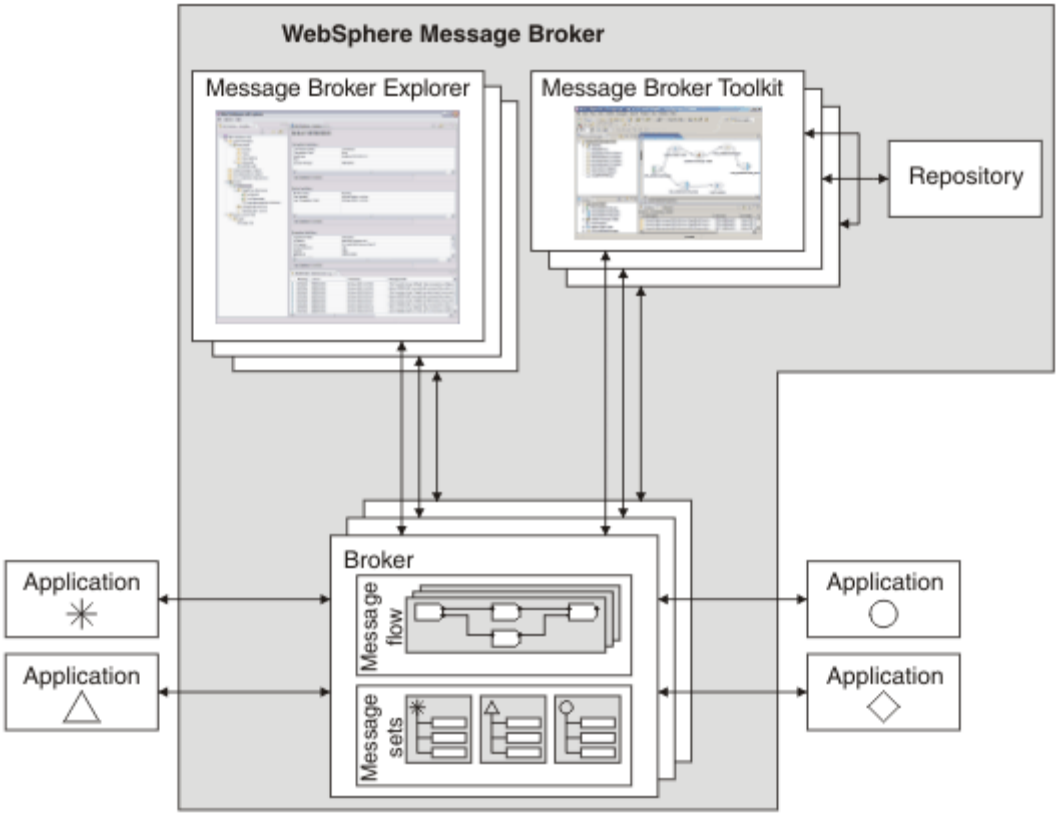
IBM WebSphere Message Broker

- WebSphere Message Broker (**WMB)** is IBM's integration broker from the WebSphere product family that allows business information to flow between disparate applications across multiple hardware and software platforms. Rules can be applied to the data flowing through the message broker to *route and transform the information*. The product is an Enterprise Service Bus providing connectivity between applications and services in a Service Oriented Architecture.The key feature of the WMB product is the ability to *abstract the business logic away from transport or protocol specifics*.



HISTORY : Originally the product was developed by NEON (New Era of Networks) Inc., a company which was acquired by Sybase in 2001. The product was later re-branded as an IBM product called '**MQSeries Integrator**' (or 'MQSI' for short). Versions of MQSI ran up to 2.0.

The product was added to the WebSphere family and rebranded '**WebSphere MQ Integrator**', at version 2.1. After 2.1 the version numbers became more synchronized with the rest of the WebSphere family and jumped to version 5.0. The name changed to '**WebSphere Business Integration Message Broker**' (WBIMB). In this version the development environment was redesigned using Eclipse and support for Web services was integrated into the product. Since **version 6.0** the product has been known as '**WebSphere Message Broker**'. WebSphere Message Broker version 7.0 was announced in October 2009,[2] and WebSphere Message Broker version 8.0 announced in October 2011[3]

In April 2013, IBM announced that the WebSphere Message Broker product was undergoing another rebranding name change.[4] The WMB product version for the new product name of **IBM Integration Bus** is **version 9** and includes new nodes such as the Decision Service node which enables content based routing based on a rules engine and requires IBM WebSphere Operational Decision Management product.[5] The IBM WebSphere Enterprise Service Bus product has been discontinued with the release of IBM Integration Bus and IBM is offering transitional licenses to move to IBM Integration Bus.[6] The WebSphere Message Broker Transfer License for WebSphere Enterprise Service Bus enables customers to exchange some or all of their WebSphere Enterprise Service Bus license entitlements for WebSphere Message Broker license entitlements. Following the license transfer, entitlement to use WebSphere Enterprise Service Bus will be reduced or cease. This reflects the WebSphere Enterprise Service Bus license entitlements being relinquished during the exchange. ***IBM announced at Impact 2013 that WESB will be end-of-life in five years and no further feature development of the WESB product will occur.***

-IBM Integration Bus (WMB V9) provides capabilities to build solutions needed to support diverse integration requirements through a set of connectors to a range of data sources, including packaged applications, files, mobile devices, messaging systems, and databases. A benefit of using IBM Integration Bus is that the tool *enables existing applications for Web Services without costly legacy application rewrites*. IIB

avoids the point-to-point strain on development resources by connecting any application or service over multiple protocols, including SOAP, HTTP and JMS.[1] *Modern secure authentication mechanisms*, including the ability to perform actions on behalf of masquerading or delegate users, through MQ, HTTP and SOAP nodes are supported such as LDAP, X-AUTH, O-AUTH, and two-way SSL.also,a *new Global Cache feature enhances overall performance capability and throughput rates*

Overview :

The WebSphere Message Broker runtime reduces cost and complexity of IT systems by unifying the method a company uses to implement interfaces between disparate systems. WMB runtime forms the Enterprise Service Bus of a Service Oriented Architecture by efficiently increasing the flexibility of connecting unlike systems into a unified, homogeneous architecture. The key feature of the WMB product is the ability to abstract the business logic away from transport or protocol specifics.

The WebSphere Message Broker Toolkit enables developers to graphically design mediations, known as message flows, and related artifacts. Once developed, these resources can be packaged into a broker archive (BAR) file and deployed into the runtime environment. At this point, the broker is able to continually process messages according to the logic described by the message flow.[7] A wide variety of data formats are supported, and may be modeled using standard XML Schema and DFDL schema. After modeling, a developer can create transformations between various formats using nodes supplied in the Toolkit, such as Mapping node, Compute nodes, or database nodes.

WebSphere Message Broker flows can be used in a Service Oriented Architecture, and if properly designed by Middleware Analysts, integrated into event-driven SOA schemas, sometimes referred to as SOA 2.0. Businesses rely on the processing of events, which might be part of a business process, such as issuing a trade order, purchasing an insurance policy, reading data using a sensor, or monitoring information gathered about IT infrastructure performance. WebSphere Message Broker includes rich complex-event-processing capabilities that enable analysis of events to perform validation, enrichment, transformation and intelligent routing of messages based on a set of business rules.

A developer creates WMB functionality in a cyclical workflow, probably more agile than most other software development. Developers will create a message flow, generate a BAR file, deploy the message flow contained in the BAR file, test the message flow and repeat

as necessary to achieve reliable functionality.

MQSI Commands are runtime commands to interact with Message Broker Toolkit.

Patterns :

A pattern captures a commonly recurring solution to a problem (example: Request-Reply pattern[14]). The specification of a pattern describes the problem being addressed, why the problem is important, and any constraints on the solution. Patterns typically emerge from common usage and the application of a particular product or technology. A pattern can be used to generate customized solutions to a recurring problem in an efficient way. We can do this pattern recognition or development through a process called service oriented modeling.
WebSphere Message Broker version 7 introduced patterns that:

Provide guidance in implementing solutions
Increase development efficiency because resources are generated from a set of predefined templates
Improve quality through asset reuse and common implementation of functions such as error handling and logging

- The patterns cover a range of categories including file processing, application integration, and message based integration.

Pattern examples

- Request-Reply (RR)
- Aggregation (Ag)
- Sequential (Seq)

What are the overlaps and main differences between the transformation technologies in WMB and WTX?

There are some overlaps between transformation capabilities, but at their core, the two products have complementary strengths — ***WMB's being routing, WTX's being complex transformation***

Message Broker is a platform for "enterprise-class data movement and message processing" building off the well proven capabilities of WebSphere MQ. WebSphere MQ is the industry-leading platform for assured transactional message delivery. On its own, it provides no capability to "manipulate" the content of messages, but in combination with WebSphere Message Broker, it offers very powerful capabilities. WMB is rich in detecting and distributing information based on message content, switching between protocols, refacing applications to perform and appear as Web services, providing high volume, high availability processing and more. It also has many different options for data transformation catering to different skill sets. While these data transformation capabilities are typically a class above those offered by other integration vendor's, ***WMB's focus has been mostly on the domain of small short messages***.

WTX comes from a different background and has different strengths. Its heritage is the ***processing of large file structures***, and in this area, it is particularly rich. It can provide complex data transformation capabilities for all and any structured and semi-structured data formats (including "mixed" types). Furthermore, WTX is better suited to handling "custom" tagged message formats or mixed tagged and binary. Finally, it is well suited for large file strucutures.

Combined, WTX and WMB provide a truly winning combination that is unique in the market in its breadth and depth.

Why not include WTXMB as integral component of WMB?

IBM prefers to give customers choice by providing a range of packaging options allowing the start simple and grow as and when required.

MB COMPONENTS :

- **Message Flow :** A message flow is a sequence of processing steps that run in the broker when an input message is received.

- **Message Set :** A message set is a container for grouping messages and associated message resources (elements, types, groups);COMPLEX TYPE CREATION .

- **Execution Group :** A grouping of message flows that have been assigned to a broker.

- **Broker :** A broker is a set of execution processes or groups that hosts one or more message flows to route, transform, and enrich in flight messages.

- **Broker Domain :** A broker domain is one or more brokers that share a common configuration, together with the single Configuration Manager that controls them.

- **Configuration Manager :** The Configuration Manager is the interface between the workbench and an executing set of brokers. It provides brokers with their initial configuration, and updates them with any subsequent changes. It maintains the broker domain configuration. It is the central runtime component that manages the components and resources that constitute the broker domain.

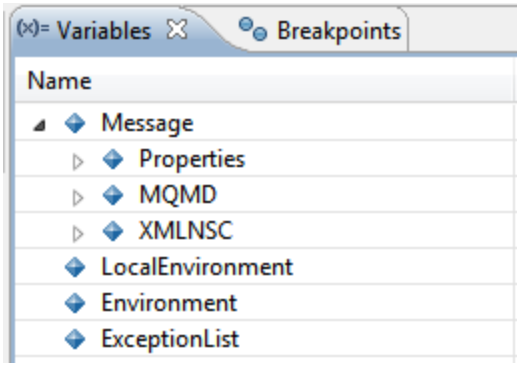
- The Configuration Manager has four main functions:

- Maintains configuration details in an internal repository. This repository provides a central record of the broker domain components.
- Deploys the broker topology and message processing operations in response to actions initiated through the workbench. Broker archive (BAR) files are deployed through the Configuration Manager to the execution groups within a broker.
- Reports on the results of deployment and the status of the broker.
- Communicates with other components in the broker domain using WebSphere® MQ transport services.

LOGICAL TREE STRUCTURE:

- The logical tree structure is the internal (broker) representation of a message. It is also known as the message assembly.

- When a message arrives at a broker, it is received by an input node that you have configured in a message flow. Before the message can be processed by the message flow, the message must be interpreted by one or more parsers that create a logical tree representation from the bit stream of the message data.



The input node creates this message assembly, which consists of four trees:

1. Message tree structure
2. Environment tree structure
3. Local environment tree structure
4. Exception list tree structure

-The input node passes the message assembly that it has created to subsequent message processing nodes in the message flow:

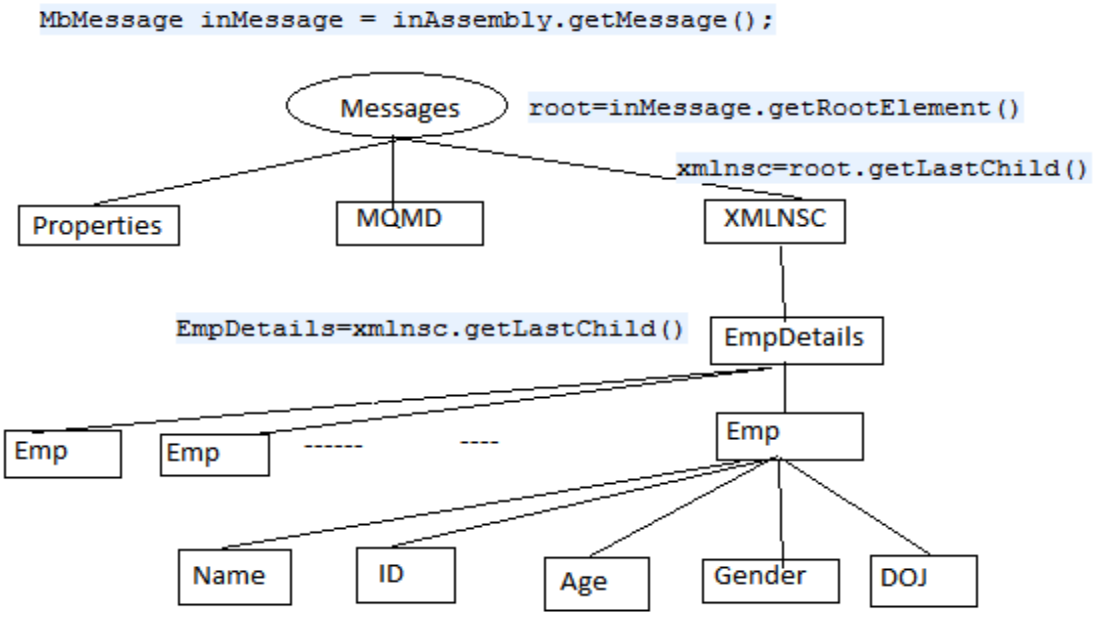
- All message processing nodes can read the four trees.
- You can code ESQL in the Database and Filter nodes, or use mappings in the nodes that support that interface to modify the Environment and LocalEnvironment trees only.
- The Compute node differs from other nodes in that it has both an input message assembly and at least one output message assembly. *Configure the Compute node to determine which trees are included in the output message assembly*; the Environment tree is an exception in that it is always retained from input message assembly to output message assembly.

To determine which of the other trees are included, you must specify a value for the Compute mode property of the node (displayed on the Advanced tab). *The default action is for only the message to be created*. You can specify any combination of message, LocalEnvironment, and ExceptionList trees to be created in the output message assembly.

- If you want the output message assembly to contain a complete copy of the input message tree, you can code a single ESQL SET statement to make the copy. If you want the output message to contain a subset of the input message tree, code ESQL to copy those parts that you want. In both cases, your choice of Compute mode must include Message.
- If you want the output message assembly to contain all or part of the input LocalEnvironment or ExceptionList tree contents, code the appropriate ESQL to copy information you want to retain in that tree. Your choice of Compute mode must include LocalEnvironment, or Exception, or both.
- You can also code ESQL to populate the output message, Environment, LocalEnvironment, or ExceptionList tree with information that is not copied from the input tree. For example, you can retrieve data from a database, or calculate content from the input message data.

1. Message tree structure :

The message tree is a part of the logical message tree in which the broker stores its internal representation of the message body.



- **Properties :**The Properties folder is the first element of the message tree and

holds information about the characteristics of the message like;

- Message Set
- Message Type
- Message Format
- Transactionality
- Persistence
- Priority

- **MQMD :** Defines Message Headers like;
  - Source Queue
  - Message Id
  - Correlation Id
  - User Id
  - Reply To Q
  - Reply To QM
  - Put time
  - Put Date
- **Body(XMLNSC) :**The Body tree is a structure of child elements that represents the message content (data), and reflects the logical structure of that content.

2. Environment tree structure :

Scope: Single instance of it is maintained throughout the message flow

- The entire contents of the input environment tree are retained in the output environment tree, subject to any modifications that you make in the node. Any changes that you make are available to subsequent nodes in the message flow, and to previous nodes if the message flows back (for example, to a FlowOrder or TryCatch node).

You could use the following ESQL statements to create the content shown above.

```
SET Environment.Variables =
    ROW('granary' AS bread, 'riesling' AS wine, 'stilton' AS cheese);
SET Environment.Variables.Colors[] =
    LIST{'yellow', 'green', 'blue', 'red', 'black'};
SET Environment.Variables.Country[] = LIST{ROW('UK' AS name, 'pound' AS currency
        ROW('USA' AS name, 'dollar' AS currency)};
```

3. Local environment tree structure :

Scope : Instance of it is maintained till the next node of the message flow

- The local environment tree to store variables that can be referred to and updated by message processing nodes that occur later(next node) in the message flow.

- You can also use the local environment tree to define destinations (that are internal and external to the message flow) to which a message is sent.

Example:

LocalEnvironment.Destination.MQ
LocalEnvironment.Destination.DestinationList
LocalEnvironment.Destination.File
LocalEnvironment.Destination.Email

4. Exception list tree structure :

- The exception list tree is a part of the logical message tree in which the message flow writes information about exceptions that occur when a message is processed.

- If an exception condition occurs, message processing is suspended and an exception is thrown. Control is passed back to a higher level; that is, an enclosing catch block. An exception list is built to describe the failure condition, and the whole message, together with the local environment tree, and the newly-populated exception list, is propagated through an exception-handling message flow path.

-The child of ExceptionList is always RecoverableException. Typically, only one child of the root is created, although more than one might be generated in some circumstances. The child of ExceptionList contains a number of children, the *last of which provides further information specific to the type of exception*. The following list includes some of the exception types that you might see:

- FatalException

- RecoverableException
- ConfigurationException
- SecurityException
- ParserException
- ConversionException
- DatabaseException
- UserException
- CastException
- MessageException
- SqlException
- SocketException
- SocketTimeoutException
- UnknownException

ESQL OVERVIEW :

Extended Structured Query Language (ESQL) is a programming language defined by WebSphere® Message Broker to define and manipulate data within a message flow.

ESQL is based on Structured Query Language (SQL) which is in common usage with relational databases such as DB2®. ESQL extends the constructs of the SQL language to provide support for you to work with message and database content to define the behavior of nodes in a message flow.

The ESQL code that you create to customize nodes within a message flow is defined in an ESQL file, typically named *<message_flow_name>.esql*, which is associated with the message flow project. You can use ESQL in the following built-in nodes:

- Compute node
- Database node
- Filter node

You can also use ESQL to create functions and procedures that you can use in the following built-in nodes:

- DataDelete node
- DataInsert node
- DataUpdate node
- Extract node
- Mapping node
- Warehouse node

To use ESQL correctly and efficiently in your message flows, you must also understand the following concepts:

- Data types
- Variables
- Field references
- Operators
- Statements
- Functions
- Procedures
- Modules

-Use the ESQL debugger, which is part of the flow debugger, to debug the code that you write. The debugger steps through ESQL code statement by statement, so that you can view and check the results of every line of code that is run.

ESQL data types

-ESQL defines the following categories of data. Each category contains one or more data types.

- Boolean
- Datetime
- Null
- Numeric
- Reference
- String

ESQL variables

-An ESQL variable is a data field that is used to help process a message.

External :

-External variables (defined with the EXTERNAL keyword) are also known as user-defined properties,*They exist for the entire lifetime of a message flow and are visible to all messages passing through the flow.* You can define external variables only at the module and schema level. You can modify the initial values of external variables (optionally set by the DECLARE statement) at design time, by using the Message Flow editor, or at deployment time, by using the Broker Archive editor. You can query and set the values of

user-defined properties at run time by using the Configuration Manager Proxy (CMP).

Normal :
-Normal variables have a *lifetime of just one message passing through a node*. They are visible to that message only. To define normal variables, omit both the EXTERNAL and SHARED keywords.

Shared :
-*Shared variables can be used to implement an in-memory cache in the message flow*, see Optimizing message flow response times. Shared variables have a long lifetime and are visible to multiple messages passing through a flow, see Long-lived variables. Shared variables exist for the *lifetime of the execution group process*, the lifetime of the flow or node, or the lifetime of the node SQL that declares the variable (whichever is the shortest). Shared variables are initialized when the first message passes through the flow or node after each broker startup.

---------- ESQL COMPUTE NODE CODING & STANDARDS : ----------

1.Declaration:

```
DECLARE i INTEGER 1;
DECLARE A CHARACTER 'Miracle software systems';
DECLARE i INTEGER CARDINALITY(someinput); // Gives total number of records in the input
DECLARE i INTEGER SUM(someinput);
```

2.Assigning a input reference to a variable or an assignment operation:

```
DECLARE inRef REFERENCE TO InputRoot.XMLNSC.DETAILS;
SET outRef.EMP[i].ENAME = inRef.EMP[i].ENAME;
```

3.Defining a while loop:

```
WHILE i <=  count  DO
  SET outRef.EMP[i].ENAME = inRef.EMP[i].ENAME;
  SET outRef.EMP[i].LOCATION = inRef.EMP[i].LOCATION;
  SET outRef.EMP[i].BATCH = inRef.EMP[i].BATCH;
  SET i = i + 1;
END WHILE;
```

4.Creating a xml Field:

```
CREATE FIELD OutputRoot.XMLNSC.DETAILS.EMP;

-Its always a best practise to use reference to the field created
DECLARE forOutRef REFERENCE TO OutputRoot.XMLNSC.DETAILS.EMP;
```

5.Creating a next sibling to a xml Field:

```
CREATE NEXTSIBLING OF forOutRef AS forOutRef
CREATE NEXTSIBLING OF OutputRoot.XMLNSC.Order.Summary.CustomerDetails
NAME 'Address';
```

6.Deleting a lastchild in a xml:

```
DELETE LASTCHILD OF OutputRoot.XMLNSC.DETAILS;
```

7.Creating a Procedure:

```
CREATE PROCEDURE  mapping (IN inputRef REFERENCE, INOUT inoutRef
REFERENCE )
BEGIN

  SET inoutRef.Designation = 'Senior Employee';
  SET inoutRef.Nationality = 'Indian';
  SET inoutRef.Company  = inputRef.COMPANY;

END;
```

8.Defining a for loop:

```
FOR forRef AS inRef.EMP[]  DO
  SET forOutRef.ENAME   = forRef.ENAME;
  SET forOutRef.LOCATION = forRef.LOCATION;
  SET forOutRef.BATCH   = forRef.BATCH;

-- Try to create procedures for reusable of  code.
  CALL mapping(forRef,forOutRef);
  CREATE NEXTSIBLING OF forOutRef AS forOutRef REPEAT;
END FOR;
```

9.Defining a Row and Select Statement to access a row in xml:

```
DECLARE Record ROW;
SET Record.val[] =  SELECT A.BATCH  FROM  OutputRoot.XMLNSC.DETAILS.EMP[]AS
A;
```

10.Casting:

```
CAST(A.BATCH AS INTEGER)
Cast(InputRoot.XMLNSC.Order.Items.Item[i].Price AS DECIMAL
CCSID InputRoot.MQMD.CodedCharSetId
```

11.Get the FieldNames and FieldValues from xml:

```
FIELDNAME(InputRoot.XMLNSC.DETAILS.EMP.ENAME);
FIELDVALUE(InputRoot.XMLNSC.DETAILS.EMP.ENAME);
```

12.Current Date:

```
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
```

13.Switch Case:

```
-- Calculating the Tax based on state name --
CASE state
WHEN 'NJ' THEN
SET tax=(sum*7)/100;
WHEN 'NY' THEN
SET tax=(sum*9)/100;
WHEN 'CY' THEN
SET tax=(sum*8)/100;
WHEN 'TX' THEN
SET tax=0;
ELSE
SET tax=NULL;
END CASE;
```

14.Create CHILD With a name

```
CREATE FIRSTCHILD OF OutputRoot.XMLNSC.Order NAME 'Summary';
```

15.Create Attributes for Tag of XML

```
DECLARE ref2 REFERENCE TO OutputRoot.XMLNSC.Order.Summary.Address;
SET ref2.(XMLNSC.Attribute)ccode =ref3.CustomerID;
SET ref2.(XMLNSC.Attribute)State =state;
```

15.Functions:

```
LENGTH(A)
LCASE(A)
```

UCASE(A)
LEFT(A,2)
RIGHT(A,2)
TRIM(A)
SUBSTRING(A FROM 2 FOR 3)
POSITION('i' IN A)
LTRIM(A)
RTRIM(A)
OVERLAY(A PLACING 'ss' FROM 2 FOR 2)
REPLACE(A,'s','ssss')
REPLICATE(A,3)
TRANSLATE(A,'Miracle','ITLOKAM')

16.Sending the same data to multiple queues

```
SET OutputLocalEnvironment.Destination.MQ.DestinationData[1].queueName = 'Q3';
SET OutputLocalEnvironment.Destination.MQ.DestinationData[2.queueName = 'Q4;
```

17.Sending the data to another terminal without deleting the original data

```
PROPAGATE TO TERMINAL 'out1' DELETE NONE;
```

18.Calling External Java Class in ESQL :

```
CALL CallExternalJavaProgram() INTO A; -- Function call

CREATE PROCEDURE  CallExternalJavaProgram() RETURNS CHARACTER
LANGUAGE JAVA
EXTERNAL NAME "com.ibm.shiva.MyJavaClassForMb.CallMe"; -- CallMe is a
methodName in JavaClass
```

---------- JAVA COMPUTE NODE CODING & STANDARDS TO CREATE A OUTPUT STRUCTURE FOR AN INPUT XML : ----------

```
package com.shiva;

import com.ibm.broker.javacompute.MbJavaComputeNode;
import com.ibm.broker.plugin.*;

public class JavaComputeNodeMsgFlow_JavaCompute extends MbJavaComputeNode
{

   public void evaluate(MbMessageAssembly inAssembly) throws MbException {
          MbOutputTerminal out = getOutputTerminal("out");
          MbOutputTerminal alt = getOutputTerminal("alternate");

          //Declaration of Variables
          String Ename = null,Location = null,Batch = null,Company = null;

          MbMessage inMessage = inAssembly.getMessage();

          // create new message
          MbMessage outMessage = new MbMessage(inMessage);
          MbMessageAssembly outAssembly = new MbMessageAssembly(inAssembly,
                        outMessage);


          //-------- INPUT PARSING ------------

          MbElement InRoot  = inMessage.getRootElement();            // MESSAGE

          MbElement InXmlNsc       = InRoot.getLastChild();          // XMLNSC

          MbElement  InDetails  = InXmlNsc.getLastChild();    //<DETAILS></DETAILS>



          //creating MbElement array to store all employee details
          MbElement Emp[]            = InDetails.getAllElementsByPath("*");    //
<EMP></EMP>



          // ------- CREATING STRUCTURE FOR OUTPUT PARSING ----------


         MbElement OutRoot= outMessage.getRootElement();          // MESSAGE
```

```
MbElement OutXmlNsc      = OutRoot.getLastChild();           // XMLNSC

 MbElement OutDetails      =
OutXmlNsc.createElementAsFirstChild(MbElement.TYPE_NAME,"EMP_DETAILS",null
;  //<EMP_DETAILS></EMP_DETAILS>
```

```
 //-- This Line used when you have Single <EMP></EMP> Record---
//MbElement OutEmp =
OutDetails.createElementAsFirstChild(MbElement.TYPE_NAME,"EMP",null);
                                                                                //<EMP></EMP>
```

```
 //Extracting child elements from <EMP>

 for(int i=0;i<Emp.length;i++){
//-- This Line used when you have Multiple <EMP></EMP> Record---
MbElement OutEmp =
OutDetails.createElementAsFirstChild(MbElement.TYPE_NAME,"EMP",null);
                                                                                //<EMP></EMP>

Ename    = Emp[i].getFirstChild().getValueAsString();
Location  = Emp[i].getFirstChild().getNextSibling().getValueAsString();
Batch      = Emp[i].getFirstChild().getNextSibling().getNextSibling().getValueAsString();
Company = Emp[i].getLastChild().getValueAsString();

 //Create Values inside Multiple <EMP></EMP> Record ,OutEmp

OutEmp.createElementAsFirstChild(MbElement.TYPE_NAME_VALUE,"ENAME",Ena
me);

OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"LOCATION",Lo
cation);

OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"BATCH",Batch
);

OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"COMPANY",C
ompany);
```

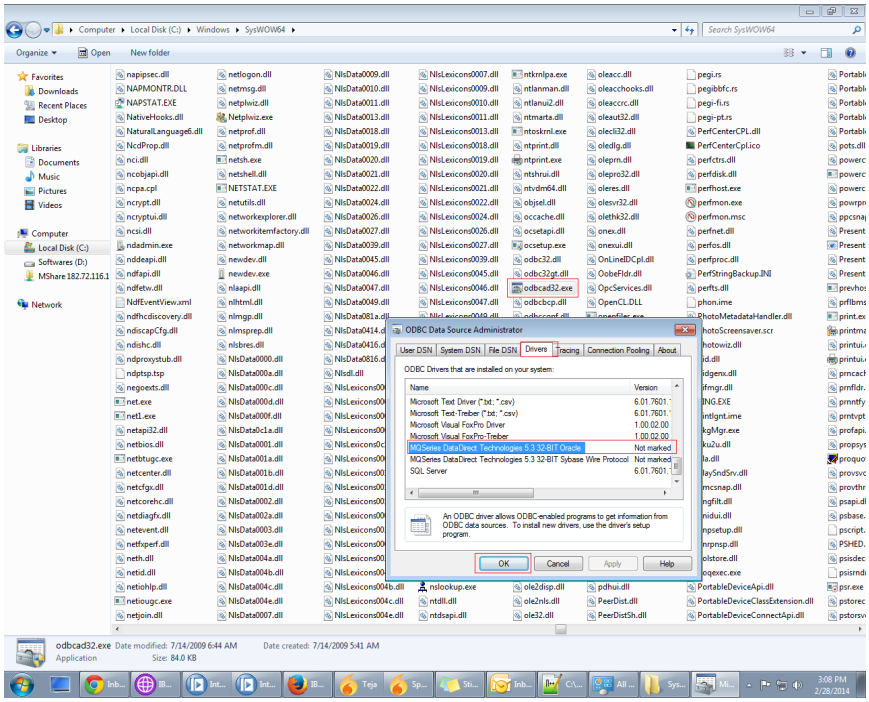
```
                } // End Of For Loop
```

```
        //Create Values inside Single <EMP></EMP> Record ,OutEmp

//OutEmp.createElementAsFirstChild(MbElement.TYPE_NAME_VALUE,"ENAME",Ena
me);

//OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"LOCATION",L
ocation);

//OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"BATCH",Batc
h);

//OutEmp.createElementAsLastChild(MbElement.TYPE_NAME_VALUE,"COMPANY",
Company);
```

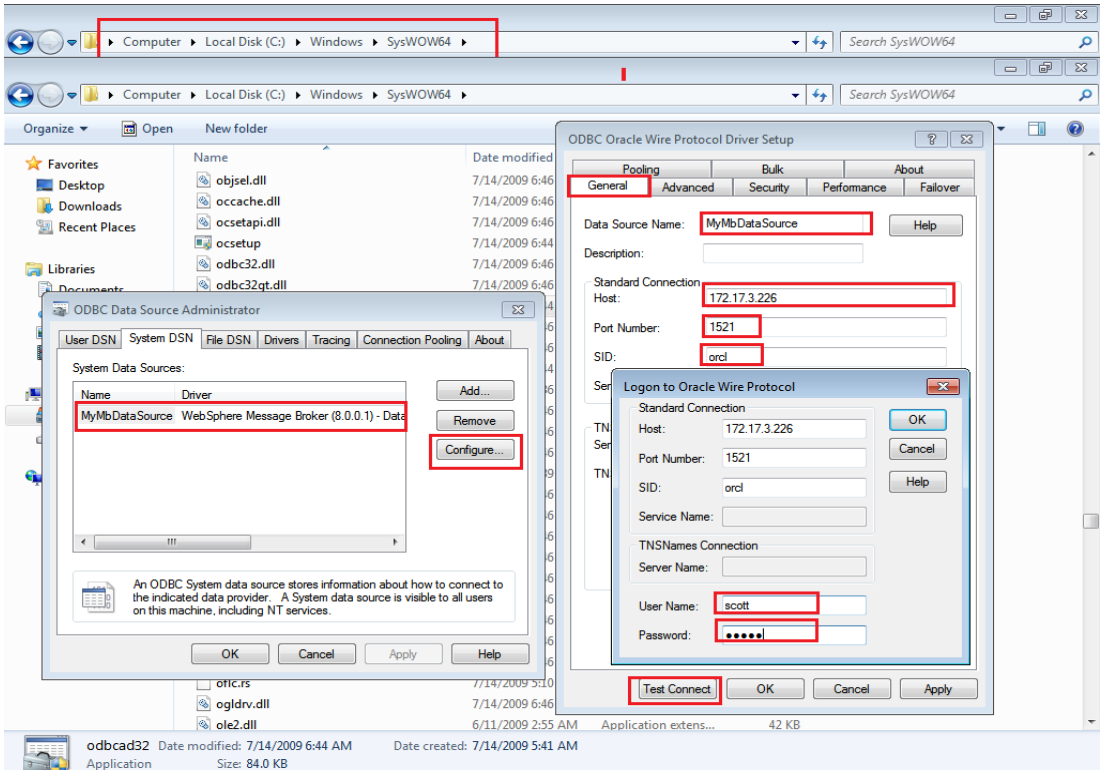
```
//detach XMLNSC last child because outMessage is created from inMessage and
already contains InMessage Structure.
   OutXmlNsc.getLastChild().detach();


//Finally Propogate the Output to OutAssembly
 out.propagate(outAssembly);

 //clear the output message
  outMessage.clearMessage();
   }
}
```

---------- Steps to follow for setting DataSource to MB ----------

1.



2.



3.Use the *mqsisetdbparms* command to associate a specific user ID and password

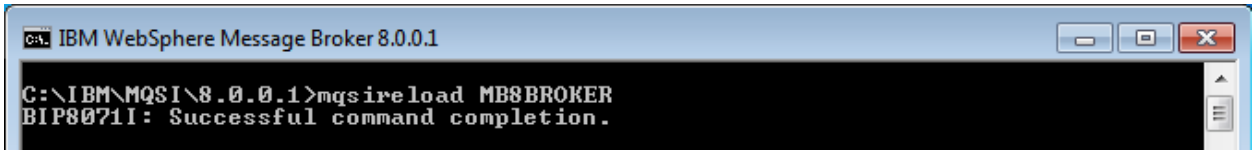
(or SSH identity file) with one or more resources(source to db) that are accessed by the broker.

> mqsisetdbparms (*BrokerName*) - n (*DataSourceName*) -u (*UserName*) -p(*Password*)

```
C:\IBM\MQSI\8.0.0.1>mqsisetdbparms MB8BROKER -n MyMbDataSource -u scott -p tiger

BIP8071I: Successful command completion.
```

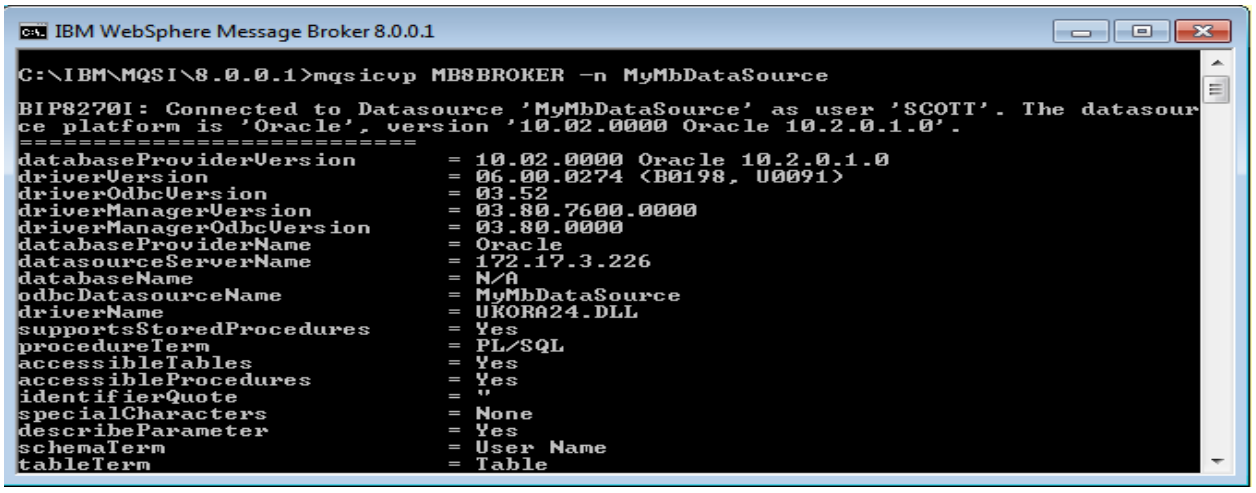
4.Use the *mqsireload* command to request the broker to stop and restart execution groups.

