Pub-Sub Models

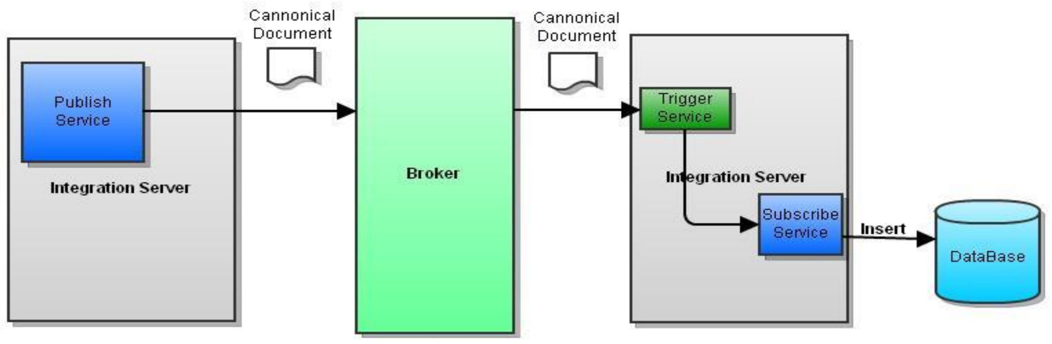
Introduction:

The publish-and-subscribe model is a specific type of message-based solution in which messages are exchanged anonymously through a message broker.
Applications that produce information that needs to be shared will make this information available in specific types of recognizable documents that they publish to the message broker.
Applications that require information subscribe to the document types they need. At run time, the message broker receives documents from publishers and then distributes the documents to subscribers. The subscribing application processes or performs work using the document and may or may not send a response to the publishing application.

1. Publish

Publishes a document locally or to a configured Broker. Any clients (triggers) with subscriptions to documents of this type will receive the document.

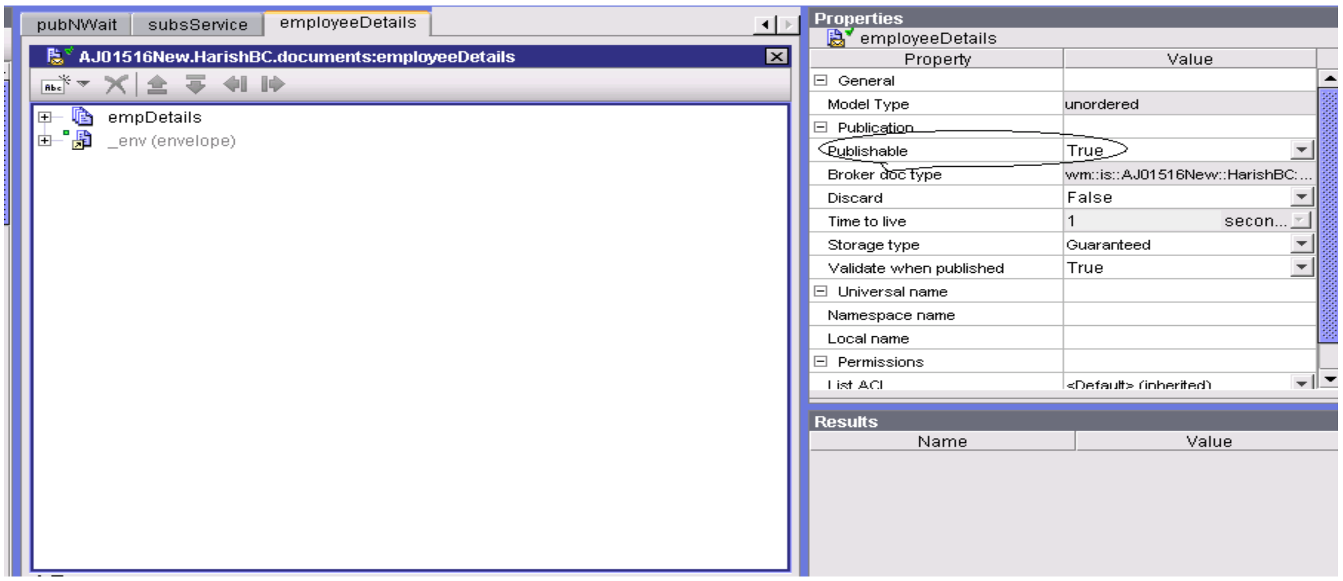
Design Diagram



Following are the steps to implement the Publish model:

Creating a Publishable Document:

Create a IS document and make it publishable by changing the property "Publishable" to "true".



AJ01516New.HarishBC.documents:employeeDetails

empDetails
_env (envelope)

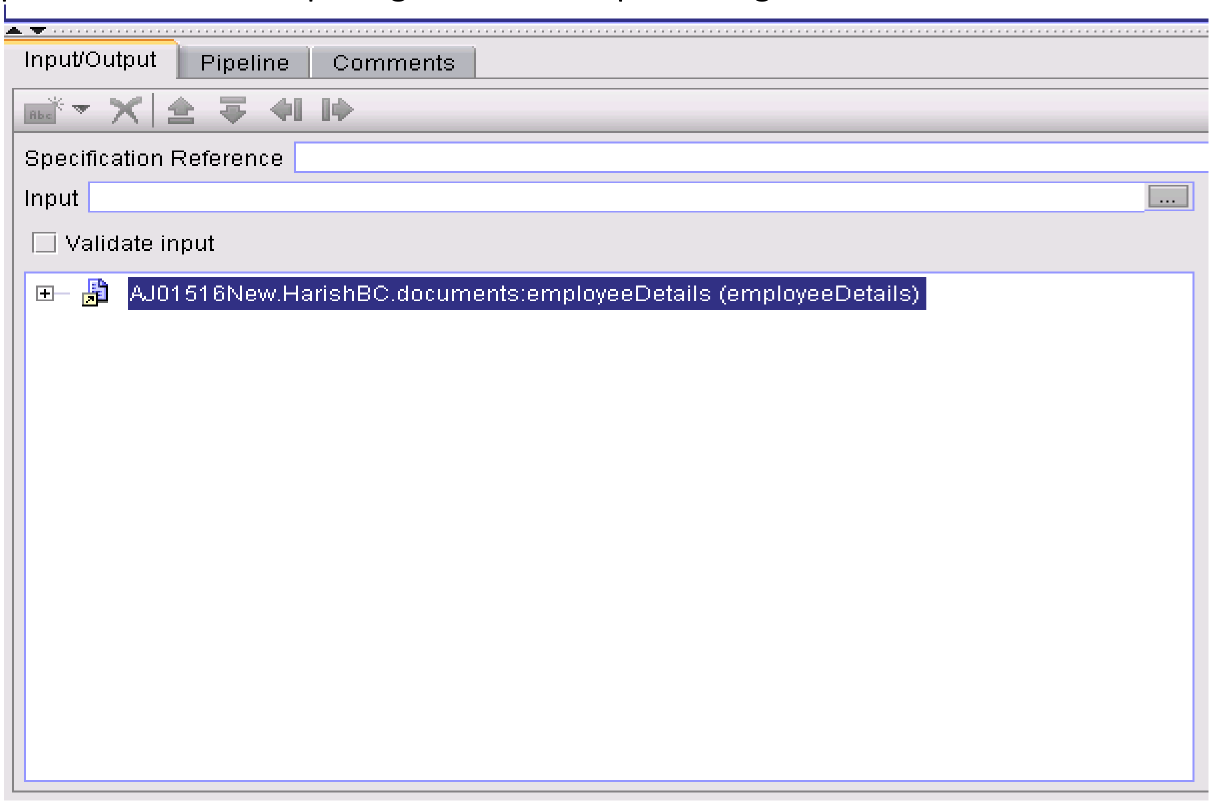
employeeDetails

Property	Value
General	
Model Type	unordered
Publication	
Publishable	True
Broker doc type	wm::is::AJ01516New::HarishBC:...
Discard	False
Time to live	1 secon...
Storage type	Guaranteed
Validate when published	True
Universal name	
Namespace name	
Local name	
Permissions	
List ACL	<Default> (inherited)

Name	Value

On Publisher Side:

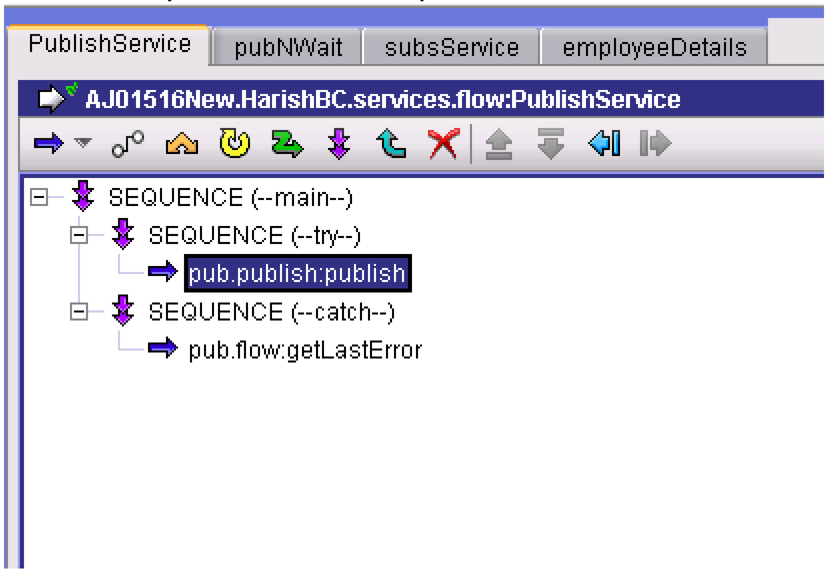
Step1: Create a flow service with declaring a IS document reference which is publishable in the input signature of the publishing service.



