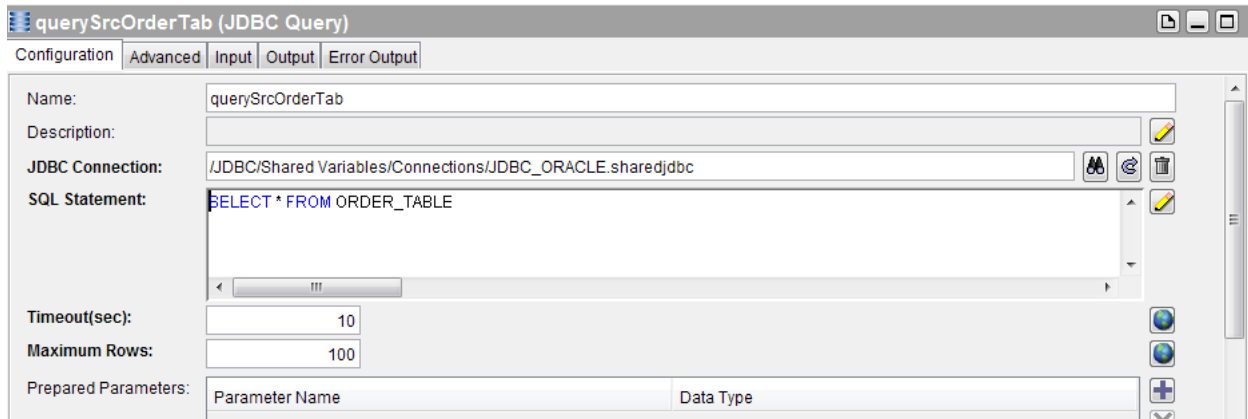
Palates required:

- Write file in sub-group
- JDBC Query and the Write File sub-group inside a parent-group

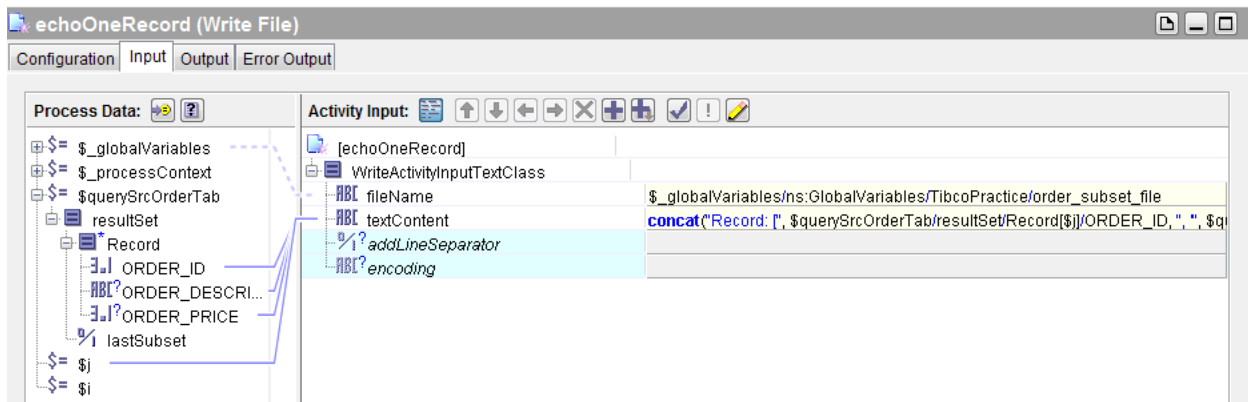


Steps to follow:

- For the JDBC Query, set the JDBC Connection and the sql statement in the configuration tab. In the advanced tab check on Process as Subsets. Next in the input tab give the subset size.



- Now in the Write File give the filename and map the record which is output of the JDBC Query as the textContent.



- For the Write File Sub-Group set group action as iterate and iterate on the Records of the JDBC Query output resultset.

**ProcessSubset (Group)**

Configuration

Name: ProcessSubset

Description:

Group Action: Iterate

Index Name: i

Variable List: \$querySrcOrderTab/resultSet/Record

Iteration Element:

Accumulate output:

- For the Parent group select group action as repeat until true and set the condition where we check whether it's the last subset for JDBC Query or not. For example, \$querySrcOrderTab/resultSet/lastSubset = "true".

**ProcessInSubsets (Group)**

Configuration

Name: ProcessInSubsets

Description:

Group Action: Repeat-Until-True

Index Name: i

Conditions: \$querySrcOrderTab/resultSet/lastSubset = "true"

Accumulate output:

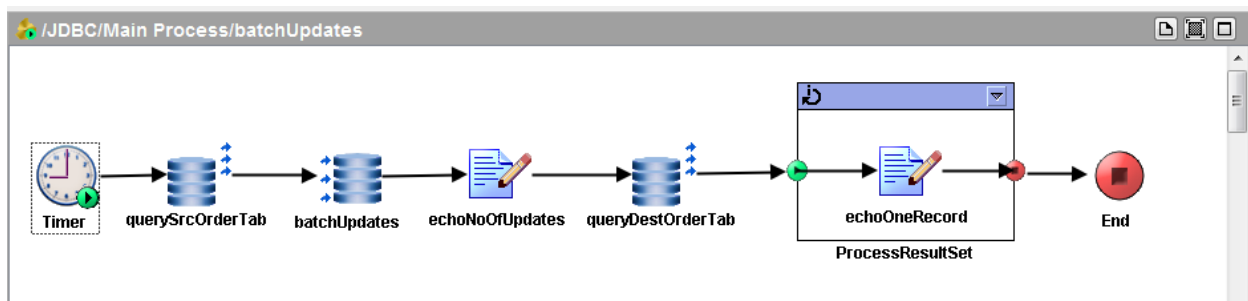
## JDBC Update

We have JDBC Update palate for update operations. We can do insert operations also.

Here we will copy the entire contents of one table into another using JDBC Update.

Palates Required:

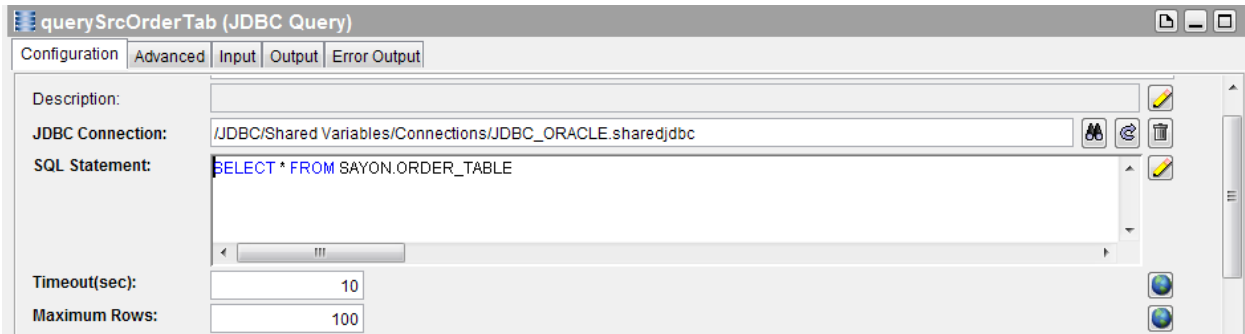
- JDBC Query
- JDBC Update
- Write File



Steps to follow:

- For JDBC Query we will do the same as we did in earlier case when we used JDBC Query in a simple way. Its output will be a resultset of multiple records.