> mqsireload (*BrokerName*)



5.Use the *mqsicvp* command to perform verification tests on a broker or Configuration Manager.

> mqsicvp (*BrokerName*) -n (*DataSourceName*)



DataBase Interaction Uses PASSTHRU statement in Compute Node :

The main use of the PASSTHRU statement is to issue administrative commands to databases (for example, to create a table).

Note: Do not use PASSTHRU to call stored procedures; instead, use the CALL statement because PASSTHRU imposes limitations (you cannot use output parameters, for example).Uses specified *ODBC data source Name*.

- Only DDL Statements (CREATE, DROP , ALTER) requires PASSTHRU in a compute node.

Examples :

The following example creates the table Customers in schema Shop in database DSN1:

```
PASSTHRU 'CREATE TABLE Shop.Customers (
  CustomerNumber INTEGER,
  FirstName      VARCHAR(256),
  LastName       VARCHAR(256),
  Street         VARCHAR(256),
  City           VARCHAR(256),
  Country        VARCHAR(256)
)' TO Database.DSN1;
```

If, as in the last example, the ESQL statement is specified as a string literal, you must put single quotation marks around it. If, however, it is specified as a variable, omit the quotation marks.
For example:

```
SET myVar = 'SELECT * FROM user1.stocktable';
SET OutputRoot.XMLNS.Data[] = PASSTHRU(myVar);
```

The following example "drops" (that is, deletes) the table Customers from schema Shop in database DSN1:

```
PASSTHRU 'DROP TABLE Shop.Customers' TO Database.DSN1;
```

DataBase Interaction Uses ESQL statements in DataBase Node :

-Use the Database node to interact with a database in the specified *ODBC data source*.

You can use specialized forms of this node to:

- Update values within a database table (the DataUpdate node)
- Insert rows into a database table (the DataInsert node)
- Delete rows from a database table (the DataDelete node)
- Store the message, or parts of the message, in a warehouse (the Warehouse node)

DataBase Interaction With DataBaseRetrieve Node :

- Use the DatabaseRetrieve node to *ensure that information in a message is up to date*.

- Use the DatabaseRetrieve node to modify a message using information from a database. For example, you can add information to a message using a key that is contained in a message; the key can be an account number*(Select Query Elements & DataElements).*

- The DatabaseRetrieve node looks up values from a database and stores them as elements in the outgoing message assembly trees. The type of information that is obtained from the database in the form of output column values, which is acquired and passed back in the result set from SQL queries, is converted first into a matching Java type, then into an internal message element value type when it is finally stored in a location in am outgoing message assembly tree. If a message element already exists in the outgoing message tree, the new value overwrites the old value. If the target element does not exist, it is created, and the value is stored.

- Uses specified *Configurable Service Name as a Data Source Name*.

- Run ***mqsisetdbparms*** command if the Configurable service is newly created.

DataBase Interaction With DataBaseRoute Node :

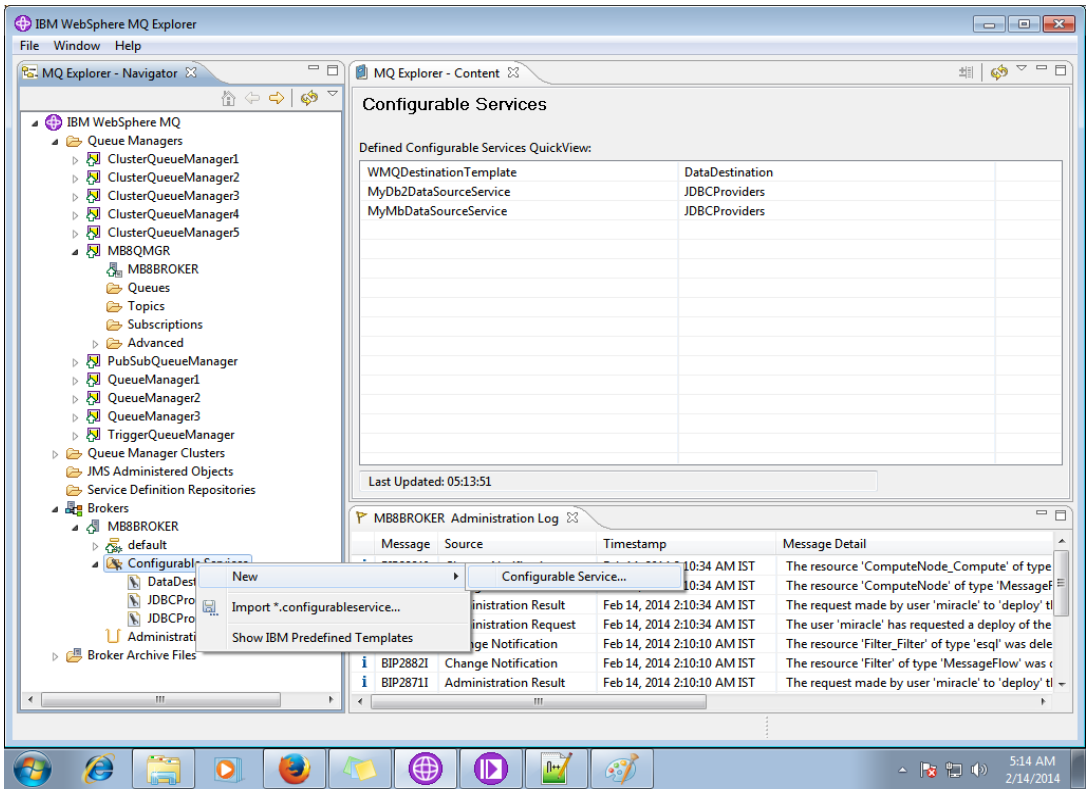
- Use the DatabaseRoute node to *route messages using information from a database* in conjunction with XPath expressions.

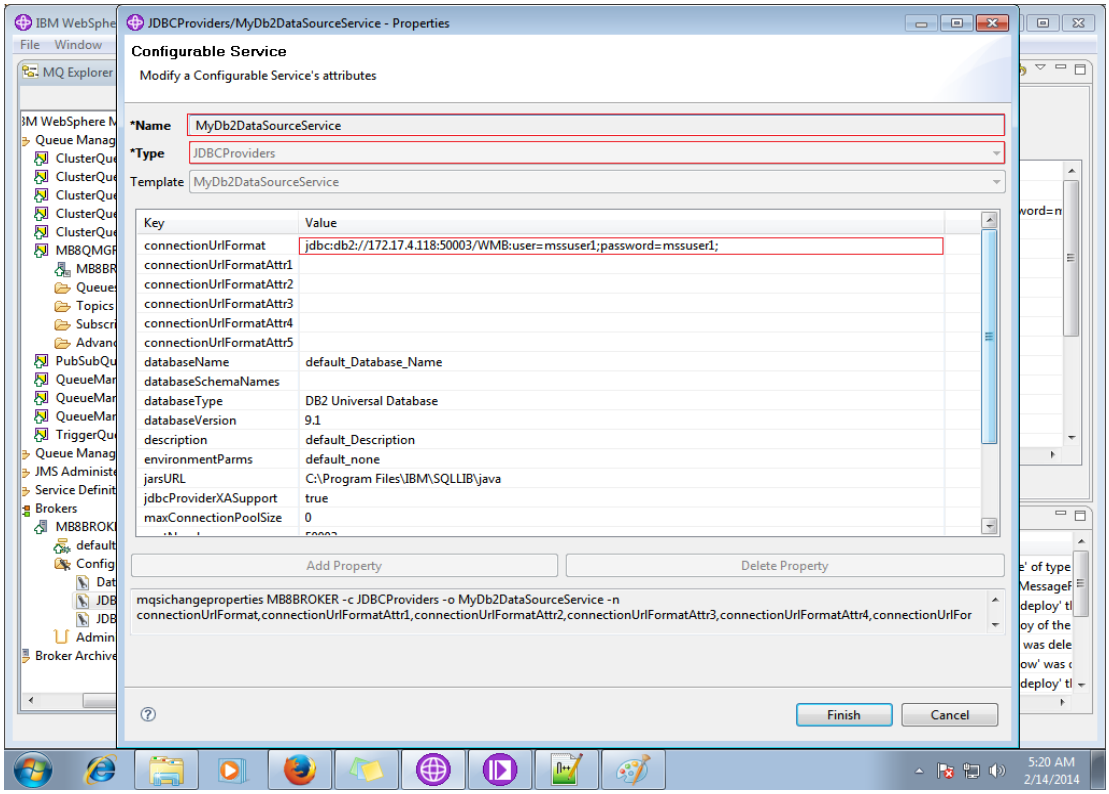
- The DatabaseRoute node uses a collection of named column values from a selected database row and synchronously applies one or more XPath expressions to these acquired values to make routing decisions.

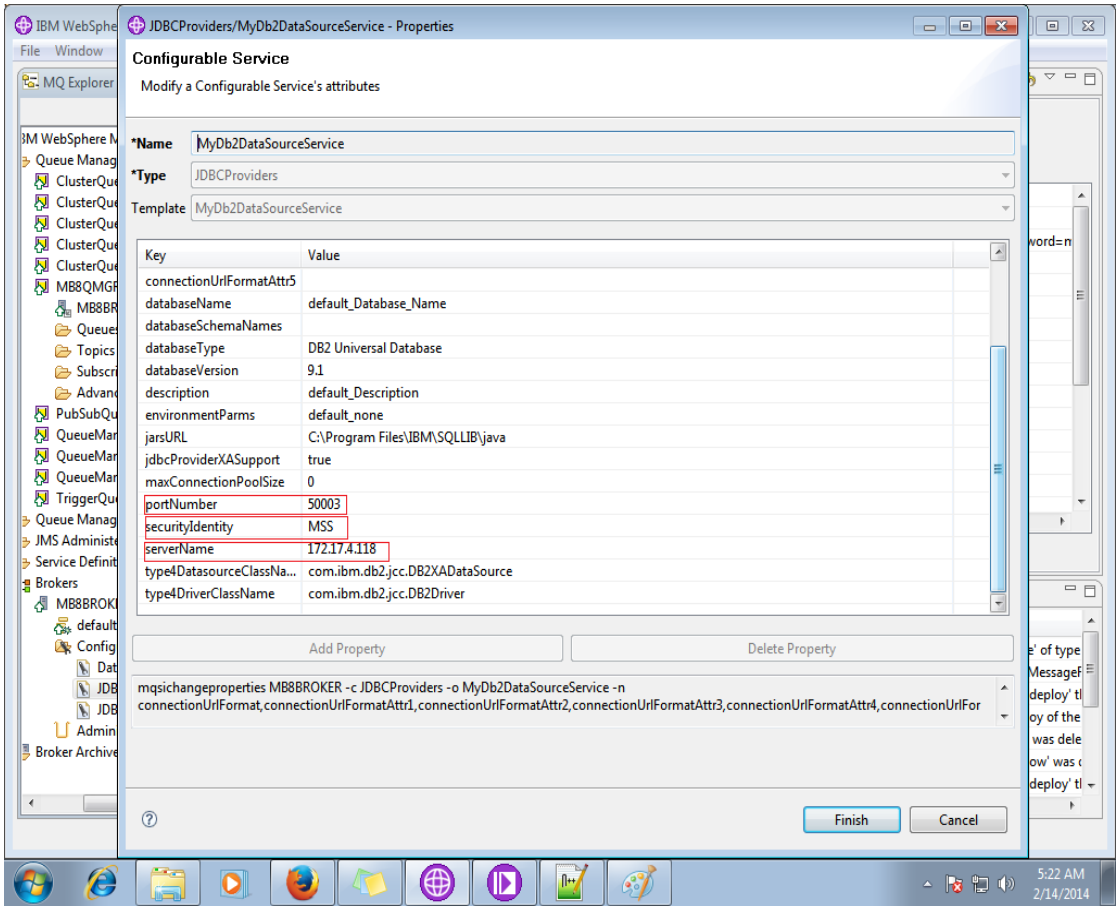
- Uses specified *Configurable Service Name as a Data Source Name*.

- Run ***mqsisetdbparms*** command if the Configurable service is newly created.

---------- Creation Of new Configurable Service in MQ ----------




JDBCProviders/MyDb2DataSourceService - Properties
Configurable Service
Modify a Configurable Service's attributes
*Name MyDb2DataSourceService
*Type JDBCProviders
Template MyDb2DataSourceService
Key Value
connectionUrlFormat jdbc:db2://172.17.4.118:50003/WMB:user=mssuser1;password=mssuser1;
connectionUrlFormatAttr1
connectionUrlFormatAttr2
connectionUrlFormatAttr3
connectionUrlFormatAttr4
connectionUrlFormatAttr5
databaseName default_Database_Name
databaseSchemaNames
databaseType DB2 Universal Database
databaseVersion 9.1
description default_Description
environmentParms default_none
jarsURL C:\Program Files\IBM\SQLLIB\java
jdbcProviderXASupport true
maxConnectionPoolSize 0
Add Property
Delete Property
mqsichangeproperties MB8BROKER -c JDBCProviders -o MyDb2DataSourceService -n
connectionUrlFormat,connectionUrlFormatAttr1,connectionUrlFormatAttr2,connectionUrlFormatAttr3,connectionUrlFormatAttr4,connectionUrlFor
Finish
Cancel
5:20 AM
2/14/2014


JDBCProviders/MyDb2DataSourceService - Properties
Configurable Service
Modify a Configurable Service's attributes
*Name MyDb2DataSourceService
*Type JDBCProviders
Template MyDb2DataSourceService
Key Value
connectionUrlFormatAttr5
databaseName default_Database_Name
databaseSchemaNames
databaseType DB2 Universal Database
databaseVersion 9.1
description default_Description
environmentParms default_none
jarsURL C:\Program Files\IBM\SQLLIB\java
jdbcProviderXASupport true
maxConnectionPoolSize 0
portNumber 50003
securityIdentity MSS
serverName 172.17.4.118
type4DatasourceClassNa... com.ibm.db2.jcc.DB2XADataSource
type4DriverClassName com.ibm.db2.jcc.DB2Driver
Add Property
Delete Property
mqsichangeproperties MB8BROKER -c JDBCProviders -o MyDb2DataSourceService -n
connectionUrlFormat,connectionUrlFormatAttr1,connectionUrlFormatAttr2,connectionUrlFormatAttr3,connectionUrlFormatAttr4,connectionUrlFor
Finish
Cancel
5:22 AM
2/14/2014

Note: *mqsisetdbparms* command, *mqsireload* command & *mqsicvp* commands are mandatory after creating a new configurable service for any in-built nodes.

In the above command console, MIRACLE is a security Identity Name provided in the Configurable service & JdbcProvider is the type you selected while creating the service.

On Another Note :

Viewing and Changing Http Local Listener Ports for HTTP Nodes :

Viewing Existing Http Listener Details :

mqsireportproperties (*BrokerName*) - e (*ExecutionGroup*) -o HTTPConnector -a

Changing Existing Http Listener Port :

mqsichangeproperties (*BrokerName*) -b HTTPListener -o HTTPConnector -n port -v 7800

IBM WebSphere MQ

An introduction to MQ :

In a nutshell, WebSphere MQ is an *assured delivery mechanism*, which consists of queues managed by Queue Managers for MB applications to Send/Receive Data or Messages. We can put messages onto, and retrieve messages from queues, and the movement of messages between queues is facilitated by components called Channels and

Transmission Queues.

There are a number of fundamental points that we need to know about WebSphere MQ:

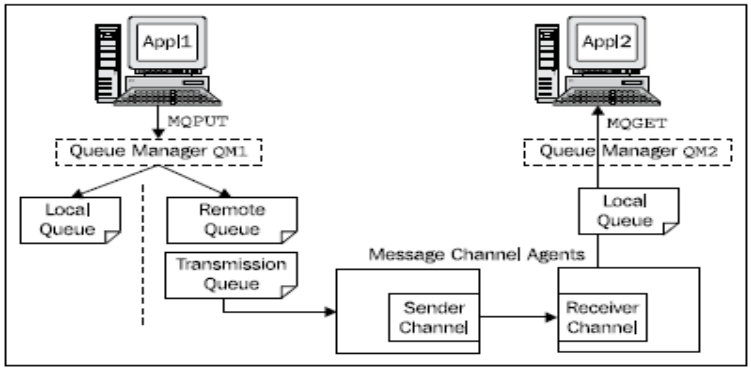
- All objects in WebSphere MQ are case sensitive
- We cannot read messages from a *Remote* Queue (only from a Local Queue)
- We can only put a message onto a *Local* Queue (not a Remote Queue)

MQ queues :

MQ queues can be thought of as conduits to transport messages between Queue Managers.

There are four different types of MQ queues and one related object. The four different types of queues are: **Local Queue** (QL), **Remote Queue** (QR), **Transmission Queue** (TQ), and **Dead Letter Queue**, and the related object is a **Channel** (CH).

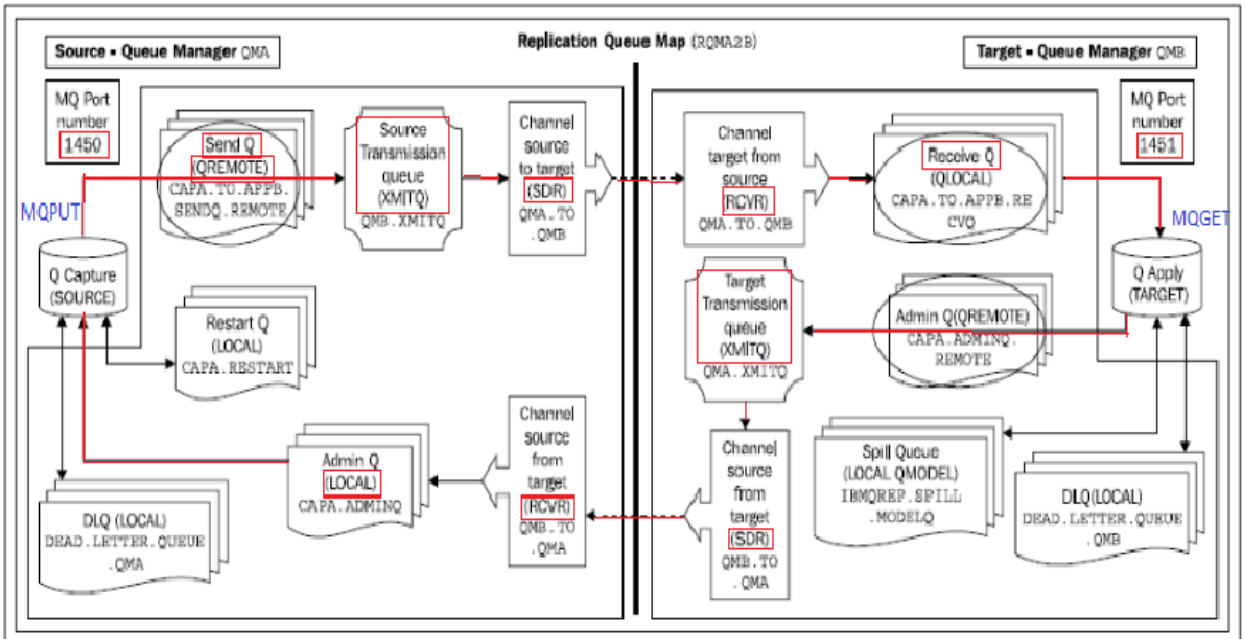
Let's take a high-level look at how messages are moved, as shown in the following diagram:



-When the application *Appl1* wants to send a message to application *Appl2*, it opens a queue - the local Queue Manager (*QM1*) determines if it is a Local Queue or a Remote Queue. When *Appl1* issues an *MQPUT* command to put a message onto the queue, then if the queue is local, the Queue Manager puts the message directly onto that queue. If the queue is a Remote Queue, then the Queue Manager puts the message onto a Transmission Queue.

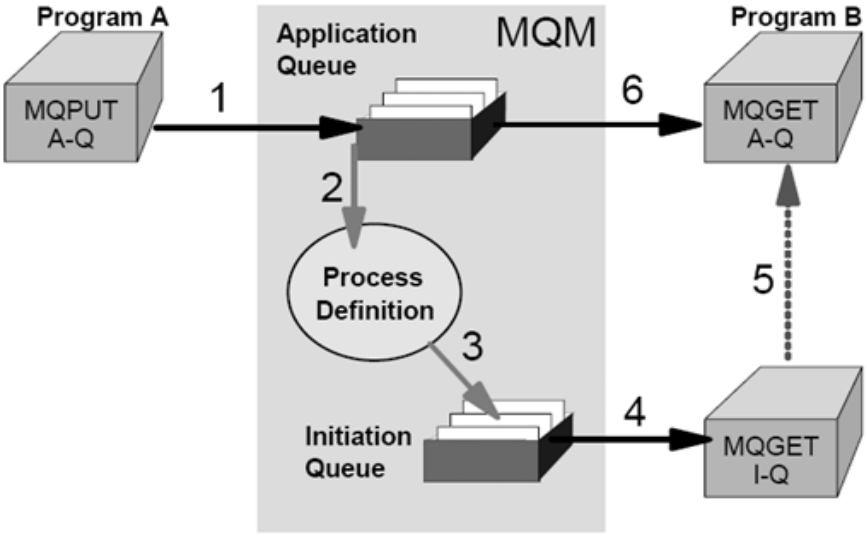
-The Transmission Queue sends the message using the Sender Channel on *QM1* to the Receiver Channel on the remote Queue Manager (*QM2*). The Receiver Channel puts the message onto a Local Queue on *QM2. Appl2* issues a *MQGET* command to retrieve the message from this queue.

-Now let's move on to look at the queues used by Q replication and in particular, unidirectional replication, as shown in the following diagram. What we want to show here is the *relationship between Remote Queues, Transmission Queues, Channels, and Local Queues*. As an example, let's look at the path a message will take from Q Capture on *QMA* to Q Apply on *QMB*.



MQ Triggering :

The queue manager defines certain conditions as constituting “trigger events”.
If triggering is enabled for a queue and a trigger event occurs, the queue manager sends a trigger message to a queue called an initiation queue. The presence of the trigger message on the initiation queue indicates that a trigger event has occurred.



1. Program A issues an MQPUT and puts a message into A-Q for Program B.

2. The queue manager processes this API call and puts the message into the application queue.

3. It also finds out that the queue is triggered. It creates a trigger message and looks in the Process Definition to find the name of the application and puts it in the trigger message. The trigger message is put into the initiation queue.

4. The trigger monitor gets the trigger message from the initiation queue and starts the program specified.

5. The application program starts running and issues an MQGET to retrieve the message from the application queue.

Process definition:

A process definition object defines an application that starts in response to a trigger event on a Web Sphere MQ queue manager.

Initiation queue:

Defining an initiation queue when a trigger event occurs, the queue manager puts a trigger message on the initiation queue specified in the application queue definition. Initiation queues have no special settings.

WebSphere MQSC commands for MQ:

Start MQ Explorer - STRMQCFG

Display Version Information - DSPMQVER

Display Queue Managers - DSPMQ

Display command server - DSPMQCSV *(Queue Manager Name)*
(display the status of the command server
for the specified queue manager;Starting,Ending
Running,Stopped)

End command server - ENDMQCSV *(Queue Manager Name)*

START command server - STRMQCSV *(Queue Manager Name)*

Create Queue Manager - CRTMQM *(Queue Manager Name)*

Delete queue manager - DLTMQM *(Queue Manager Name)*

Start Queue Manager - STRMQM *(Queue Manager Name)*

End Queue Manager - ENDMQM *(Queue Manager Name)*

Run MQSC commands - RUNMQSC *(Queue Manager Name)*
(defining, altering, or deleting a local queue object)

Define a Local Queue - DEFINE QLOCAL (*Queue Name*)
Define a Transmission Queue - DEFINE QLOCAL (*Transmission queue Name*)
USAGE (XMITQ)

Define a Remote Queue Definition - DEFINE QREMOTE (*Remote Queue Definition Name*) RQMNAME (*Remote Queue Manager Name*) RNAME (*Remote Queue Name*) XMITQ(*Transmission Queue Name*)

Run Channel Initiator - RUNMQCHI [-m QMgrName] [-q InitQ]

Define a Sender Channel - DEFINE CHANNEL (*Channel name)*

CHLTYPE (SDR) TRPTYPE (TCP) CONNAME (*'LOCALHOST(Remote Queue Manager Listener)'*) XMITQ (*Transmission Queue Name*)

Define a Receiver Channel - DEFINE CHANNEL (*Channel name*) CHLTYPE (RCVR) TRPTYPE(TCP)

Define a Listener - DEFINE LISTENER (*Listener Name*) TRPTYPE (TCP) PORT (*Port Number*)

Start a Listener - RUNMQLSR -t (TCP)

End Listener - ENDMQLSR

Start Trigger Monitor - RUNMQTRM -q(*Initiation Queue Name)* -m(*Queue Manager Name*)

Run Dead-Letter Queue Handler - RUNMQDLQ

Start Trace - STRMQTRC

End Trace - ENDMQTRC

Display Formatted Trace - DSPMQTRC

WebSphere MQSI commands for MB:

Broker Commands

FOR ADMINISTRATOR COMMANDS - RUNMQSICOMMANDCONSOLE

Create a broker and its associated resources - MQSICREATEBROKER (*brokerName)* -i (*serviceUserId)* -a (*servicePassword)* -q (*queueManagerName)* -n (*dataSourceName)*

Change one or more of the configuration parameters of the broker - MQSICHANGEBROKER (*brokerName*)

Delete a broker - MQSIDELETEBROKER (*BrokerName*)

Request the broker to stop and restart execution groups - MQSIRELOAD(*BrokerName*)

Create deployable broker archive files containing message flows and dictionaries - MQSICREATEBAR -data (*WorkSpace*) -o (FilePathOfMsgFlow) -b (*BarName*)

Deployment request to the Configuration Manager - MQSIDEPLOY -b (brokerName) -e (executionGroupName) -a (BARFileName)

ConfigManager Commands

Create a Configuration Manager and its associated resources - MQSICREATECONFIGMGR (*ConfigmgrName)* -i (*ServiceUserID)* -a (*ServicePassword)* -q (*QueueManagerName)*

Changes one or more of the properties of the Configuration Manager - MQSICHANGECONFIGMGR (*ConfigmgrName)*

Delete a named Configuration Manager - MQSIDELETECONFIGMGR (*ConfigmgrName*)

Properties commands

Modify broker properties and properties of broker resources - MQSICHANGEPROPERTIES (*BrokerName)* -o (*ObjectName)* -n (*PropertyName)* -v (*PropertyValue)*

Display properties that relate to a broker, an execution group, or a configurable service

MQSIREPORTPROPERTIES -o *(ObjectName)*

Trace Commands

Display the trace options currently in effect - MQSIREPORTTRACE (Bro*ker Name*) -u -e (*ExecutionGroupName*) -f (*FlowName*) -o (*OutFileName*)

Set the tracing characteristics for a component - MQSICHANGETRACE (*Broker Name*) -u - e (*ExecutionGroupName*) -f (*FlowName*) -r -l (*Level*)

Retrieve the trace log for the specified component - MQSIREADLOG (*Broker Name*) -u -e (*ExecutionGroupName*) -f (*FlowName*) -o (*OutFileName*)

Process the XML log created by mqsireadlog - MQSIFORMATLOG -i (InputFileName) -o (OutFileName)

Note : In the above trace commands Broker Name can be replaced by any component you wish to generate and read logs

WebSphere MB NODES

WebSphere MQ :

MQInput node : 

- MQInput node is used to *receive messages from clients* that connect to the broker by using the WebSphere® MQ Enterprise Transport, and that use the MQI and AMI application programming interfaces.

The MQInput node handles messages in the following message domains:

- XMLNSC

- DataObject
- JSON
- BLOB
- MIME
- MRM
- JMSMap
- JMSStream
- XMLNS

.

Terminals:

Out : The MQInput node routes each *message that it retrieves successfully* to the Out terminal.

Failure : If the *backout count is exceeded* (as defined by the BackoutThreshold attribute of the input queue), the message is routed to the Failure terminal.If you have not connected the Failure terminal, the message is written to the backout queue.

Catch : If the message is caught by this Input node after an *exception has been thrown further on in the message flow*, the message is routed to the Catch terminal.*If you have not connected the Catch terminal, the message loops continually through the node until the problem is resolved.*

Note:
-You must define a backout queue or a dead-letter queue (DLQ) to prevent the message from looping continually through the node.

Transaction Mode:

Automatic : The message is received under sync point if the incoming message is marked as persistent; otherwise, it is not received under sync point. Any message that is sent later by an output node is put under sync point, as *determined by the incoming persistence property*, unless the output node has overridden this property explicitly.

Yes (the default) : The message is received under sync point; that is, within a WebSphere MQ unit of work. Any messages that are sent later by an output node in the same instance of the message flow are put under sync point, unless the output node has overridden this explicitly.

No : The message is not received under sync point. Any messages that are sent later by an output node in the message flow are not put under sync point, unless an individual output node has specified that the message must be put under sync point.The MQOutput node is the only

output node that you can configure to override this option.

Parse timing :

On Demand : On-demand parsing, referred to as partial parsing, is used to *parse an input message bit stream only as far as is necessary to satisfy the current reference*(Delayed).An input message can be of any length. *To improve performance of message flows*, a message is parsed only when necessary to resolve the reference to a particular part of its content. If none of the message content is referenced within the message flow (for example, the entire message is stored in a database by the DataUpdate node, but no manipulation of the message content takes place), the message body is not parsed.Therefore, fields might not be parsed until late in the message flow, or never. This restriction applies to both the message body and the message headers.

Immediate and **Complete** : Both override partial parsing and *parse the entire message*, including any message headers, except when the MRM parser encounters an element with a complex type with Composition set to Choice or Message that cannot be resolved at the time; for example, the content must be resolved by the user in ESQL. If Composition is set to Choice, the data is added to the message tree as an unresolved item, and parsing continues with the next element. If Composition is set to Message, parsing terminates at that point. *The only difference in behavior between Immediate and Complete occurs when MRM validation is enabled.*

Order mode :

-The order in which messages are retrieved from the input queue and processed.*This property has an effect only if the message flow property Additional instances on the Instances tab, is set to greater than zero; that is, if multiple threads read the input queue.*

Default : Messages are retrieved in the *order that is defined by the queue attributes, but this order is not guaranteed* because the messages are processed by the message flow.

User ID : Messages that have the same UserIdentifier value in the MQMD are retrieved and processed in *the order that is defined by the queue attributes; this order is guaranteed to be preserved when the messages are processed*. A message that is associated with a particular user identifier that is being processed by one thread, is completely processed before the same thread, or another thread, can start to process another message with the same user identifier. Ensure that each message has a unique message ID in the MQMD of the incoming message. No other ordering is guaranteed to be preserved.

Queue Order : Messages are retrieved and processed by this node in the *order that is defined*

by the queue attributes; this order is guaranteed to be preserved when the messages are processed. This behavior is identical to the behavior that is exhibited if the message flow property Additional instances is set to zero. However, if you set Order mode to By Queue Order then redeploy the message flow, additional instances that are already running are not released. Therefore, when you set Order mode to By Queue Order, either stop and restart the message flow, or run the mqsireload command for the execution group after you redeploy the flow.

User Defined : You can specify a message element using the Order field location property. An XPath or ESQL expression property to control which part of the message is used to impose ordering on incoming messages when Order mode is User Defined. If the field is missing, an exception is raised, and the message is rolled back.

Logical order :

If you select this check box, *messages are received in logical order, as defined by WebSphere MQ*.If you clear the check box, messages that are sent as part of a group are not received in a predetermined order. If a broker expects to receive messages in groups, and you have not selected this check box, either the order of the input messages is not significant, or you must design the message flow to process them appropriately.

All messages available :

-Select All messages available if you want message retrieval and processing to be done only *when all messages in a single group are available*.

Match message ID,Match correlation ID :

- A message ID,Match correlation ID that must *match the message ID in the MQMD of the incoming message*. Enter a message identifier if you want the input node to receive only messages that contain a matching message identifier value in the MsgId field of the MQMD.

Browse Only :

-This property controls whether a message is removed from the queue when it is read. If you select this check box, the message is not removed from the queue when it is read. If you select this option, OutputLocalEnvironment.MQ.GET.Browsed is set to true when a message is propagated to the output terminal of the MQInput node.

MQOutput node : 

Use the MQOutput node to send messages to clients that connect to the broker using the WebSphere® MQ Enterprise Transport and that use the MQI and AMI application programming interfaces.

Terminals:

In : Connect the In terminal to the node from which *outbound messages bound are routed*.

Out/Failure : Connect the Out or Failure terminal of this node to another node in this message flow to *process the message further, process errors, or send the message to an additional destination*.If you use aggregation in your message flows, you must use the output terminals.

Transaction Mode:

When you define an MQOutput node, the option that you select for the Transaction Mode property defines whether the message is written under sync point:

- If you select **Yes**, the message is written under sync point (that is, within a WebSphere MQ unit of work).the message is put *transactionally.*
- If you select **Automatic** (the default), the message is written under sync point if the incoming input message is marked as persistent.
- If you select **No**, the message is not written under sync point.

Persistence Mode :

Persistence Mode, defines whether the output message is marked as persistent when it is put to the output queue:

- If you select **Yes**, the message is marked as persistent.the message is put *persistently*.
- If you select **Automatic** (the default), the message persistence is determined from the properties of the incoming message, as set in the MQMD.
- If you select **No**, the message is not marked as persistent.
- If you select As Defined for Queue, the message persistence is set as defined in the WebSphere MQ queue by the MQOutput node specifying the MQPER_PERSISTENCE_AS_Q_DEF option in the MQMD.

Destination Mode :

-The queues to which the output message is sent.

Queue Name(the default) : If you select Queue Name (the default), the *message is sent to the queue that is named in the Queue Name property*. If you select this option, you must set the Queue Manager Name and Queue Name properties.

Reply To Queue : If you select Reply To Queue, the *message is sent to the queue that is named in the ReplyToQ field in the MQMD*.When you select this value, the MQOutput node constructs a WebSphere MQ reply message.

Destination List : The *message is sent to the list of queues that are named in the local environment* that is associated with the message.

New Message ID/New Correlation ID :
- If you select this check box, WebSphere MQ generates a new message identifier/new correlation identifier to replace the contents of the MsgId field in the MQMD.

Segmentation Allowed : If you select this check box, WebSphere MQ breaks the message into segments in the queue manager. Clear the check box if you do not want segmentation to occur.

Message Context : When a security profile is associated with the node and is configured to perform identity propagation, the chosen context can be overridden to ensure that the outgoing identity is set.

Request : If you select the check box, each output message in the MQMD is generated as a request message and the message identifier field is cleared so that WebSphere MQ generates a new identifier.
Reply-to Queue : The name of the WebSphere MQ queue to which to put a reply to this request. This name is inserted into the MQMD of each output message as the reply-to queue.

MQReply node : 

The MQReply node is a specialized form of the MQOutput node that puts the output message to the WebSphere® MQ queue that is identified by the ReplyToQ field of the input message header.

-Use the MQReply node to send a response to the originator of the input message. The MQReply node is a specialized form of the MQOutput node that puts the output message to the WebSphere® MQ queue that is identified by the ReplyToQ field of the input message header.

- Terminals(In,Failure & Out) , Segmentation , Persistence mode , Transaction mode ,Validation properties are same as defined for MQOutput node.

HTTP :

HTTPInput node : 

Use the HTTPInput node to *receive an HTTP message from an HTTP client for processing by a message flow*.

-When the HTTPInput node receives a message from a web service client, the node starts the appropriate parsers to interpret the headers and the body of the message, and to create the message tree that is used internally by the message flow. The node creates a unique identifier for the input message and stores it as a binary array of 24 bytes in the local environment tree at LocalEnvironment.Destination.HTTP.RequestIdentifer. This value is used by the HTTPReply node, therefore you must not modify it.

- HTTP messages are always non-persistent, and have no associated order.
- HTTP uses port 7080, which is the default HTTP port for the broker-wide listener.
- The default port numbers for the embedded execution group listener are 7800 for HTTP and 7843 for HTTPS.

Note:You can change these port numbers, and port ranges used by the execution group listeners, by using the *mqsichangeproperties* command.

Terminals:

Failure : The output terminal to which the message is routed if an error occurs(If message validation fails).If you have not connected the Failure terminal, the message is discarded, the Maximum client wait time expires, and an error is returned to the client.

Out : The output terminal to which the message is routed if it is successfully retrieved.

Catch : The output terminal to which the message is routed if an exception is thrown downstream and caught by this node.If you have not connected the Catch terminal, the message is discarded, the Maximum client wait time expires, and an error is returned to the client.

HTTP Timeout : The output terminal to which a timeout fault message is routed if the HTTP Reply node that is connected to the Out terminal does not respond within the time interval specified by the Maximum client wait time property.

Path suffix for URL :

This property identifies the *location from where web service requests are retrieved*. Do not use the full URL. If the URL that you want is http://hostname[:port]/[path], specify either /path or /path fragment/* where * is a wildcard that you can use to mean match any.

Use HTTPS : This property identifies whether the node is to accept secure HTTP. If the node is to accept secure HTTP, select the check box.