Specification Reference
Input
Validate input

AJ01516New.HarishBC.documents:employeeDetails (employeeDetails)

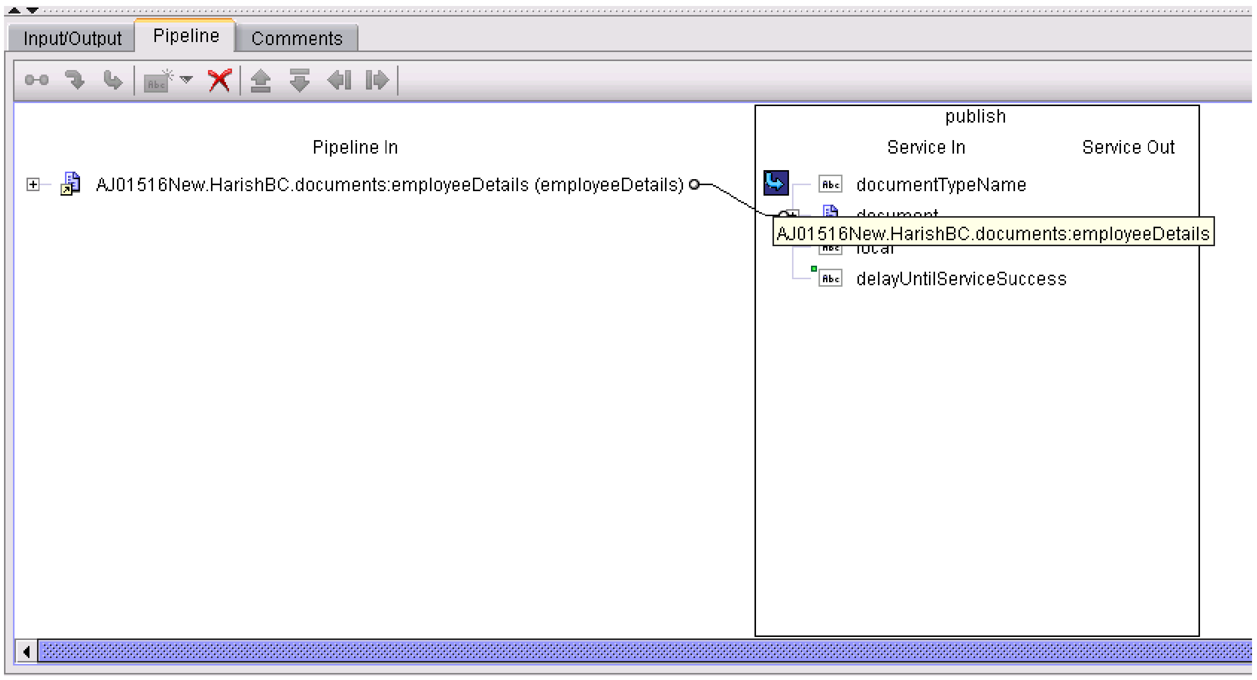
Step2: Invoke **pub.publish:publish** to publish the document. This service takes the document you created and publishes it.



AJ01516New.HarishBC.services.flow:PublishService

```
SEQUENCE (--main--)
    SEQUENCE (--try--)
        pub.publish:publish
    SEQUENCE (--catch--)
        pub.flow:getLastError
```

Step3: Map the publishable document to the **document** which is defined in the Input signature and specify the fully qualified document name in the **documentTypeName** options.



AJ01516New.HarishBC.documents:employeeDetails (employeeDetails)

documentTypeName

document

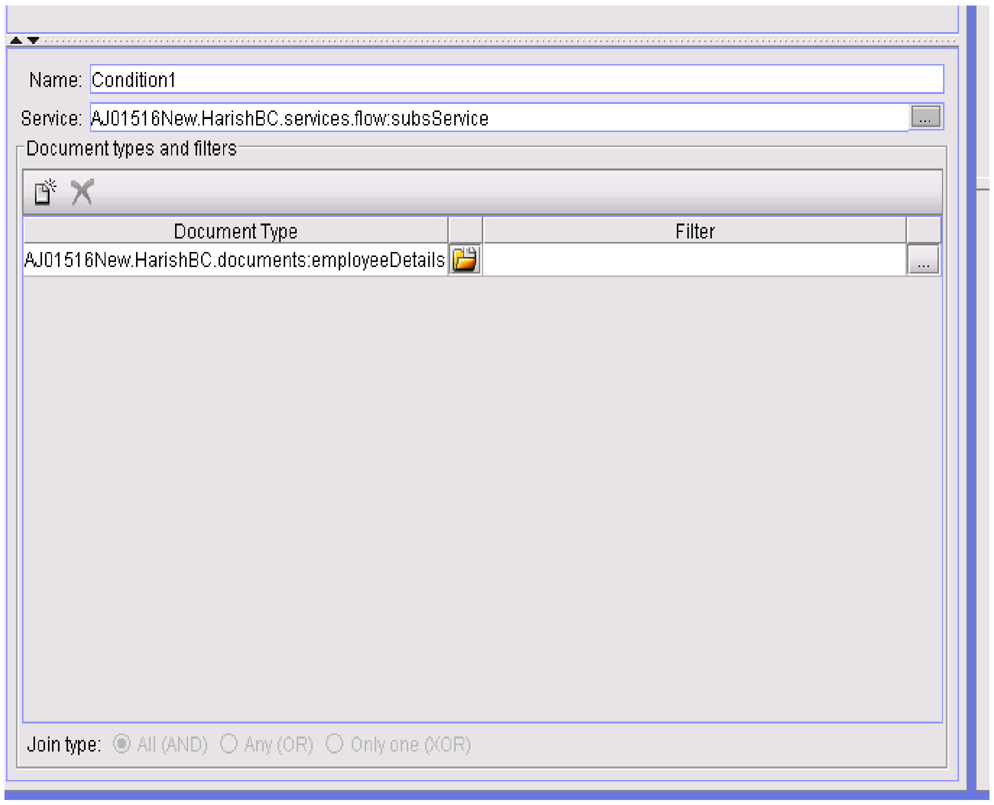
local

delayUntilServiceSuccess

AJ01516New.HarishBC.documents:employeeDetails

Creating a trigger:

- Create a new Broker/Local trigger which subscribes to the published document. Select the service that needs to be invoked when the published document matches with the defined document type in the trigger.



Name: Condition1

Service: AJ01516New.HarishBC.services.flow:subsService

Document types and filters

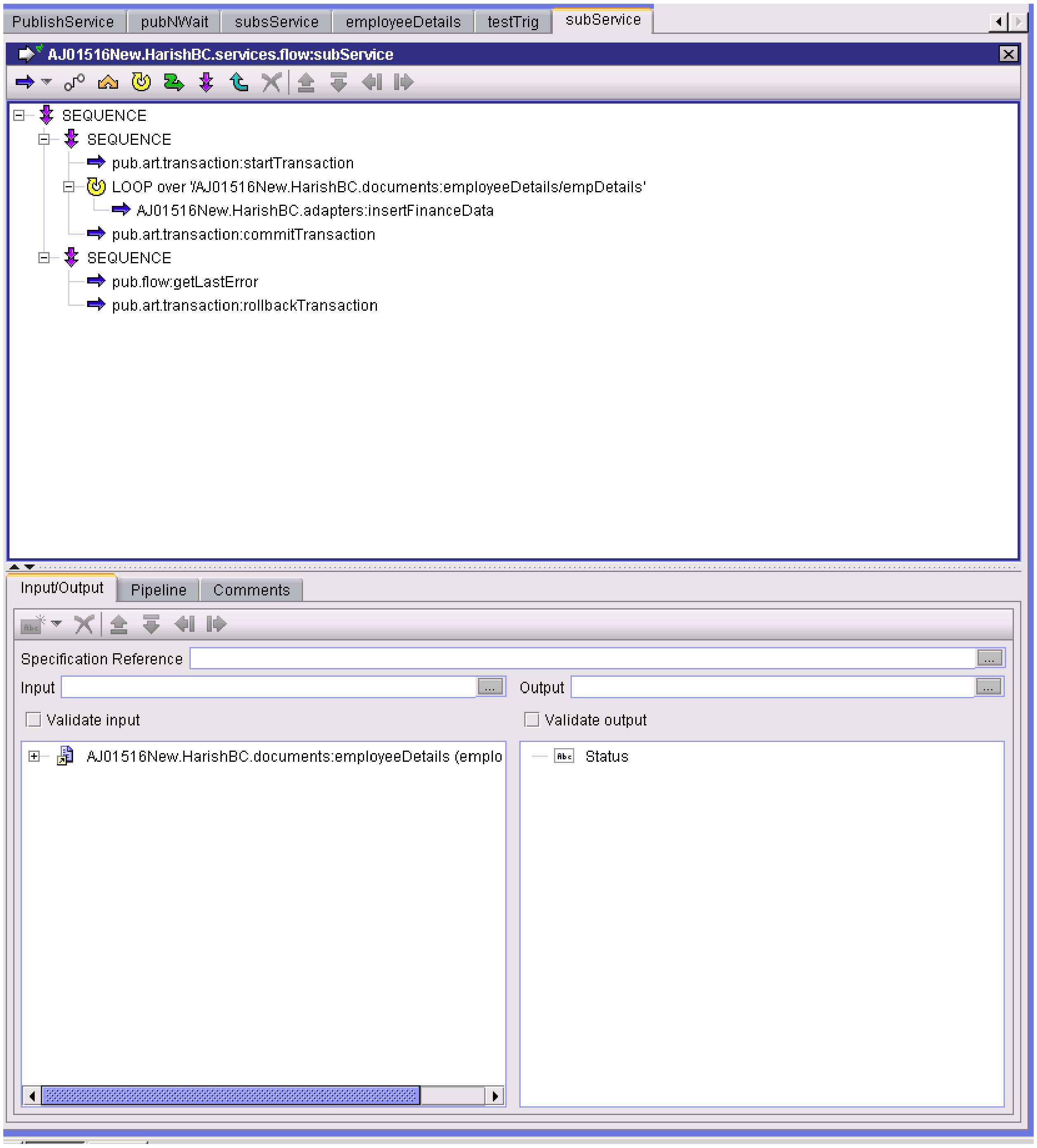
Document Type	Filter
AJ01516New.HarishBC.documents:employeeDetails	

Join type: [✓ All (AND)] Any (OR) Only one (XOR)

On the Subscriber Side:

Step1: Create a subscriber flow service. Define the publishable document in the input tab of the flow service.

Step2: Create a insert JDBC adapter service which will insert the published records into the DB.