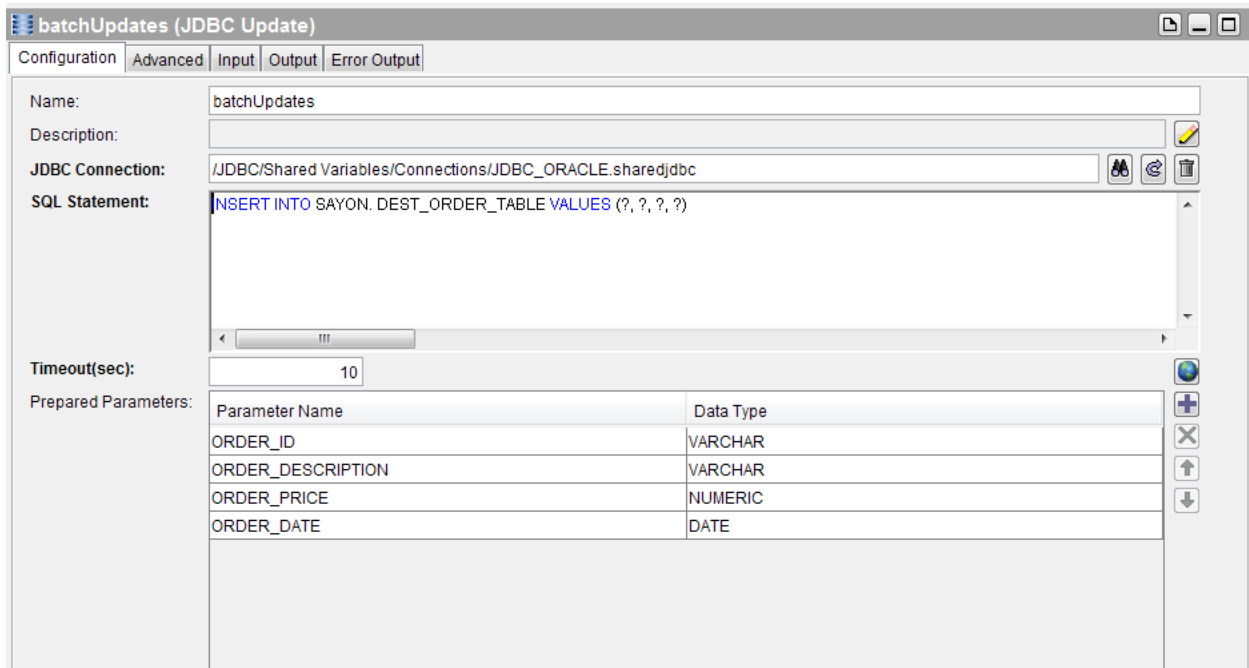




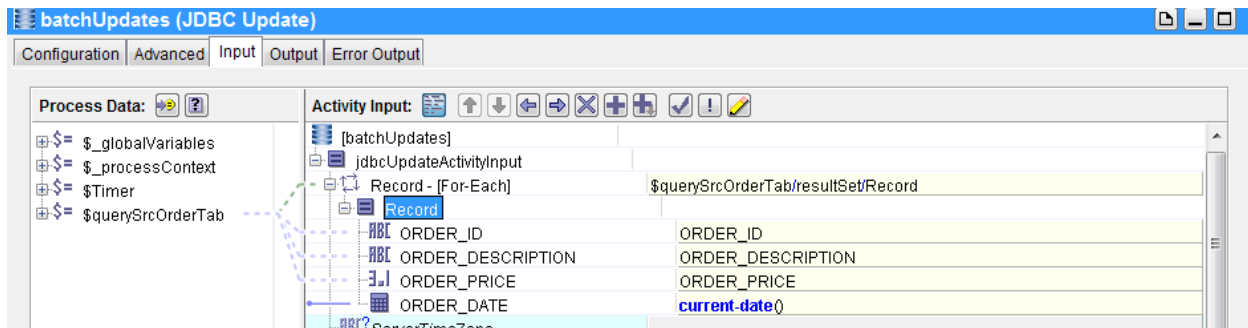
- For JDBC Update we will set a connection then write the insert query like

INSERT INTO DEST\_ORDER\_TABLE VALUES (?, ?, ?, ?)  
and set four parameters for four ? marks like this →

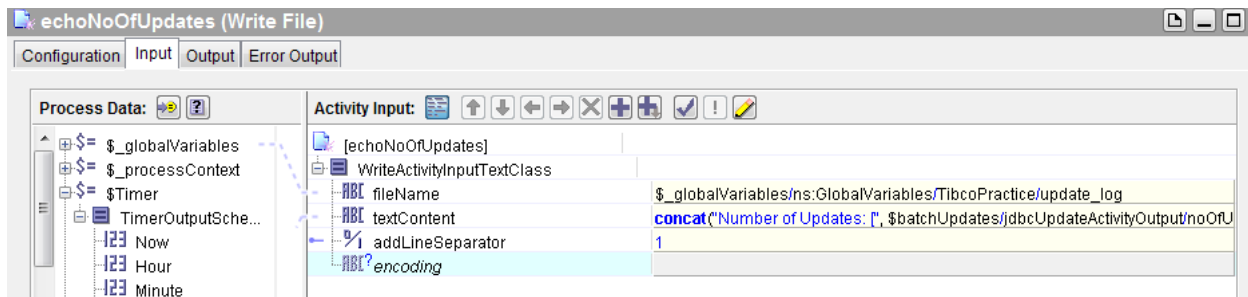
|                   |         |
|-------------------|---------|
| ORDER_ID          | VARCHAR |
| ORDER_DESCRIPTION | VARCHAR |
| ORDER_PRICE       | NUMERIC |
| ORDER_DATE        | DATE    |



- Now go to the input tab and map the record of the JDBC Query output to the record of the JDBC Update and make it for each.



- For the write file node, give a file name and set the textContent with output no of Updates of the JDBC update node.