Advanced Properties:

- **Set destination list :** This property specifies whether to *add the method binding name to the route to label destination list*. If you select this check box, the method binding name is added so that you can use a RouteToLabel node in the message flow after the HTTPInput node.

- **Label prefix :** The prefix to add to the method name when routing to label. Add a label prefix to avoid a clash of corresponding label nodes when you include multiple WebSphere Message Broker input nodes in the same message flow. By default, there is no label prefix, therefore the method name and label name are identical.
- **Parse Query String :** This property causes any query string that is present with an incoming message to be parsed and decoded (according to http://tools.ietf.org/html/rfc3986) into the following location in the local environment as a series of name-value elements that match the names and values present in the query string:
  LocalEnvironment.HTTP.Input.QueryString
  For example, for this query string:
  ?myParam1=my%22Value%221&myParam2=my%22Value%222
  -The following elements are placed into the local environment under the QueryString folder:
  - myParam1 with a value of my"Value"1
  - myParam2 with a value of my"Value"2

- **Decompress input message :**This property indicates whether an inbound HTTP request is decompressed or not.If this option is selected, and the HTTP header Content-Encoding field is "gzip" or "deflate", the input message is decompressed and propagated to the Out terminal, and the Content-Encoding field is removed.

Note : "Input Message Parsing" , "Parse Options" & Validation for HttpInput node are same as MQInput node.

Error handling properties :

- **Maximum client wait time (sec) :** Default(180)

-The length of time, in seconds, for which the TCP/IP listener that received the input message from the web service client waits for a response from the HTTPReply node in the message flow. The valid range is zero (which means a short wait) through (2^{31})-1. If a response is received within this time, the listener propagates the response to the client. If a response is not received in this time, a fault message is generated indicating that the timeout has expired. This fault message is either sent by the listener or timeout terminal processing.

- **Fault format :** The format of any HTTP errors that are returned to the client. Valid values are SOAP 1.1, SOAP 1.2, and HTML.

HTTP Reply node : 

--Use the HTTPReply node to *return a response from the message flow to an HTTP client*. This node generates the response to an HTTP client from which the input message was received by the HTTPInput node, and waits for confirmation that it has been sent.

-The HTTPReply node *constructs a reply message for the Web service client from the entire input message tree, and returns it to the requester*. If the message was initially received by an HTTPInput node in another message flow, the response is associated with the reply by a request identifier that is stored in the local environment of the message by the HTTPInput node.

Terminals:

Failure : The output terminal to which the message is routed if a failure is detected when the message is propagated.

In : The input terminal that accepts a message for processing by the node.

Out : The output terminal to which the message is routed if it has been propagated successfully, and if further processing is required in this message flow.

Ignore transport failures : Select Ignore transport failures if you want transport-related failures to be ignored (for example, *if the client is disconnected*). If you clear the check box, and a transport-related error occurs, the input message is propagated to the Failure terminal. If you

clear the check box, you must supply a value for Reply send timeout (sec).

Reply send timeout (sec) : Set the Reply send timeout (sec) value *if you are not ignoring transport failures*. This property specifies the length of time, in seconds, that the node *waits for an acknowledgment that the client has received the reply*. If the acknowledgment is received within this time, the input message is propagated through the Out terminal to the rest of the message flow, if it is connected. If an acknowledgment is not received within this time, the input message is propagated through the Failure terminal, if it is connected. If the Failure terminal is not connected, and an acknowledgment is not received in time, an exception is generated.

Generate default HTTP headers from reply or response : Select Generate default HTTP headers from reply or response if you want the default Web service headers to be created using values from the HTTPReplyHeader or the HTTPResponseHeader. If the appropriate header is not present in the input message, default values are used.

HTTP Request node :

-The HTTPRequest node *interacts with a web service, using all or part of the input message as the request that is sent to that service*.You can also configure the node to create an output message from the contents of the input message, augmented by the contents of the web service response/reply, before you propagate the message to subsequent nodes in the message flow.

-Depending on the configuration, this node *constructs an HTTP or an HTTP over SSL (HTTPS)* ***request*** from the specified contents of the input message, and sends this request to the web service. The node *receives the* ***response*** *from the web service*, and parses the response for inclusion in the output tree. The node generates HTTP headers if they are required by your configuration.

Note: You can use this node in a message flow that does or does not contain an HTTPInput or HTTPReply node because provider flow can be anywhere but only WebService URL to access that provider is required.

- An HTTP request has two parts:
    1. The URL of a service.

2. A stream of data that the remote server processes, then sends back a response, which is often a SOAP or other web service message in XML.The data must be in CCSID 1208 format for most requests.

Terminals:

Out : If the *request is successful*, the HTTPResponse is inserted into the front of the message tree, the reply placed in the specified location in the tree, and the request propagated to the Out terminal.

Failure : If the *HTTPRequest node is not able to issue the request*, an ExceptionList is inserted into the message tree and the tree is propagated to the Failure terminal.You can specify a *timeout interval*, so that if the request takes longer than the specified duration, the request is propagated to the Failure terminal with an appropriate message

Error : If the request is sent successfully by the HTTPRequest node, but the *web service is not successful*, the HTTPResponse is inserted into the message tree, and propagated to the Error terminal.

Handling errors :

The node interacts directly with an external service using TCP/IP; it can, therefore, experience the following types of error:

- *Errors that are generated by TCP/IP*, for example *no route to host or connection refused.*
If the node detects these errors, it generates an exception, populates the exception list with the error information that is received, and routes the input message unchanged to the *Failure terminal.*

-*Errors that are returned by the web server.* These errors are represented by HTTP status codes that are outside the range 100 - 299. If the node detects these errors, it routes the reply to the *Error terminal* while following the properties specified on the Error tab.

Replace input with error : For the web service error message to be included in the output message with part of the input message content, clear Replace input with error and set the Error message location property(say,OutputRoot.XMLNSC.ABC.DEF).

HTTP Response Codes :

100 - 'continue' response
200 - success
300 - redirection

HTTP Settings :

- **HTTP(S) proxy location :** set the location of the proxy server to which requests are sent.
- **Follow HTTP(S) redirection :** To specify how the node handles web service responses that contain an HTTP status code of 300 to 399(If you select the check box, the node follows the redirection that is provided in the response, and *reissues the web service request to the new URL* (included in the message content).)
- **Use Compression :** To specify the compression of the content of the HTTP request.

Advanced Properties :

Use whole input message as *request* : For the *request message to be the whole input message body*, leave Use whole input message as request selected (the default setting).

***Request* message location in tree :** For the request message to contain a *subset of the input message*, clear Use whole input message as request and set the Request message location in tree property(Say,InputRoot.XMLNSC.ABC).

Replace input message with web-service *response* : For the whole web service response message to be propagated as the output message, leave Replace input message with web-service response selected (the default setting).

***Response* message location in tree :** For the web service response message to be included in the output message with part of the input message content, clear Replace input message with web-service response and set the Response message location in tree property(say,OutputRoot.XMLNSC.ABC.DEF).

Accept compressed responses : Select the Accept compressed responses by default property to indicate whether the request accepts compressed responses. If you select this option, the request can receive responses with a Content-Encoding of gzip or deflate.

Routing :

Filter node : 

Use the Filter node to route a message according to message content.

- Create a filter expression in ESQL to define the route that the message is to take. You can include elements of the input message or message properties in the filter expression, and you can use data that is held in an external database to complete the expression. The output terminal to which the message is routed depends on whether the expression evaluates to *true, false, or unknown*.

Note : The Filter node accepts ESQL statements in the same way as the Compute and Database nodes.If your message flow requires more ***complex routing options, use the RouteToLabel and Label nodes*****.**

Terminals:

In : The input terminal that accepts a message for processing by the node.

Failure : The output terminal to which the message is routed if a failure is detected during the computation.

Unknown : The output terminal to which the message is routed if the specified filter expression evaluates to unknown or a null value.

False : The output terminal to which the message is routed if the specified filter expression evaluates to false.

True : The output terminal to which the message is routed if the specified filter expression evaluates to true.

Basic Properties :

Data Source : The *ODBC data source name of the database that contains the tables* to which you refer in the ESQL that is associated with this node (identified by the Filter Expression property). This name identifies the appropriate database on the system on which this message flow is to execute. The broker connects to this database with user ID and password information that you have specified.

Transaction : The transaction mode for the node. The values are:

- **Automatic (the default)**. The message flow, of which the Filter node is a part, is committed if it is successful. That is, the actions that you define in the ESQL module are performed and the message continues through the message flow. *If the message flow fails, it is rolled back*. Therefore, if you choose Automatic, the ability to commit or roll back the action of the Filter node on the database *depends on the success or failure of the entire message flow.*
- **Commit**. To commit any uncommitted actions that are performed in this message flow on the database that is connected to this node, irrespective of the success or failure of the message flow as a whole, select Commit. *The changes to the database are committed even if the message flow itself fails*.

Filter Expression : The *name of the module within the ESQL resource (file)* that contains the statements to execute against the message that is received in the node The ESQL file, which by default has the name <message_flow_name>.esql

Important Note : You can use all the ESQL statements including SET, WHILE, DECLARE, and IF in this module, but (unlike the Compute node) the Filter node propagates the message that it receives at its input terminal to its output terminal unchanged. Therefore, in the Filter node, like the Database node, you have only one message to which to refer.

You cannot modify any part of any message, so the assignment statement (the SET statement, not the SET clause of the INSERT statement) can assign values only to temporary variables. The scope of actions that you can take with an assignment statement is therefore limited.

Treat warnings as errors : For database warning messages to be treated as errors, and to propagate the output message from the node to the Failure terminal, select Treat warnings as errors.
-If you do not select the check box, the node treats warnings as normal return codes and does not raise any exceptions. *The most significant warning raised is not found*, which can be handled safely as a normal return code in most circumstances.

Label node : 

-Use the Label node to process a message that is propagated by a RouteToLabel node to *dynamically determine the route* that the message takes through the message flow.

-The RouteToLabel node interrogates the LocalEnvironment of the message to determine the identifier of the Label node to which the message must be routed next.You can propagate the message by coding ESQL in a Compute node, or by coding Java™ in a JavaCompute or user-defined node.

-*Precede the RouteToLabel node in the message flow with a Compute node or JavaCompute* node and populate the LocalEnvironment of the message with the identifiers of one or more Label nodes that introduce the next sequence of processing for the message.

- *Typically, a Label node connects to a subflow that processes each message in a specific way*, and either ends in an output node or in another RouteToLabel node.

Terminals:

Out : The output terminal to which the message is routed.

RouteToLabel node : 

- Use the RouteToLabel node in combination with one or more Label nodes to *dynamically determine the route* that a message takes through the message flow, based on its content.

-The RouteToLabel node interrogates the local environment of the message to determine the identifier of the Label node to which to route the message.

- You must precede the RouteToLabel node in the message flow with a Compute node that populates the local environment of the message with the identifiers of one or more Label nodes that introduce the next sequence of processing for the message. *The destinations are set up as a list of label names in the local environment tree in a specific location.*

Example:

```
IF InputRoot.XMLNSC.PassengerQuery.ReservationNumber<>'' THEN
  SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelName = 'L1';
ELSE
  SET OutputLocalEnvironment.Destination.RouterList.DestinationData[1].labelName = 'L2';
END IF;
```

Note :

- A label name in the local environment must match the Label Name property of a corresponding Label node.
- When you configure the Compute node, you must also select a value for the Compute Mode property from the list that includes LocalEnvironment.

Terminals:

In : The input terminal that accepts a message for processing by the node.

Failure : The output terminal to which the message is routed if a failure is detected during processing.

Mode :

This property controls how the RouteToLabel node processes the items in the local environment that is associated with the current message . Valid values are:

- **Route To First:** removes the first item from the local environment. The current message is routed to the Label node that is identified by labelName in that list item.
- **Route To Last (the default):** removes the last item from the local environment. The current message is routed to the Label node that is identified by labelName in that list item.

Route node : 

-Use the Route node *to direct messages that meet certain criteria* down different paths of a message flow.

-As an example, you can *forward a message to different service providers*, *based on the request details*. You can also use the Route node to bypass unnecessary steps. For example, you can

check to see if certain data is in a message, and perform a database lookup operation only if the data is missing. If you set the Distribution Mode property to All, you can trigger multiple events that each require different conditions. For example, you can log requests that relate to a particular account identifier, and send requests that relate to a particular product to be audited.

-You use *XPath filter expressions to control processing*. A result of a filter expression is cast as Boolean, so the result is guaranteed to be either true or false.

Consider the following example input message:

```
<EmployeeRecord>
        <EmployeeNumber>00001</EmployeeNumber>
        <FamilyName>Smith</FamilyName>
        <Wage>20000</Wage>
</EmployeeRecord>
```

and the following XPath filter expressions:

```
$Root/XMLNSC/EmployeeRecord/EmployeeNumber="00002"|Match
$Root/XMLNSC/EmployeeRecord/EmployeeNumber="00001"|out_exp2
```

Terminals:
In : The static input terminal that accepts a message for processing by the node.

Match : A dynamic output terminal to which the original message can be routed when processing completes successfully. You can create additional dynamic terminals;

Default : The static output terminal to which the message is routed if *no filter expression resolves to true*.

Failure : The static output terminal to which the message is routed if a failure is detected during processing.

Filter table :

- A table where all rows are expressions and associated terminal names that define the switching that is performed by this node following evaluation of each filter expression. The full expression is in the format
XPath filter expression, terminal name
- All XPath expressions must start with ***$Root, $Properties, $LocalEnvironment, $DestinationList, $ExceptionList, or $Environment***. If you are creating an expression by hand, you can also start the expression with $Body. However, the XPath Expression Builder and

associated validation in the WebSphere Message Broker Toolkit do not support use of the $Body variable. If you are using the XPath Expression Builder, use the $Root variable instead.

Distribution Mode :

-This property determines the routing behavior of the node when an inbound message matches multiple filter expressions. If you set the Distribution Mode property to **First**, the message is propagated to the associated output terminal of the first expression in the table that resolves to true. If you set this property to **All**, the message is propagated to the associated output terminal for each expression in the table that resolves to true. If no output terminal matches, the message is propagated to the Default terminal.

Aggregate Control node : 

- Use the AggregateControl node to mark the beginning of a fan-out of requests that are part of an aggregation.

- Aggregation is an extension of the request/reply application model. It combines the generation and *fan-out of a number of related requests with the fan-in of the corresponding replies, and compiles those replies into a single aggregated reply message*.

- It sends a control message that is used by the AggregateReply node to match the different requests that have been made.

Terminals:

In : The input terminal that accepts a message for processing by the node.

Out : The output terminal to which the original message is routed when processing completes successfully.

Control : The output terminal to which a control message is routed. The control message is sent to a corresponding AggregateReply node. The Control terminal is deprecated in Version 6.0.

Timeout (sec) :

-The amount of time, in seconds, that it waits for replies to arrive at the fan-in.

-The default value is **zero**; if you accept this default value, the *timeout is disabled* for fan-outs from this node (that is, it waits for replies indefinitely). If not all responses are received, the message flow continues to wait, and does not complete. Set a value **greater than zero** to ensure that t*he message flow can complete, even if not all responses are received*.

Timeout location :

- Default('$LocalEnvironment/Aggregation/Timeout')

-The *location in the message tree where the aggregation timeout value is defined*. The value specified in the message tree *overrides the Timeout (sec) property* of the AggregateControl node and the timeoutSeconds property of the Aggregation configurable service.

Aggregate Request node : 

- Use the AggregateRequest node to *record the fact that request messages have been sent*. This node also collects information that helps the AggregateReply node to construct the compound response message.

- The information that is added to the message Environment by the AggregateRequest node must be preserved, otherwise the aggregation fails.

Terminals:

In : The input terminal that accepts messages sent as part of an aggregate request.

Out : The output terminal to which the input message is routed when processing completes successfully.

Folder Name : The name that is used as a folder in the AggregateReply node's compound message to store the reply to this request. You must enter a value for this property, but the value does not need to be unique.

Aggregate Reply node : 

- Use the AggregateReply node to mark the end of an aggregation fan-in. This node collects replies and combines them into a single compound message.

Terminals:

In : The input terminal that accepts a message for processing by the node.

Control : The input terminal that accepts control messages that are sent by a corresponding AggregateControl node. The Control terminal is *deprecated* in Version 6.0.

Failure : The output terminal to which the message is routed if a failure is detected during processing.

Unknown : The output terminal to which messages are routed when they *cannot be identified as valid reply messages*.

Out : The output terminal to which the compound message is routed when processing completes successfully.

Timeout : The output terminal to which the incomplete compound message is routed when the timeout interval that is specified in the corresponding AggregateControl node has expired.

Catch : The output terminal to which the message is routed if an exception is thrown

downstream and then caught by this node.

Unknown Message Timeout : The amount of time, in seconds, for which messages that cannot be identified as replies are held before they are propagated to the Unknown terminal.
The default value is zero; if you accept this default value, the timeout is disabled, and unknown messages are propagated to the Unknown terminal upon receipt.

Transaction Mode : If you select the check box (the default), the subsequent message flow is under transaction control. This setting remains true for messages that derive from the output message and are produced by an MQOutput node, unless the MQOutput node explicitly overrides the transaction status. No other node can change the transactional characteristics of the output message.

Collector node : 

-Use the Collector node to create *message collections from one or more sources* based on rules that you configure in the node.

-For example, you might need to extract, combine, and transform information from three different sources. The messages from these different sources might arrive at the input terminals at different times and in an unknown order.A collection is defined by configuring an event handler for each input terminal. Each event handler controls the acceptance of a message into a collection according to the following properties:

- Number of messages
- Collect messages for a set period
- Match the contents of a correlation path
- Match the contents against a correlation pattern

- Use the Collector node to group messages from different input sources for further

processing. A message collection can be processed by the following nodes only:

- Compute
- JavaCompute
- PHPCompute

- You can add and *configure as many input terminals as required to the Collector node*. You can configure the properties of each input terminal separately to control how the messages received on each input terminal are added to the appropriate message collection.

- You can use the *Control terminal* to trigger the output of completed message collections from the Collector node. Configure the *Event coordination property* to set the behavior of the Collector node when messages are received on the Control terminal.

- If you use additional instances of a message flow or multiple inputs to the Collector node, you can use the ***Correlation path and Correlation pattern*** properties to ensure that *related messages are added to the same message collection*.

Terminal	Description
Control	The static input terminal that accepts control messages. A message received by the Control terminal is treated as a cont message.
Out	The output terminal to which the *complete message collection routed* if the received messages satisfy the configured conditions f the message collection.
Expire	The output terminal to which the incomplete message collectior routed if the received messages do not satisfy the configur conditions within the time specified on the Collection expiry proper If you have not set a value for the Collection expiry property t

	terminal is not used.
Failure	The output terminal to which the message collection is routed i failure is detected during processing.
Catch	The output terminal to which the message collection is routed if exception is thrown downstream and caught by this node.

Collection expiry : The amount of time, in seconds, that the Collector node waits for messages to arrive. After this time an incomplete message collection is expired and propagated to the Expire output terminal.
-If you set this property to zero, the collection expiry is disabled and the Collector node waits for messages indefinitely. Set a value greater than zero to ensure that the message collection is processed, even if not all messages are received. A warning is issued if this property is not set.

Persistence mode : This property specifies whether messages are stored on the queues of the Collector node persistently.

Channels :

WebSphere® MQ uses two different types of channels:

- A message channel, which is a unidirectional communications link between two queue managers. WebSphere MQ uses message channels to transfer messages between the queue managers. To send messages in both directions, you must define a channel for each direction.
- An MQI channel, which is bidirectional and connects an application (MQI client) to a queue manager on a server machine. WebSphere MQ uses MQI channels to transfer MQI calls and responses between MQI clients and queue managers.

Do not confuse these two distinct types of channels.
When discussing message channels, the word channel is often used as a synonym for a channel definition. It is usually clear from the context whether we are talking about a complete channel, which has two ends, or a channel definition, which has only one end.

Message channels

Message channel definitions can be one of the following types:

Message channel definition type	Description
Sender	A sender channel is a message channel that the queue manag uses to send messages to other queue managers. To se messages using a sender channel, you must also create, on t other queue manager, a receiver channel with the same name the sender channel. You can also use sender channels w requester channels if you are implementing a "callback" mechanis
Server	A server channel is a message channel that the queue manag uses to send messages to other queue managers. To se messages using a server channel, you must also create, on t other queue manager, a receiver channel with the same name the server channel. You can also use server channels with reques channels. In that case, the requester channel definition at the oth end of the channel requests the server channel definition to sta The server sends messages to the requester. The server can a initiate the communication as long as the server knows t connection name of the partner channel.
Receiver	A receiver channel is a message channel that the queue manag uses to receive messages from other queue managers. To rece

	messages using a receiver channel, you must also create, on t other queue manager, a sender or a server channel with the sa name as this receiver channel.
Requester	A requester channel is a message channel that the queue mana uses to send messages to other queue managers. To se messages using a requester channel, you must also create, on t other queue manager, a sender channel if you are implementing callback mechanism, or a server channel.
Cluster-se nder	A cluster-sender (CLUSSDR) channel definition defines the sendi end of a channel on which a cluster queue manager can se cluster information to one of the full repositories. The cluster-send channel is used to notify the repository of any changes to the que manager's status, for example the addition or removal of a queue is also used to transmit messages. The full repository que managers themselves have cluster-sender channels that point each other. They use them to communicate cluster status chang to each other. It is of little importance which full repository a que manager's CLUSSDR channel definition points to. After the ini contact has been made, further cluster queue manager objects a defined automatically as required so that the queue manager c send cluster information to every full repository, and messages every queue manager. For more information, see Queue manag clusters.
Cluster-rec eiver	A cluster-receiver (CLUSRCVR) channel definition defines t receiving end of a channel on which a cluster queue manager c receive messages from other queue managers in the cluster. cluster-receiver channel can also carry information about th cluster—information destined for the repository. By defining th cluster-receiver channel, the queue manager indicates to the oth cluster queue managers that it is available to receive messages. Y need at least one cluster-receiver channel for each cluster que manager. For more information, see Queue manager clusters.

For each channel you must define both ends so that you have a channel definition for

each end of the channel. The two ends of the channel must be compatible types. You can have the following combinations of channel definitions:

- Sender–Receiver
- Server–Receiver
- Requester–Server
- Requester–Sender (callback)
- Cluster-sender–Cluster-receiver

Message channel agents

Each channel definition that you create belongs to a particular queue manager. A queue manager can have several channels of the same or different types. At each end of the channel is a program, the message channel agent (MCA). At one end of the channel, the caller MCA takes messages from the transmission queue and sends them through the channel. At the other end of the channel, the responder MCA receives the messages and delivers them to the remote queue manager.

A caller MCA can be associated with a sender, server, or requester channel. A responder MCA can be associated with any type of message channel.

WebSphere MQ supports the following combinations of channel types at the two ends of a connection:

Caller		Direction of message flow	Responder	
Channel type	Listener required?		Listener required?	Channel type
Sender	No	Caller to Responder	Yes	Receiver
Server	No	Caller to Responder	Yes	Receiver
Server	No	Caller to Responder	Yes	Requester
Requester	No	Responder to	Yes	Server

		Caller		
Requester	Yes	Responder to Caller	Yes	Sender

MQI channels

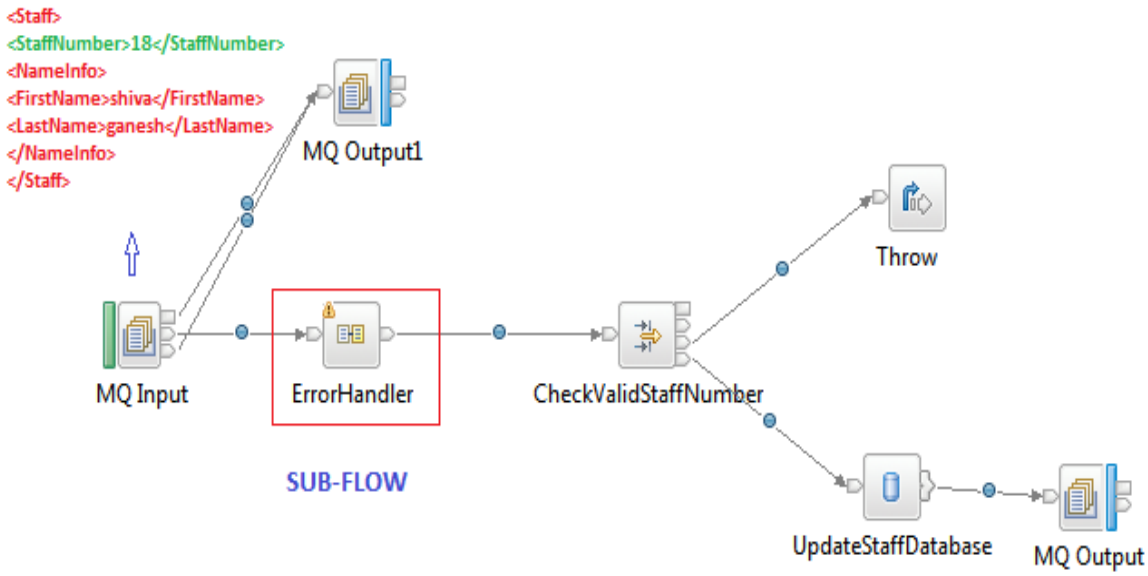
MQI channels can be one of the following types:

MQI channel type	Description
Server connection	A server connection channel is a bidirectional MQI channel that used to connect a WebSphere MQ client to a WebSphere M server. The server connection channel is the server end of t channel.
Client connection	A client connection channel is a bidirectional MQI channel that used to connect a WebSphere MQ client to a WebSphere M server. WebSphere MQ Explorer also uses client connections connect to remote queue managers. The client connection chan is the client end of the channel. When you create client-connection channel, a file is created on the computer th hosts the queue manager. You must then, copy t client-connection file to the WebSphere MQ Client computer.

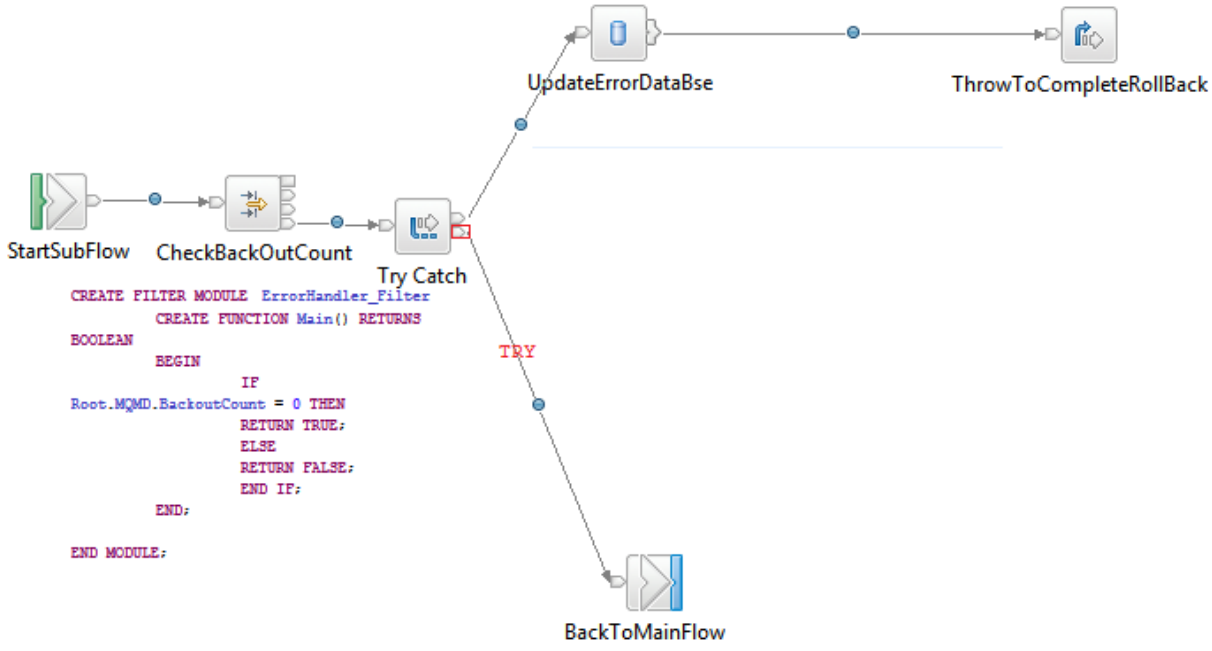
For more information, see WebSphere MQ Intercommunication.

ERROR HANDLING SCENARIO

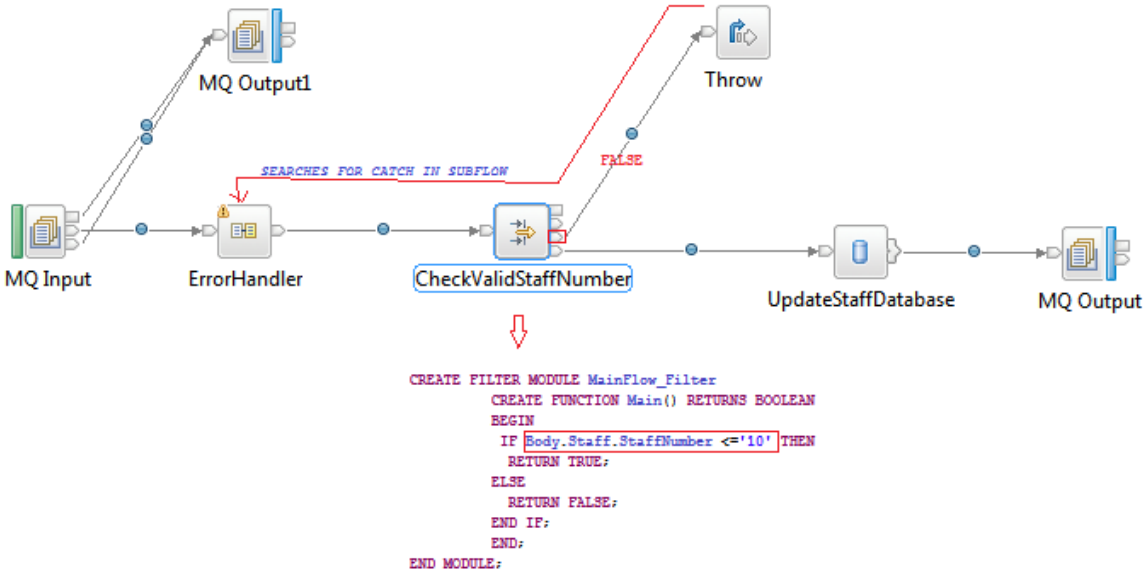
1.INPUT



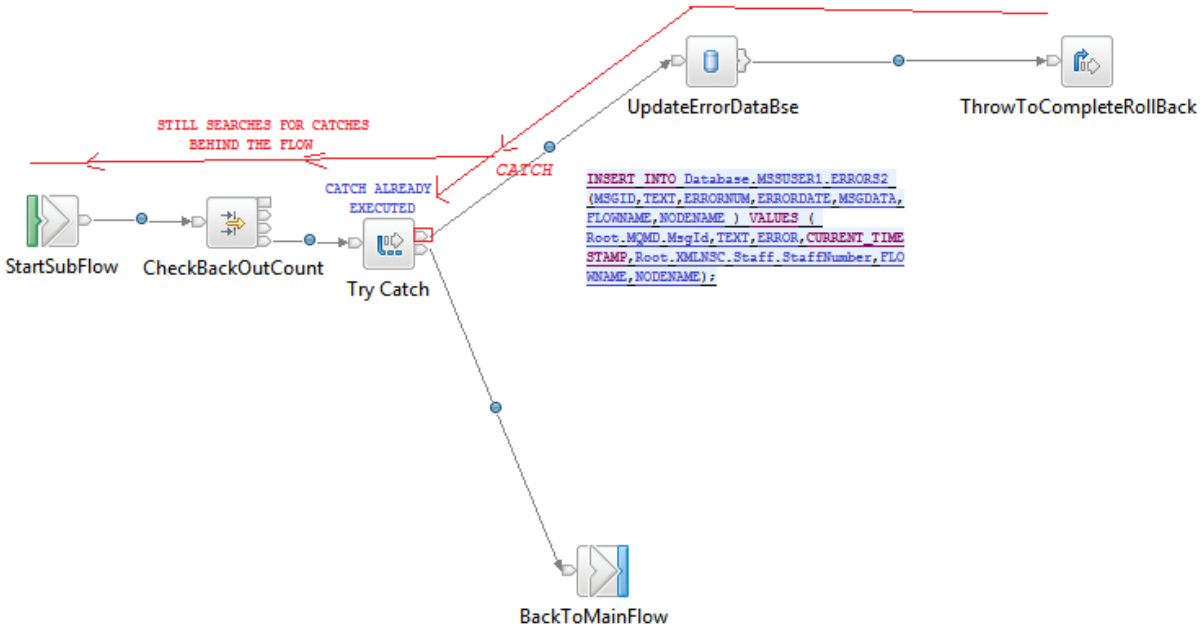
2. SUB-FLOW



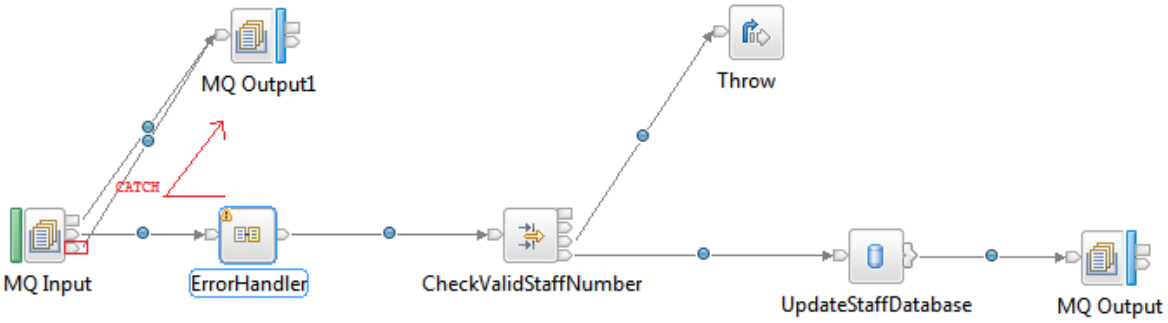
3.BACK TO MAIN FLOW



4.UPDATES THE ERROR IN THE DATABASE WITHIN SUBFLOW AND ROLLS BACK BY THROW



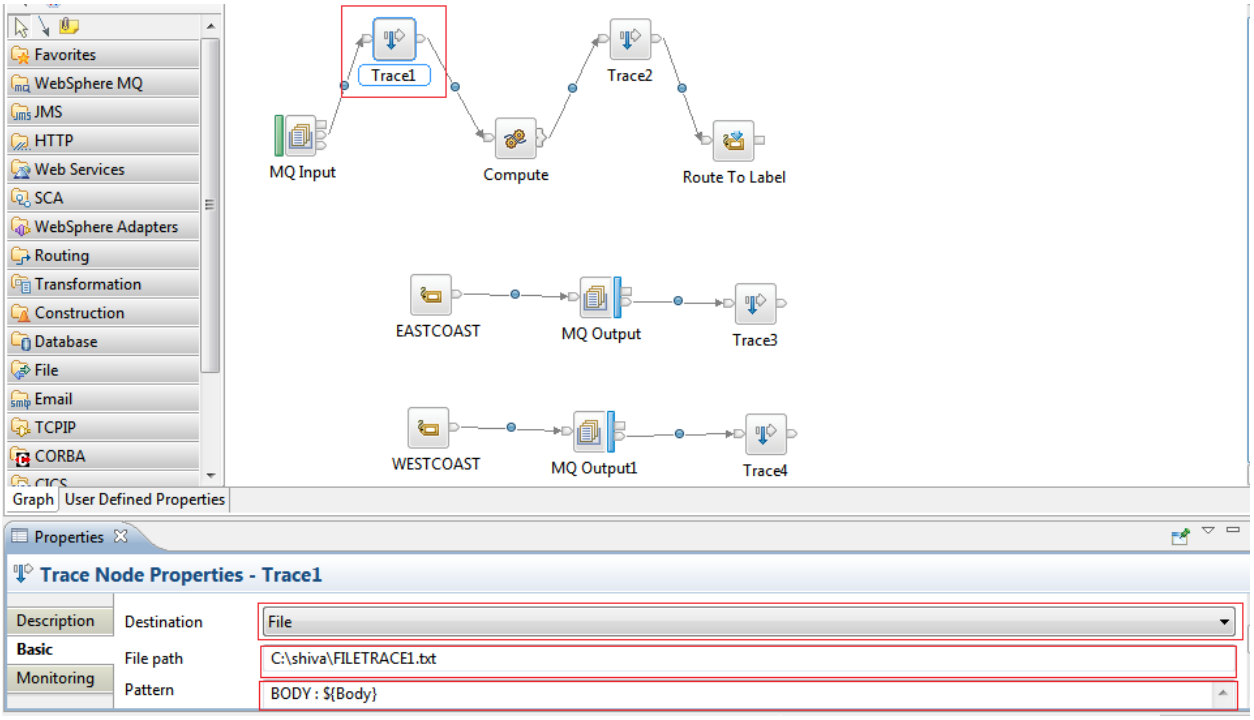
5.RETURNS TO CATCH TERMINAL OF INPUT UPON ROLLBACK AND UPDATES OUTPUT1 QUEUE WITH THE FAILED MESSAGE



NOTE: IF THE CONDITION *Body.Staff.StaffNumber <='10'* IN CheckValidStaffNumber NODE IS SATISFIED THEN UpdateStaffDatabase node hits and computes accordingly.