AJ01516New.HarishBC.services.flow:subService

```
SEQUENCE
    SEQUENCE
        pub.art.transaction:startTransaction
        LOOP over '/AJ01516New.HarishBC.documents:employeeDetails/empDetails'
            AJ01516New.HarishBC.adapters:insertFinanceData
        pub.art.transaction:commitTransaction
    SEQUENCE
        pub.flow:getLastError
        pub.art.transaction:rollbackTransaction
```

Specification Reference

Input

Output

Validate input

Validate output

AJ01516New.HarishBC.documents:employeeDetails (emplo

Status

Publishing a Document and Waiting for a Reply

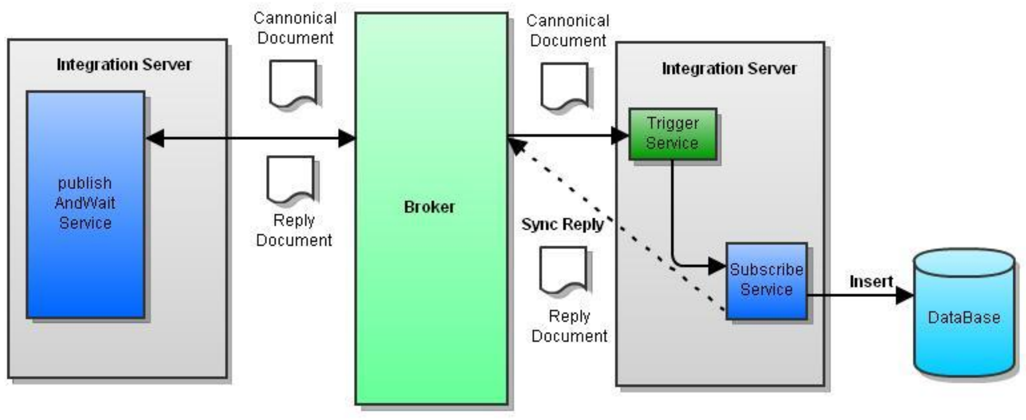
A service can request information from other resources in the webMethods system by publishing a document that contains a query for the information. Services that publish a request document, wait for and then process a reply document follow the *request/reply* model. The request/reply model is a variation of the publish and subscribe model. In the request/reply model, a publishing client broadcasts a request for information. Subscribers retrieve the broadcast document, process it, and send a reply document that contains the requested information to the publisher.

A service can implement a synchronous or asynchronous request/reply.

- In a **synchronous** request/reply, the publishing service stops executing while it waits for a response to a published request. The publishing service resumes execution when a reply document is received or the specified waiting time elapses.
- In an **asynchronous** request/reply, the publishing service continues to execute after publishing the request document. The publishing service must invoke another service to wait for and retrieve the reply document.

Following are the steps to implement the Publish And Wait Synchronously:

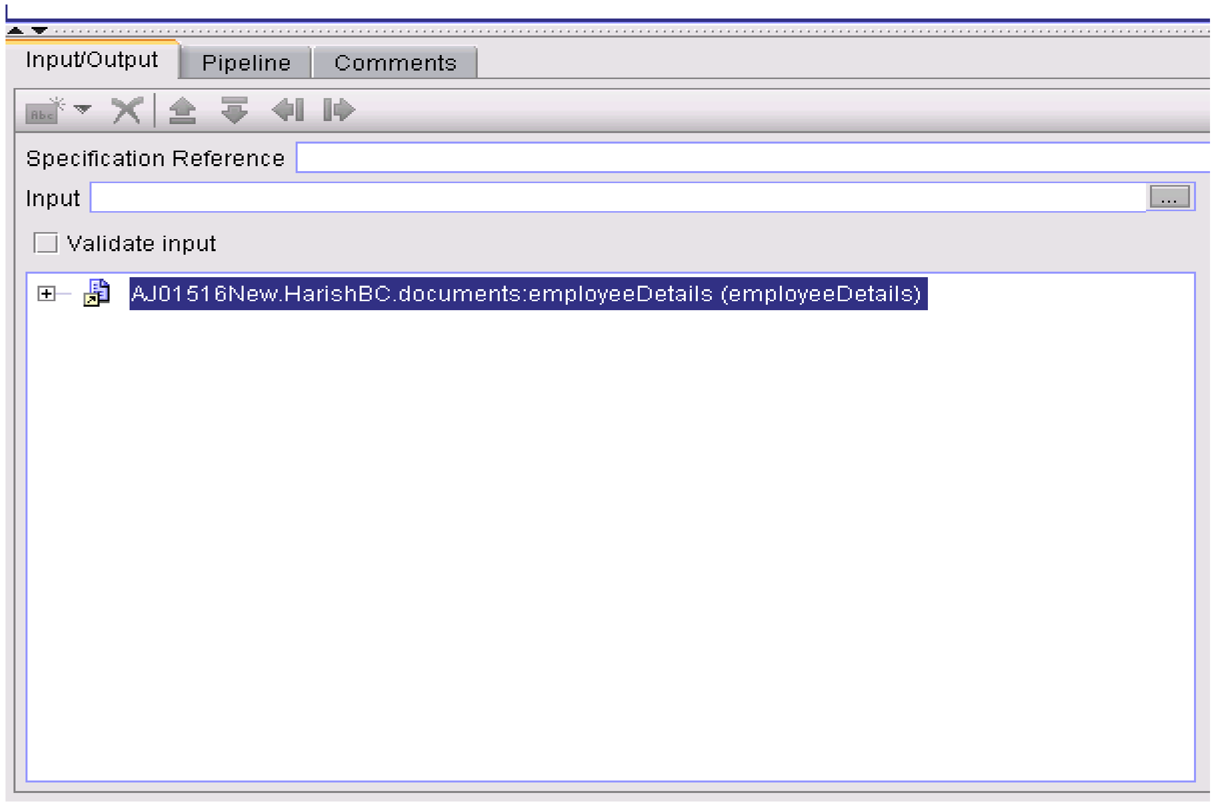
Design Diagram



On the Publisher Side:

Step1: Create a flow service with declaring a IS document reference to the publishable document type that you want to publish.
You can accomplish this by:

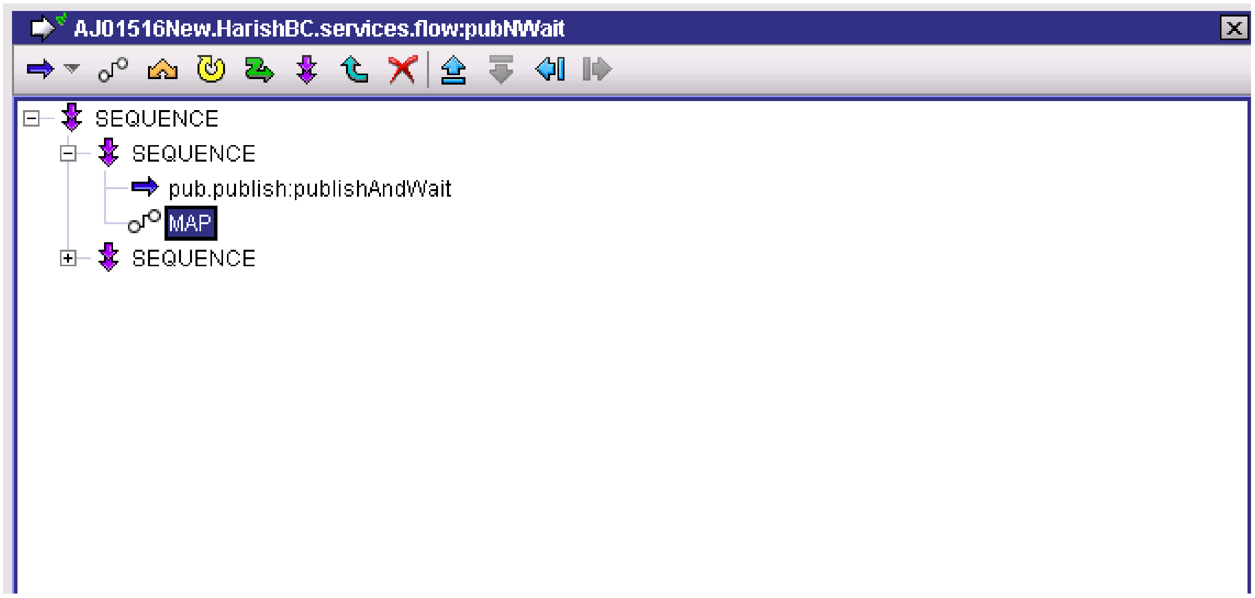
- Declaring a document reference in the input signature of the publishing service.
  —OR—
- Inserting a MAP step in the publishing service and adding the document reference to **Pipeline Out**. You must link or assign a value to the document reference *immediately*. If you do not, Developer automatically clears the document reference the next time it refreshes the **Pipeline** tab.



Specification Reference
Input
Validate input

AJ01516New.HarishBC.documents:employeeDetails (employeeDetails)

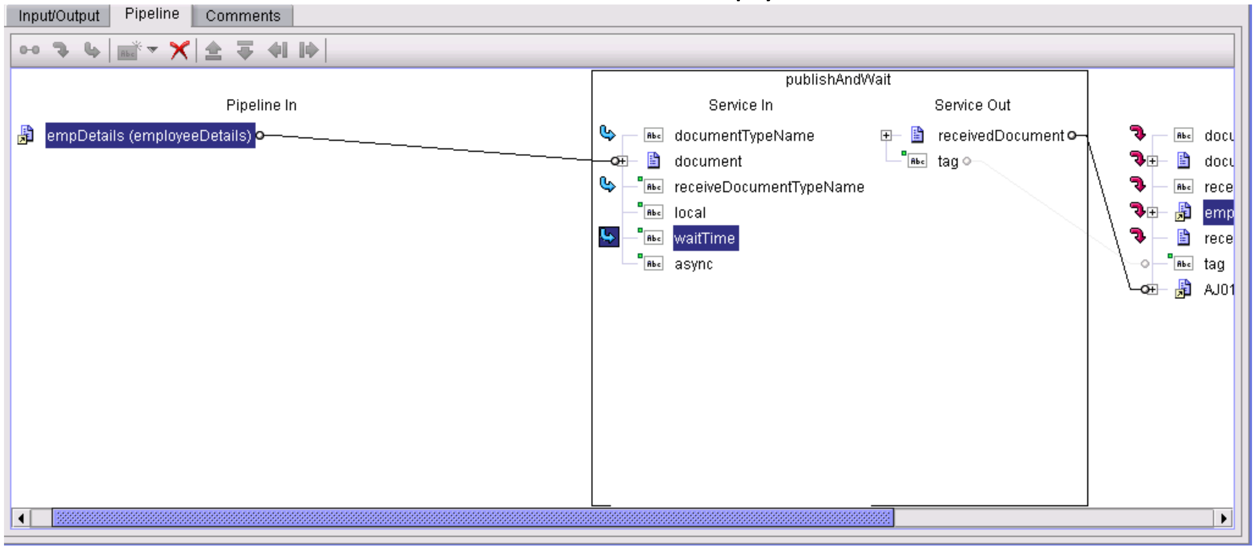
Step2: Invoke **pub.publish:publishAndWait** to publish the document. This service takes the document you defined and publishes it.