After running the flow, we will get the total update count written in the file.

## SOAP/SERVICE

Trend now is now shifting to web services. TIBCO provides us with the facility to create web services. Web services are of two types SOAP and REST. SOAP uses XML as its message format and HTTP or JMS as its transport. SOAP has a well-defined structure for each message. Like SOAP Headers, body and trailer all enclosed in a SOAP envelope. For error messages, we have SOAP Fault messages. Such message structure makes it more secure but its little heavy. A SOAP Service can have multiple operations and based upon the operations we have its input or output. All these information of a SOAP Service is construction in a WSDL Document.

We will here show a basic example of creating a web service to add two numbers.

For this basic operation our input structure would be

- inputA(number)
- inputB(number)

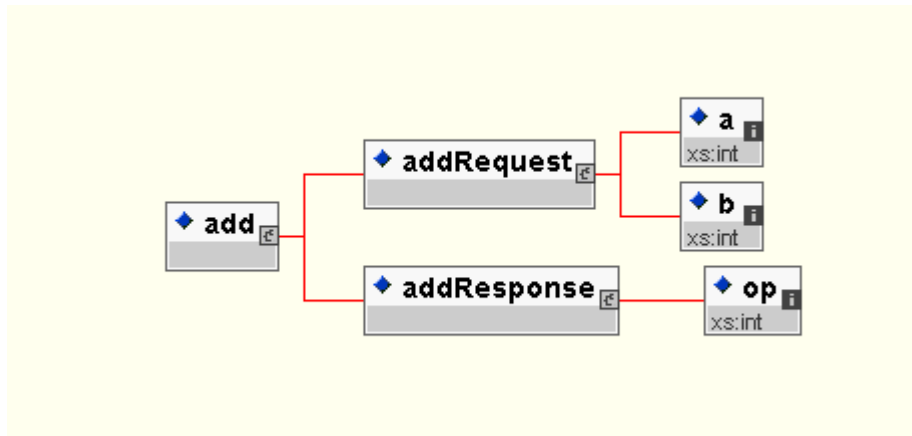
and output structure will be

- output(number)

We can create a XML Schema for our input and output like below:

- root (Element)

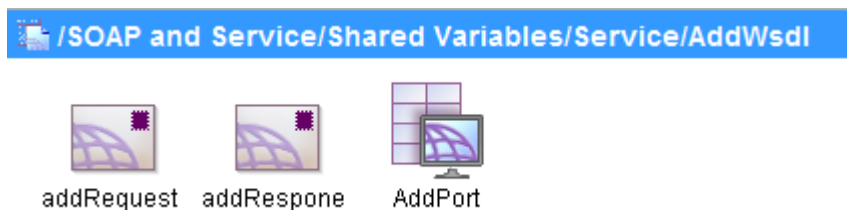
- input (Element)
  - inputA (int)
  - inputB (int)
- output(Element)
  - op (int)



After creating the schema we can create the Abstract WSDL.

### Building an abstract WSDL

An abstract WSDL is one where we just define the operations name and its input and output message and the data types of the I/O parameters of the I/O messages. No soap binding or port binding is defined here. Simple speaking, here we just say what are the operations are there and its input and output but we don't say how to call those operations or what transport they use. We don't say the address where we host this web service.



Steps to create an abstract WSDL:

- Choose WSDL from palate.
- Then inside that WSDL add two messages and a portType.
- Map one message to the input element of the schema we made.

**addRequest (Message)**

Configuration

Message Name:

Description:

Part Table:

| Name       | Type          |
|------------|---------------|
| addRequest | ns:addRequest |

- Map the other message to the output element of the schema we made.

**addResponse (Message)**

Configuration

Message Name:

Description:

Part Table:

| Name        | Type           |
|-------------|----------------|
| addResponse | ns:addResponse |

- Now go inside the portType and add a operation in it.

**/SOAP and Service/Shared Variables/Service/AddWSDL/AddPort**

add

- For the operation, set the input message and output message by clicking on the + button and setting the messages we created in the previous steps.

**add (Operation)**

Configuration

Operation Name:

Description:

Message Table:

| Message Kind | Name | Message         |
|--------------|------|-----------------|
| input        |      | tns:addRequest  |
| output       |      | tns:addResponse |

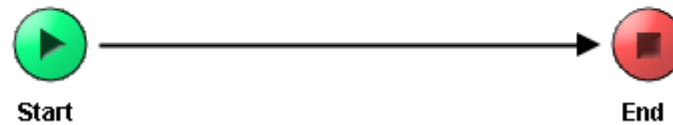
So in an abstract WSDL we have input and output message of the operations and then port type and operations tagged with their I/O messages.

### Creating a web service from the abstract WSDL and Building a concrete WSDL

After creating the abstract WSDL, we have to create web methods (Process Definition) that would be the working process for the operations defined.

Here we have one addOperation so we create a simple process definition with a Start and End.

### /SOAP and Service/Main Process/AddProcess



To the start node, we map the abstract WSDL input message as its output.

**Start (Start)**

Configuration | **Output Editor** | Output | Error Output

Content: WSDL Message

Cardinality: Required

Wsd: http://add.sayon.com

Message: addRequest

For End node, we map the abstract WSDL output message as its input.

**End (End)**

Configuration | **Input Editor** | Input | Error Schemas | Error Output

Content: WSDL Message

Cardinality: Required

Wsd: http://add.sayon.com

Message: addResponse

Now in the end node go to the input tab and map the two inputs of the start node with a plus sign to the output field to perform addition.

**End (End)**

Configuration | **Input** | Error Schemas | Error Output

Process Data:   
 \$globalVariables  
 \$\_processContext  
 \$Start

Activity Input:   
 [End]  
 addResponse  
 addResponse  
 addResponse  
 op  
 \$Start/px2.addRequest/addRequest/ns1:addRequest/ns1:a + \$Start/px2.addRequ...