DEBUGGING USING TRACE SCENARIO

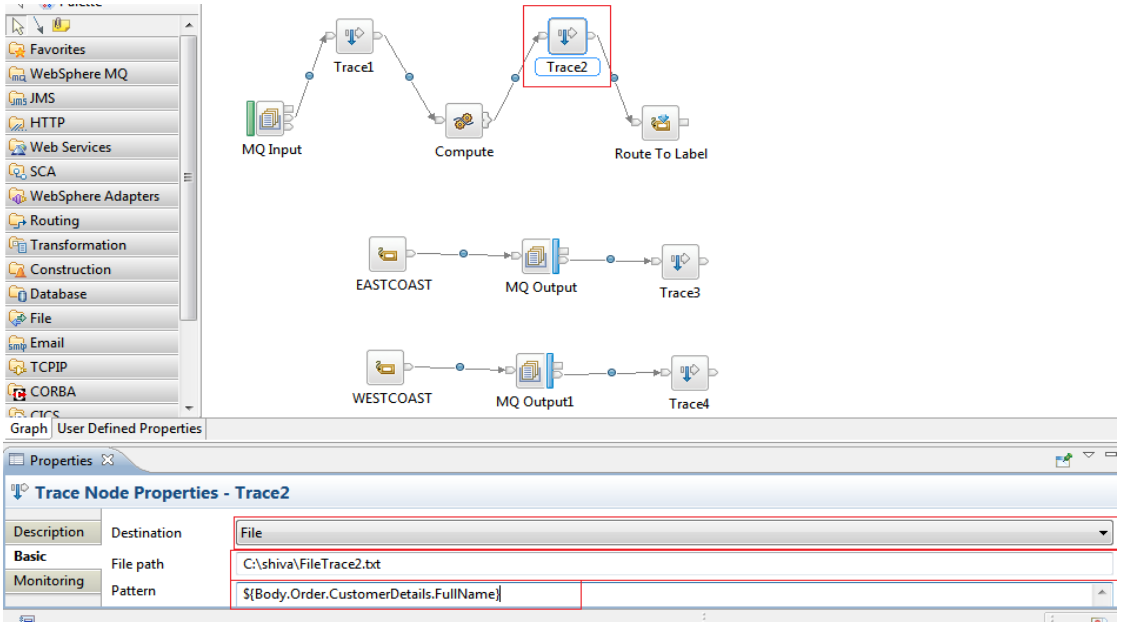
1.Trace node – Destination property:
File: If we select "File" in Destination property of Trace node, the behavior of the message flow will be recorded in a file (file name is given in "Filepath").${Body}: It writes only the message data.



2.Trace node – Destination property:

File: If we select "File" in Destination property of Trace node, the behavior of the message flow will be recorded in a file (file name is given in "Filepath").

${Body.Order.CustomerDetails.FullName}: It records the data in Body.Order.CustomerDetails.FullName (In one message fields will record in to file)



3.Trace node – Destination property:

User Trace: This is also same as LocalErrorLog.If we select User Trace as destination the log files will store in "C:\Documents and Settings\All Users\ApplicationData\IBM\MQSI\common\log".

USER TRACE COMMANDS:-

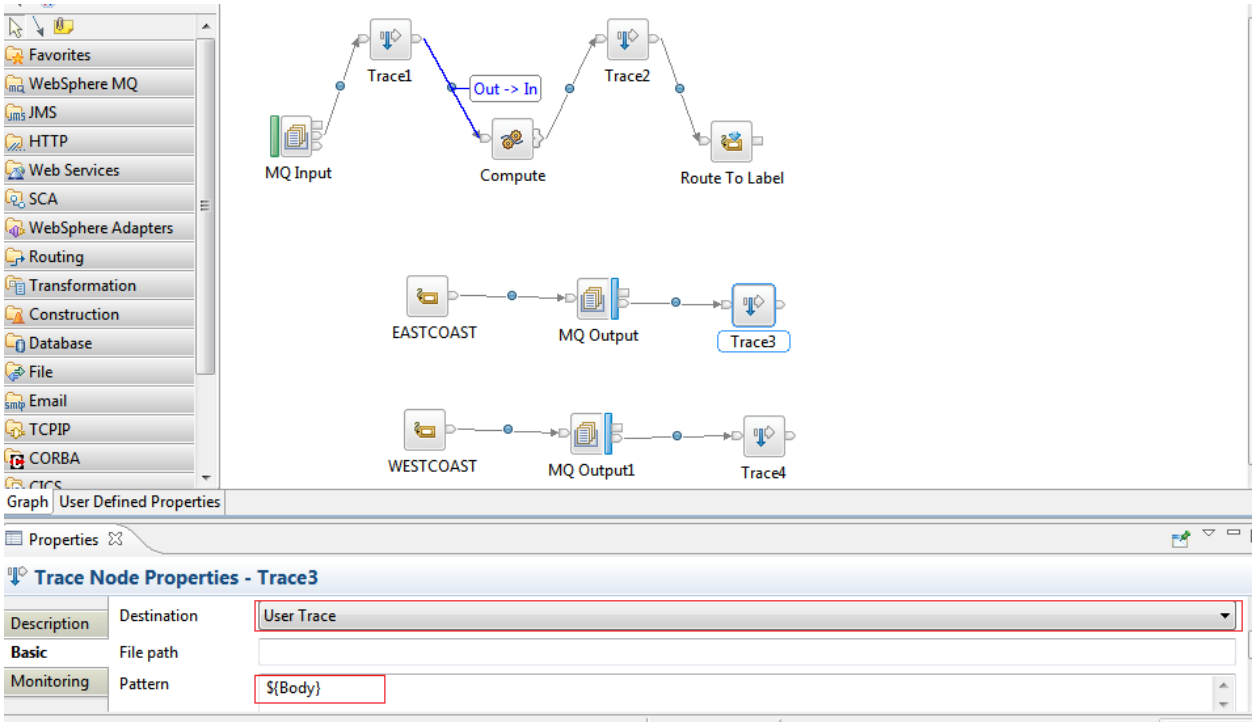
On/Off the TRACE:
mqsichangetrace *BrokerName* -n on/off -e default

Change the TRACE Mode:
mqsichangetrace *BrokerName* -u -e default -l debug/normal

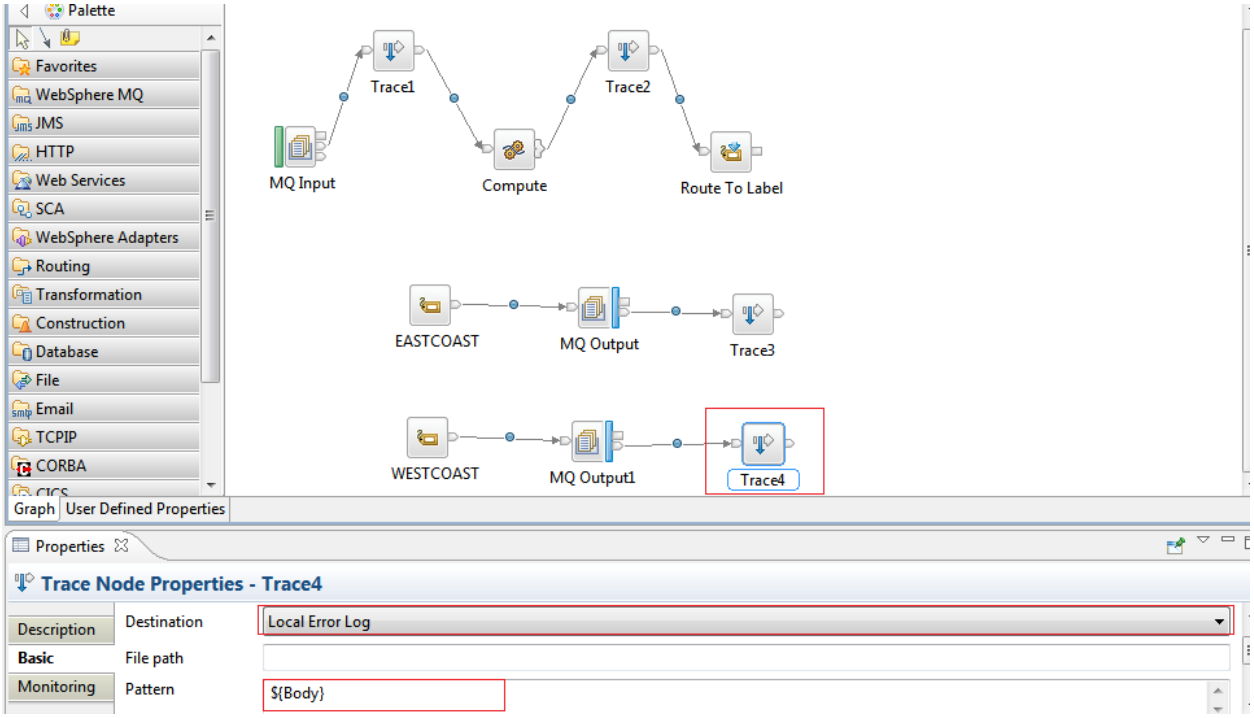
Read the TRACE to some file:
mqsireadlog *BrokerName* -u -e default -o outputfile.log

Format the TRACE LOG:
mqsiformatlog -i inputfile.txt -o outputfile.log

-where default is the execution group name.

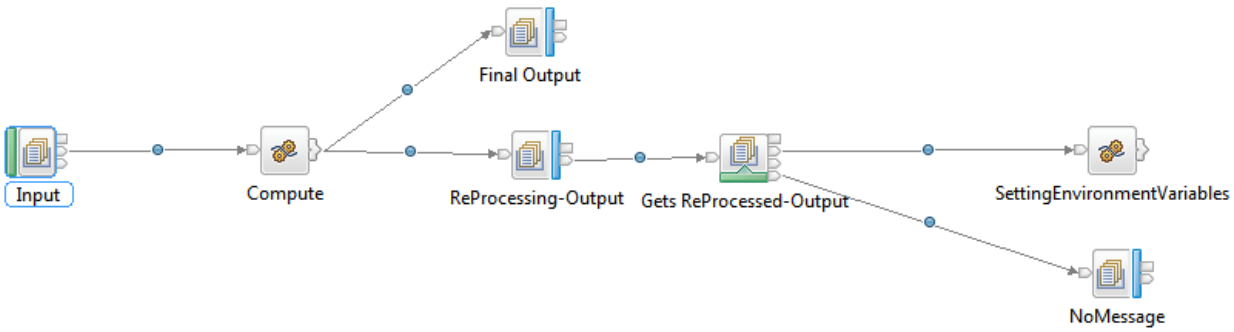


4.Trace node – Destination property:
LocalErrorLog: If we select LocalErrorLog as Destination then the behavior of MessageFlow will be recorded in log files.That log files will be stores "C:\DocumentsandSettings\AllUsers\ApplicationData\IBM\MQSI\common\log".

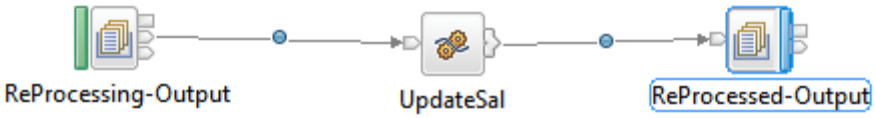


SCENARIO ON USAGE OF TRANSACTION MODE(sync -point),CORRELATIONID,ENVIRONMENT VARIABLES AND GET NODE

Message Flow 1(Main):



Message Flow 2(Sub):

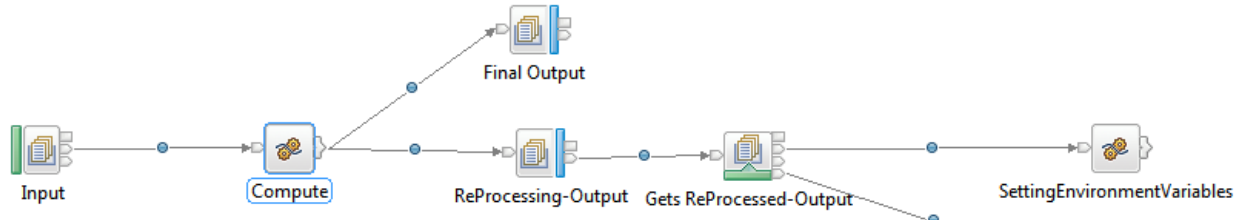


INPUT:

```
<EmployeeDetails>
<Employee>
<esal>10</esal>
</Employee>
<Employee>
<esal>20</esal>
</Employee>
<Employee>
<esal>30</esal>
</Employee>
<Employee>
<esal>40</esal>
</Employee>
<Employee>
<esal>50</esal>
</Employee>
</EmployeeDetails>
```

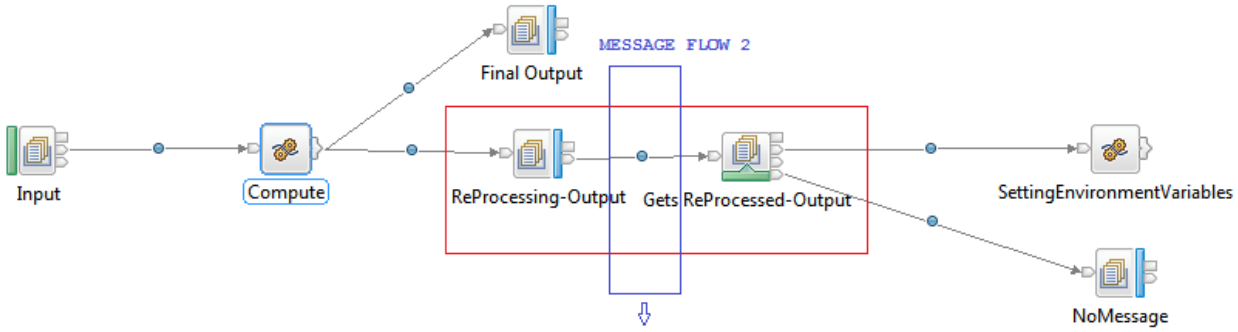
(1)STANDARD INPUT INITIALIZATION:

```
DECLARE ptr REFERENCE to InputRoot.XMLNSC.EmployeeDetails.Employee[1];
DECLARE i INTEGER 1;
```




```
STANDARD INPUT PROCESSING
WHILE LASTMOVE(ptr) DO
SET
OutputRoot.MQMD.CorrelId=CAST
('000000000000000000000000000
0000000000000000000' ||CAST
(i as CHARACTER) AS BLOB);
SET i= i+1;
SET
OutputRoot.XMLNSC.Employee =
ptr;
move ptr NEXTSIBLING;
Propagate to TERMINAL 'out1';
END WHILE;
SET
OutputRoot.XMLNSC=Environment
;
RETURN TRUE;
```

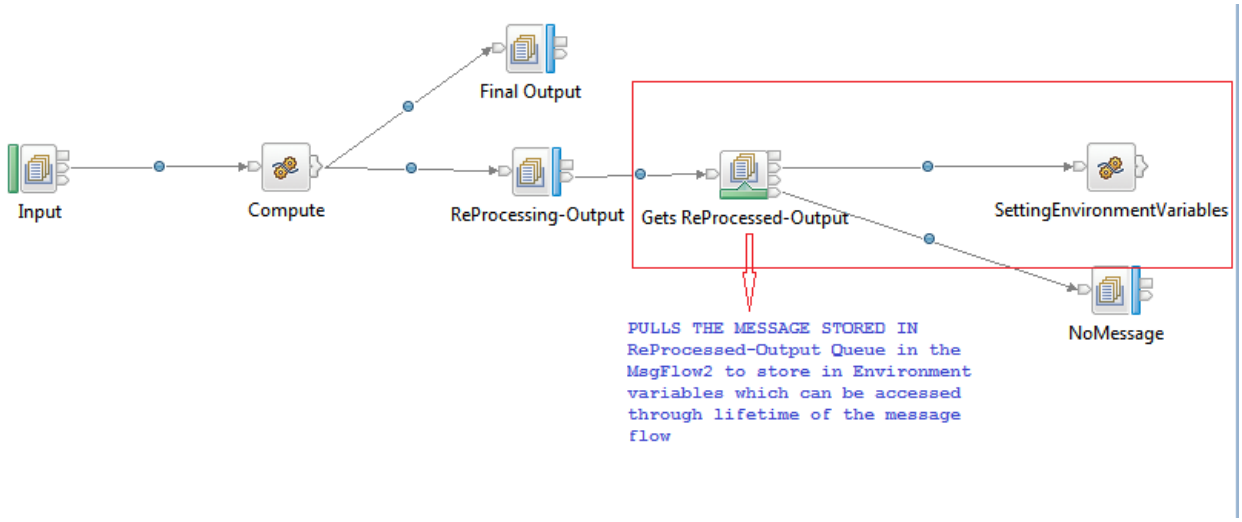
(2)MESSAGE FLOWS THROUGH 'out1' TERMINAL AND IN TURN LINKS TO MESSAGEFLOW 2(NOT SUBFLOW) BY HAVING SAME QUEUE NAMES TO FURTHER PROCESS THE STANDARD INPUT MESSAGE.



(3)RE-PROCESSING THE STANDARD INPUT.


```
DECLARE a INTEGER
InputRoot.XMLNSC.Employee.
esal;
SET
OutputRoot.XMLNSC.Employee
.esal= a + 100;
RETURN TRUE;
```

(4)GETS THE RE-PROCESSED INPUT (*say,a+100*) INTO GETNODE,WHOSE PURPOSE IS TO GET THE INPUT IN THE MIDDLE OF THE FLOW AND FINALLY SETS TO ENVIRONMENT VARIABLES.



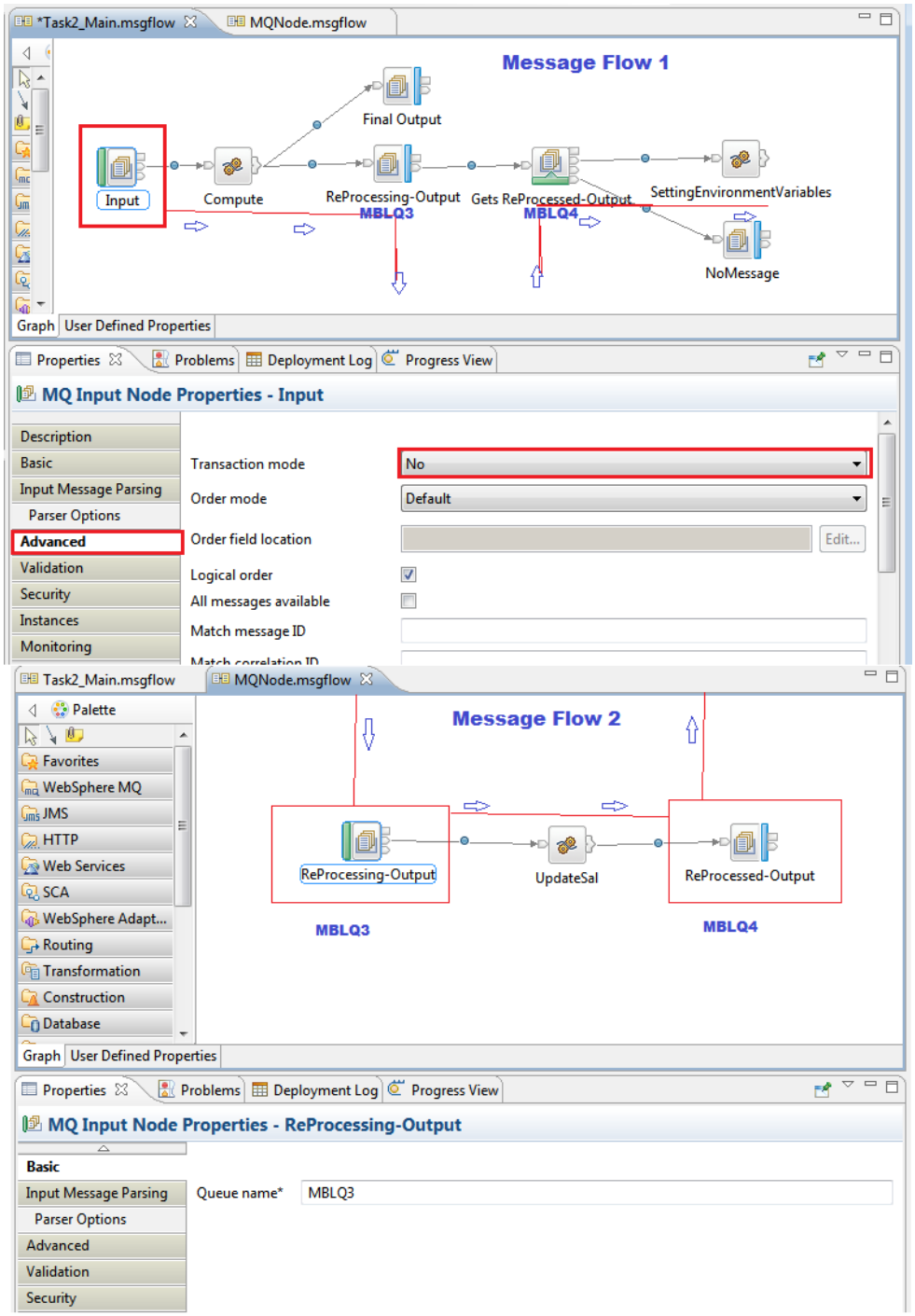
CODE TO SET THE RE-PROCESSED INPUT TO ENVIRONMENT VARIABLES:

```
CREATE FIELD Environment.EmployeeDetails;
  DECLARE envPtr REFERENCE to Environment.EmployeeDetails;
  CREATE LASTCHILD OF envPtr AS envPtr NAME 'Employee';
  set envPtr = InputRoot.XMLNSC.Employee;
  RETURN TRUE;
```

-ONCE THE BOTH THE FLOWS END,THE OUTPUT MESSAGE CREATED IN ENVIRONMENT TREE WILL BE PROPAGATED TO OUT TERMINAL(*Final Output Node in the first step*).HERE THE POINT IS, JUST BECAUSE THESE ARE ENVIRONMENT VARIABLES,WE COULD EASILY ABLE TO RE-GET THE INPUT RE-PROCESSED MESSAGE IN TO THE OUTPUT STRUCTURE WITHOUT SCOPING THEM TILL NEXT NODE(*localEnvironment*) USING SET OutputRoot.XMLNSC=Environment;

USE OF TRANSACTION MODE:

BY PUTTING THE INPUT NODE ADVANCED PROPERTY,TRANSACTION MODE TO "NO" THE WHOLE FLOW WILL NOT BE TREATED AS A TRANSACTION AND WILL NOT TERMINATE EVEN THERE IS SHIFT FROM Message Flow 1(Main) TO Message Flow 2(Sub) DUE TO SAME QUEUE NAMES.



EXCEPTION CASES : IF TRANSACTION MODE IS 'YES' then GETNODE(Gets ReProcessed-Output) Terminates through 'No Message' terminal and IF 'Get by correlation ID' is unchecked in request property of GETNODE.

IMPORTANT NOTE:-
When working with MQGET node for our surprise we observed that when the MQOutput node name and the MQGet node names are similar the message is appearing in the MQGet node Queue even though the Queue names configured to both the nodes are different.

MQGet node can be used anywhere in a message flow to store message temporarily.Using Compute node we copied the message id of the incoming message to correlation id of output message which helps to retrieve the temporarily message from the queue.

By using the statement like **SET OutputRoot.MQMD.CorrelId = InputRoot.MQMD.MsgId**

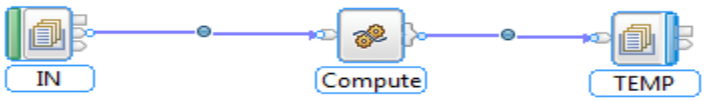
This node name activity was a bug in this WMB V 8.0.0.1 this was being cleared when the same PI is being deployed in the IIB9 it was generating the error if any two nodes have the similar name with in the flow.

**** Sample Scenario For combining the Temporary message in the Queue with the new message that is obtained in the middle of the flow using the MQGet node ***

we know MQGet node can be used anywhere in a message flow to store message in intermediate state & afterwords in another thread of flow we can aggregate this temporary result to form final output (Getting message by correl/message id..)

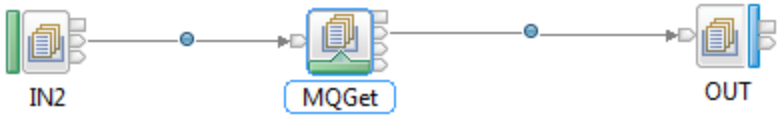
Following example illustrates how an intermediate message can be combined to incoming message to form final message.

First Flow : IN (MQInput) ,Compute, TEMP (MQOutput)

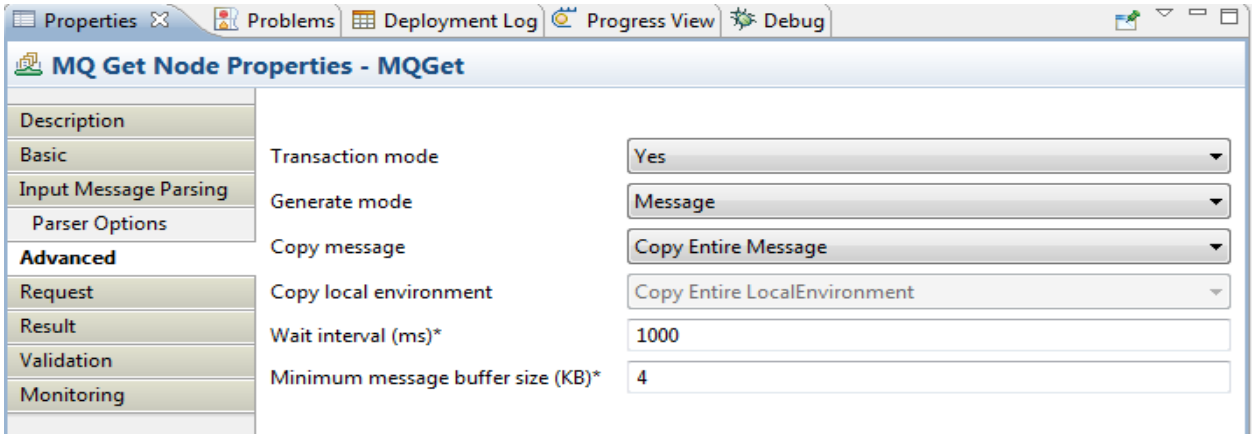


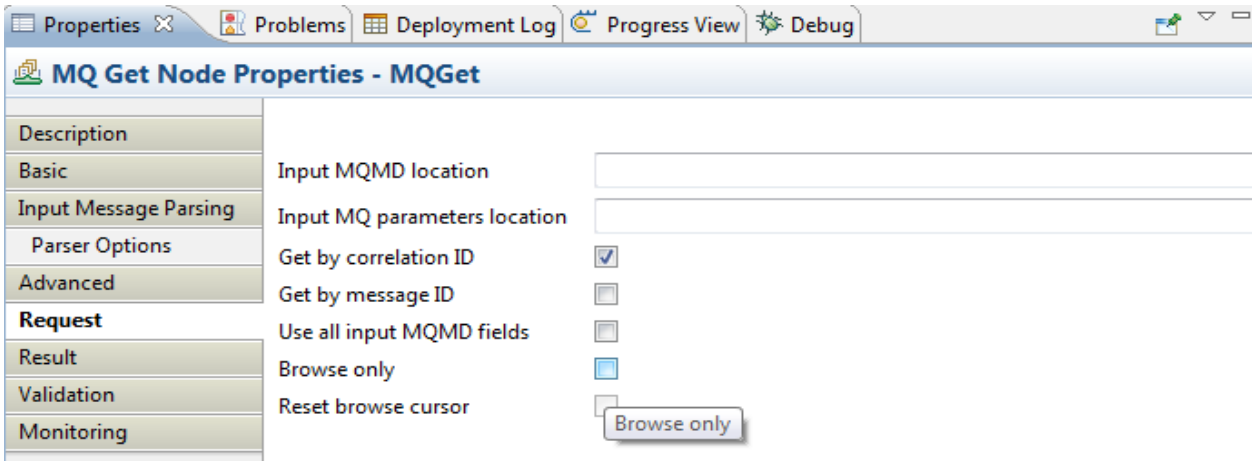
Here the Compute node copies the message id of the incoming message to correlation id of output message, so that we can retrieve this intermediate message based on correlation id.
SET OutputRoot.MQMD.CorrelId = InputRoot.MQMD.MsgId;

Second Flow : IN2 (MQInput),MQGet (MQGet), OUT (MQOutput)



Set the properties of MQGet Node as

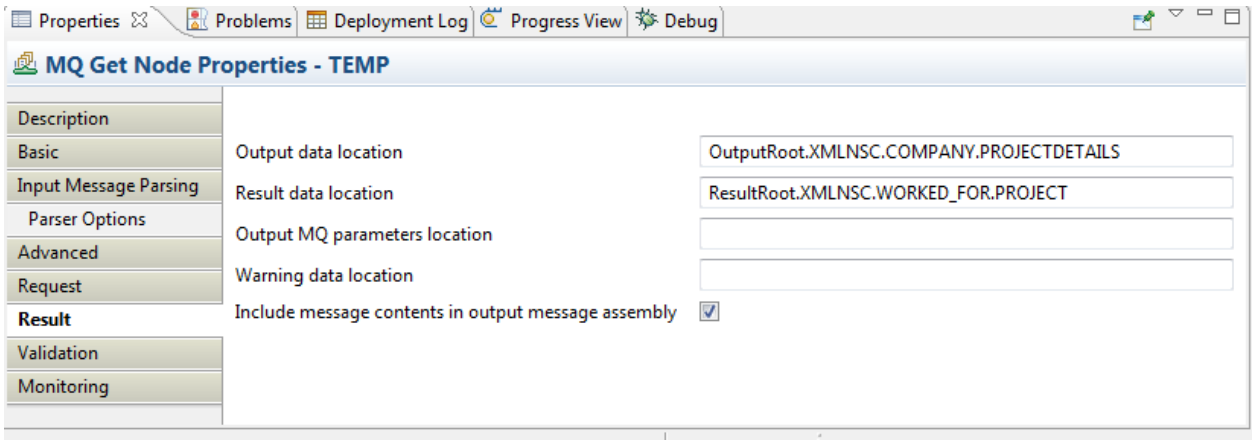




INPUT MESSAGE FOR FIRST FLOW IN IS :-
<WORKED_FOR><PROJECT>IRCTC_MB_NEW</PROJECT></WORKED_FOR>

INPUT MESSAGE FOR SECOND FLOW IN2 IS:-
<COMPANY>
<NAME>NEW MB WAVES</NAME>
<PROJECTDETAILS>----------</PROJECTDETAILS>
</COMPANY>

Motto:-
Now i am trying to combine both the messages from IN,IN2 and make a single output message so for this purpose i will make the MQGet Node properties be set as



Give the queue name for Temp Node in First flow and MQGet node in second flow as similar (for example let it be as :-OUT)

The final result obtained at the end of the flow is

<COMPANY>
<NAME>NEW MB WAVES</NAME>
<PROJECTDETAILS>IRCTC_MB_NEW</PROJECTDETAILS>
</COMPANY>

IMPORTANT SNIPPETS FOR JAVA COMPUTE NODE:-

1)
create an ESQL module in the **mqsi schema**. The module is then assigned to a new Compute node by using the setComputeExpression() method:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
ESQLModule module = new ESQLModule();
module.setBrokerSchema("mqsi");
module.setEsqlMain("MyESQLMain");
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(module);
mf1.addNode(compNode);
```

2)
create an ESQL module in the **default schema**. The setBrokerSchema() method is not required.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
ESQLModule module = new ESQLModule();
module.setEsqlMain("MyESQLMain");
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(module);
mf1.addNode(compNode);
```

3)
Discover an ESQL module from within an ESQL file by using the getEsqlModules() method. You can then use the ESQL module to set the compute expression on a Compute node by using the setComputeExpression() method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
File esql = new File("FileBatchProcessingSample_Branch.esql");
ESQLFile esqlFile = new ESQLFile(esql);
Vector<ESQLModule> esqlModules = esqlFile.getEsqlModules();
ComputeNode compNode = new ComputeNode();
compNode.setNodeName("My Compute Node");
compNode.setComputeExpression(esqlModules.get(0));
mf1.addNode(compNode);
```

4)
To **load** a message flow into memory, create a File object and use the read() method of the FlowRendererMSGFLOW, which makes the message flow available for your Java code. The read() method takes the message flow project containing the required message flow file and the relative path to the message flow file from this project.

```
File msgFlow = new File("../mqsi/main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
```

5)

To **rename** a node, you must first access the required node. You can access the node by using the existing name of the node and the getNodeByName() method. You can then rename the node by using the setNodeName() method, which takes the new node name as a parameter.

For example:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Node mqinNode = mf1.getNodeByName("My Input Node");
mqinNode.setNodeName("New Input Node");
```

6)

Example to show how to **add** a new built-in node:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
mf1.addNode(mqinNode);
```

Example to show how to add a new subflow node to a message flow:

1. A new subflow node is created and assigned to object sfNode.
2. The subflow node name is set to My Sub Flow Node.
3. The subflow node is linked to the subflow message flow by using the setSubFlow() method.
4. The new subflow node is added to the message flow held in object mf1.

The subflow can be stored in a .msgflow format:

-
```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
File subFlow = new File("subflow.msgflow");
MessageFlow sub1 = FlowRendererMSGFLOW.read(subFlow);
SubFlowNode sfNode = new SubFlowNode();
sfNode.setNodeName("My Sub Flow Node");
sfNode.setSubFlow(sub1);
mf1.addNode(sfNode);
```

The subflow can be stored in a .subflow format:

-
```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
sFile subFlow = new File("subflow.subflow");
MessageFlow sub1 = FlowRendererMSGFLOW.read(subFlow);
```

```
SubFlowNode sfNode = new SubFlowNode();
sfNode.setNodeName("My Sub Flow Node");
sfNode.setSubFlow(sub1);
mf1.addNode(sfNode);
```

7)

Set the **position** of a node on the canvas by using the setLocation() method of the node object. The following example sets the position of a new MQOutput node to coordinates x=300 pixels, y=100 pixels:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQOutputNode mqoutNode = new MQOutputNode();
mqoutNode.setLocation(300, 100);
```

8)

Copy a built-in or subflow node by using the clone() method. In the following example, a new MQInput node mqinNode is created and properties on the node are set. A new MQInput node mqinNode1 is then created by copying mqinNode by using the clone() method. When the node is copied, the node properties are also copied:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
MQInputNode mqinNode1 = (MQInputNode) mqinNode.clone();
mqinNode1.setNodeName("Copy of My Input Node");
mf1.addNode(mqinNode1);
```

9)

Remove a node you must first get the required node from the message flow object. In the following example, the getNodeByName() method is used to get the required node from message flow object mf1. The node is then removed by using the removeNode() method. When a node is removed, any connections to or from the node are also removed:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Node mqinNode = mf1.getNodeByName("My Input Node");
mf1.removeNode(mqinNode);
```

10)

Example to shows how to **connect** two built-in nodes:

- A MQInput node and a Collector node are created.
- The getInputTerminal() method is used to create a dynamic Input terminal called NEWIN on the Collector node.
- The Input terminal is connected to the Output terminal of the MQInput node by using the

connect() method of the message flow object mf1.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("INPUTQ");
CollectorNode colNode = new CollectorNode();
colNode.getInputTerminal("NEWIN");
mf1.connect(mqinNode.OUTPUT_TERMINAL_OUT, colNode.getInputTerminal("NEWIN"));
```

Example to show how to connect a subflow node to a built-in node.

The main message flow, main.msgflow, and a Compute node in the main message flow are loaded into memory.

- The subflow message flow, subflow.msgflow, and the subflow node in the main message flow are loaded into memory.
- The setSubFlow() method of the subflow node is used to link the subflow message flow sub1 to the subflow node sfNode.
- The getOutputTerminal() method is used to get the Process terminal of the subflow node. The connect() method of the message flow object is used to connect this terminal to the Input terminal of the Compute node.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
ComputeNode compNode = (ComputeNode)mf1.getNodeByName("My Compute Node");
File subFlow = new File("subflow.msgflow");
MessageFlow sub1 = FlowRendererMSGFLOW.read(subFlow);
SubFlowNode sfNode = (SubFlowNode)mf1.getNodeByName("My Subflow Node");
sfNode.setSubFlow(sub1);
mf1.connect(sfNode.getOutputTerminal("Process"), compNode.INPUT_TERMINAL_IN);
```

11)

Example to show how to add a user-defined node to a message flow and connect it to a built-in node:

1. An MQInput node is created and added to the message flow.
2. A user-defined node is created by using the GenericNode class and is added to the message flow object.
3. The static output terminal of the MQInput is assigned to the variable outputTerminal.
4. The input terminal of the user-defined node is assigned to the variable inputTerminal by using the getInputTerminal() method with the known terminal name In.
5. The nodes are connected by using the connect() method.
6. The final section of code shows that the input node is now available for use in the message flow, by using the getInputTerminals() method of the user-defined node instance.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
```

```
MQInputNode mqinNode = new MQInputNode();
mqinNode.setNodeName("My Input Node");
mqinNode.setQueueName("IN");
mf1.addNode(mqinNode);

GenericNode myNode = new GenericNode("MyUserDefinedNode");
myNode.setNodeName("MyNode");
mf1.addNode(myNode);

OutputTerminal outputTerminal = mqinNode.OUTPUT_TERMINAL_OUT;
InputTerminal inputTerminal = myNode.getInputTerminal("In");
mf1.connect(outputTerminal, inputTerminal);

InputTerminal[] inputTerminals = myNode.getInputTerminals();
System.out.println("Input terminals on my node:");
for (int i = 0; i < inputTerminals.length; i++)
 {
   InputTerminal inputTerminal = inputTerminals[i];
   System.out.println(inputTerminal.getName());
}
```

12)

Disconnect two nodes by using the disconnect() method of the message flow object. You must provide this method with the names of the terminal instances that you want to disconnect.

For example:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
MQInputNode mqinNode = (MQInputNode)mf1.getNodeByName("My Input Node");
MQOutputNode mqoutNode = (MQOutputNode)mf1.getNodeByName("My Output Node");
mf1.disconnect(mqinNode.OUTPUT_TERMINAL_OUT, mqoutNode.INPUT_TERMINAL_IN);
```

13)

Example to show how to create a UDP and add it to a message flow:

1. A UDP called Property1 is created in parameter group Group1. The data type of the UDP is defined as a string and the UDP is given the default value Hello World!
2. The UDP is then added to the message flow by using the addFlowProperty() method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
UserDefinedProperty udp = new UserDefinedProperty("Group1", "Property1",
UserDefinedProperty.Usage.MANDATORY, UserDefinedProperty.Type.STRING, "Hello World!");
mf1.addFlowProperty(udp);
```

UDPs in a message flow are discovered by using the getFlowProperties() method on the message flow. The setName() method is then used to set the name of the first UDP to Property3:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
Vector<FlowProperty> flowProperties = mf1.getFlowProperties();
flowProperties.get(0).setName("Property3");
```

14)

Rename a message flow by using the setName() method. In the following example, a message flow file called main.msgflow is renamed to mainGenerated.msgflow:

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
mf1.setName(mf1.getName()+"Generated");
```

15)

Add and update rows on the filter table of a Route node.

Adding a new row

The following example shows you how to add a new row to a filter table by using the createRow() method:

1. The message flow and the Route node are loaded into memory.
2. The filter table of the Route node is loaded into memory by using the getFilterTable() method of the RouteNode object.
3. A new filter table row is created by using the createRow() method.
4. The value of the filter pattern property on this new row is set to value="123" by using the setFilterPattern() method.
5. The routing output terminal property is set to NEWOUT by using the setRoutingOutputTerminal() method.
6. The new row is then added to the filter table by using the addRow() method.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
RouteNode routeNode = (RouteNode)mf1.getNodeByName("My Route Node");
RouteNode.FilterTable filterTable = (RouteNode.FilterTable)routeNode.getFilterTable();
RouteNode.FilterTableRow newRow = filterTable.createRow();
newRow.setFilterPattern("value=\"123\"");
newRow.setRoutingOutputTerminal("NEWOUT");
filterTable.addRow(newRow);
```

Updating a row

The following example shows you how to update rows on the filter table of a Route node.