AJ01516New.HarishBC.services.flow:pubNWait

```
SEQUENCE
  SEQUENCE
    pub.publish:publishAndWait
    MAP
  SEQUENCE
```

Step3: Map the publishable document to the **document** which is defined in the Input signature and specify the fully qualified document name in the **documentTypeName** options.

- If you define the **waitTime**, then the publishing service will wait till the specified time interval for the reply document. If you do not specify it then the service will wait until it receives the reply document.



Pipeline In: empDetails (employeeDetails)

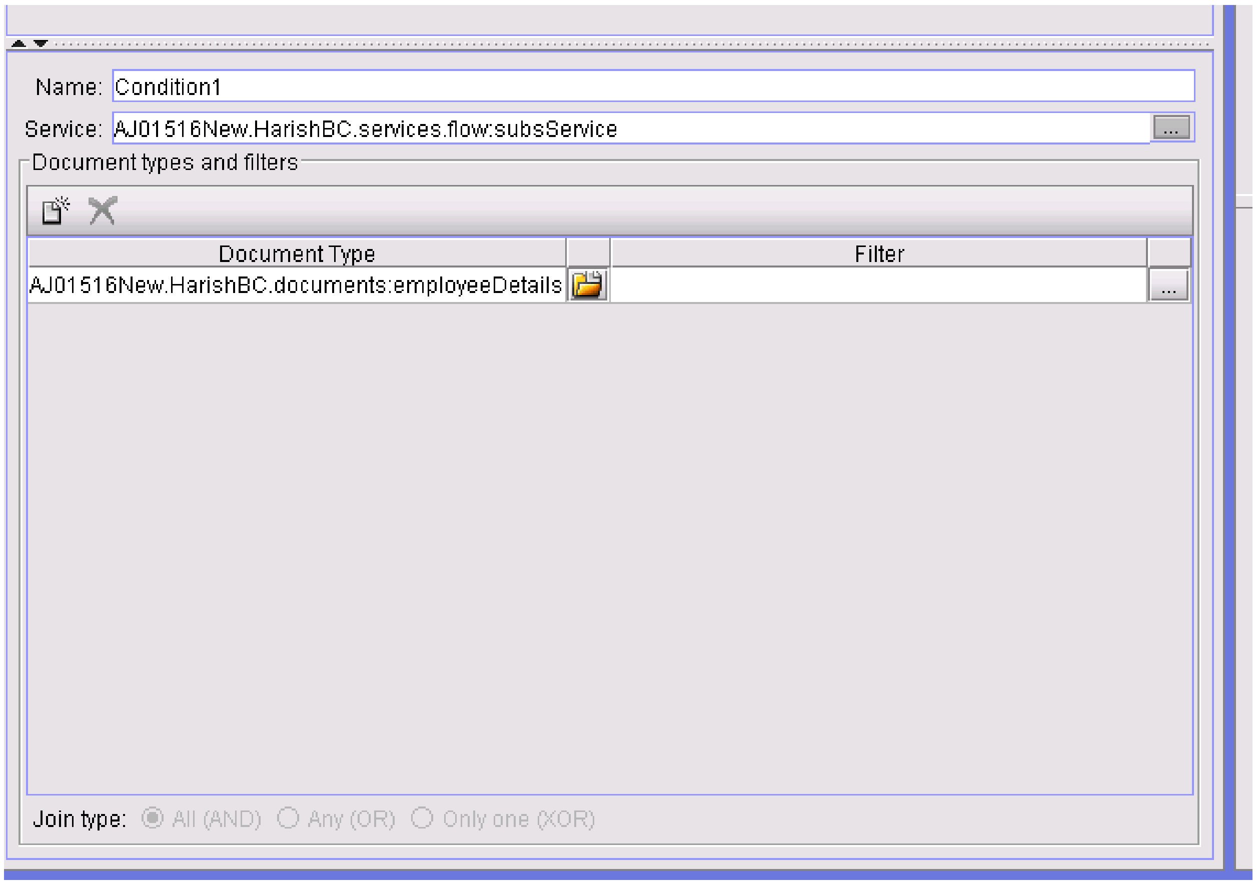
Service In: documentTypeName, document, receiveDocumentTypeName, local, waitTime, async

Service Out: receivedDocument, tag

docu, docu, rece, emp, rece, tag, AJ01

Creating a trigger:

- Create a new Broker/Local trigger which subscribes to the published document. Select the service that needs to be invoked when the published document matches with the defined document type in the trigger.



Name: Condition1

Service: AJ01516New.HarishBC.services.flow:subsService

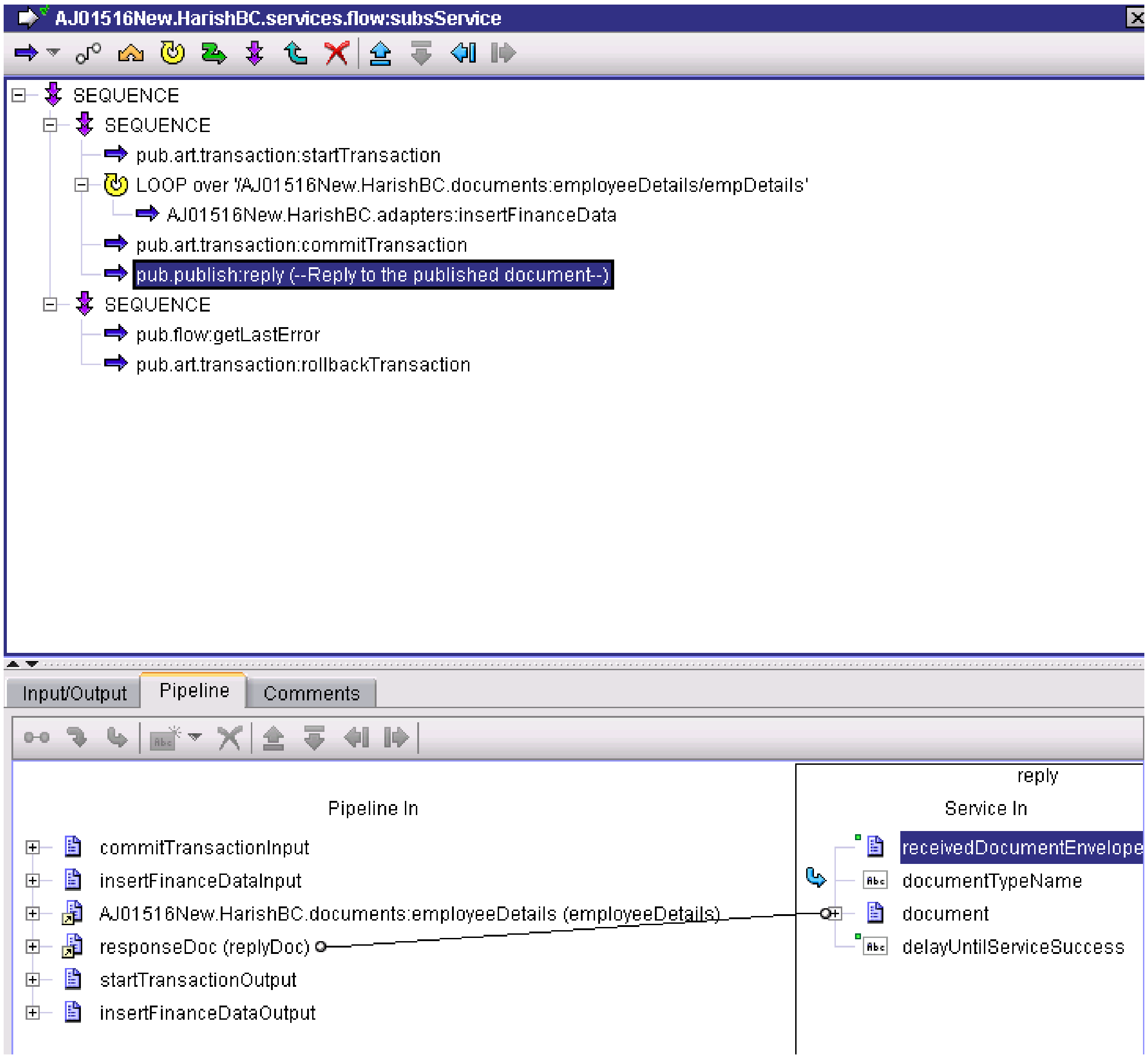
Document types and filters

Document Type		Filter
AJ01516New.HarishBC.documents:employeeDetails		

Join type: [✓ All (AND)] Any (OR) Only one (XOR)

On the Subscriber Side:

Step1: Create a subscriber flow service. Define the publishable document in the input tab of the flow service.
Step2: Create a insert JDBC adapter service which will insert the published records into the DB. Once it is successfully inserted then send the reply back to the publishing service by invoking the **pub.publish:reply** service.

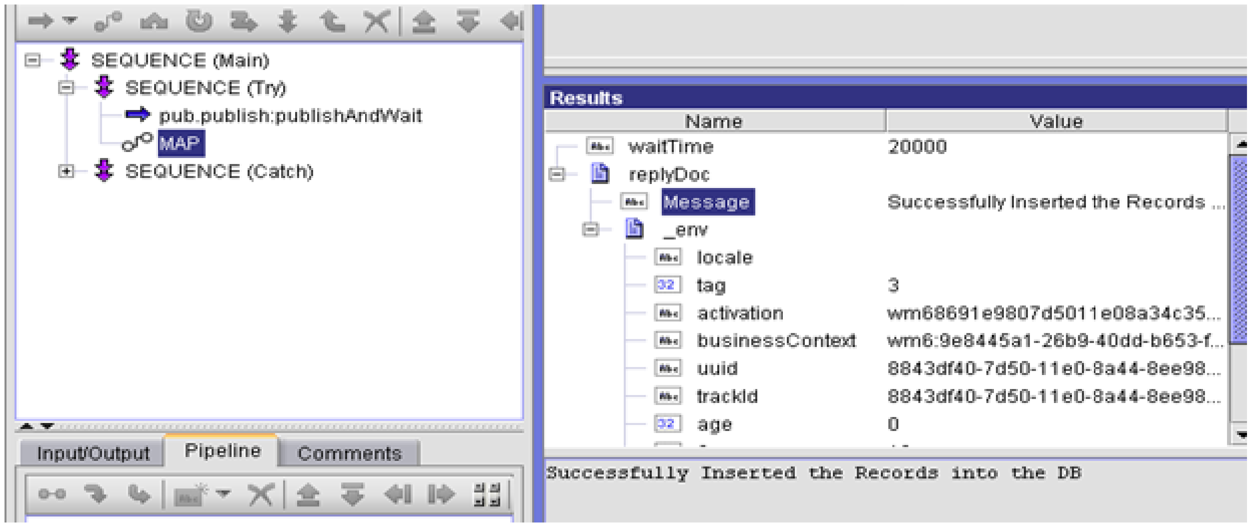


AJ01516New.HarishBC.services.flow:subsService

```
SEQUENCE
  SEQUENCE
    pub.art.transaction:startTransaction
    LOOP over '/AJ01516New.HarishBC.documents:employeeDetails/empDetails'
      AJ01516New.HarishBC.adapters:insertFinanceData
    pub.art.transaction:commitTransaction
    pub.publish:reply (--Reply to the published document--)
  SEQUENCE
    pub.flow:getLastError
    pub.art.transaction:rollbackTransaction
```