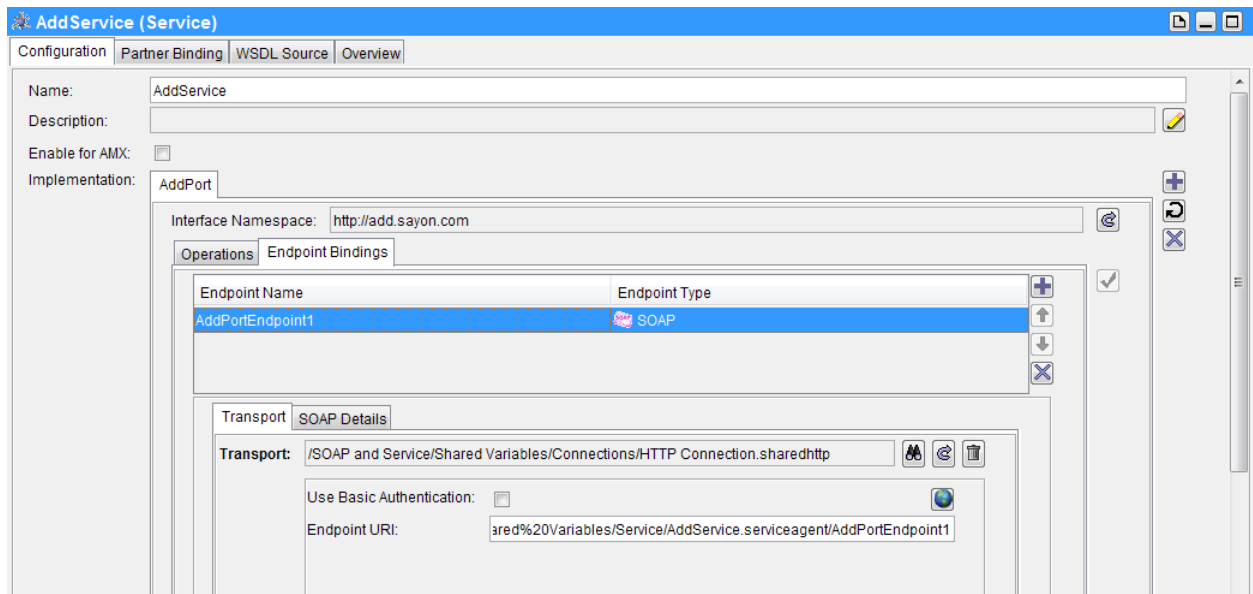
Now our addition process is done. Now we are set to create a service.

Steps to create a service:

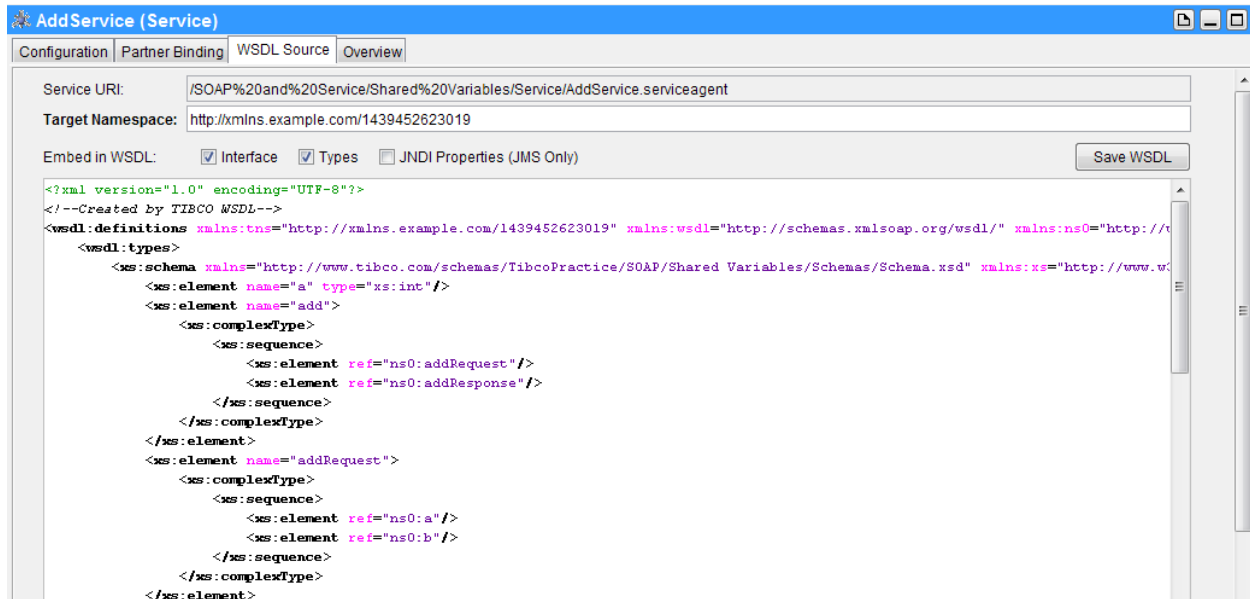
- Select Service from the palate.
- Now in the configuration add the abstract WSDL in the implementation.
- In the operation implementation select the process definition.



- Go to next tab endpoint bindings and add an endpoint.
- For that endpoint give SOAP as the endpoint type.
- Now below in the transport tab select a HTTP Connection and click apply.



Now in WSDL source we can find the full WSDL will get generated automatically and this is our concrete WSDL with the port bindings and transport details.



So we will save it for later use. When we have to call a service this concrete WSDL will be necessary as it provides the complete information about the service and its operation.

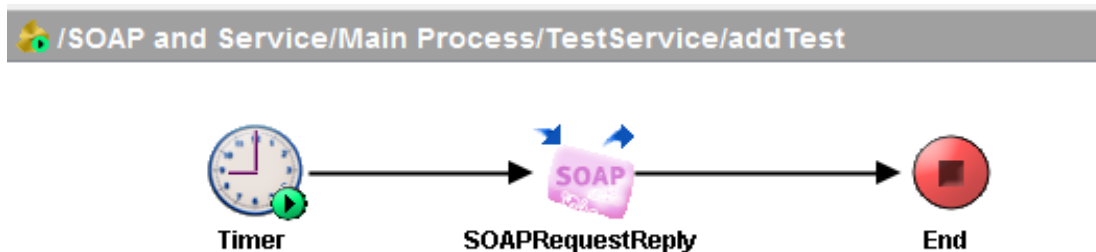
### Using a SOAP Request Reply for calling the service

Our concrete WSDL and service is ready. It's now time to test it.

For this we will create a process and call our web service by a SOAP Request Reply Node.

Palates Required:

- SOAP Request Reply



Steps to follow:

- For the SOAP Request Reply, set the service by locating the concrete WSDL, then select the endpoint and operation to call.

**SOAPRequestReply (SOAP Request Reply)**

Configuration | Transport Details | Advanced | Input | Output | Error Output

Name: SOAPRequestReply

Description:

Service: <http://xmlns.example.com/1439452623019>

AddService

Port: AddPortEndpoint1

Operation: add

SoapAction: /SOAP%2520and%2520Service/Shared%2520Variables/Service/AddService.serviceagent/AddPortEndpoint1/add

Timeout(sec): 0

Timeout Unit: Seconds

Attachment Style: SwA

- In the input tab give the values of the input for the operation.