1. The message flow, Route node, and filter table of the Route node are loaded into memory.
2. The rows of the filter table are loaded into memory by using the getRows() method.
3. The filter pattern property of the first row of the filter table is set to value2="456".
4. The routing output terminal property of the first row of the filter table is set to NEWOUT2.

```
File msgFlow = new File("main.msgflow");
MessageFlow mf1 = FlowRendererMSGFLOW.read(msgFlow);
RouteNode routeNode = (RouteNode)mf1.getNodeByName("My Route Node");
```

```
RouteNode.FilterTable filterTable = (RouteNode.FilterTable)routeNode.getFilterTable();
Vector<RouteNode.FilterTableRow> filterTableRows = filterTable.getRows();
filterTableRows.get(0).setFilterPattern("value2=\"456\"");
filterTableRows.get(0).setRoutingOutputTerminal("NEWOUT2");
```

Working with Timer Nodes:-

TIMER NODES PRESENT IN WMB ARE :
1.TIMER NOTIFICATION NODE,
2.TIMER CONTROL.

WE CAN USE TIMER NOTIFICATION NODE INDEPENDENTLY BUT WHEN WE USE TIMER CONTROL NODE WE SHOULD USE TIMER NOTIFICATION NODE.

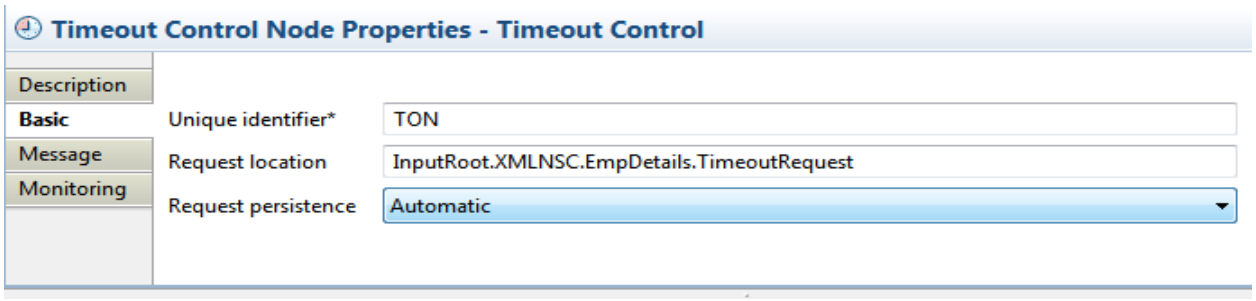
WE CAN USE TIMER NOTIFICATION NODE WITH MULTIPLE TIMER CONTROL NODES.(i.e ONE TIMER NOTIFICATION NODE AND ONE OR MORE TIMER CONTROL NODES.)

WHEN WE USE TIMER NOTIFICATION WITH TIMER CONTROL "UNIQUE IDENTIFIER" SHOULD BE COMMON WHICH IS COMMON PROPERTY FOR TIMER NOTIFICATION AND TIMER CONTROL NODES.

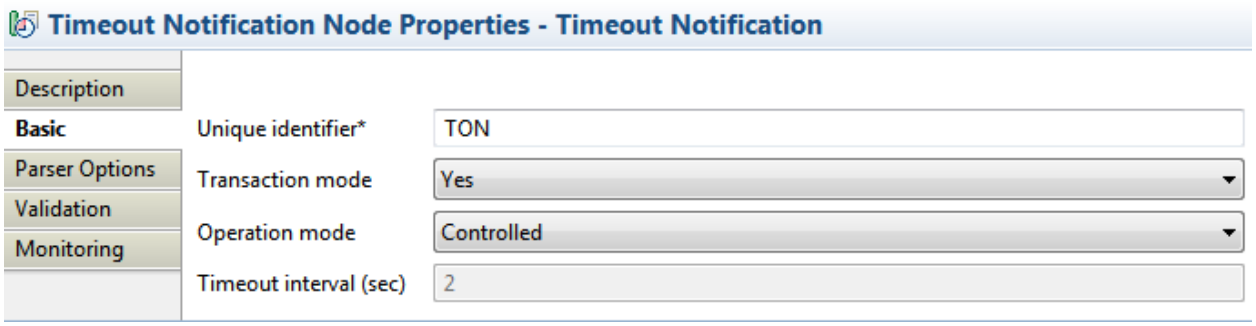
SAMPLE OF TIMER NOTIFICATION SCENARIO

Properties for the nodes:-
MQInput,MQOutput :-The properties for this nodes are as usual.

TimeOutControl:-

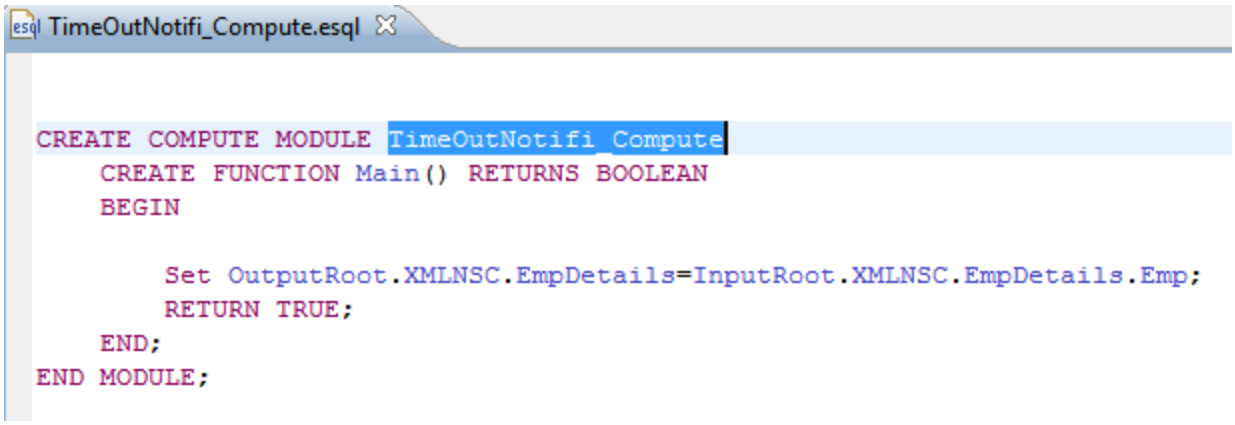


TimeOutNotification:-



Set the operation mode as controlled to establish a connection between TimeOutControl and TimeOutNotification.

Compute Node:-



Input Message:-

```
<EmpDetails>
<TimeoutRequest>
 <Action>SET</Action>
 <Identifier>ThreeTimes</Identifier>
 <StartDate>TODAY</StartDate>
 <StartTime>NOW</StartTime>
 <Interval>5</Interval>
 <Count>3</Count>
 <IgnoreMissed>TRUE</IgnoreMissed>
 <AllowOverwrite>TRUE</AllowOverwrite>
</TimeoutRequest>
 <Emp>
 <EmpName>EddieNorton</EmpName>
 <EmpID>9966</EmpID>
 <EmpAge>31</EmpAge>
 <EmpSex>M</EmpSex>
 <EmpDoj>05/26/1999</EmpDoj>
 </Emp>
 </EmpDetails>
```

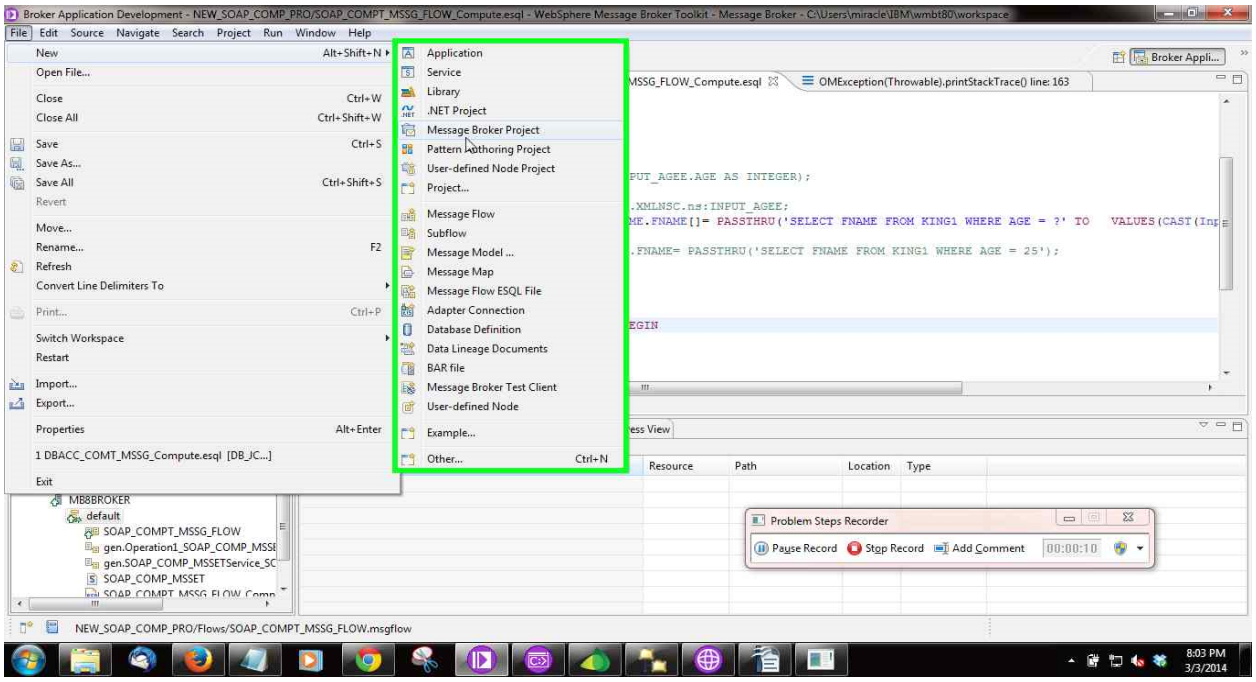
This message can also be taken as the below but the main thing is that some of the tag names must be given with same tag names to make the Timer node to have its functionality in a proper way.

```
<EmpDetails>
 <Emp>
  <Action>SET</Action>
  <Identifier>ThreeTimes</Identifier>
  <StartDate>TODAY</StartDate>
  <StartTime>NOW</StartTime>
  <Interval>5</Interval>
  <Count>3</Count>
  <IgnoreMissed>TRUE</IgnoreMissed>
  <AllowOverwrite>TRUE</AllowOverwrite>
  <EmpName>EddieNorton</EmpName>
  <EmpID>9966</EmpID>
  <EmpAge>31</EmpAge>
  <EmpSex>M</EmpSex>
  <EmpDoj>05/26/1999</EmpDoj>
 </Emp>
 </EmpDetails>
```

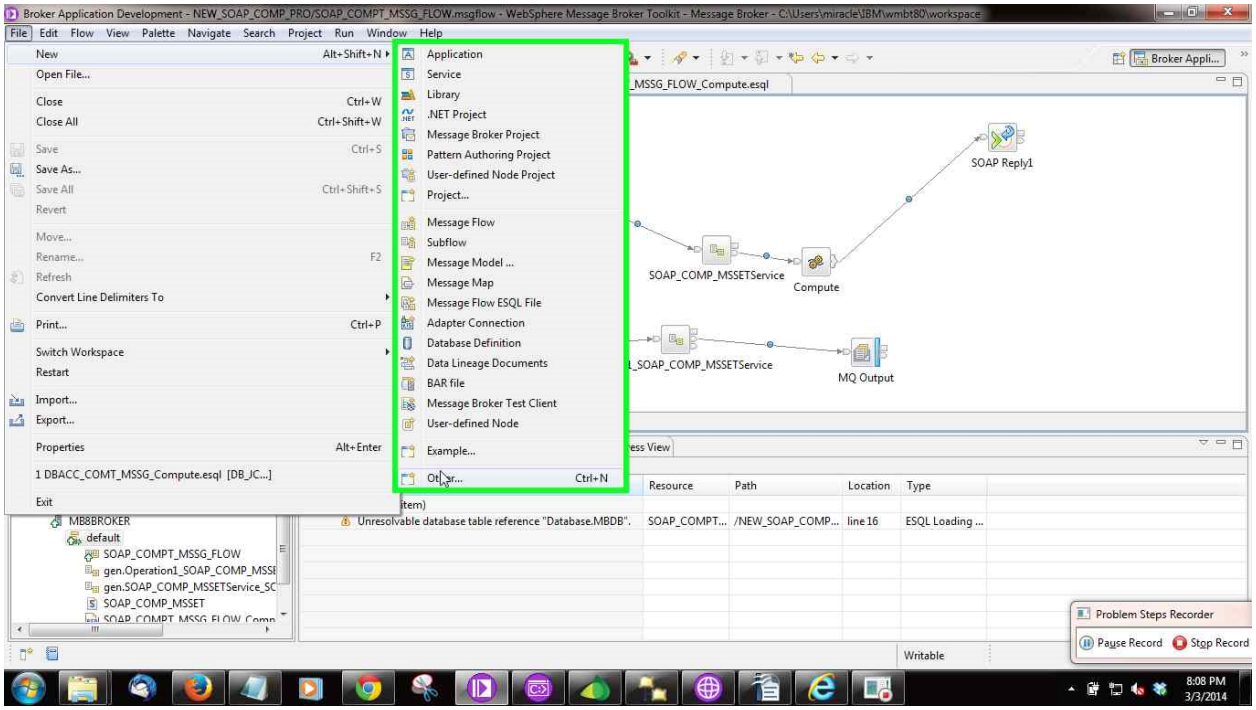
According to the above messages the message will be sent to the Output Queue for every 5 seconds and for 3 times.

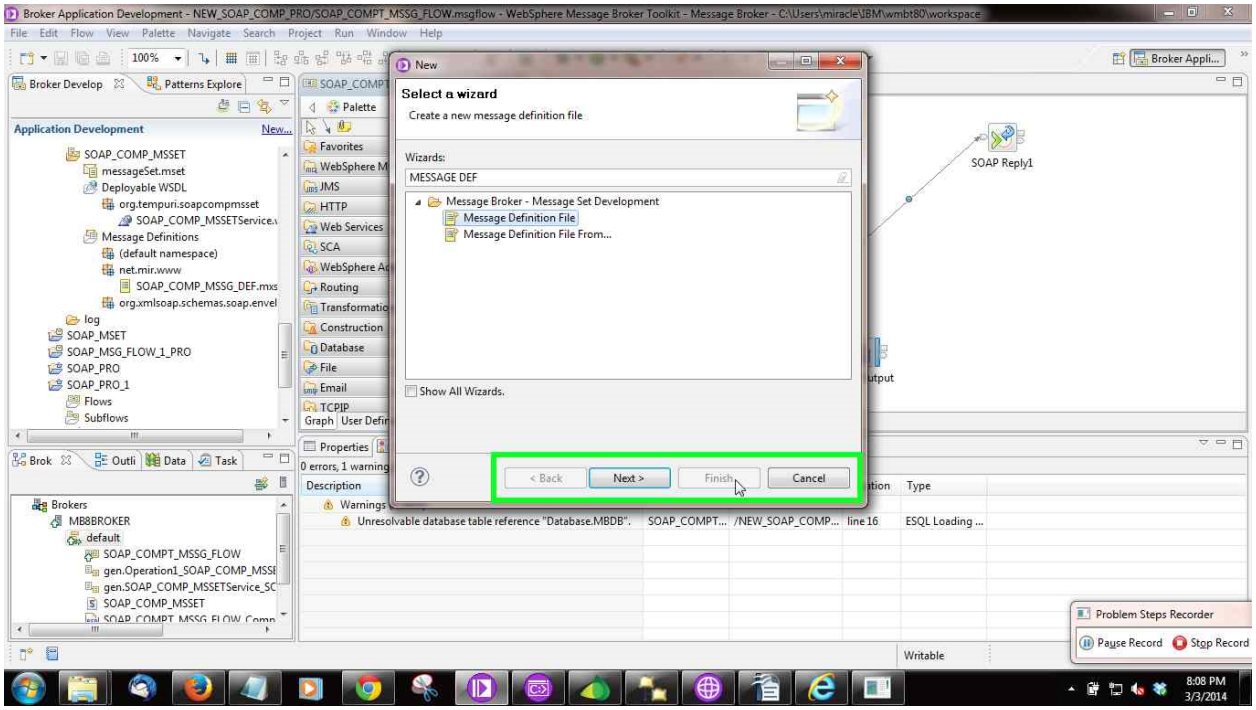
Scenerio :Acessing Database using compute node with soap web services by takeing 'AGE as input giveing FNAME of that person as output'. - by ARUN

STEP1:CREATE NEW MESSAGE BROKER PROJECT.

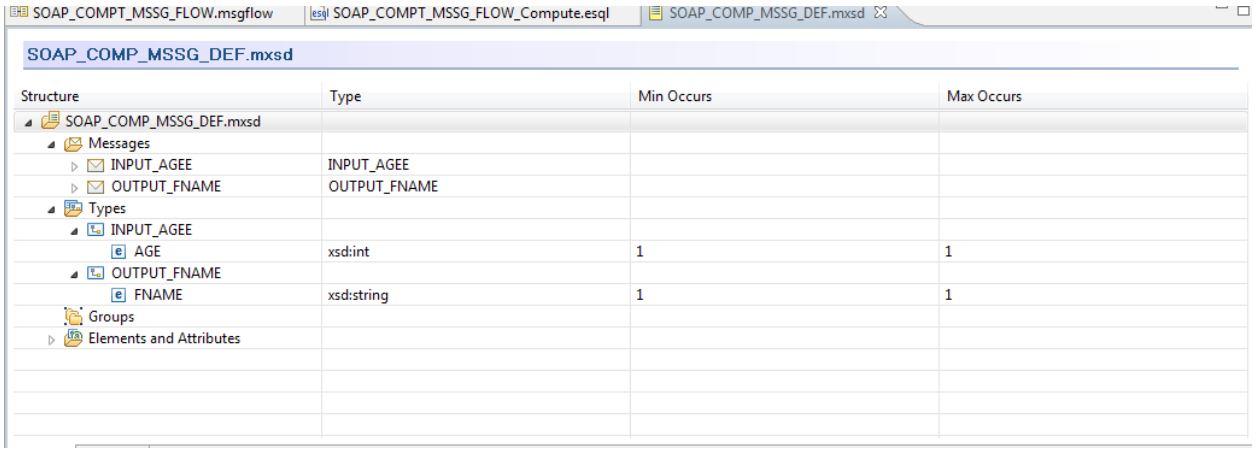


STEP2:CREATE NEW MESSAGE FLOW.

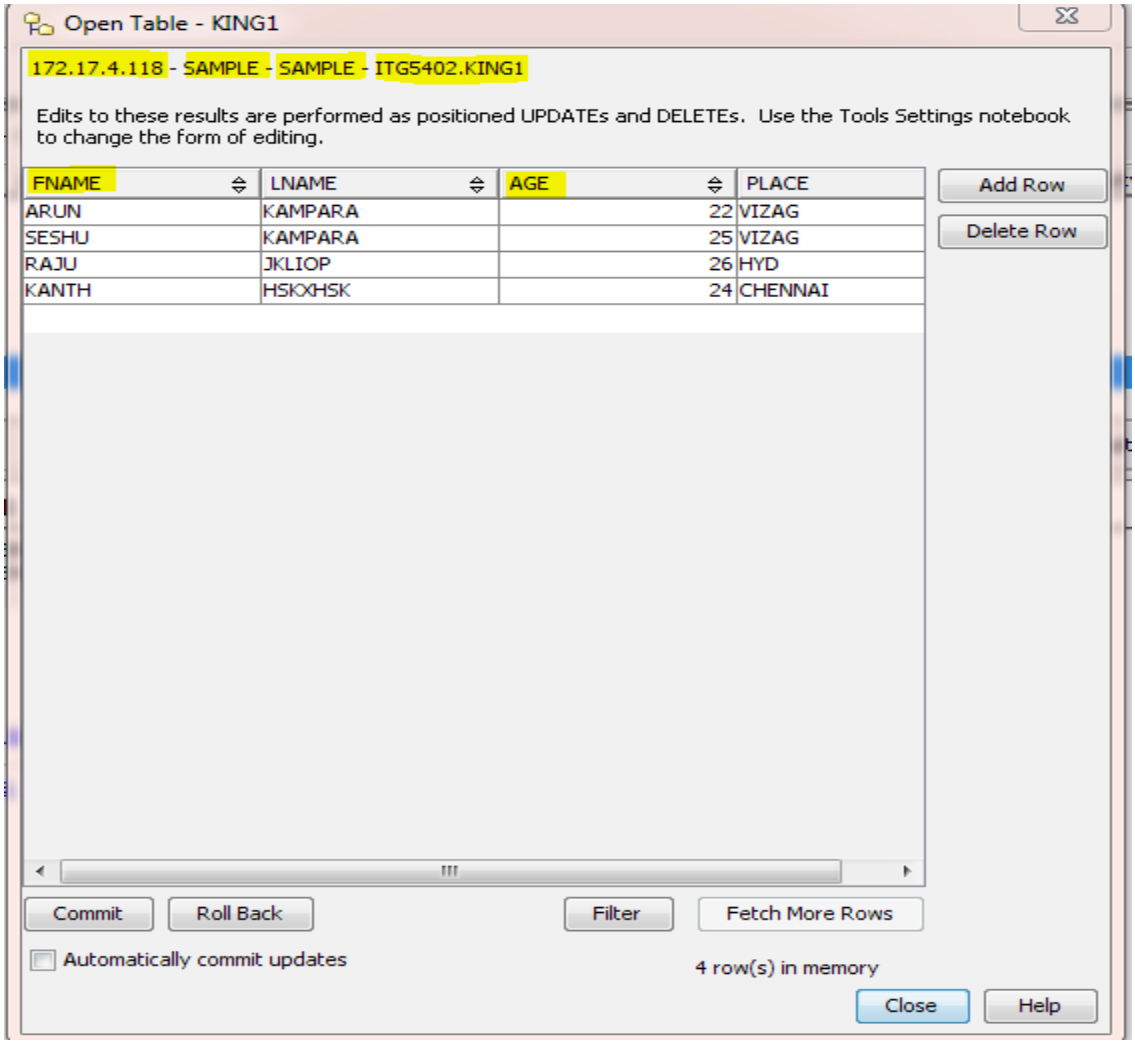




STEP3:CREATE NEW MESSAGE DEFNITION .



Creation of message set definition need database information according to current scenario.here age and fname are columns of database.observe below figure carefully



Here AGE and FNAME are columns of table .accroding to scnerio AGE act as input and FNAME act as output based on this u created message defination.

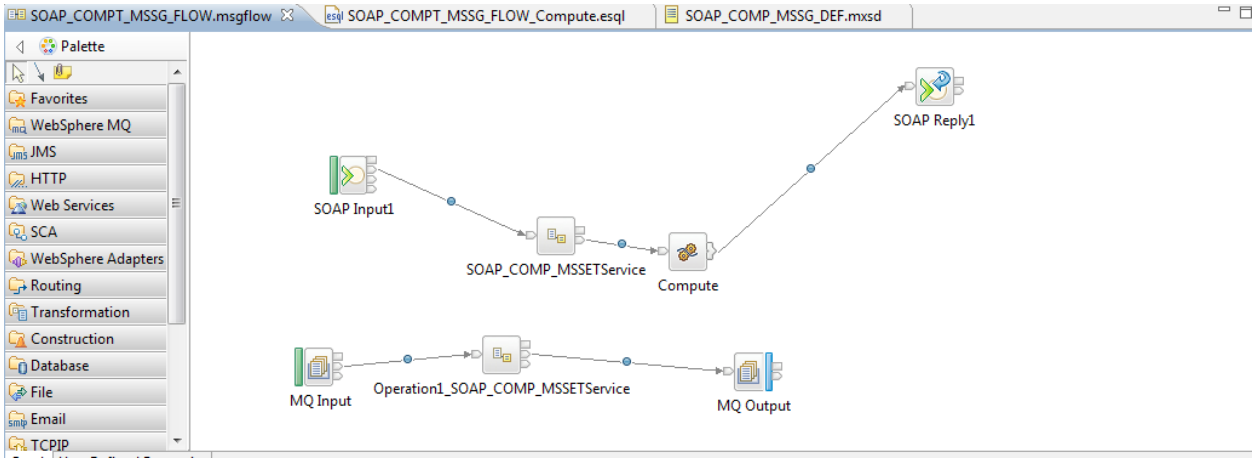
NOTE:

1.BEFORE THIS YOU NEED TO CREATE DATA SOURCE AND YOU NEED TO CONFIGURE DATASOURCE WITH THE BROKER .

2.WHEN YOU ARE CREATING WSDL FILE--
'SELECT THE SOAP WEB SERVICE' in drop down.
'TICK THE XMLNSC PARSER'
because soap return object so its difficult to take input data even though input came it works only when we are copying Input as it is to Output.(SO TICK THE XMLNSC PARSER)

MESSAGE FLOW:

COMPUTE NODE FIGURE:



COMPUTE NODE CODE:

```
DECLARE ns NAMESPACE 'http://www.mir.net';


CREATE COMPUTE MODULE SOAP_COMPT_MSSG_FLOW_Compute
	CREATE FUNCTION Main() RETURNS BOOLEAN
	BEGIN
		-- CALL CopyMessageHeaders();
		-- CALL CopyEntireMessage();

		--DECLARE A INTEGER;
```

```
            --SET A=CAST(InputRoot.XMLNSC.ns:INPUT_AGEE.AGE AS INTEGER);

            --DECLARE B REFERENCE TO InputRoot.XMLNSC.ns:INPUT_AGEE;
SET OutputRoot.XMLNSC.ns:OUTPUT_FNAME.FNAME[]= PASSTHRU('SELECT FNAME FROM
KING1 WHERE AGE = ?' TO Database.MBDB
VALUES(CAST(InputRoot.XMLNSC.ns:INPUT_AGEE.AGE AS INTEGER)));

(OR)

SET A=CAST(InputRoot.XMLNSC.ns:INPUT_AGEE.AGE AS INTEGER);

SET OutputRoot.XMLNSC.ns:OUTPUT_FNAME.FNAME[]= PASSTHRU('SELECT FNAME FROM
KING1 WHERE AGE = ?' TO Database.MBDB VALUES(A);


      --SET OutputRoot.XMLNSC.ns:OUTPUT_FNAME.FNAME= PASSTHRU('SELECT FNAME
FROM KING1 WHERE AGE = 25');

            RETURN TRUE;
      END;

      CREATE PROCEDURE CopyMessageHeaders() BEGIN
            DECLARE I INTEGER 1;
            DECLARE J INTEGER;
            SET J = CARDINALITY(InputRoot.*[]);
            WHILE I < J DO
                  SET OutputRoot.*[I] = InputRoot.*[I];
                  SET I = I + 1;
            END WHILE;
      END;

      CREATE PROCEDURE CopyEntireMessage() BEGIN
            SET OutputRoot = InputRoot;
      END;
END MODULE;
```

SEQUENCE NODE EXAMPLE

REFRENCE LINK: http://webspheremb.blogspot.in/2011/09/sequence-and-re-sequence-nodes.html

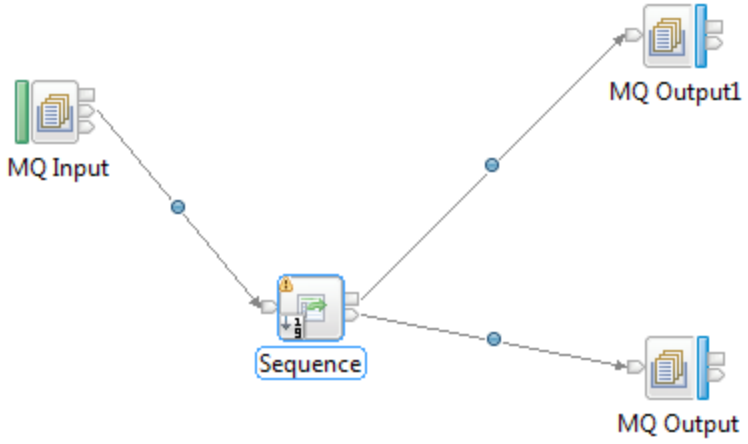
Sequence used to assign sequence number and sequncegroup id to message which is travelled from source to destination nodes.

When we are going for sequence nodes 4 things we need to set

1.sequence number path. **Mandatory**
2.sequence group identifier. **Optional**(but useful with large messages)
3.start of sequence. **Mandatory**
4.end of sequence. **Mandatory**

(observe below properties figure).

STEP1:MESSAGE FLOW WITH MANDATORY PROPERTIES OF SEQUENCE NODE



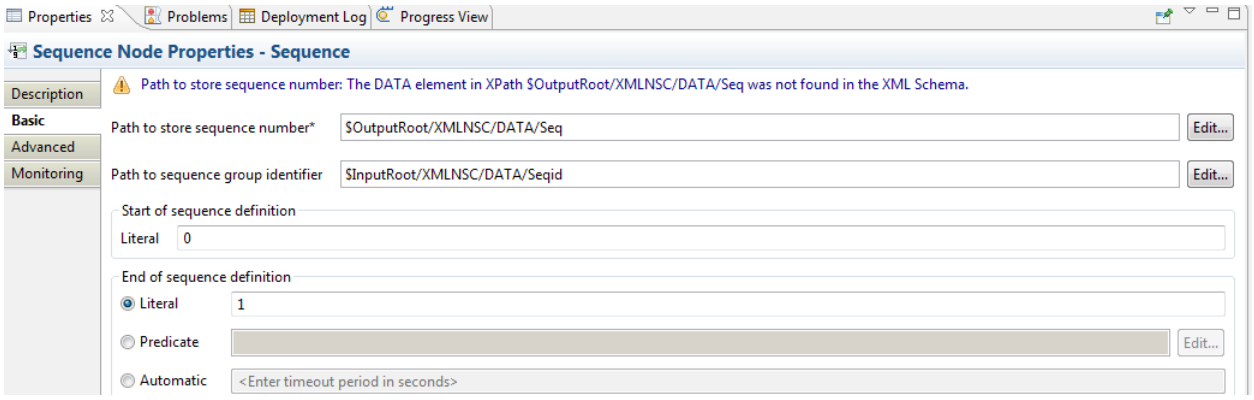
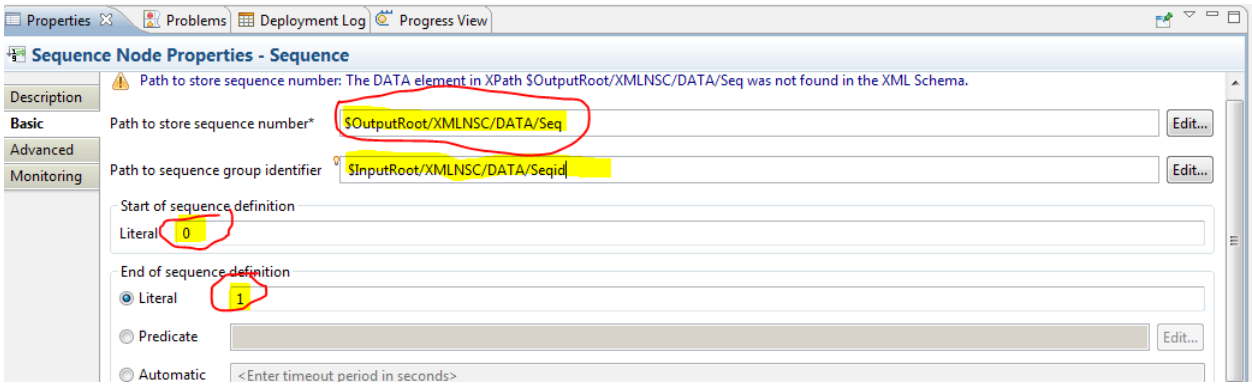


FIGURE2:SEQUENCE NODE PROPERTIES

ERRORS:

Erro1:we nedd to specify following **Red color rounded properties** otherwise error will come.