Pipeline In:
- commitTransactionInput
- insertFinanceDataInput
- AJ01516New.HarishBC.documents:employeeDetails (employeeDetails)
- responseDoc (replyDoc)
- startTransactionOutput
- insertFinanceDataOutput

reply — Service In:
- receivedDocumentEnvelope
- documentTypeName
- document
- delayUntilServiceSuccess



```
SEQUENCE (Main)
  SEQUENCE (Try)
    pub.publish:publishAndWait
    MAP
  SEQUENCE (Catch)
```

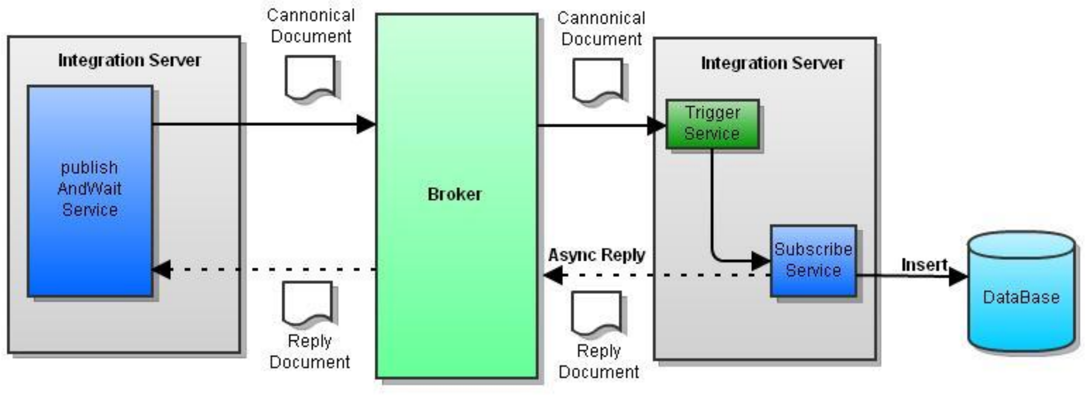
Results

Name	Value
waitTime	20000
replyDoc	
Message	Successfully Inserted the Records ...
_env	
locale	
tag	3
activation	wm68691e9807d5011e08a34c35...
businessContext	wm6:9e8445a1-26b9-40dd-b653-f...
uuid	8843df40-7d50-11e0-8a44-8ee98...
trackId	8843df40-7d50-11e0-8a44-8ee98...
age	0

```
Successfully Inserted the Records into the DB
```

Following are the steps to implement the Publish And Wait Asynchronously:

Design Diagram:



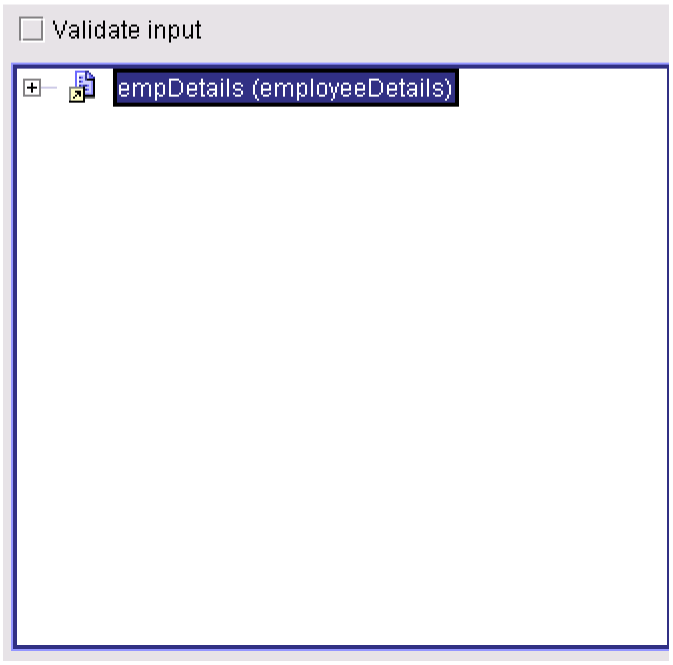
To implement Publish and Wait in asynchronous way, we need to use **pub.publish:waitForReply** service. This service retrieves the reply document for a specific request when ever required in the flow service.

On the Publisher Side:

Step1: Create a flow service with declaring a IS document reference to the publishable document type that you want to publish.
You can accomplish this by:

- Declaring a document reference in the input signature of the publishing service.
  —OR—

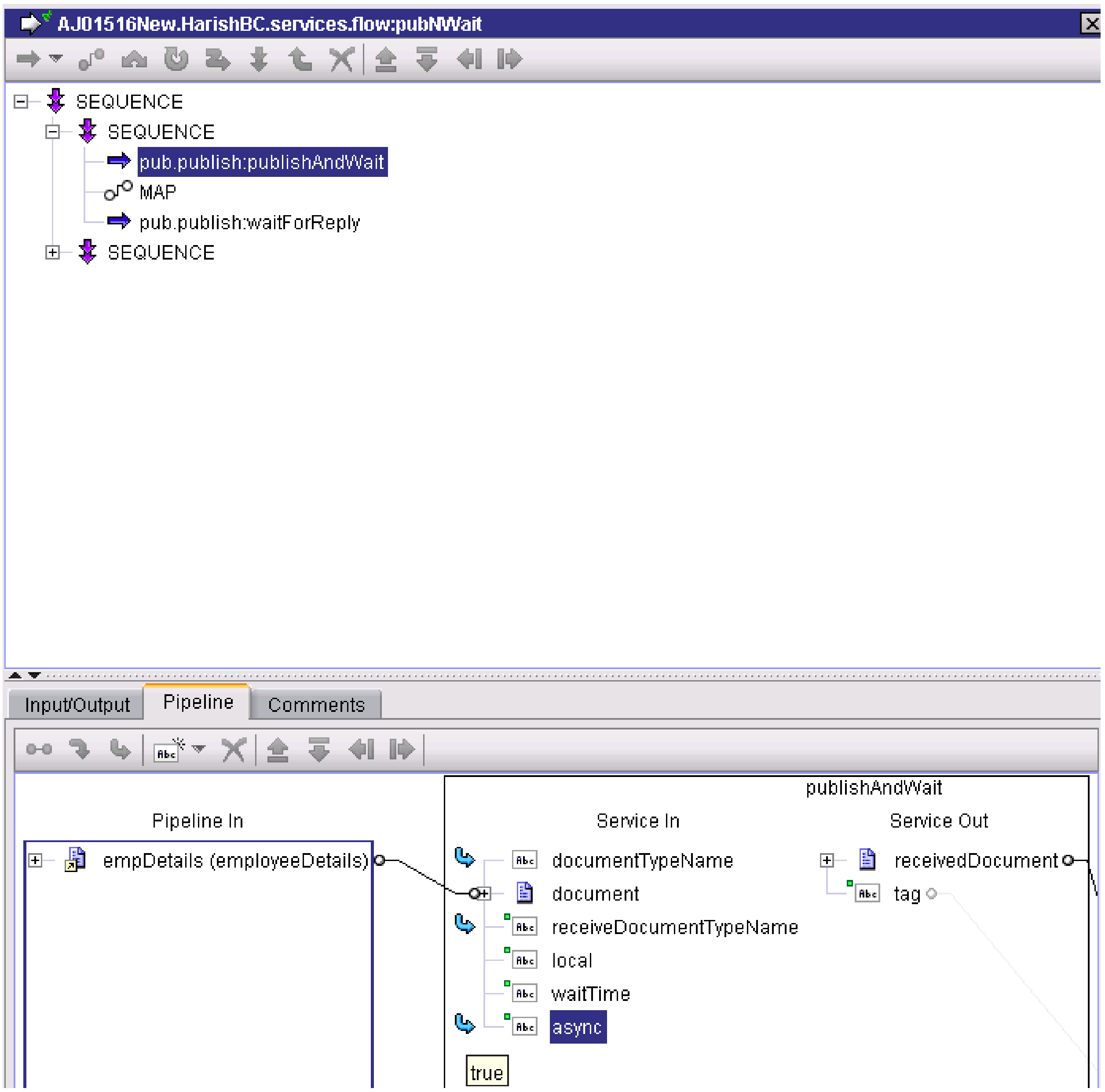
Inserting a MAP step in the publishing service and adding the document reference to **Pipeline Out**. You must link or assign a value to the document reference *immediately*. If you do not, Developer automatically clears the document reference the next time it refreshes the **Pipeline** tab.


Validate input

empDetails (employeeDetails)

Step2: Invoke **pub.publish:publishAndWait** to publish the document. This service takes the document you defined and publishes it.

Step3: Map the publishable document to the **document** which is defined in the Input signature and specify the fully qualified document name in the **documentTypeName** options.