**SOAPRequestReply (SOAP Request Reply)**

Configuration | Transport Details | Advanced | Input | Output | Error Output

Process Data:   
 \$ = \$globalVariables  
 \$ = \$\_processContext  
 \$ = \$Timer

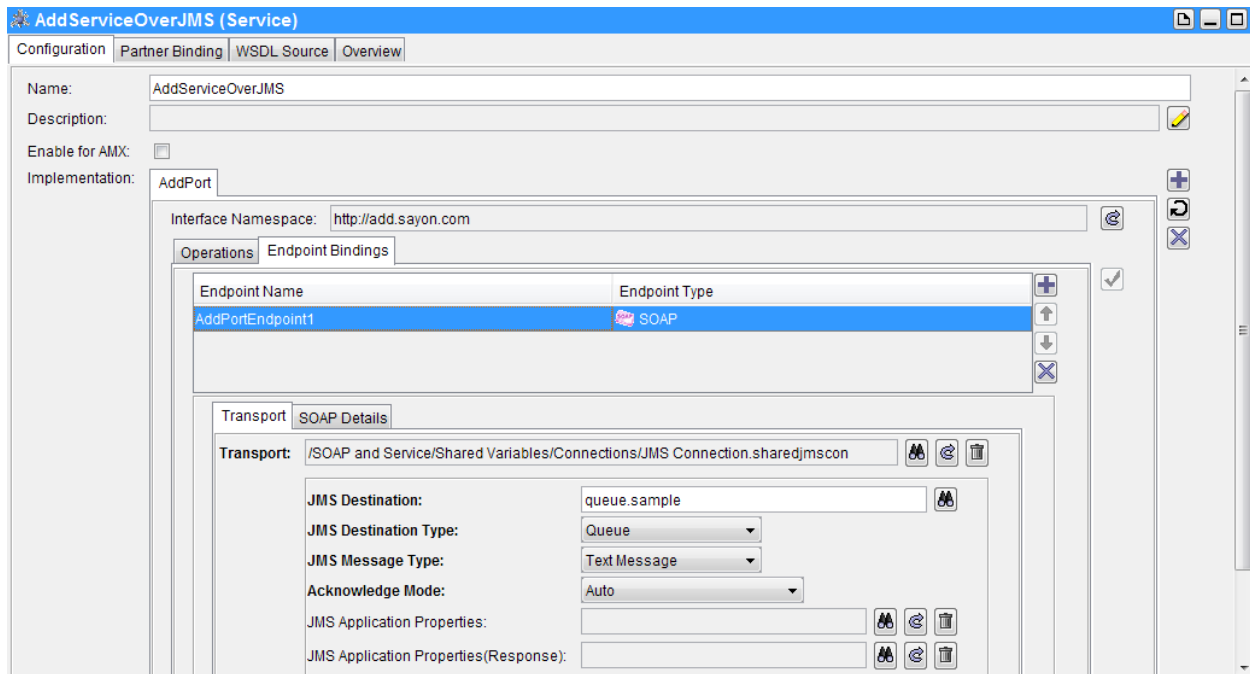
Activity Input:

|                     |    |
|---------------------|----|
| [SOAPRequestReply]  |    |
| inputMessage        |    |
| addRequest          |    |
| a                   | 20 |
| b                   | 20 |
| mimeEnvelopeElement |    |
| _configData         |    |

## SOAP over JMS

To use JMS as our transport we have to give Transport for the endpoint binding as a JMS Connection and then give a destination queue name and message type and acknowledgement mode. Then a JMS Queue will be used as transport and not HTTP.





Rest remains the same as we did for HTTP. In our concrete WSDL generated the transport will be JMS. While calling by a SOAP Request Reply Node we can change the queue and connection factory details if required.

### SOAP Event Source/ Reply and Fault

Earlier we created a service where we have our operation and then tagged a process to that operation.

But for just one operation we needn't have to create a service instead we could have created a SOAP Process which will listen for a particular event and then will give a SOAP response for that event. All we need to do is to create a process.

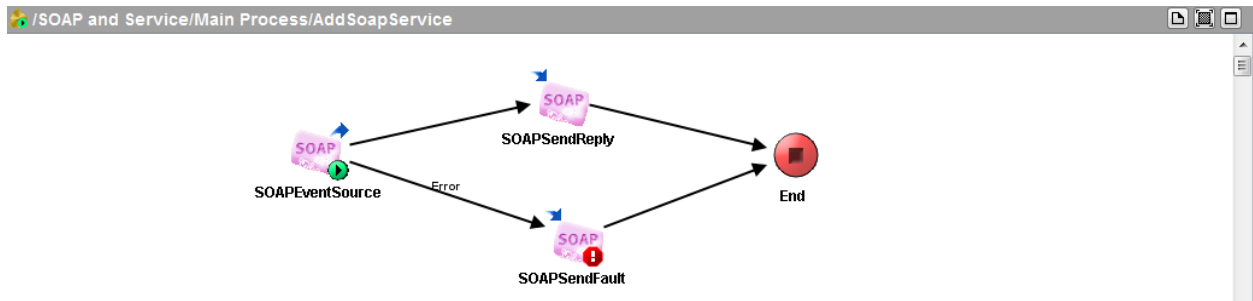
So if there is only one operation we are looking for then we can use the SOAP Event Source node and SOAP Reply and SOAP Fault node. Creating service is always the best option when we want to have multiple operations in that.

Perquisites:

- Abstract WSDL
- HTTP/JMS Connection

Palates required:

- SOAP Event Source
- SOAP Reply
- SOAP Fault



Steps to follow:

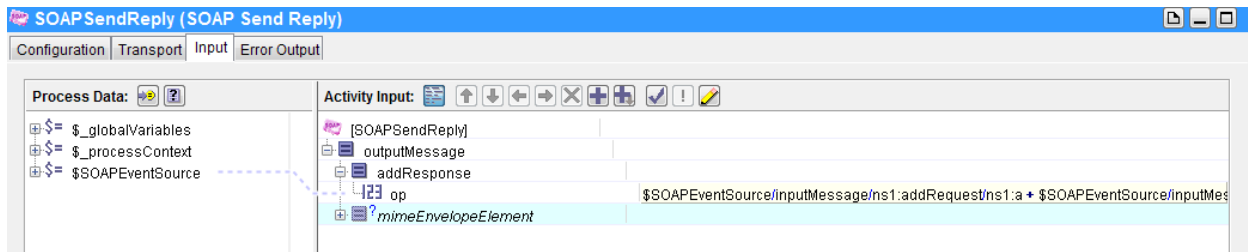
- In SOAP Event Source, set the location of the abstract WSDL, the portType, the operation and the transport. Now, in the WSDL Source a concrete WSDL is auto generated. Copy its contents and save it as a WSDL file. This concrete WSDL will be required when we will call this process.

```

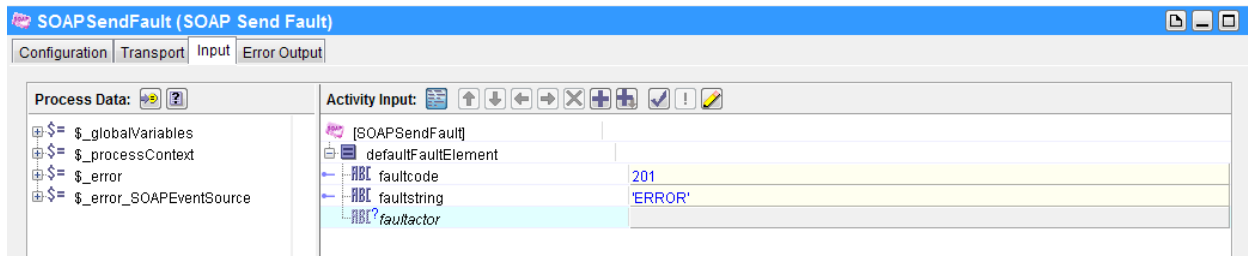
<?xml version="1.0" encoding="UTF-8"?>
<!--Created by TIBCO WSDL-->
<wSDL:definitions xmlns:tns="http://add.sayon.com/addImpl/SOAP_sp_and_sp_Service/Main_sp_Process" xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/" >
  <wSDL:types>
    <xs:schema xmlns="http://www.tibco.com/schemas/TibcoPractice/SOAP/Shared Variables/Schemas/Schema.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema" >
      <xs:element name="a" type="xs:int"/>
      <xs:element name="add">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="tns:addRequest"/>
            <xs:element ref="tns:addResponse"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="addRequest">
        <xs:complexType>
          <xs:sequence>
            <xs:element ref="ns0:a"/>
            <xs:element ref="ns0:b"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="addResponse">
        <xs:complexType>
          <xs:sequence>

```

- Set two transition one success to SOAP Reply and the other one error to SOAP Fault from SOAP Event Source.
- For the SOAP Reply, set reply for the SOAP Event Source and then in the input tab set the output message as the addition of the two inputs.

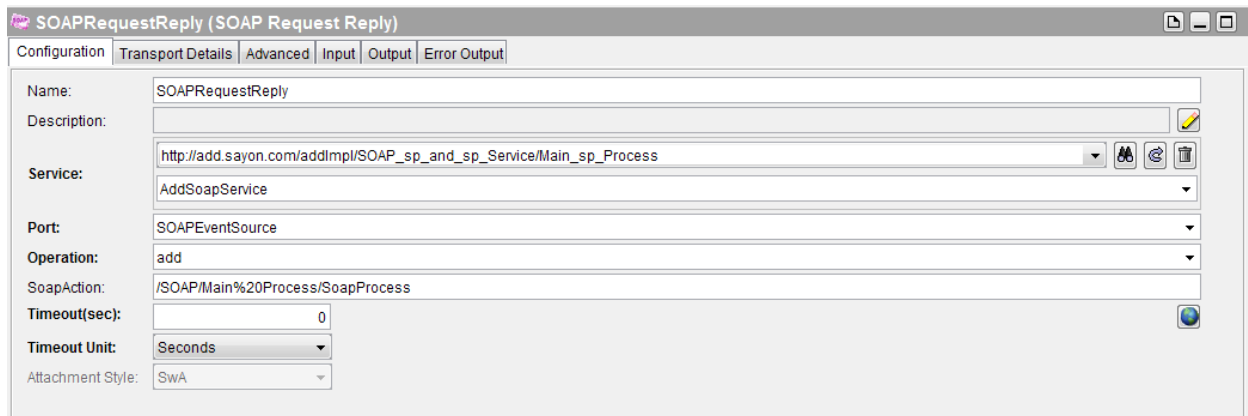


- For the SOAP Fault, set faultCode and faultString.



Invoking the Service:

- To call this process, we will use the same SOAP Request Reply node we used earlier but this time we will refer to the concrete WSDL generated in the SOAP Event Source node.



- Now run both the process together and we will get the output in the output of SOAP Request Reply node.

### SOAP with Attachments (SwA)

SwA means we will send a file as an attachment along with a SOAP request or response.

We can do this in TIBCO in two ways:

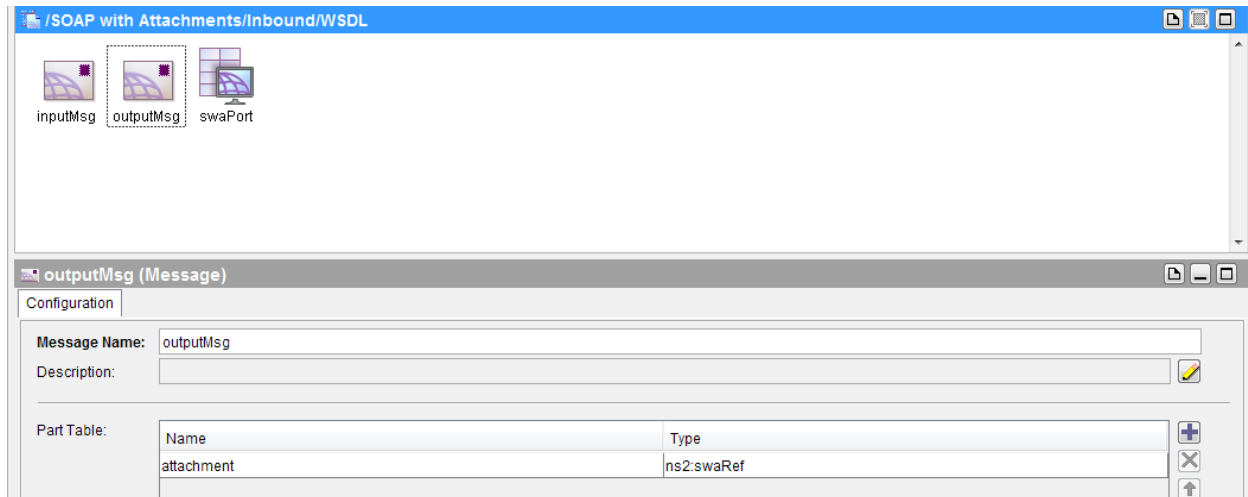
#### Inbound

In this method, the attachment is mentioned in the abstract WSDL as the message with type swaRef (SOAP with Attachment Reference). It's a special type of reference for SOAP Attachment.