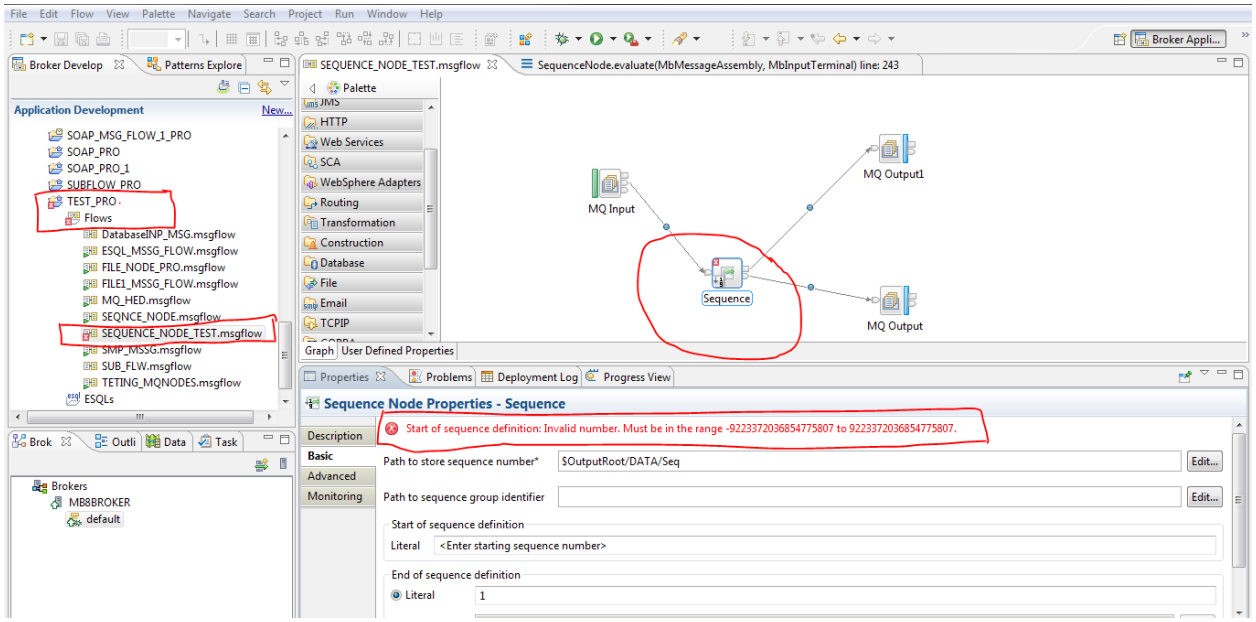
If we doesn't specify the

sequence number path,

start sequence,

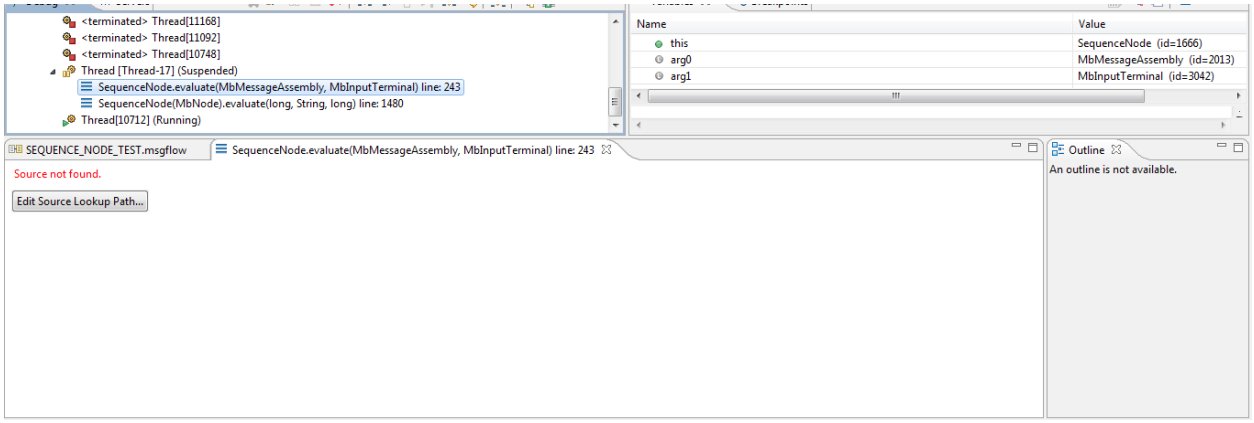
end sequence.

Then error will come it wont deploy consider next figure for example



see red color blocks in the above figure cross option (i.e error) came,so it is impossible to deploy in to the broker. Observe clearly their i didn't mention start sequence,end sequence.

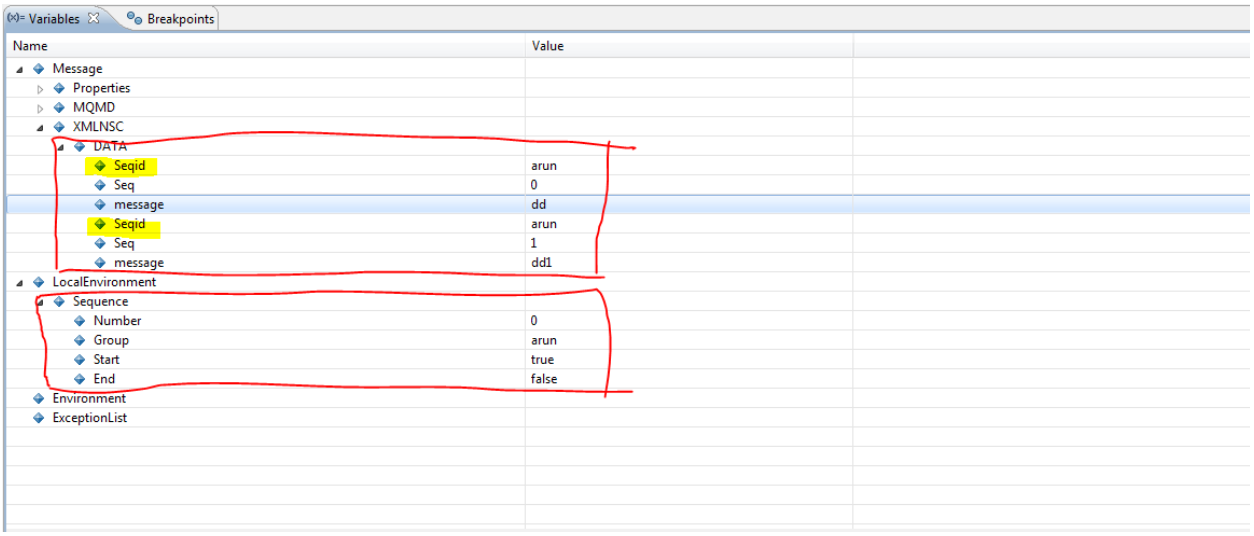
Erro2: when we doesn't mention the incoming parser type in the sequence number path then error won't come but output will not come.see the below figure.(OBSERVE FIGURE 2 sequence number path)



Even if we add broker project to debug it will come again and again.

IMPORTANT NOTE: If we any internal error came message reverse back to sequence node failure terminal if node is connected otherwise it reverse back to mq input node failure terminal if node is connected otherwise it will be in mq input node.

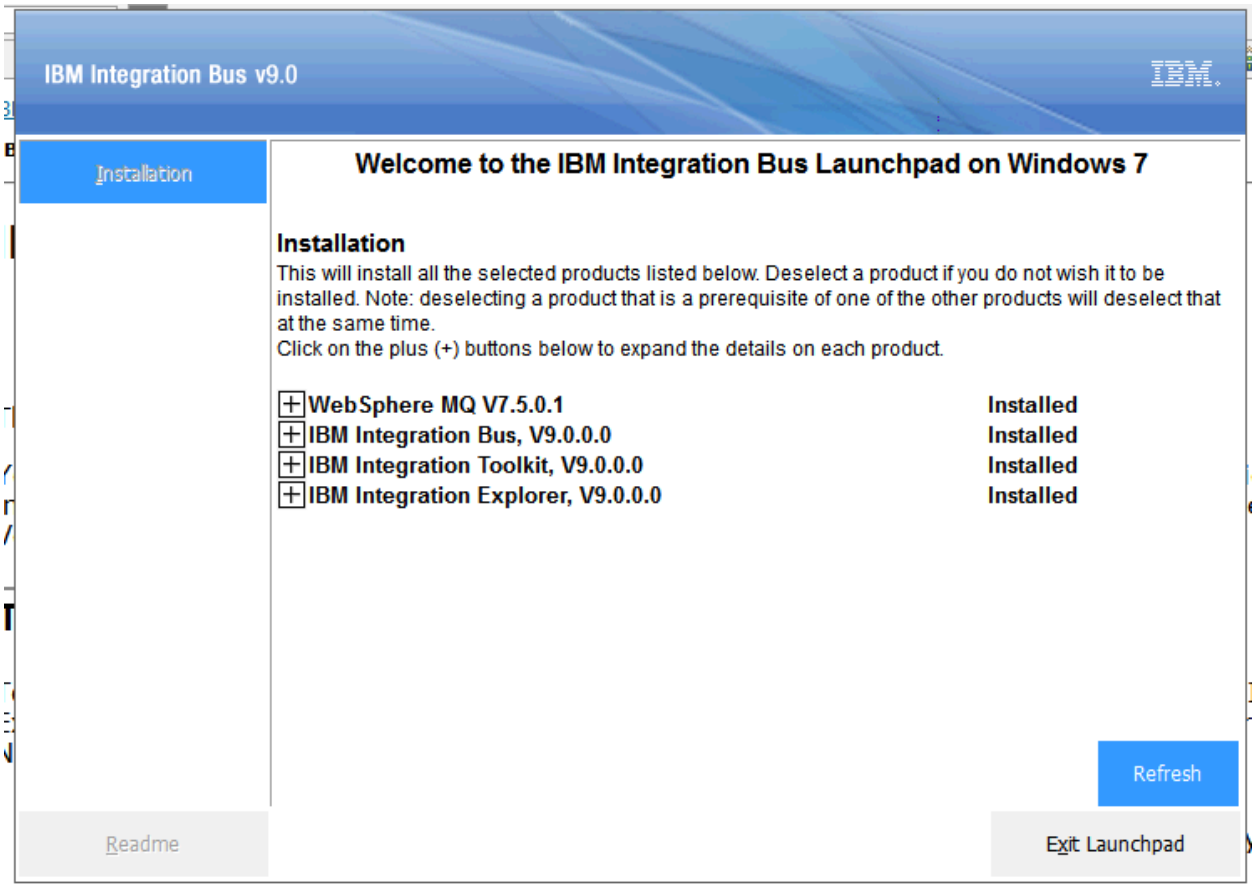
OUTPUT: first observe FIGURE 2 where we given the **sequence id also** then observe the output



INPUT:

<DATA><Seqid>arun</Seqid><Seq>0</Seq><message>dd</message><Seqid>arun</Seqid><Seq>1</Seq><message>dd1</message></DATA>

WMB INSTALLATION AND THEIR PURPOSES :



<u>WebSphere MQ</u>:

IBM® WebSphere® MQ can transport any type of data as messages, enabling businesses to build flexible, reusable architectures such as service-oriented architecture (SOA) environments. It works with a broad range of computing platforms, applications, web services and communications protocols for security-rich message delivery. WebSphere MQ provides a communications layer for visibility and control of the flow of messages and data inside and outside your organization.

WebSphere MQ provides:

- **Versatile messaging integration from mainframe to mobile** that provides a single, robust messaging backbone for dynamic heterogeneous environments.
- **Message delivery with security-rich features** that produce auditable results.
- **High-performance message transport** to deliver data with improved speed and reliability.

- **Administrative features** that simplify messaging management and reduce time spent using complex tools.
- **Open standards development tools** that support extensibility and business growth.

IBM Integration Bus V9 :

IBM Integration Bus is IBM's strategic integration product for Java, Microsoft .NET, and heterogeneous integration scenarios. It represents a significant evolution of the WebSphere Message Broker technology base, and includes new features such as policy-based workload management, business rules, and integration with Business Process Management (BPM) and Microsoft .NET. It also incorporates WebSphere Enterprise Service Bus (ESB) use cases, and WebSphere ESB capabilities will be folded into IBM Integration Bus over time, with conversion tools for initial use cases built-in from day one.

IBM Integration Toolkit :

Application developers work in separate instances of the IBM Integration Toolkit to develop resources associated with message flows. The IBM Integration Toolkit connects to one or more brokers to which the message flows are deployed.

When you start the IBM Integration Toolkit, a single window is displayed. This window is the IBM Integration Toolkit, which contains one or more perspectives.

A perspective is a collection of views and editors that you use to complete a specific task, or work with specific types of resource. The two significant perspectives in the IBM Integration Toolkit are the Integration Development perspective for application development, and the Debug perspective for debugging message flows. The first time that you start the IBM Integration Toolkit, the Integration Development perspective is displayed.

An additional stand-alone component, the *IBM Integration Explorer*, is supplied for advanced administrative users, and enables additional administration tasks that you cannot perform in the IBM Integration Toolkit.

IBM Integration Explorer :

The IBM Integration Explorer is an extension to the WebSphere® MQ Explorer.

To use the IBM Integration Explorer, you must start the WebSphere MQ Explorer. The IBM Integration Explorer adds the Integration Nodes folder and Broker Archive Files folder to the MQ Explorer - Navigator view:

- Use the Integration Nodes folder to create, view, and modify integration nodes
- Use the Broker Archive Files folder to import, view, and modify BAR files before deploying them to your integration nodes

The IBM Integration Explorer provides several QuickViews that you can use to view the properties of integration nodes and their resources. These QuickViews are automatically displayed when you click the resource in the Integration Nodes folder in the MQ Explorer - Navigator view. A QuickView is also available for viewing the details of BAR files that you have imported into the IBM Integration Explorer.

The following views and editors are provided for working with integration nodes in the IBM Integration Explorer:

Broker Archive editor

Use the Broker Archive editor to create and manage broker archive (BAR) files.

Broker Statistics and Broker Statistics Graph views

Use the Broker Statistics and Broker Statistics Graph views to view snapshot accounting and statistics data as it is produced by the broker.

Policy Sets and Policy Set Bindings editor

Use the Policy Sets and Policy Set Bindings editor to edit, save, import, and export policy sets or bindings.

Security Profiles editor

Use the Security Profiles editor to create a security profile for use with Lightweight Directory Access Protocol (LDAP) or Tivoli® Federated Identity Manager (TFIM).

DataPower® Security wizard

Use the DataPower Security wizard to configure an external DataPower appliance to handle the WS-Security Policy for your HTTP, HTTPS, and SOAP nodes within your message flow.

Administration Log view

Use the Administration Log view to view the results of deployment actions on integration nodes.

Activity Log view

Use the Activity Log view to view recent activities affecting your message flows, and related external resources.

WSRR INSTALLATION AND THEIR PURPOSE :

WebSphere Service Registry and Repository(WSRR)

- WebSphere Service Registry and Repository (WSRR) provides both a registry of Services as well as providing a repository capability to store documents that hold further metadata to detail them.

WSRR is not limited to Web Services, and can support services defined using a variety of other protocols and programming models. By using the customization capabilities provided it can also be used to define and store Service at a more conceptual level, such as Business Services. Customization can also be used to define the metadata and documents to describe just about any SOA asset type so these can also be stored in WSRR as well as their relationships to the Services.

WSRR also provides a customizable SOA lifecycle, together with Policy Management function that can be used to provide governance over the Service metadata and lifecycle within WSRR.
wants to receive. The request defines the topic, the filter, and the subscription point of each publication.

Messages that are published by a publisher can be received by more than one subscriber, and a subscriber can receive messages, on the same or different topics, from more than one publisher.

Uses of WebSphere Service Registry and Repository(WSRR)

- Registry and repository is used to find, publish, manage, and subscribe to services with the assurance that the underlying policies associated with correct usages of these services are enforced and governed. It supports the following functions:

(I) Publish : Add new services through an approval process so they are available and managed on an enterprise-wide scale. Services can be WSDL, XML, or schema definition.

(II) Find : Search for services based on any of the metadata associated with the service.

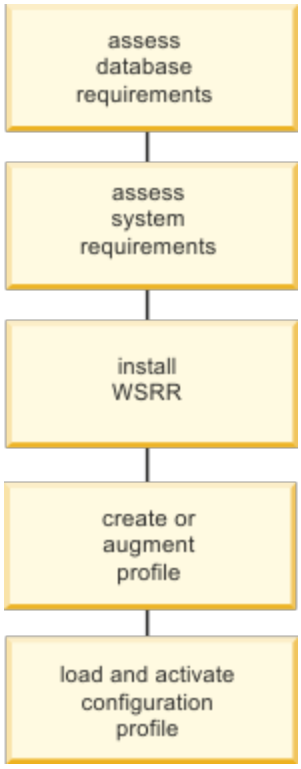
(III) Enrich : Add value to your SOA processing by adding information to enrich service artifacts. This information can be used during the Find stage.

(IV) Manage : Provide customizable processes to manage the lifecycle of services in the registry, enabling access control, promote/retire, and change analysis to services through impact analysis.

(V) Govern : Provide a central point of overall governance within your enterprise-wide SOA.

Installing and configuring WSRR is a multistep process that requires you to plan and make decisions before you start to install. Use the guidance here to help you make these decisions.

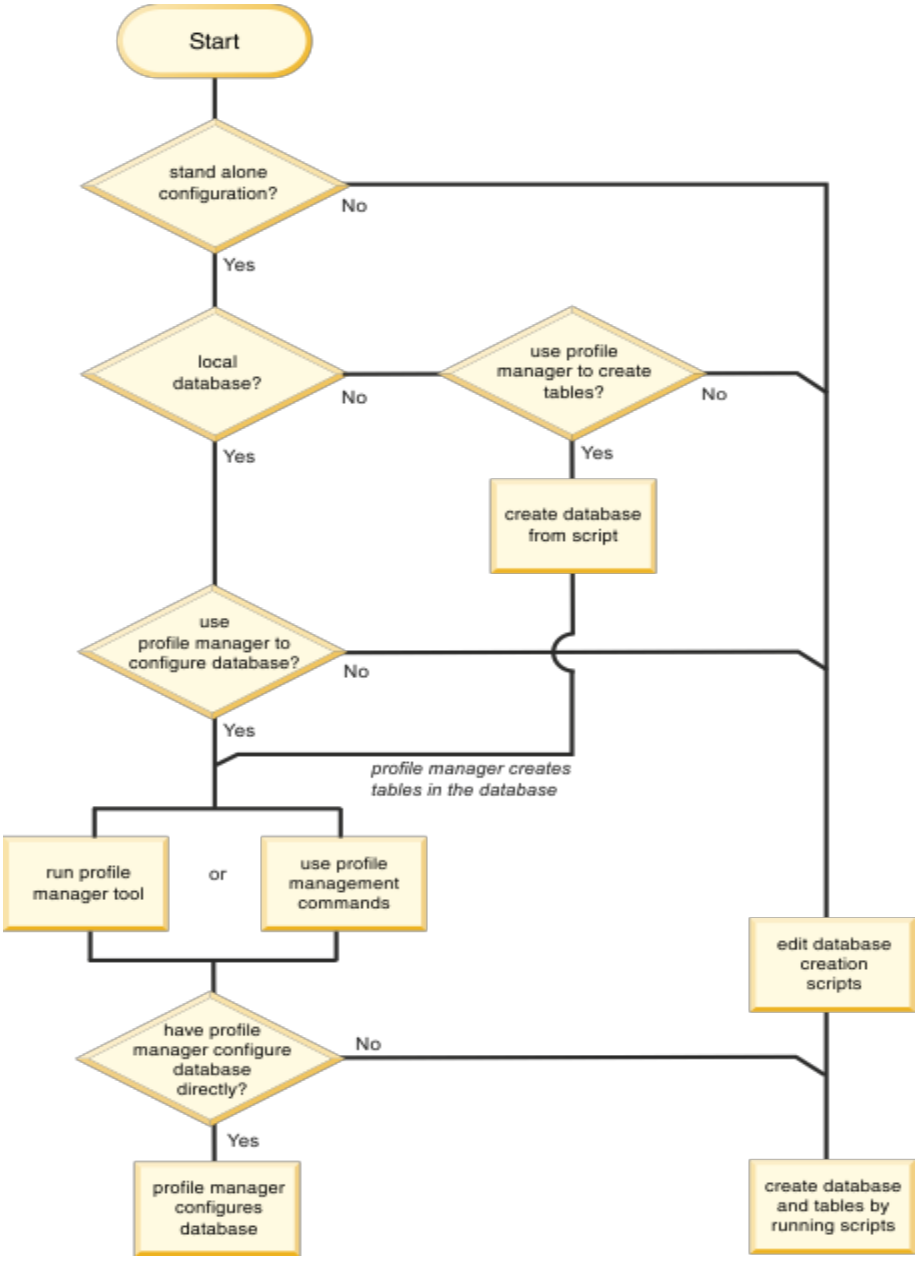
The following diagram shows the major steps required for getting WSRR installed and configured.

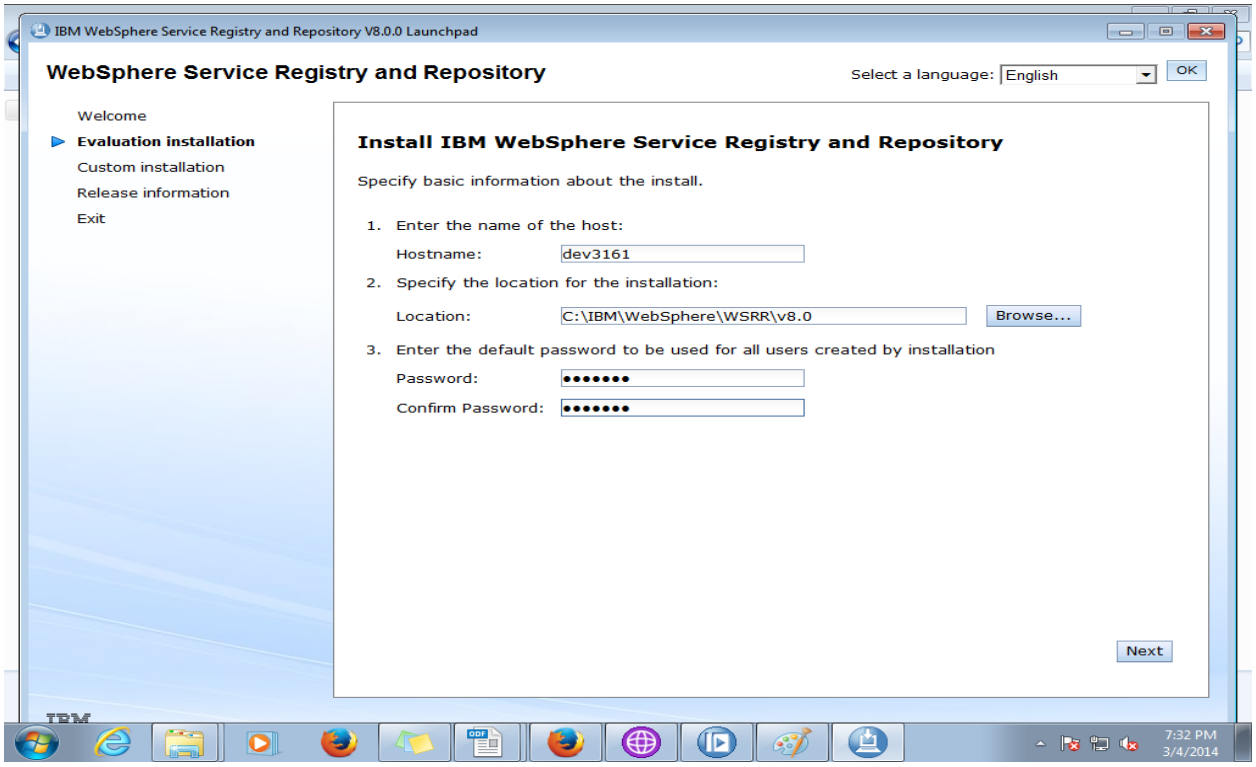


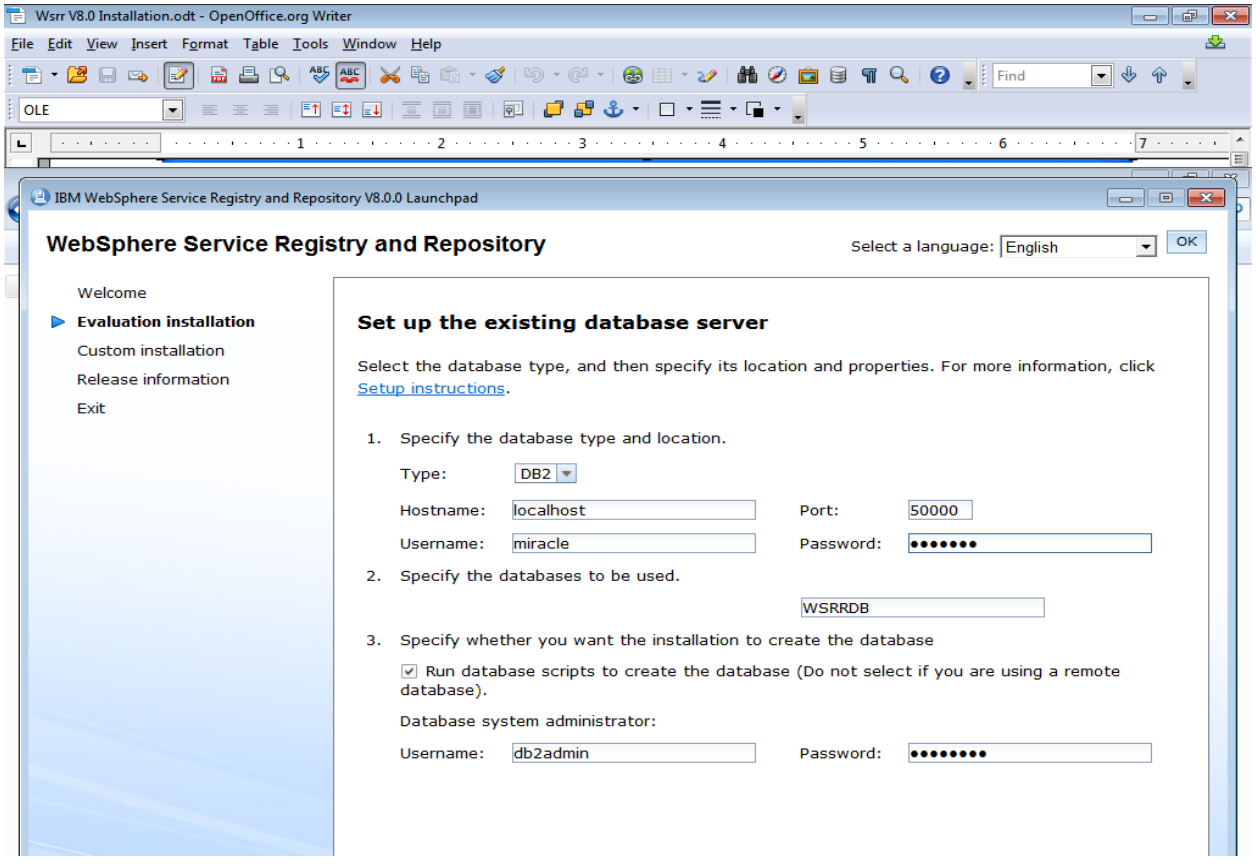
1. Assess database requirements

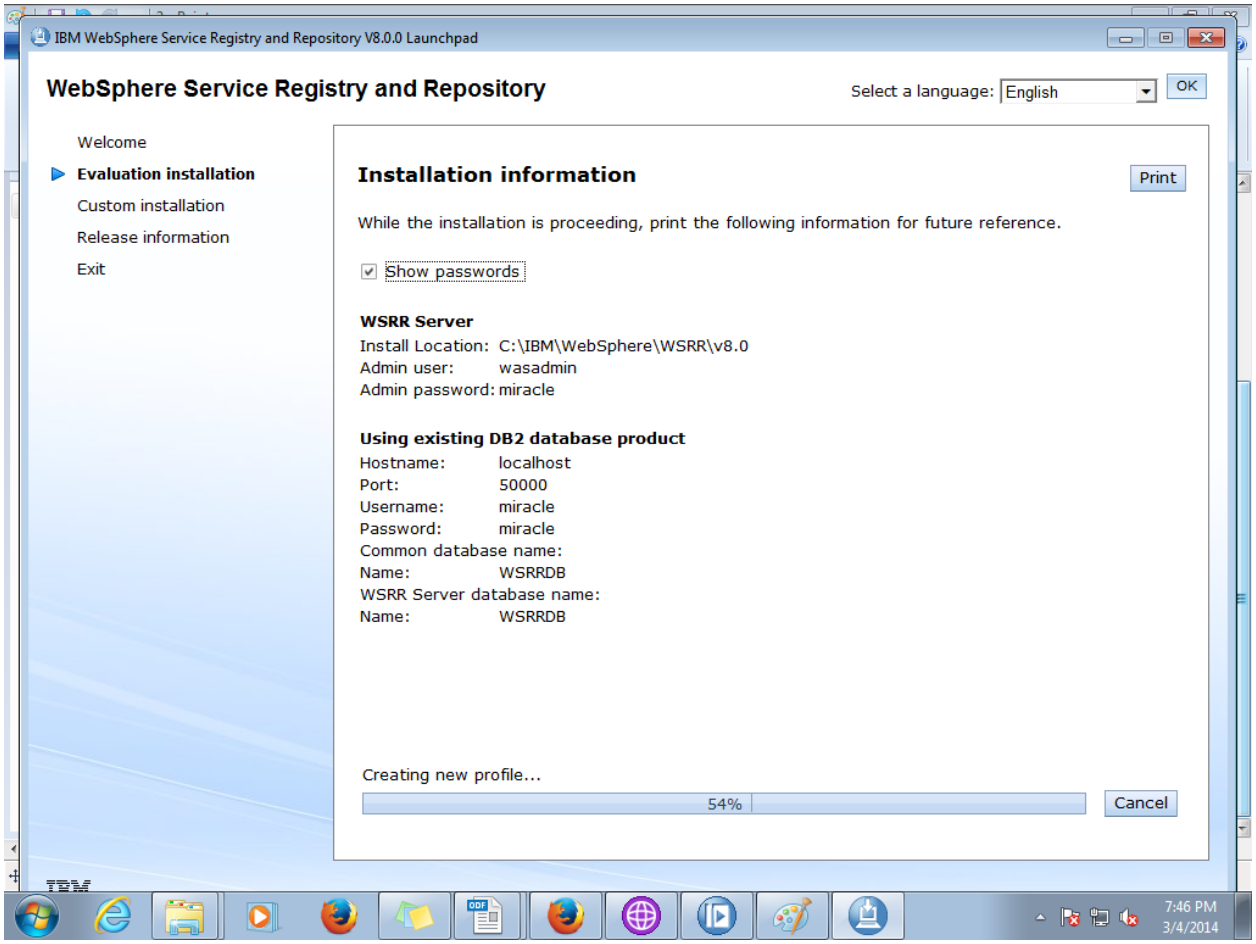
WSRR uses an RDBMS to host the registry and repository. You can use one of the following RDBMS for this purpose:

- DB2 Universal Database™
- DB2® for z/OS® v8
- DB2 for z/OS v9
- Microsoft SQL Server
- Oracle 10g
- Oracle 11g
- Derby (stand-alone server only)


Start
stand alone configuration?
No
Yes
local database?
No
use profile manager to create tables?
No
Yes
Yes
create database from script
use profile manager to configure database?
No
Yes
profile manager creates tables in the database
run profile manager tool
or
use profile management commands
edit database creation scripts
have profile manager configure database directly?
No
Yes
profile manager configures database
create database and tables by running scripts


IBM WebSphere Service Registry and Repository V8.0.0 Launchpad
WebSphere Service Registry and Repository
Select a language: English
OK
Welcome
Evaluation installation
Custom installation
Release information
Exit
Install IBM WebSphere Service Registry and Repository
Specify basic information about the install.
1. Enter the name of the host:
Hostname: dev3161
2. Specify the location for the installation:
Location: C:\IBM\WebSphere\WSRR\v8.0
Browse...
3. Enter the default password to be used for all users created by installation
Password: •••••••
Confirm Password: •••••••
Next
IBM
7:32 PM
3/4/2014


Wsrr V8.0 Installation.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
Find
OLE
IBM WebSphere Service Registry and Repository V8.0.0 Launchpad
WebSphere Service Registry and Repository
Select a language: English
OK
Welcome
Evaluation installation
Custom installation
Release information
Exit
Set up the existing database server
Select the database type, and then specify its location and properties. For more information, click Setup instructions.
1. Specify the database type and location.
Type: DB2
Hostname: localhost
Port: 50000
Username: miracle
Password: •••••••
2. Specify the databases to be used.
WSRRDB
3. Specify whether you want the installation to create the database
Run database scripts to create the database (Do not select if you are using a remote database).
Database system administrator:
Username: db2admin
Password: ••••••••

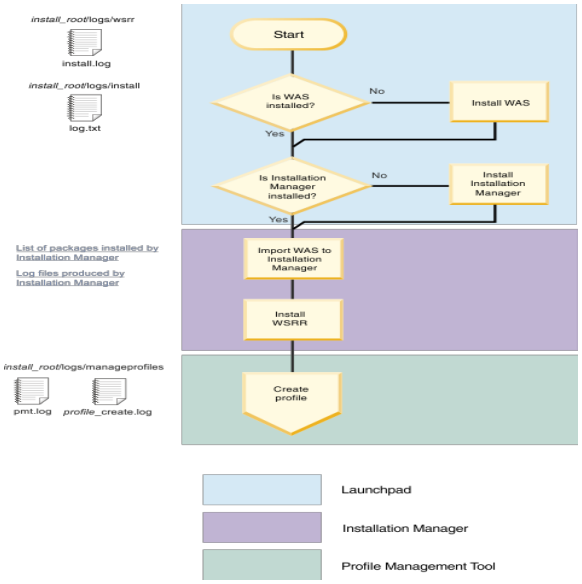


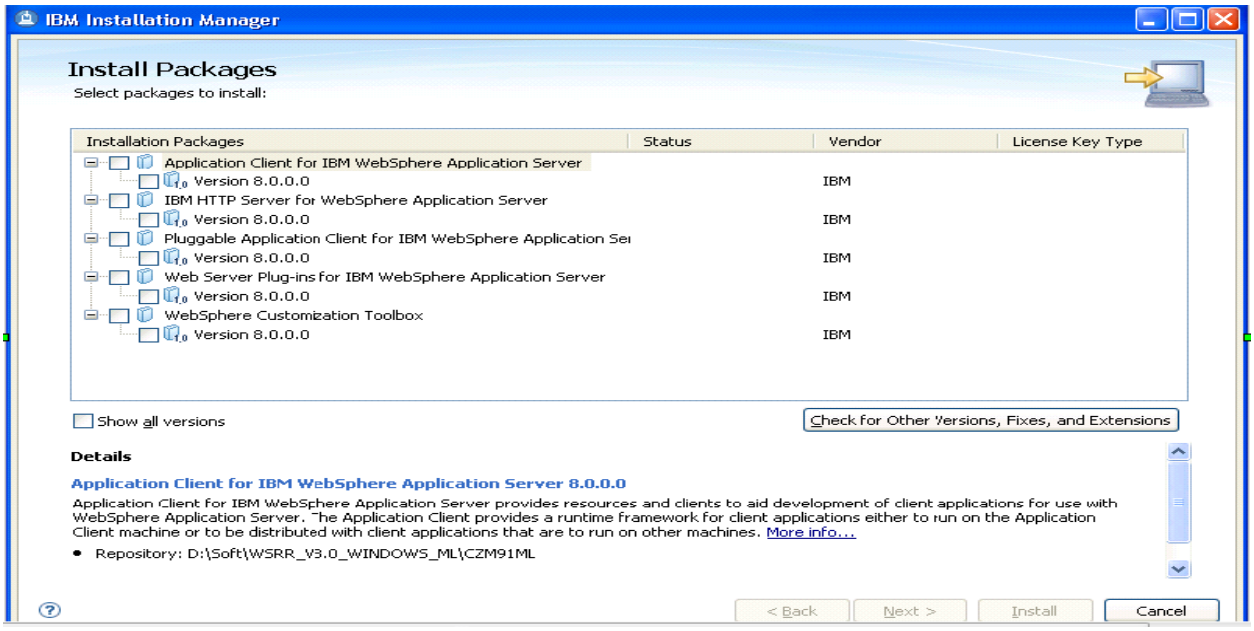
2. Assess system requirements

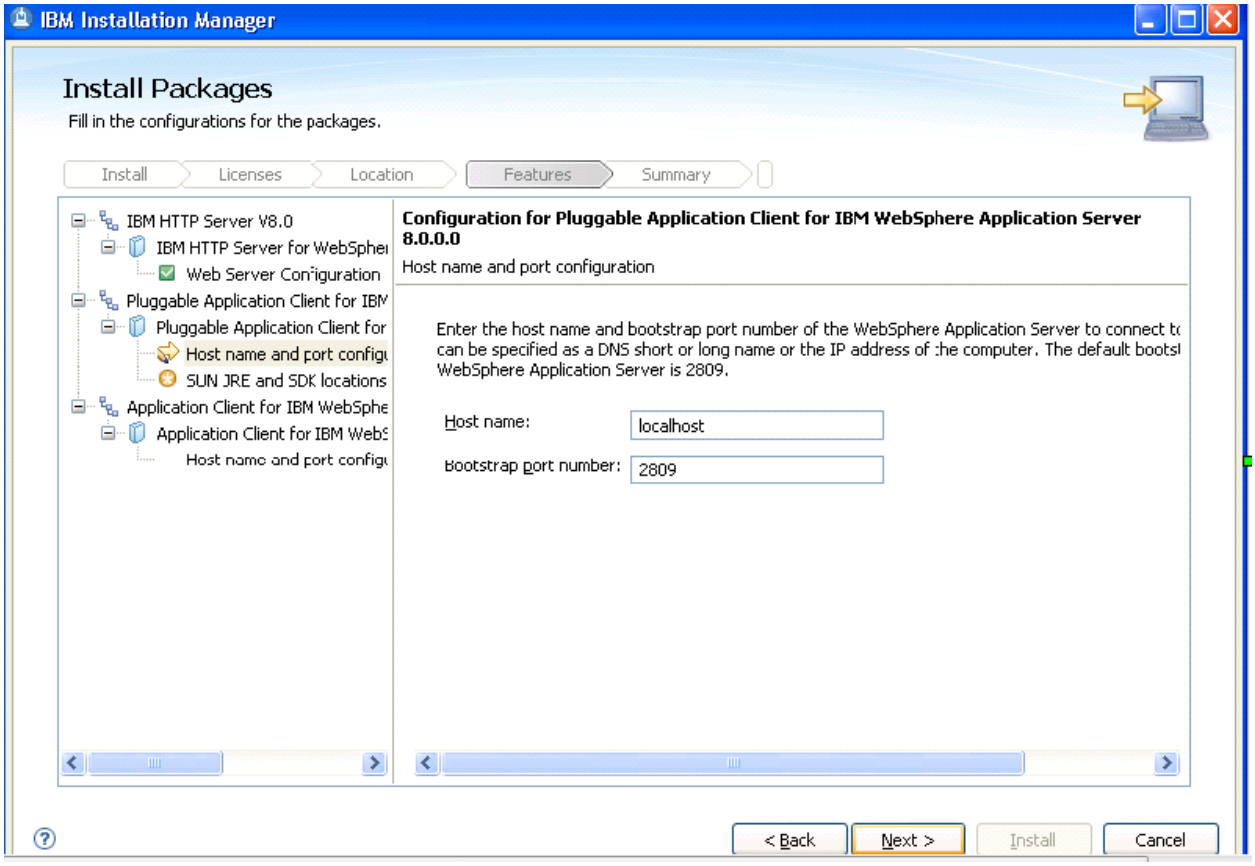
You can adopt a number of configurations for WSRR: stand-alone, stand-alone with remote database, managed server, or cluster. The configuration is implemented by WebSphere Application Server, and you need knowledge of WebSphere Application Server to set up more complex configurations such as clusters.