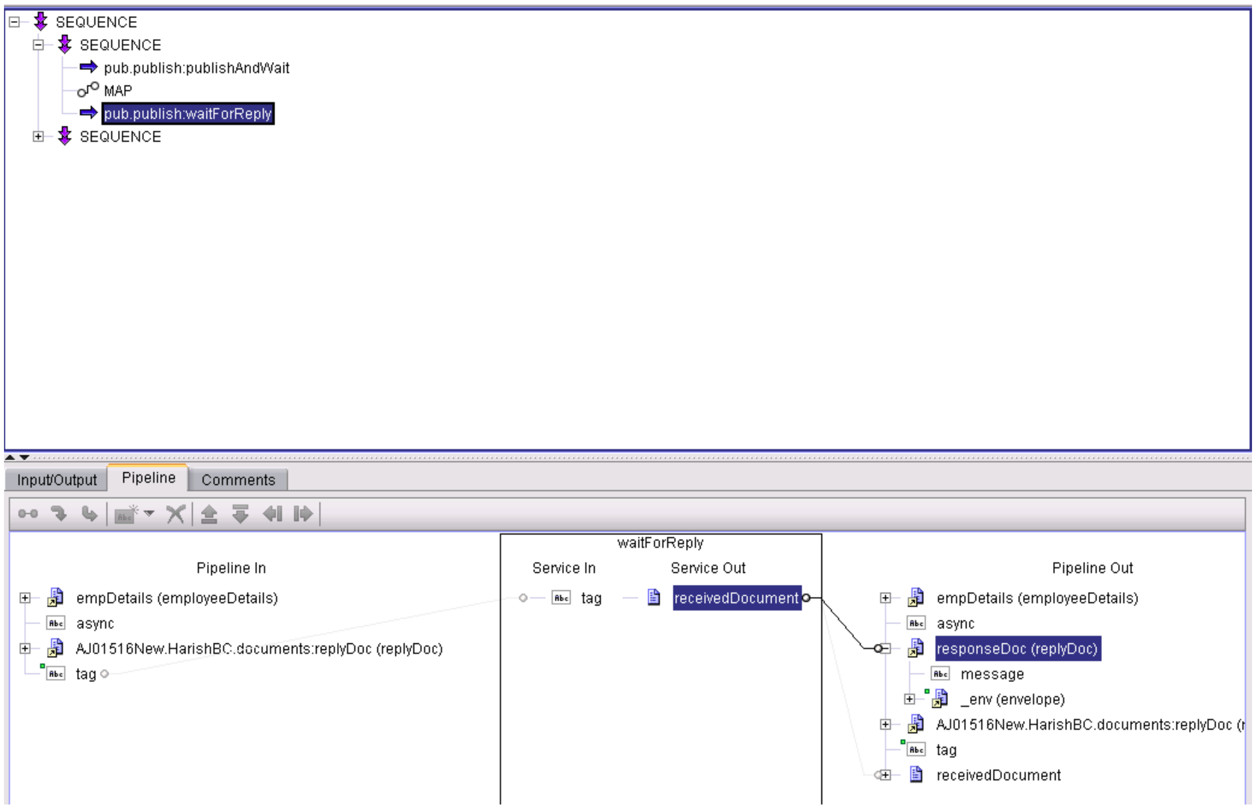


Step4: Map the tag value to a separate variable in the pipeline to perform publish and wait in an asynchronous manner.

- The **tag** field contains a unique identifier that the Integration Server uses to match the request document with a reply document.

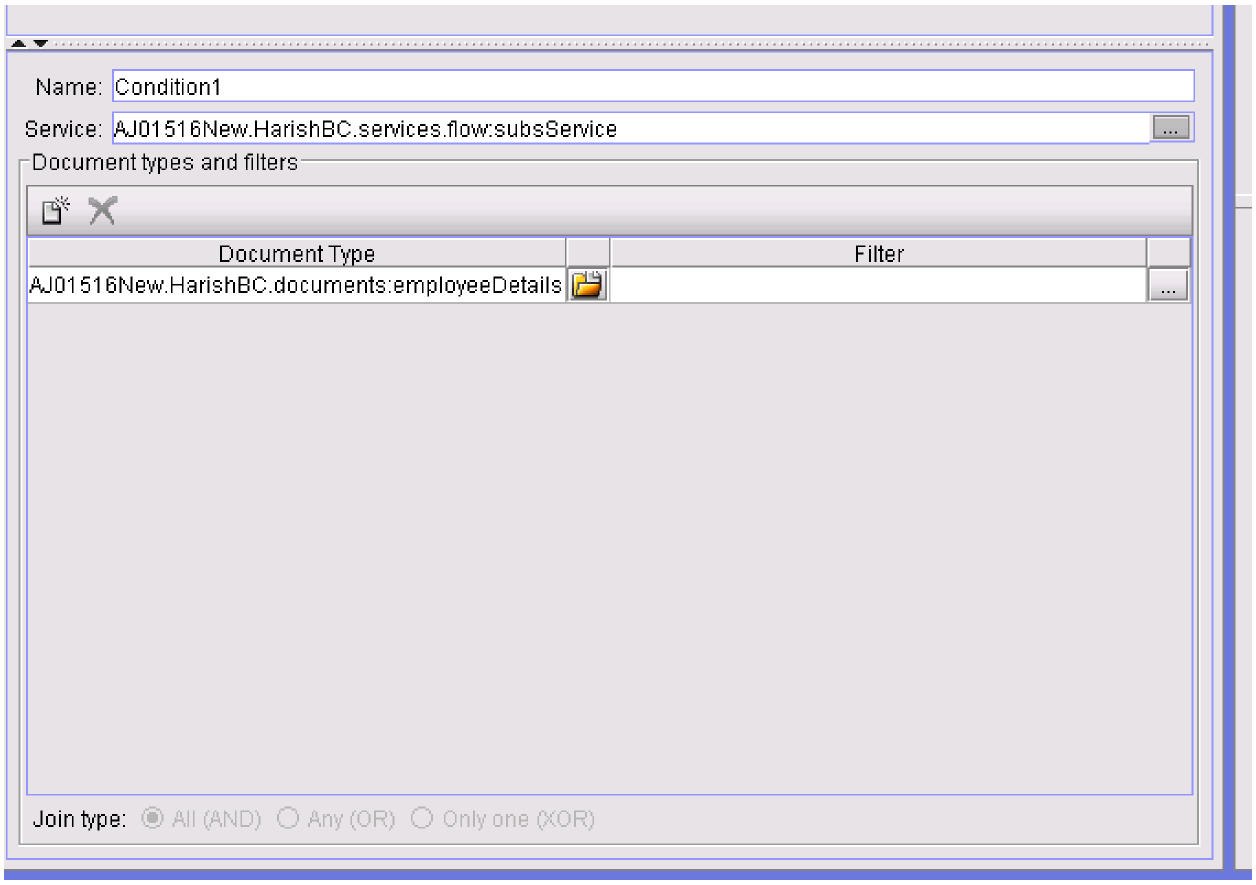
Step5: Invoke the **pub.publish:waitForReply** service when we require the response document from the subscribing service.

- If you define the **waitTime**, then the publishing service will wait till the specified time interval for the reply document. If you do not specify it then the service will wait until it receives the reply document.
- We need to set true to async option to implement publish and wait asynchronously.



Creating a trigger:

- Create a new Broker/Local trigger which subscribes to the published document. Select the service that needs to be invoked when the published document matches with the defined document type in the trigger.


Name: Condition1

Service: AJ01516New.HarishBC.services.flow:subsService

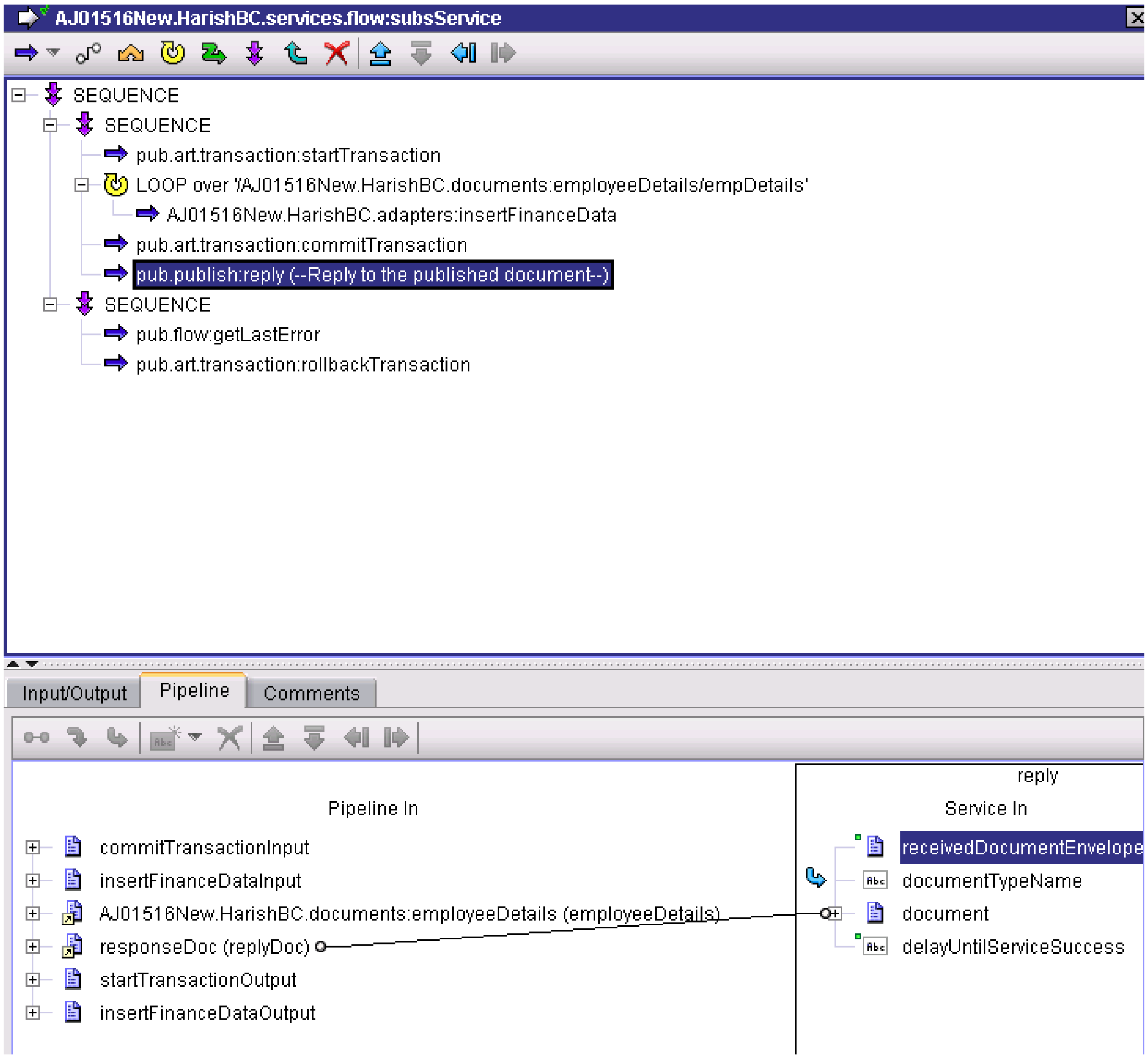
Document types and filters

Document Type	Filter
AJ01516New.HarishBC.documents:employeeDetails	

Join type: [✓ All (AND)] Any (OR) Only one (XOR)

On the Subscriber Side:

- **Step1:** Create a subscriber flow service. Define the publishable document in the input tab of the flow service.
- **Step2:** Create a insert JDBC adapter service which will insert the published records into the DB. Once it is successfully inserted then send the reply back to the publishing service by invoking the **pub.publish:reply** service.