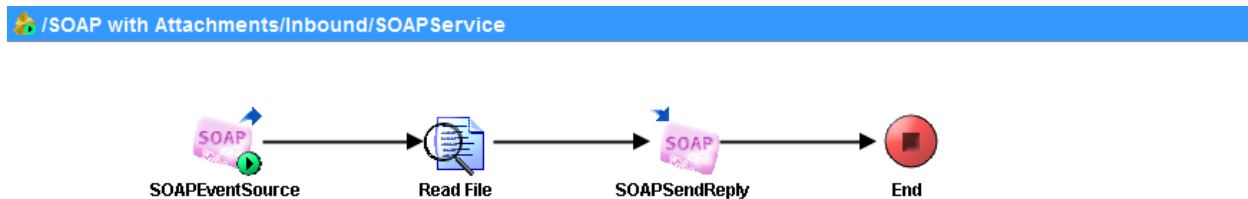
Now when we create a SOAP service (as in the previous example) and wanted attachment , we will just define a part in the input or output message of a operation as swaRef.

Working:

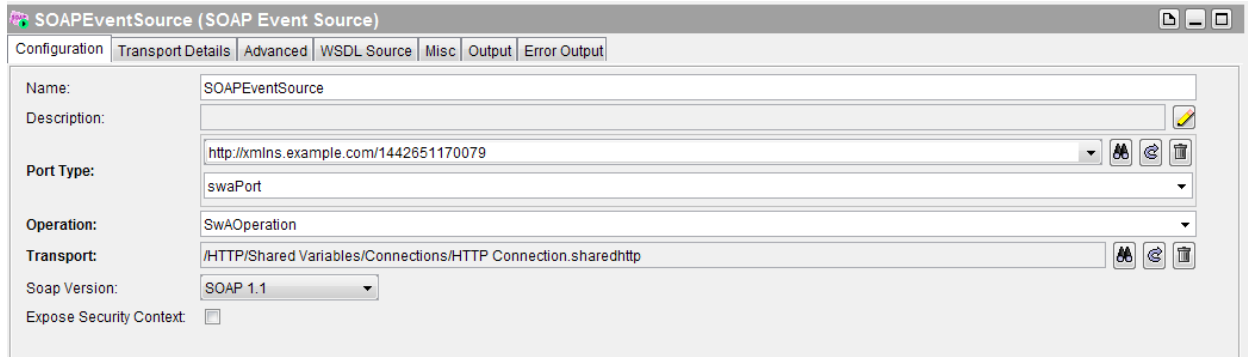
- Two processes a Client and a Server.
- An abstract WSDL with a operation and its output message is a swaRef attachment.



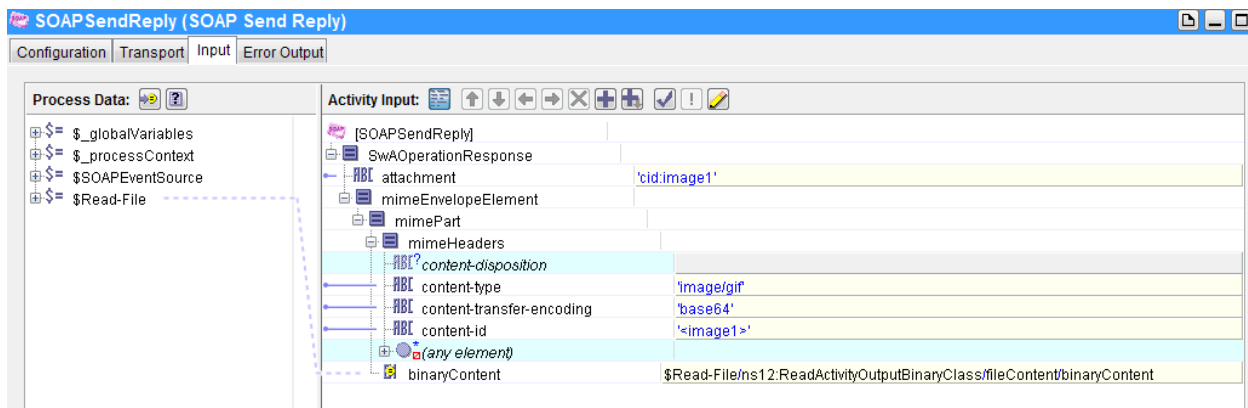
- Now in the server process, we need a SOAP Event Source, Read File, SOAP Reply nodes.



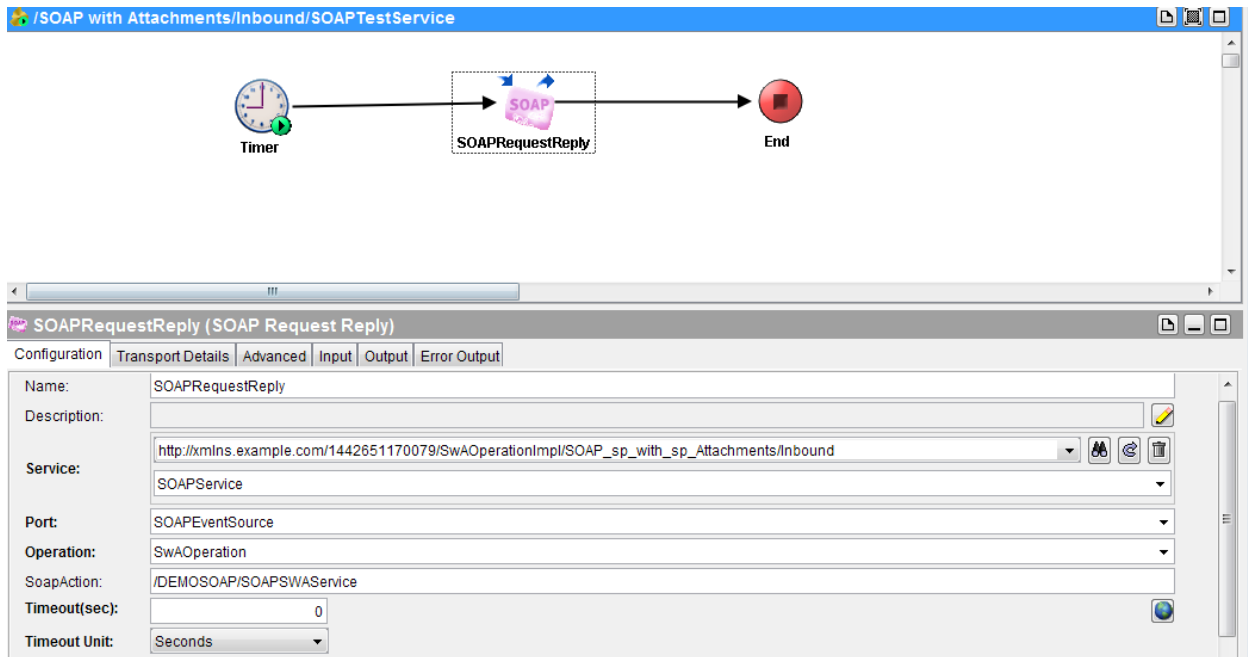
- In the SOAP Event Source, set the abstract WSDL and select the operation and port type in the configuration. Now in the advanced tab, add your input and output attachments its type and which message part should it consider as attachment. A concrete WSDL is automatically generated. Save it.