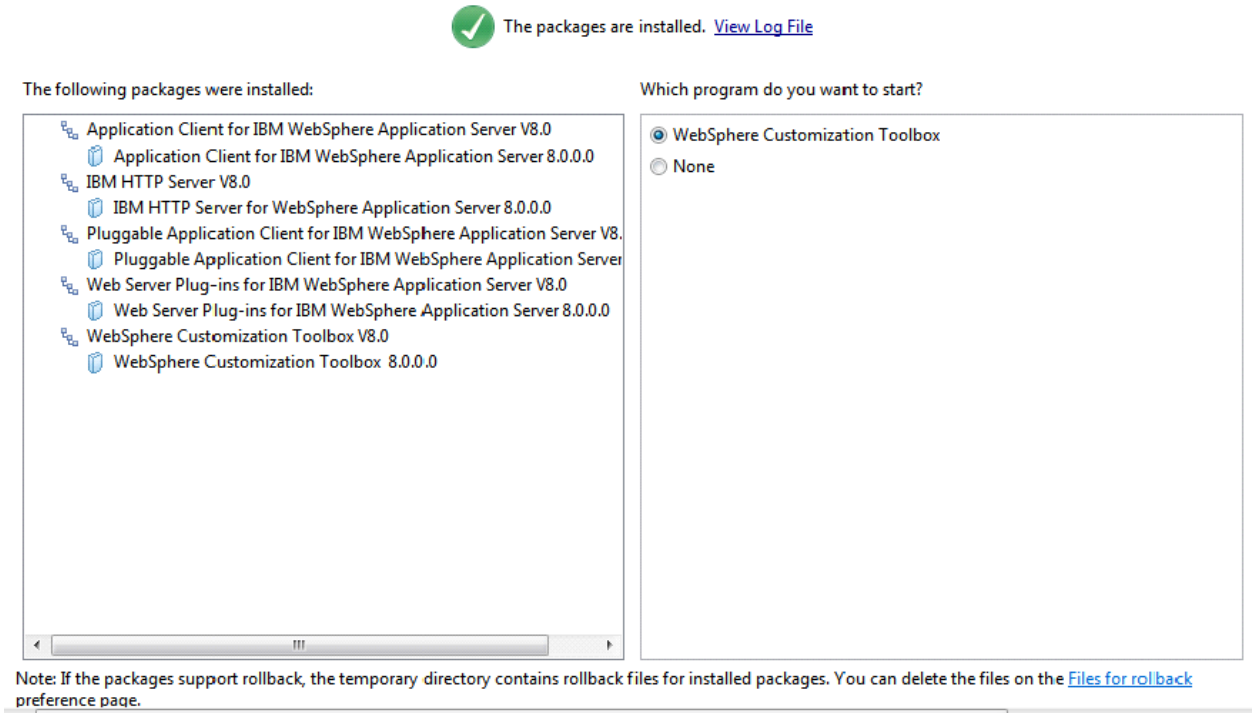
3. Install WSRR

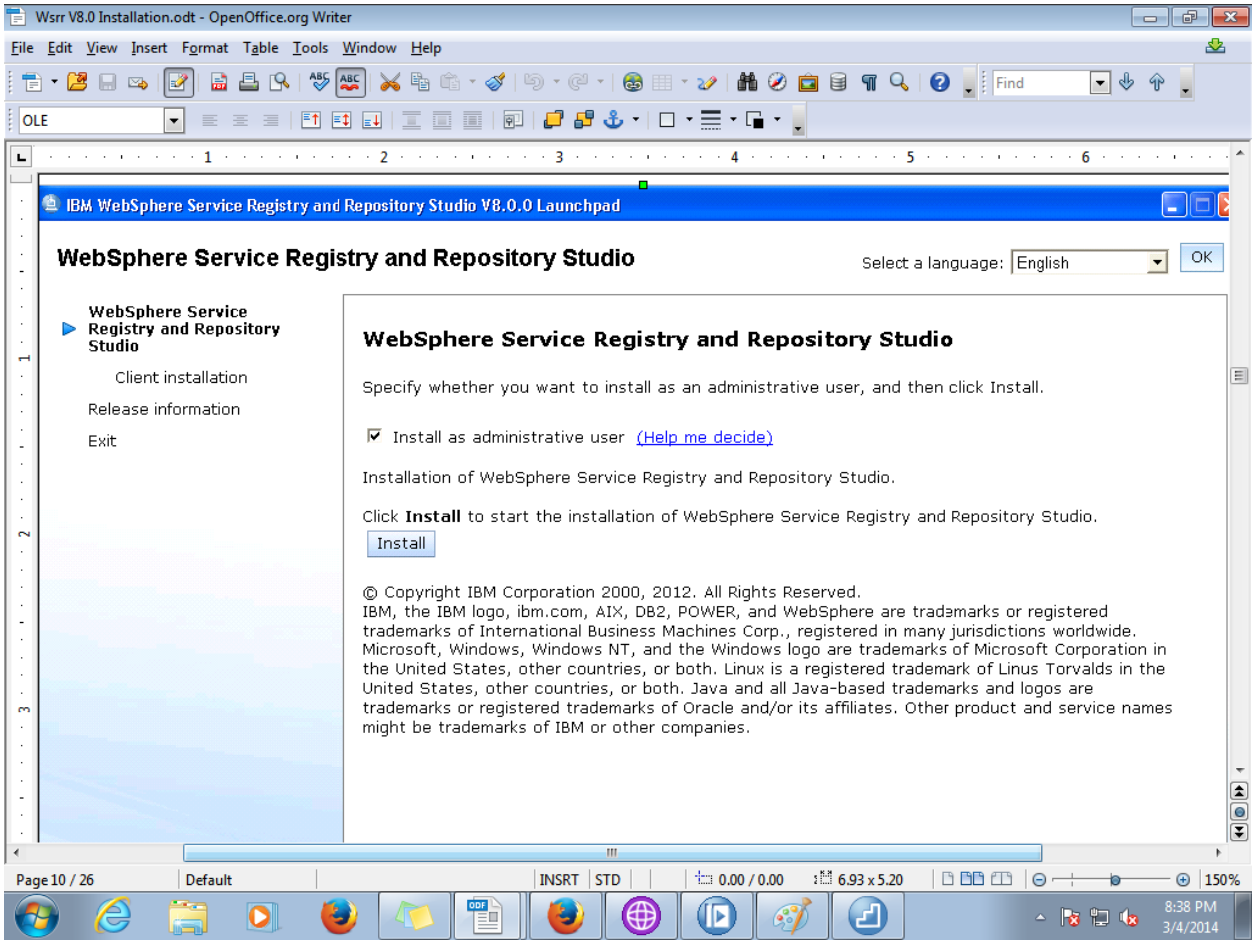
The following figure illustrates the process of installing WSRR on a stand-alone system and identifies the tools that you use to complete the different stages of installation and configuration if you install interactively.

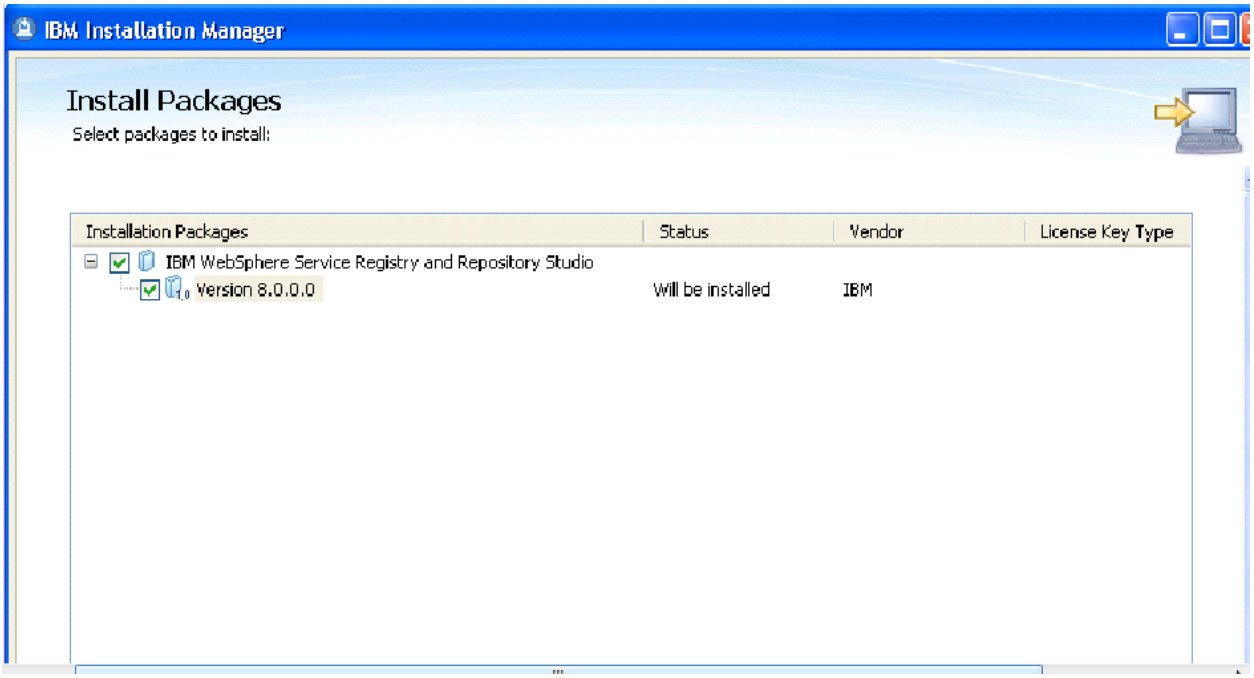


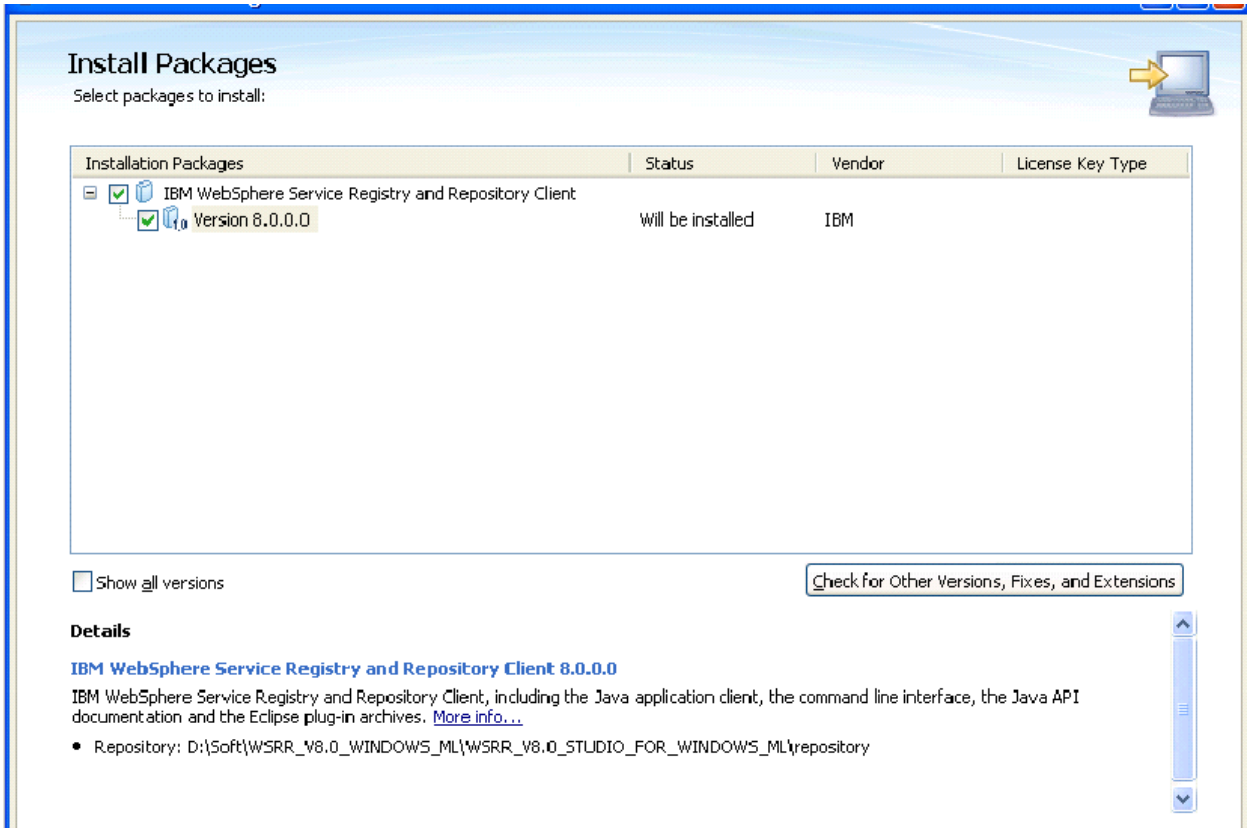



IBM Installation Manager
Install Packages
Fill in the configurations for the packages.
Install
Licenses
Location
Features
Summary
IBM HTTP Server V8.0
IBM HTTP Server for WebSphe
Web Server Configuration
Pluggable Application Client for IBM
Pluggable Application Client for
Host name and port configu
SUN JRE and SDK locations
Application Client for IBM WebSphe
Application Client for IBM Webs
Host name and port configu
Configuration for Pluggable Application Client for IBM WebSphere Application Server 8.0.0.0
Host name and port configuration
Enter the host name and bootstrap port number of the WebSphere Application Server to connect t
can be specified as a DNS short or long name or the IP address of the computer. The default bootst
WebSphere Application Server is 2809.
Host name:
localhost
Bootstrap port number:
2809
< Back
Next >
Install
Cancel


The packages are installed. View Log File
The following packages were installed:
Application Client for IBM WebSphere Application Server V8.0
Application Client for IBM WebSphere Application Server 8.0.0.0
IBM HTTP Server V8.0
IBM HTTP Server for WebSphere Application Server 8.0.0.0
Pluggable Application Client for IBM WebSphere Application Server V8.
Pluggable Application Client for IBM WebSphere Application Server
Web Server Plug-ins for IBM WebSphere Application Server V8.0
Web Server Plug-ins for IBM WebSphere Application Server 8.0.0.0
WebSphere Customization Toolbox V8.0
WebSphere Customization Toolbox 8.0.0.0
Which program do you want to start?
WebSphere Customization Toolbox
None
Note: If the packages support rollback, the temporary directory contains rollback files for installed packages. You can delete the files on the Files for rollback preference page.


Wsrr V8.0 Installation.odt - OpenOffice.org Writer
File Edit View Insert Format Table Tools Window Help
IBM WebSphere Service Registry and Repository Studio V8.0.0 Launchpad
WebSphere Service Registry and Repository Studio
Select a language: English
OK
WebSphere Service Registry and Repository Studio
Client installation
Release information
Exit
WebSphere Service Registry and Repository Studio
Specify whether you want to install as an administrative user, and then click Install.
Install as administrative user (Help me decide)
Installation of WebSphere Service Registry and Repository Studio.
Click Install to start the installation of WebSphere Service Registry and Repository Studio.
Install
© Copyright IBM Corporation 2000, 2012. All Rights Reserved.
IBM, the IBM logo, ibm.com, AIX, DB2, POWER, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. Other product and service names might be trademarks of IBM or other companies.
Page 10 / 26
Default
INSRT STD
0.00 / 0.00
6.93 x 5.20
150%
8:38 PM
3/4/2014


IBM Installation Manager
Install Packages
Select packages to install:
Installation Packages
Status
Vendor
License Key Type
IBM WebSphere Service Registry and Repository Studio
Version 8.0.0.0
Will be installed
IBM

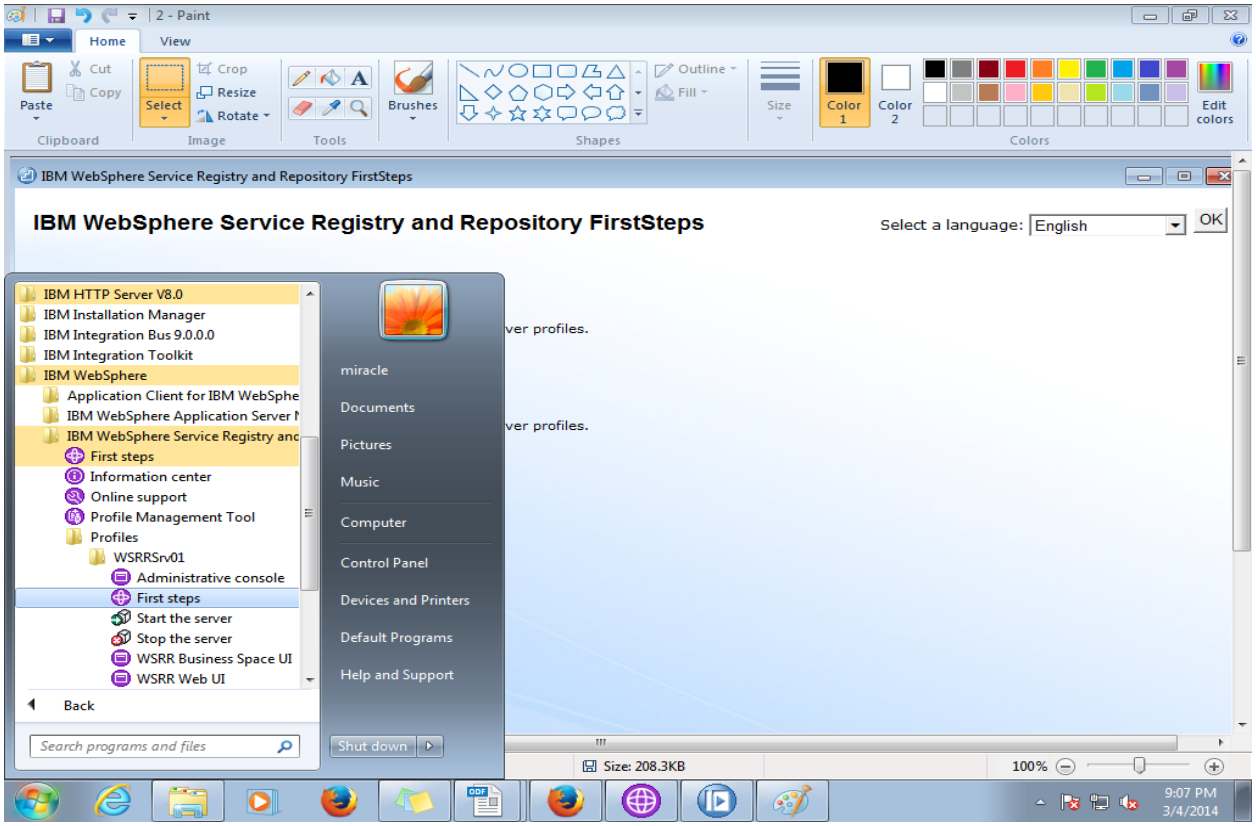


4. Create or augment profile

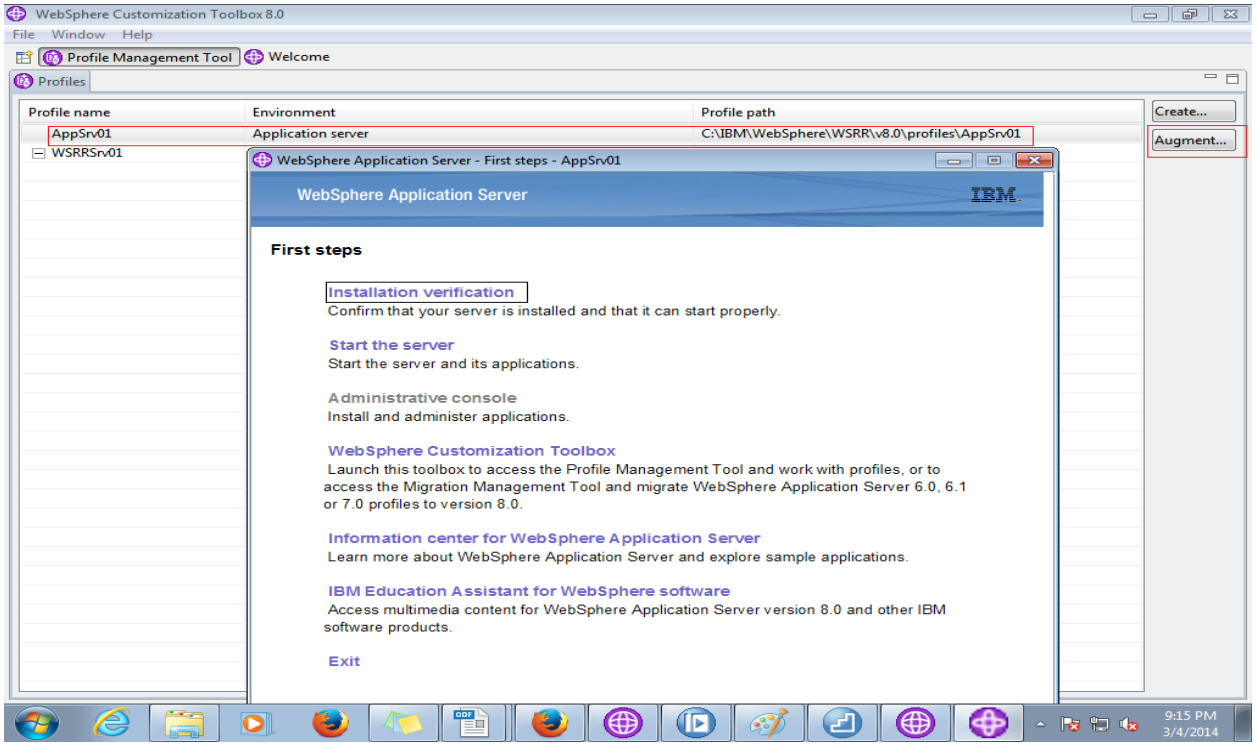
After you have installed WSRR on a standalone system, you can create a WebSphere Application Server profile, or you can augment an existing WebSphere Application Server profile to include WSRR. You might, for example, already have a WebSphere Process Server profile on your system, and you can augment this profile to add WSRR.

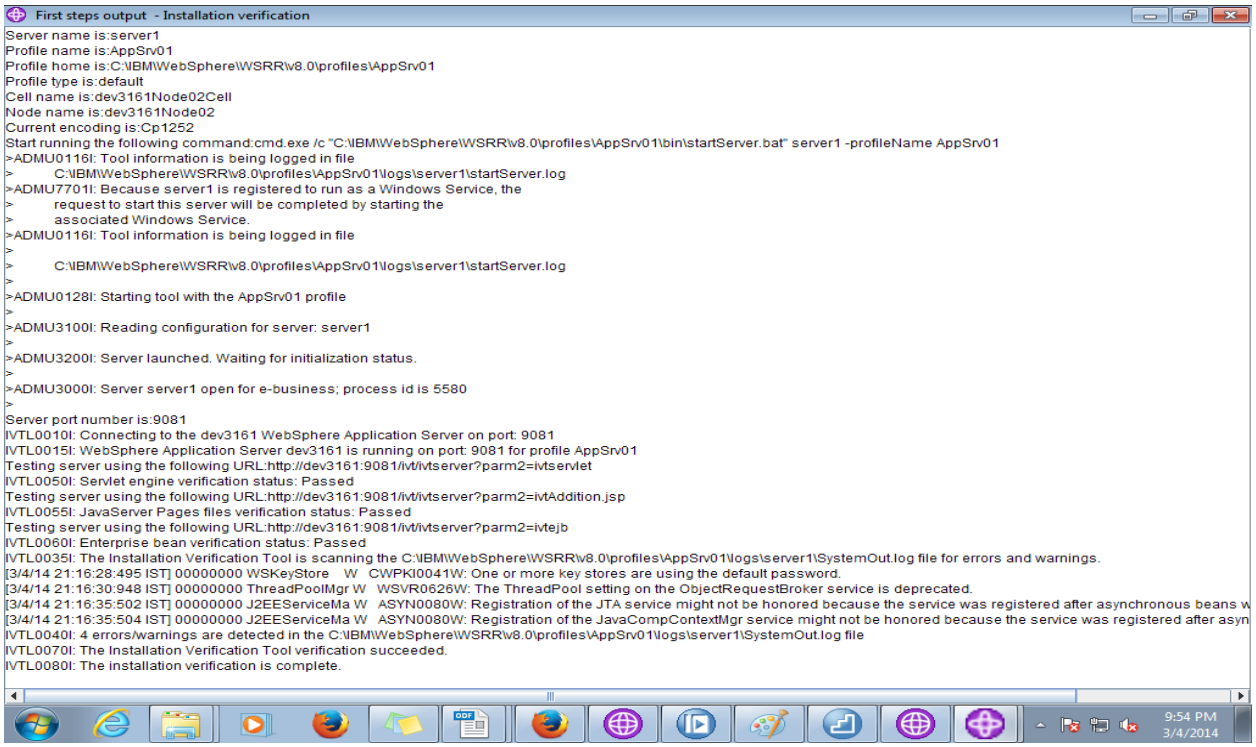
The simplest scenario is to create a *WebSphere Application Server profile* on a stand-alone system that uses Derby for the WSRR database. In this case you can use the Typical profile creation in the Profile Management Tool. If you are configuring a stand-alone system with one of the other supported RDBMS, then you must use the Advanced profile creation, so that you can supply details about the RDBMS that you are using.

When you install WSRR on a cluster or federated node system, you must set up the system first. You then augment the existing Deployment Manager profile to include WSRR. To augment the profile, you use the Advanced profile creation facilities of the Profile Management Tool.


2 - Paint
Home
View
Paste
Cut
Copy
Select
Crop
Resize
Rotate
Brushes
Outline
Fill
Size
Color 1
Color 2
Edit colors
Clipboard
Image
Tools
Shapes
Colors
IBM WebSphere Service Registry and Repository FirstSteps
IBM WebSphere Service Registry and Repository FirstSteps
Select a language:
English
OK
IBM HTTP Server V8.0
IBM Installation Manager
IBM Integration Bus 9.0.0.0
IBM Integration Toolkit
IBM WebSphere
Application Client for IBM WebSphe
IBM WebSphere Application Server
IBM WebSphere Service Registry and
First steps
Information center
Online support
Profile Management Tool
Profiles
WSRRSrv01
Administrative console
First steps
Start the server
Stop the server
WSRR Business Space UI
WSRR Web UI
Back
Search programs and files
miracle
Documents
Pictures
Music
Computer
Control Panel
Devices and Printers
Default Programs
Help and Support
Shut down
ver profiles.
ver profiles.
Size: 208.3KB
100%
9:07 PM
3/4/2014


IBM WebSphere Service Registry and Repository FirstSteps
IBM WebSphere Service Registry and Repository FirstSteps
Select a language:
English
OK
Profile Actions
Load and activate the WSRR Governance Enablement Profile.
Load WSRR Governance Enablement Profile
Stop the server and its applications.
Stop Server
Install and administer applications
Administrative Console
Perform Optional Actions
Use the WebSphere Service Registry and Repository (WSRR) Web user interface to use and manage all aspects of WSRR from a compatible Web browser.
WSRR Web UI
Use the WebSphere Service Registry and Repository (WSRR) Business Space user interface to use and manage WSRR content.
WSRR Business Space UI
Work with WebSphere Application Server profiles.
Profile Management Tool
View Documentation
Information Center
Installation Guide
9:08 PM
3/4/2014


WebSphere Customization Toolbox 8.0
File Window Help
Profile Management Tool
Welcome
Profiles
Profile name
Environment
Profile path
AppSrv01
Application server
C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01
WSRRSrv01
Create...
Augment...
WebSphere Application Server - First steps - AppSrv01
WebSphere Application Server
IBM
First steps
Installation verification
Confirm that your server is installed and that it can start properly.
Start the server
Start the server and its applications.
Administrative console
Install and administer applications.
WebSphere Customization Toolbox
Launch this toolbox to access the Profile Management Tool and work with profiles, or to access the Migration Management Tool and migrate WebSphere Application Server 6.0, 6.1 or 7.0 profiles to version 8.0.
Information center for WebSphere Application Server
Learn more about WebSphere Application Server and explore sample applications.
IBM Education Assistant for WebSphere software
Access multimedia content for WebSphere Application Server version 8.0 and other IBM software products.
Exit
9:15 PM
3/4/2014


First steps output - Installation verification
Server name is:server1
Profile name is:AppSrv01
Profile home is:C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01
Profile type is:default
Cell name is:dev3161Node02Cell
Node name is:dev3161Node02
Current encoding is:Cp1252
Start running the following command:cmd.exe /c "C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\bin\startServer.bat" server1 -profileName AppSrv01
>ADMU0116I: Tool information is being logged in file
> C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\logs\server1\startServer.log
>ADMU7701I: Because server1 is registered to run as a Windows Service, the
> request to start this server will be completed by starting the
> associated Windows Service.
>ADMU0116I: Tool information is being logged in file
>
> C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\logs\server1\startServer.log
>
>ADMU0128I: Starting tool with the AppSrv01 profile
>
>ADMU3100I: Reading configuration for server: server1
>
>ADMU3200I: Server launched. Waiting for initialization status.
>
>ADMU3000I: Server server1 open for e-business; process id is 5580
>
Server port number is:9081
IVTL0010I: Connecting to the dev3161 WebSphere Application Server on port: 9081
IVTL0015I: WebSphere Application Server dev3161 is running on port: 9081 for profile AppSrv01
Testing server using the following URL:http://dev3161:9081/ivt/ivtserver?parm2=ivtservlet
IVTL0050I: Servlet engine verification status: Passed
Testing server using the following URL:http://dev3161:9081/ivt/ivtserver?parm2=ivtAddition.jsp
IVTL0055I: JavaServer Pages files verification status: Passed
Testing server using the following URL:http://dev3161:9081/ivt/ivtserver?parm2=ivtejb
IVTL0060I: Enterprise bean verification status: Passed
IVTL0035I: The Installation Verification Tool is scanning the C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\logs\server1\SystemOut.log file for errors and warnings.
[3/4/14 21:16:28:495 IST] 00000000 WSKeyStore W CWPKI0041W: One or more key stores are using the default password.
[3/4/14 21:16:30:948 IST] 00000000 ThreadPoolMgr W WSVR0626W: The ThreadPool setting on the ObjectRequestBroker service is deprecated.
[3/4/14 21:16:35:502 IST] 00000000 J2EEServiceMa W ASYN0080W: Registration of the JTA service might not be honored because the service was registered after asynchronous beans w
[3/4/14 21:16:35:504 IST] 00000000 J2EEServiceMa W ASYN0080W: Registration of the JavaCompContextMgr service might not be honored because the service was registered after asyn
IVTL0040I: 4 errors/warnings are detected in the C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\logs\server1\SystemOut.log file
IVTL0070I: The Installation Verification Tool verification succeeded.
IVTL0080I: The installation verification is complete.
9:54 PM
3/4/2014

5.Load and activate a configuration profile

The final action to take before you start using WSRR is to *load and activate* a configuration profile. See the topic Configuration profiles for guidance.

The configuration profile defines the capabilities of your WSRR system. Two configuration profiles are supplied with WSRR: the governance enablement profile and the basic profile. The governance enablement profile provides component models, lifecycles, governance policies, security controls, and web user interface commands that enable you to implement a complete governance process. You can load and activate the governance enablement profile from the First Steps console that opens when you have installed WSRR.

The basic profile is suitable for use with a runtime WSRR system, such as can be used in conjunction with a product like WebSphere Enterprise Service Bus.

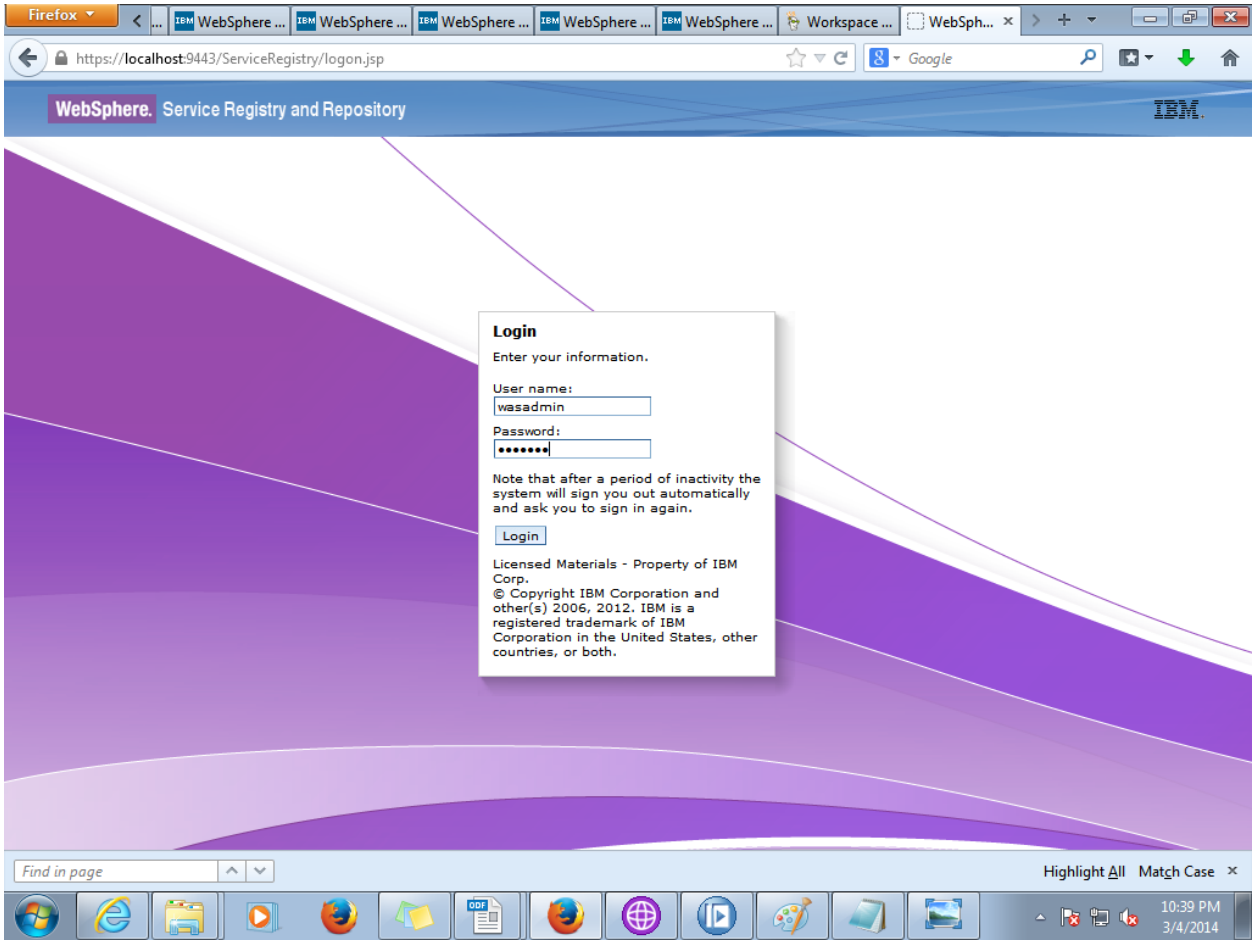
You can also provide your own customized configuration profile. For example you could add your own lifecycles and governance policies to the governance enablement profile. (WSRR Studio provides a graphical interface for designing customized configuration profiles.)