AJ01516New.HarishBC.services.flow:subsService

```
SEQUENCE
    SEQUENCE
        pub.art.transaction:startTransaction
        LOOP over '/AJ01516New.HarishBC.documents:employeeDetails/empDetails'
            AJ01516New.HarishBC.adapters:insertFinanceData
        pub.art.transaction:commitTransaction
        pub.publish:reply (--Reply to the published document--)
    SEQUENCE
        pub.flow:getLastError
        pub.art.transaction:rollbackTransaction
```

Pipeline In

- commitTransactionInput
- insertFinanceDataInput
- AJ01516New.HarishBC.documents:employeeDetails (employeeDetails)
- responseDoc (replyDoc)
- startTransactionOutput
- insertFinanceDataOutput

reply

Service In

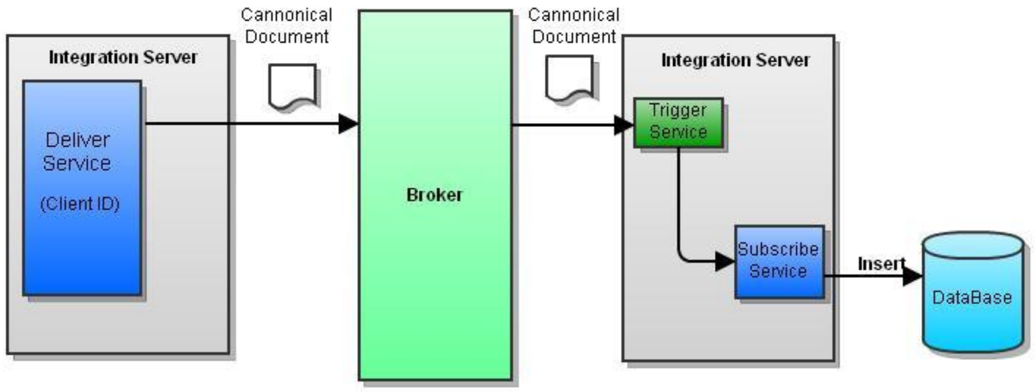
- receivedDocumentEnvelope
- documentTypeName
- document
- delayUntilServiceSuccess

Deliver:

Delivering a document is much like publishing a document, except that you specify the client that you want to receive the document. The Broker routes the document to the specified subscriber. Because only one client receives the document, delivering a document essentially bypasses all the subscriptions to the document on the Broker.

Following are the steps to implement the Deliver model:

Design Diagram:



On the Publisher Side:

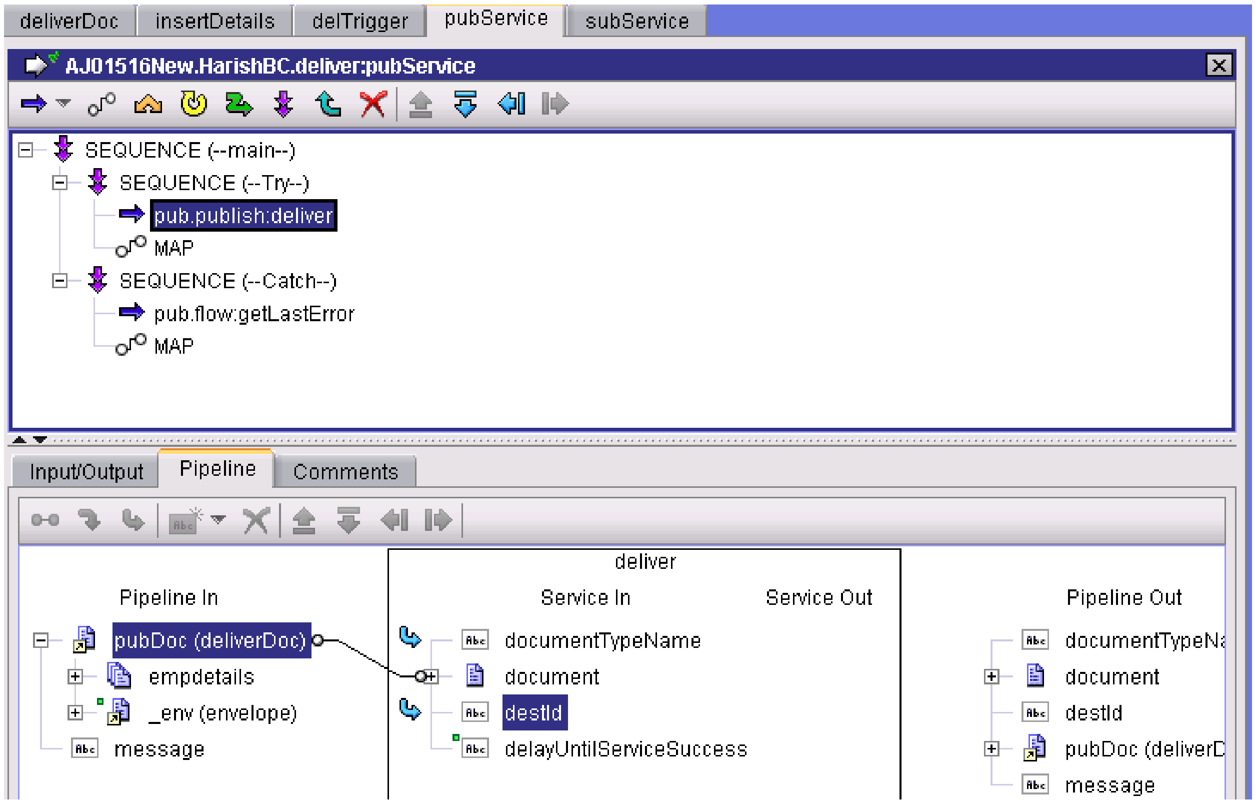
Step1: Create a flow service with declaring a IS document reference to the publishable document type that you want to deliver.
You can accomplish this by:

- Declaring a document reference in the input signature of the publishing service
  —OR—
- Inserting a MAP step in the publishing service and adding the document reference to Pipeline Out. You must *immediately* link or assign a value to the document reference. If you do not, Developer automatically clears the document reference the next time it refreshes the Pipeline tab.

Step2: Invoke **pub.publish:deliver** to publish the document. This service takes the document you defined and publishes it.

Step3: Map the publishable document to the **document** which is defined in the Input signature and specify the fully qualified document name in the **documentTypeName** options.

- Specify the **destId** (Client ID) to which you want to deliver the document. The trigger having the same client id will only able to subscribe this document.



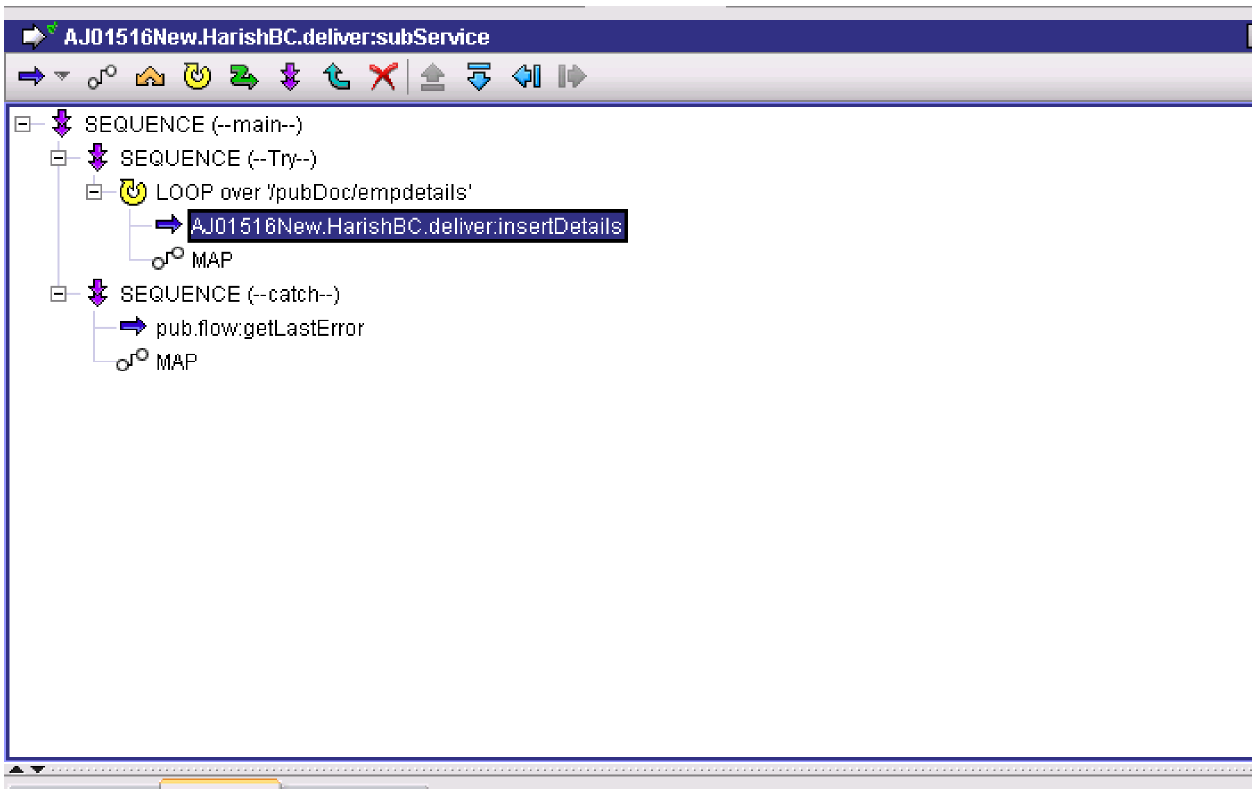
Creating a trigger:

- Create a new Broker/Local trigger which subscribes to the published document. Select the service that needs to be invoked when the published document matches with the defined document type in the trigger.

On the Subscriber Side:

Step1: Create a subscriber flow service. Define the publishable document in the input tab of the flow service.