- Now in the Read File node, read a file and then make a transition to SOAP Send Reply.
- In SOAP Send Reply, reply for the SOAP Event Source. In the input tab, map the fileContent of the Read File node to the mimePart of the SOAP Reply and in the mimeHeaders give the content type and content id (cid) like '<image1>' and in the attachment field write cid:image1 so as to refer to the mime attachment.



- Now create a client process with a SOAP Request Reply node and configure it with the concrete WSDL and then give the inputs to the operation. In the output, we will get the SOAP Response with attachments after we run the processes.

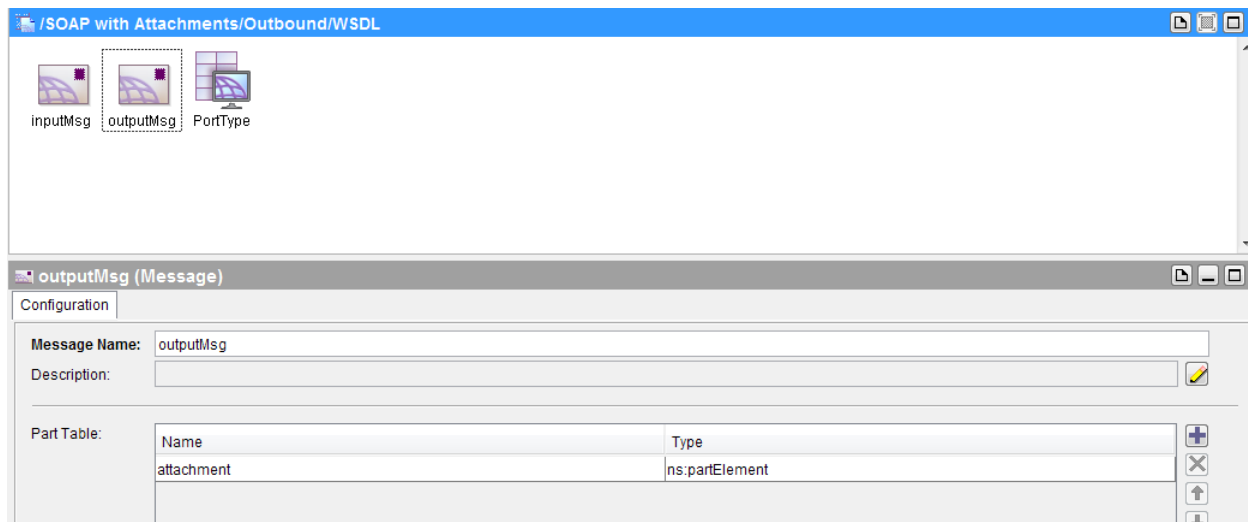


### Outbound

In this method, we don't need to state anything about the attachments in the WSDL but the partElement (float type) reference for our input and output. For attachments, we simply attach it to the mimePart of the request and reply without any cid reference.

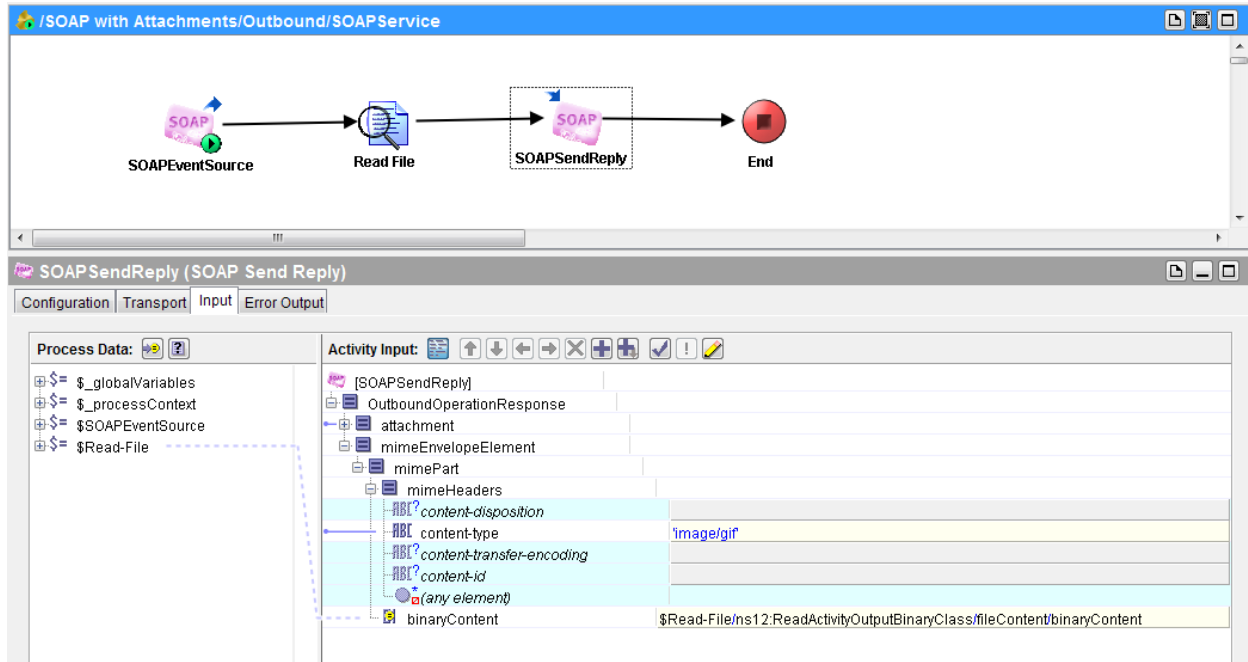
Working:

- The processes remain more or like same. The only changes are that no swaRef needed. Just refer to the WSDL itself for both input and output.



- In the SOAP Event Source, no advanced settings for attachments to done and no cid reference in the SOAP Send Reply node.
- The Client Process remains the same.

- Attachments are attached to the mimePart for both request and response.



Look at the example for better understanding.

## Conclusion

All the above examples in this tutorial is directed to help as a step by step know how on how to use the TIBCO palates and understand the transitions and more importantly getting used to the features or configurations provided by TIBCO for each palate. With increased complexity or complex business logic, the flow will become a lot complicated but the basic for each node will be the same as described.

Hope this tutorial is beneficial.

All the examples mentioned in this tutorial are available in zip format. Just import it for reference.