Step2: Create a insert JDBC adapter service which will insert the published records into the DB.



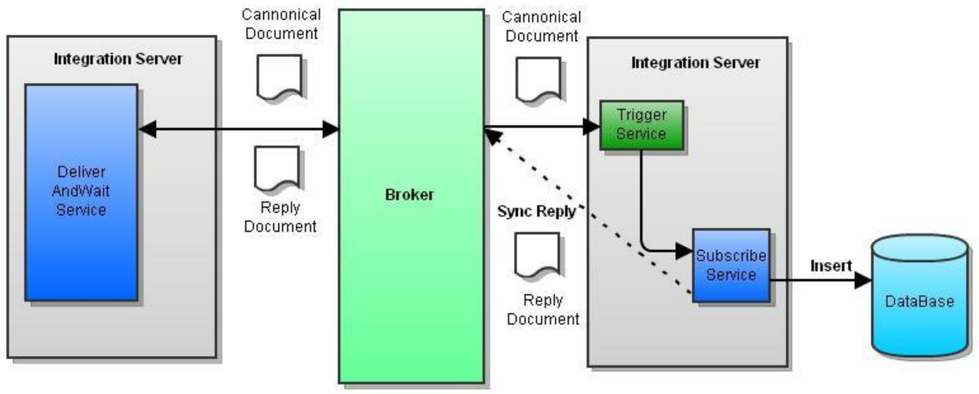
```
AJ01516New.HarishBC.deliver:subService

SEQUENCE (--main--)
    SEQUENCE (--Try--)
        LOOP over '/pubDoc/empdetails'
            AJ01516New.HarishBC.deliver:insertDetails
            MAP
    SEQUENCE (--catch--)
        pub.flow:getLastError
        MAP
```

Deliver and Wait:

Delivers a document to a specified destination and waits for a response.

Following are the steps to implement the Deliver model synchronously:

Design Diagram:

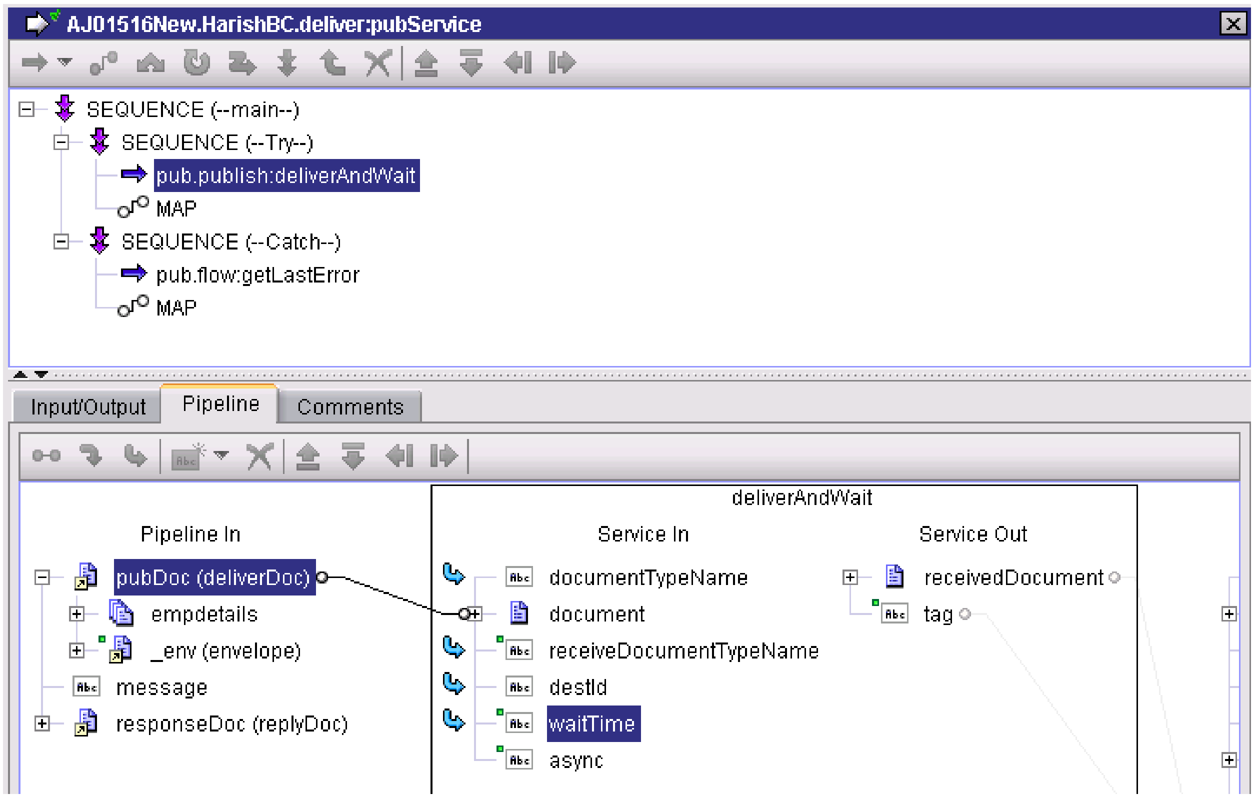


On the Publisher Side:

Step1: Create a flow service with declaring a IS document reference which is publishable in the input signature of the publishing service.
Step2: Invoke **pub.publish:deliverAndWait** to publish the document. This service takes the document you defined and publishes it and waits for the reply document from the subscribing service.

Step3: Map the publishable document to the **document** which is defined in the Input signature and specify the fully qualified document name in the **documentTypeName** options.

- Specify the **destId** (Client ID) to which you want to deliver the document. The trigger having the same client id will only able to subscribe this document.

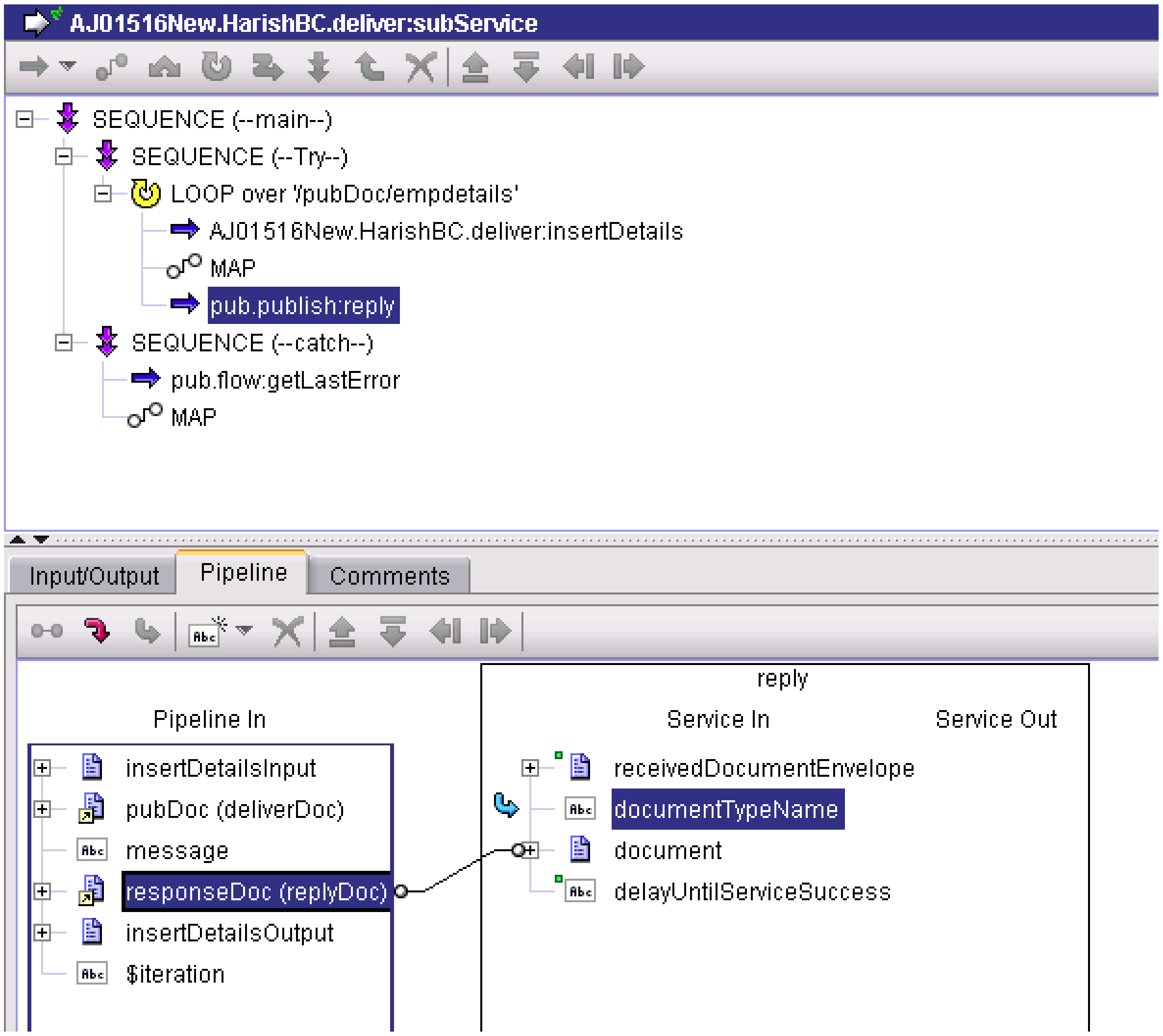


AJ01516New.HarishBC.deliver:pubService

```
SEQUENCE (--main--)
    SEQUENCE (--Try--)
        pub.publish:deliverAndWait
        MAP
    SEQUENCE (--Catch--)
        pub.flow:getLastError
        MAP
```

On the Subscriber Side:

- **Step1:** Create a trigger which subscribes to the published document. Select the service that needs to be invoked when the published document matches with the defined document type in the trigger.
- **Step2:** Create a subscriber flow service. Define the publishable document in the input tab of the flow service.
- **Step3:** Create a insert JDBC adapter service which will insert the published records into the DB.
-
- **Step4:** Once the records are inserted into the DB successfully, the reply document will be sent back to the publishing service by invoking the **pub.publish:reply** service.



AJ01516New.HarishBC.deliver:subService

```
SEQUENCE (--main--)
    SEQUENCE (--Try--)
        LOOP over '/pubDoc/empdetails'
            AJ01516New.HarishBC.deliver:insertDetails
            MAP
            pub.publish:reply
    SEQUENCE (--catch--)
        pub.flow:getLastError
        MAP
```