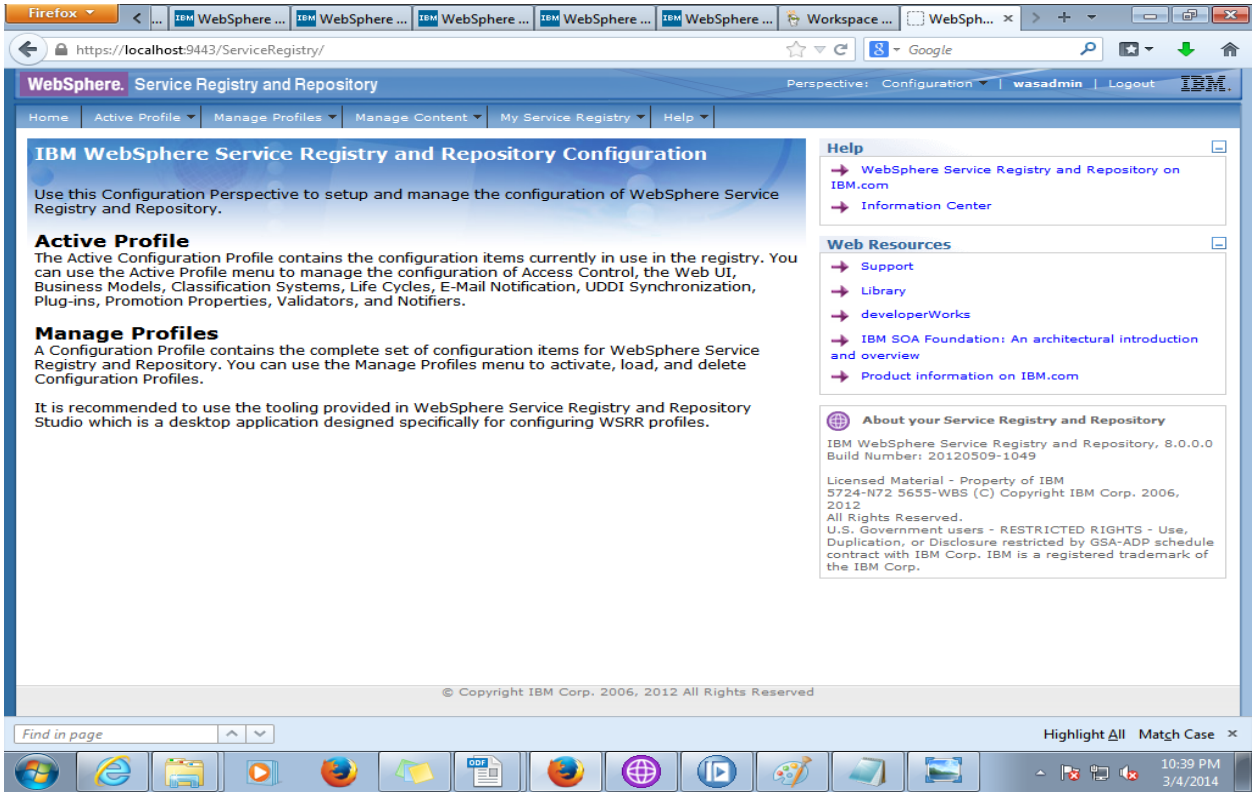
Configuration profiles :

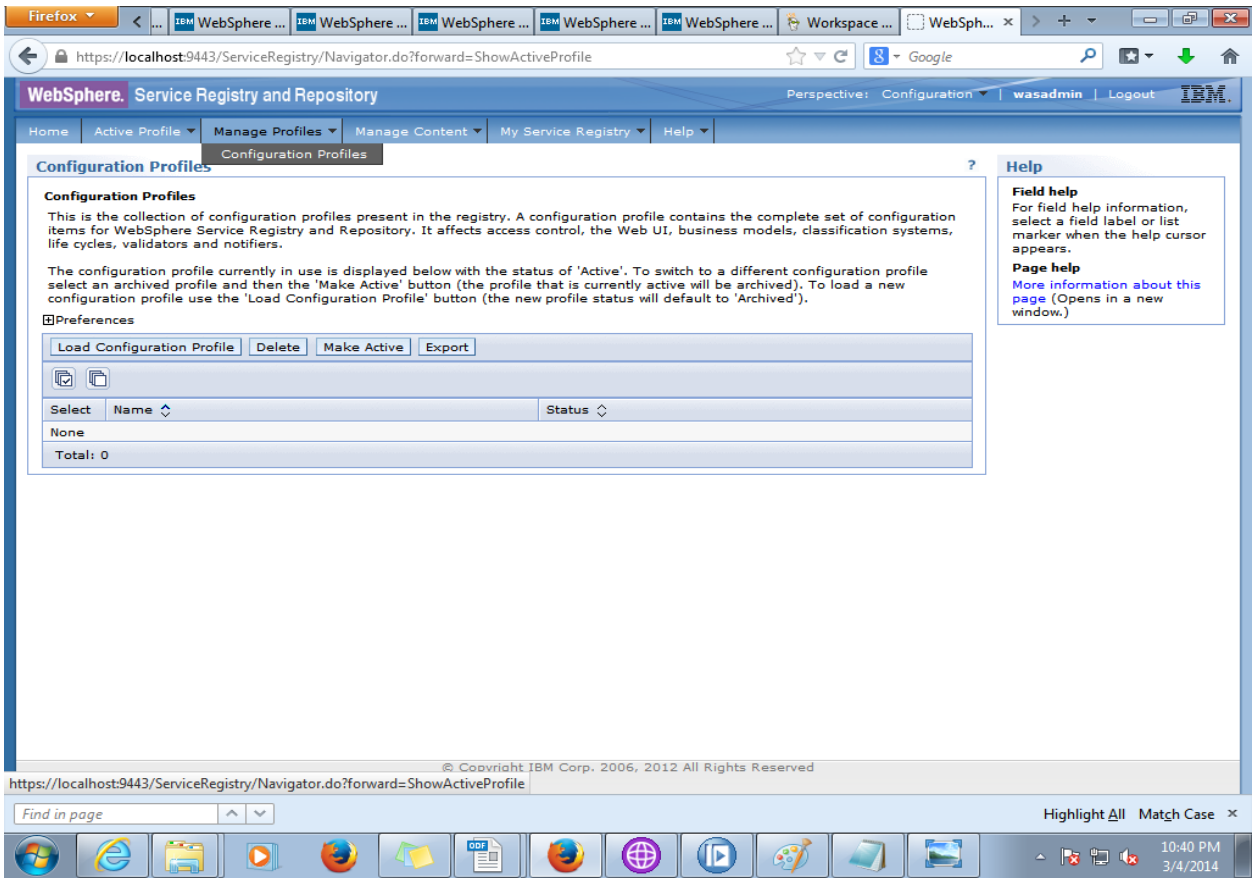
A WSRR configuration profile contains a complete set of WSRR configuration files. Configuration profiles are used for backup, restore and management of entire sets of WSRR configuration.
A WSRR configuration profile contains:
Web UI views
To display registry content in a way that is suitable for its intended use.
Roles and perspectives
User roles, and web UI perspectives to support those user roles.
Classification systems
To support your user-defined models, business domains and technical domains of relevance.
Lifecycles
Appropriate to the service lifecycle and its governance.
Configuration of user-defined validators, modifiers, and notifiers
To implement governance policies.

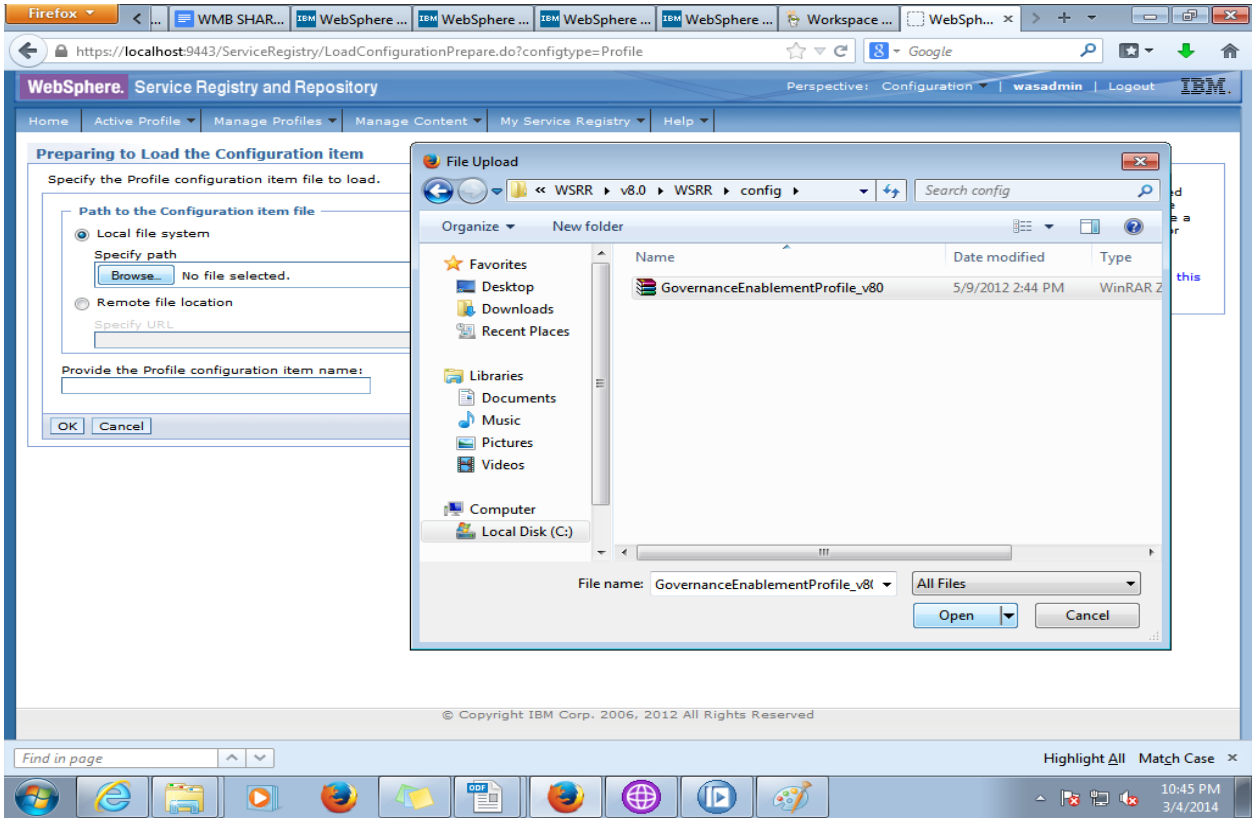
Loading a configuration profile :

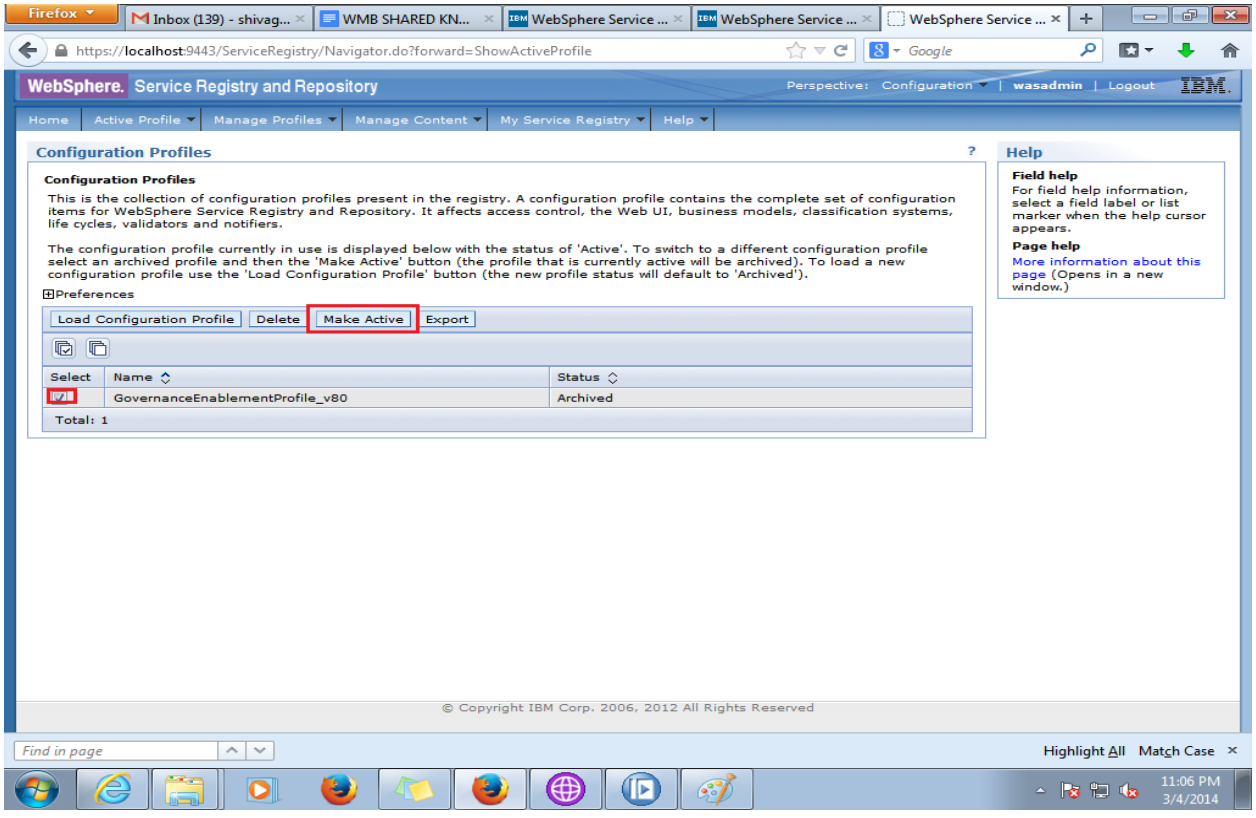
http://pic.dhe.ibm.com/infocenter/sr/v7r0/topic/com.ibm.sr.doc/twsr_configrn_config_profiles_load.html




Firefox
https://localhost:9443/ServiceRegistry/
WebSphere. Service Registry and Repository
Perspective: Configuration | wasadmin | Logout
IBM
Home
Active Profile
Manage Profiles
Manage Content
My Service Registry
Help
IBM WebSphere Service Registry and Repository Configuration
Use this Configuration Perspective to setup and manage the configuration of WebSphere Service Registry and Repository.
Active Profile
The Active Configuration Profile contains the configuration items currently in use in the registry. You can use the Active Profile menu to manage the configuration of Access Control, the Web UI, Business Models, Classification Systems, Life Cycles, E-Mail Notification, UDDI Synchronization, Plug-ins, Promotion Properties, Validators, and Notifiers.
Manage Profiles
A Configuration Profile contains the complete set of configuration items for WebSphere Service Registry and Repository. You can use the Manage Profiles menu to activate, load, and delete Configuration Profiles.
It is recommended to use the tooling provided in WebSphere Service Registry and Repository Studio which is a desktop application designed specifically for configuring WSRR profiles.
Help
WebSphere Service Registry and Repository on IBM.com
Information Center
Web Resources
Support
Library
developerWorks
IBM SOA Foundation: An architectural introduction and overview
Product information on IBM.com
About your Service Registry and Repository
IBM WebSphere Service Registry and Repository, 8.0.0.0
Build Number: 20120509-1049
Licensed Material - Property of IBM
5724-N72 5655-WBS (C) Copyright IBM Corp. 2006, 2012
All Rights Reserved.
U.S. Government users - RESTRICTED RIGHTS - Use, Duplication, or Disclosure restricted by GSA-ADP schedule contract with IBM Corp. IBM is a registered trademark of the IBM Corp.
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
Find in page
Highlight All
Match Case
10:39 PM
3/4/2014

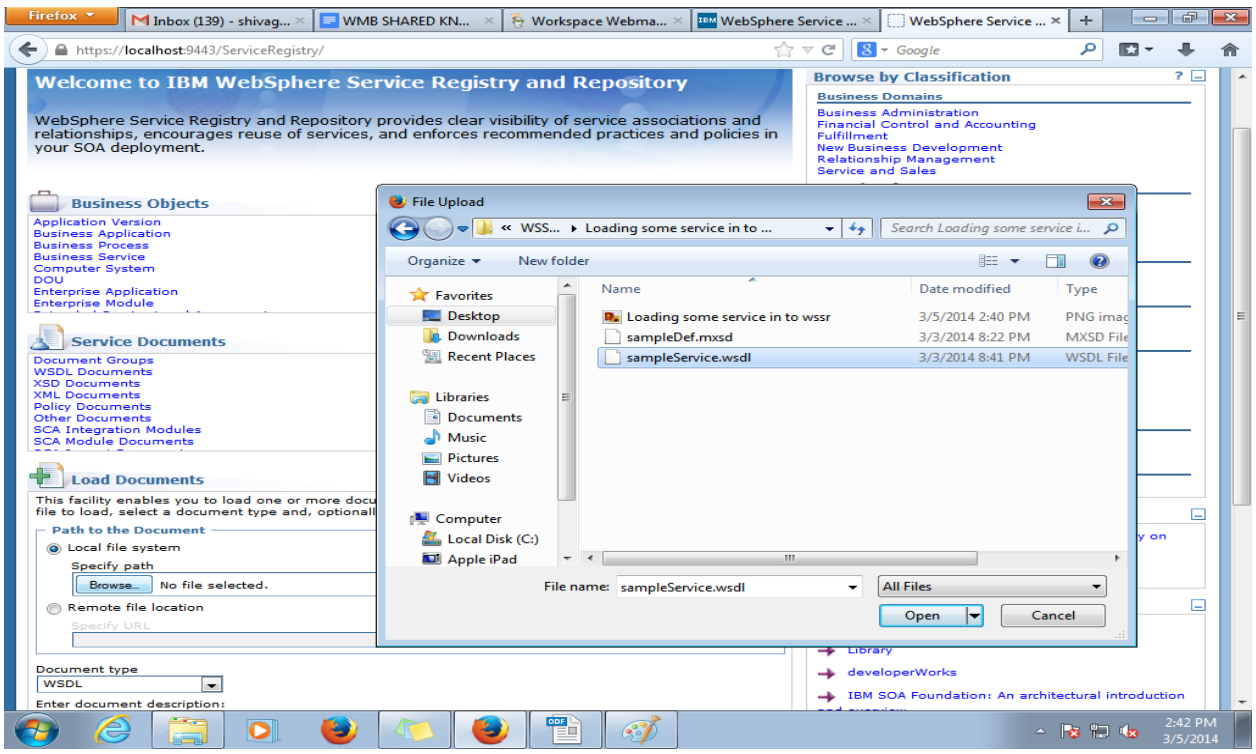

Firefox
https://localhost:9443/ServiceRegistry/Navigator.do?forward=ShowActiveProfile
WebSphere. Service Registry and Repository
Perspective: Configuration | wasadmin | Logout
IBM
Home
Active Profile
Manage Profiles
Manage Content
My Service Registry
Help
Configuration Profiles
Configuration Profiles
Configuration Profiles
This is the collection of configuration profiles present in the registry. A configuration profile contains the complete set of configuration items for WebSphere Service Registry and Repository. It affects access control, the Web UI, business models, classification systems, life cycles, validators and notifiers.
The configuration profile currently in use is displayed below with the status of 'Active'. To switch to a different configuration profile select an archived profile and then the 'Make Active' button (the profile that is currently active will be archived). To load a new configuration profile use the 'Load Configuration Profile' button (the new profile status will default to 'Archived').
Preferences
Load Configuration Profile
Delete
Make Active
Export
Select
Name
Status
None
Total: 0
Help
Field help
For field help information, select a field label or list marker when the help cursor appears.
Page help
More information about this page (Opens in a new window.)
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
https://localhost:9443/ServiceRegistry/Navigator.do?forward=ShowActiveProfile
Find in page
Highlight All
Match Case
10:40 PM
3/4/2014

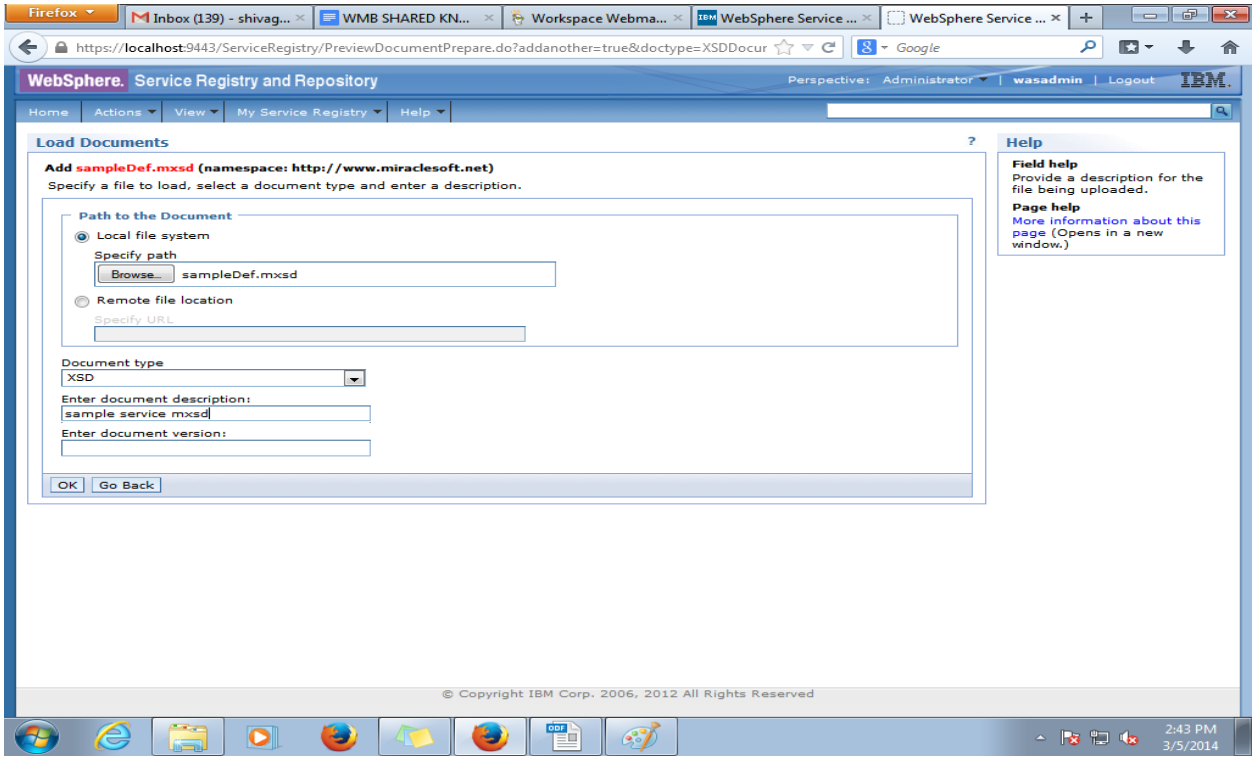

WebSphere. Service Registry and Repository
Perspective: Configuration | wasadmin | Logout
Home | Active Profile | Manage Profiles | Manage Content | My Service Registry | Help
Preparing to Load the Configuration item
Specify the Profile configuration item file to load.
Path to the Configuration item file
Local file system
Specify path
Browse...
No file selected.
Remote file location
Specify URL
Provide the Profile configuration item name:
OK
Cancel
File Upload
« WSRR ▸ v8.0 ▸ WSRR ▸ config ▸
Search config
Organize
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
Local Disk (C:)
Name
Date modified
Type
GovernanceEnablementProfile_v80
5/9/2012 2:44 PM
WinRAR Z
File name: GovernanceEnablementProfile_v8(
All Files
Open
Cancel
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
Find in page
Highlight All
Match Case
10:45 PM
3/4/2014


WebSphere. Service Registry and Repository
Perspective: Configuration | wasadmin | Logout
Home | Active Profile | Manage Profiles | Manage Content | My Service Registry | Help
Configuration Profiles
Configuration Profiles
This is the collection of configuration profiles present in the registry. A configuration profile contains the complete set of configuration items for WebSphere Service Registry and Repository. It affects access control, the Web UI, business models, classification systems, life cycles, validators and notifiers.
The configuration profile currently in use is displayed below with the status of 'Active'. To switch to a different configuration profile select an archived profile and then the 'Make Active' button (the profile that is currently active will be archived). To load a new configuration profile use the 'Load Configuration Profile' button (the new profile status will default to 'Archived').
Preferences
Load Configuration Profile
Delete
Make Active
Export
Select
Name
Status
GovernanceEnablementProfile_v80
Archived
Total: 1
Help
Field help
For field help information, select a field label or list marker when the help cursor appears.
Page help
More information about this page (Opens in a new window.)
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
Find in page
Highlight All
Match Case
11:06 PM
3/4/2014

Loading Some Service(wsdl) in to WSSR :

Configure WSDL in WSRR




Firefox
Inbox (139) - shivag...
WMB SHARED KN...
Workspace Webma...
WebSphere Service ...
WebSphere Service ...
https://localhost:9443/ServiceRegistry/PreviewDocumentPrepare.do?addanother=true&doctype=XSDDocur
Google
WebSphere. Service Registry and Repository
Perspective: Administrator | wasadmin | Logout
IBM.
Home
Actions
View
My Service Registry
Help
Load Documents
Add sampleDef.mxsd (namespace: http://www.miraclesoft.net)
Specify a file to load, select a document type and enter a description.
Path to the Document
Local file system
Specify path
Browse...
sampleDef.mxsd
Remote file location
Specify URL
Document type
XSD
Enter document description:
sample service mxsd
Enter document version:
OK
Go Back
Help
Field help
Provide a description for the file being uploaded.
Page help
More information about this page (Opens in a new window.)
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
2:43 PM
3/5/2014


Firefox
Inbox (139) - shivag...
WMB SHARED KN...
Workspace Webma...
WebSphere Service ...
WebSphere Service ...
https://localhost:9443/ServiceRegistry/PreviewDocumentAdd.do
Google
WebSphere. Service Registry and Repository
Perspective: Administrator | wasadmin | Logout
IBM.
Home
Actions
View
My Service Registry
Help
Load Documents
Documents to be Loaded
When all required documents are listed below select either 'Finish' to complete the load or 'Save as a group' if you want to refer to these documents as a document group.
Add Another Document
Finish
Save as a Group
Cancel
sampleService.wsdl (ready to load) | Remove | Replace |
sampleDef.mxsd (ready to load) | Remove | Replace |
Add Another Document
Finish
Save as a Group
Cancel
Help
Field help
For field help information, select a field label or list marker when the help cursor appears.
Page help
More information about this page (Opens in a new window.)
© Copyright IBM Corp. 2006, 2012 All Rights Reserved
2:44 PM
3/5/2014

WSRR_INTEGRATION_WITH_MB

Scenario : Based on input coming into our consumer flow , We will get the corresponding Porttype and Name Space and Version details ,which are configured in WSRR , and based on end point URL our flow need to request the Service Provider and get the response.

Accessing a secure WSRR repository from MB

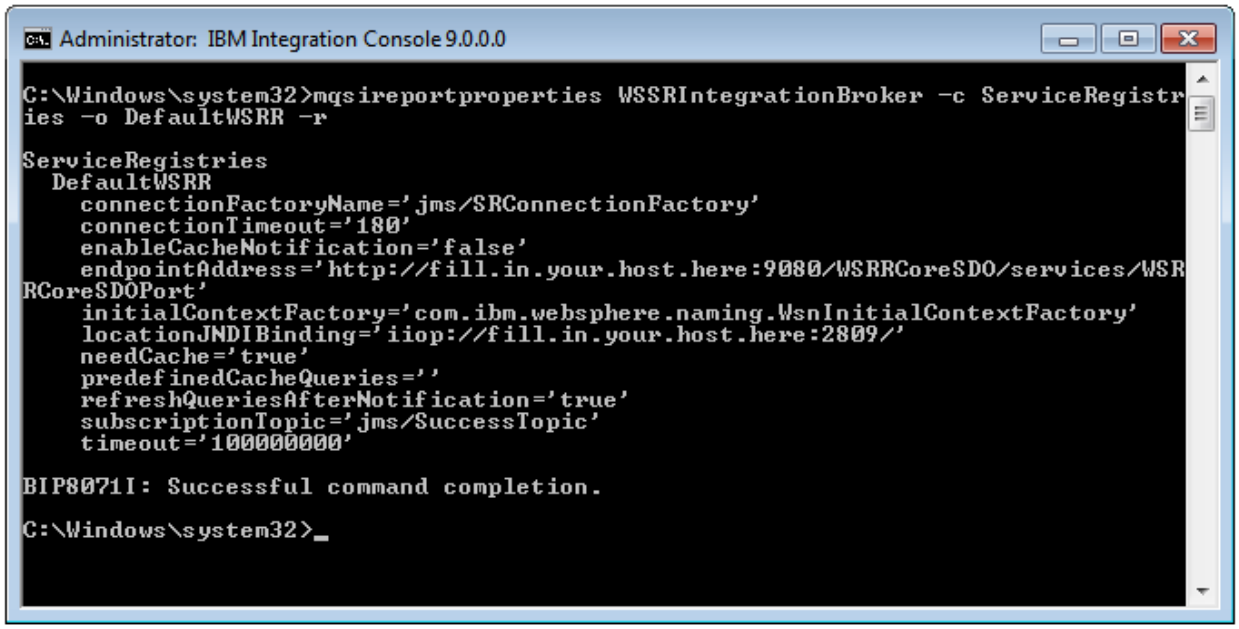
To access a secure WebSphere® Service Registry and Repository (WSRR) repository, set the configuration parameters by using the mqsichangeproperties command.

You must connect over HTTPS, not HTTP, which is specified in the endpointAddress configuration parameter of the default WSRR profile, **DefaultWSRR**.

To access a secure WebSphere Service Registry and Repository, enter the following sequence of commands:

1. Ensure that the broker is running. If it is not, use the mqsistart command to start it.
2. Use the ServiceRegistries configurable service to configure the broker to use HTTPS to communicate with the WSRR server. You can view the current configuration parameters for the ServiceRegistries configurable service by using the following command:

mqsireportproperties WSSRIntegrationBroker -c ServiceRegistries -o DefaultWSRR -r

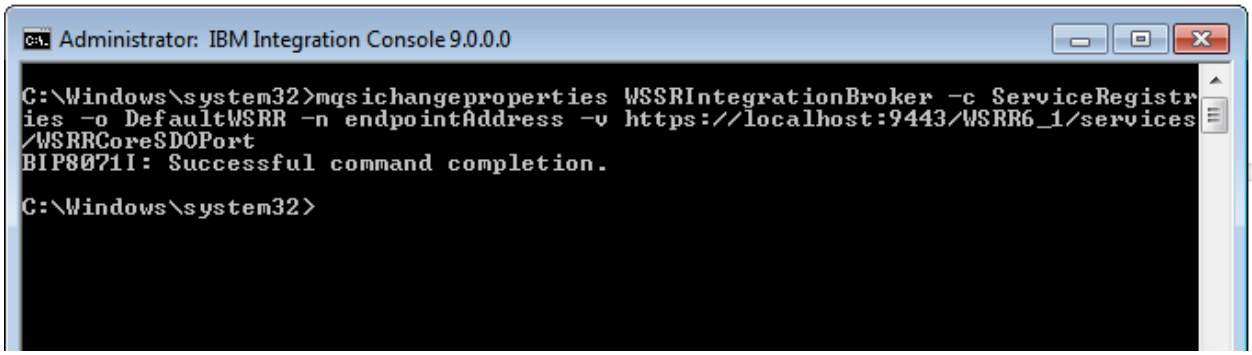


where:
-c specifies the configurable service (in this case, ServiceRegistries)

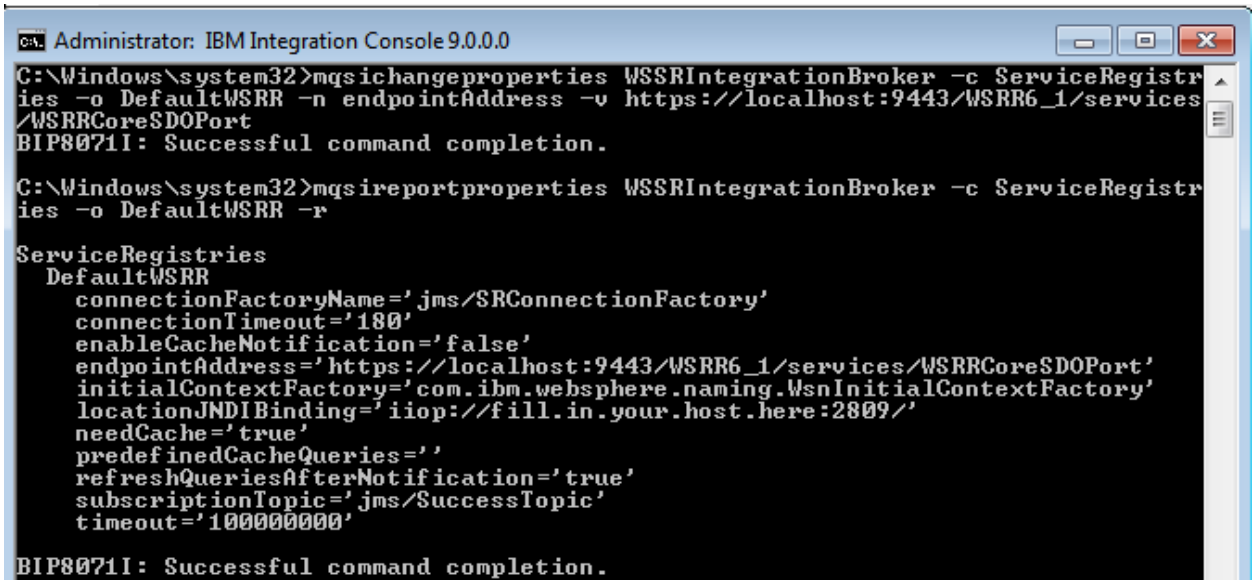
-o specifies the name of the object (in this case, BrokerRegistry)
-r specifies that all property values of the object are displayed, including the child values, if appropriate.

To change the endpointAddress configuration parameter to specify HTTPS and the secure port for the DefaultWSRR of the ServiceRegistries configurable service, use the following command.

mqsichangeproperties WSRRIntegration -c ServiceRegistries -o DefaultWSRR -n endpointAddress -v https://172.17.3.161:9443/WSRR6_1/services/WSRRCoreSDOPort



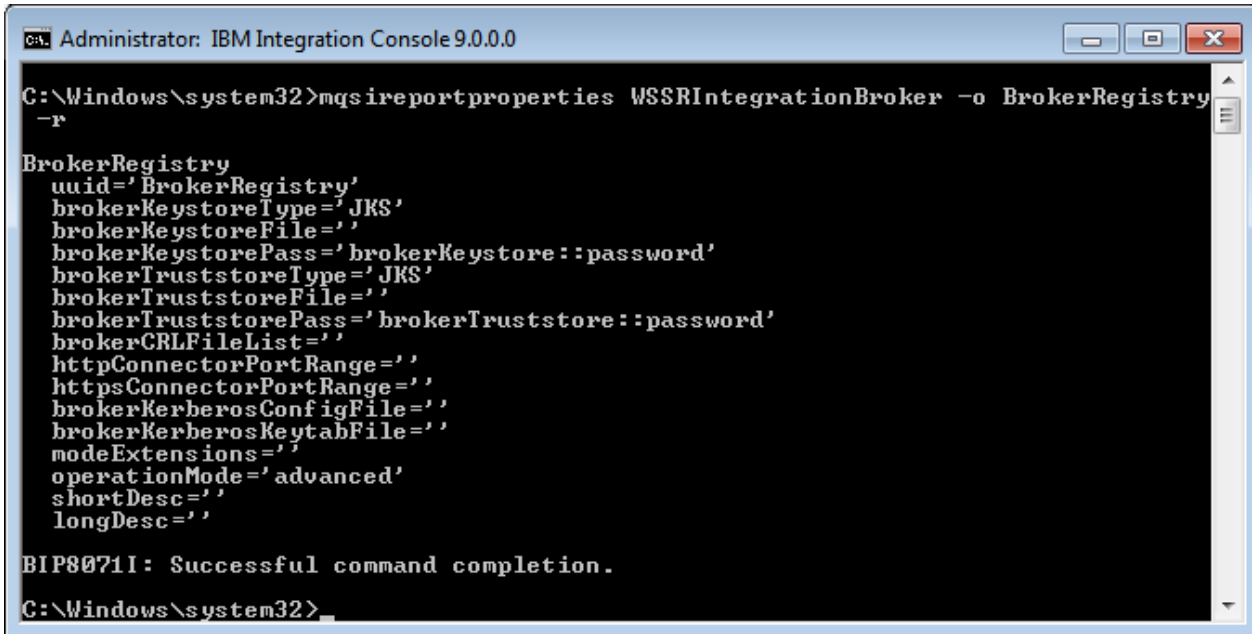
-n specifies the names of the properties to be changed
(in this case, endpointAddress)
-v specifies the values of properties defined by the -n parameter
(in this case, https://localhost:9443/WSRR6_1/services/WSRRCoreSDOPort)



3. Configure the broker keystore to contain your WSRR server certificate keys;Obtain these certificate keys from the installation of the WebSphere Application Server that hosts your WSRR server. The broker uses a single keystore, therefore, if your broker also

implements WS-Security, HTTPS, or SSL-secured WebSphere MQ, you might need to merge the provided keys into an existing keystore file. The broker keystore is configured by using the mqsichangeproperties command to change configuration parameters for the broker. Display the current configuration parameters of the broker by using the following command:

mqsireportproperties WSRRIntegration -o BrokerRegistry -r

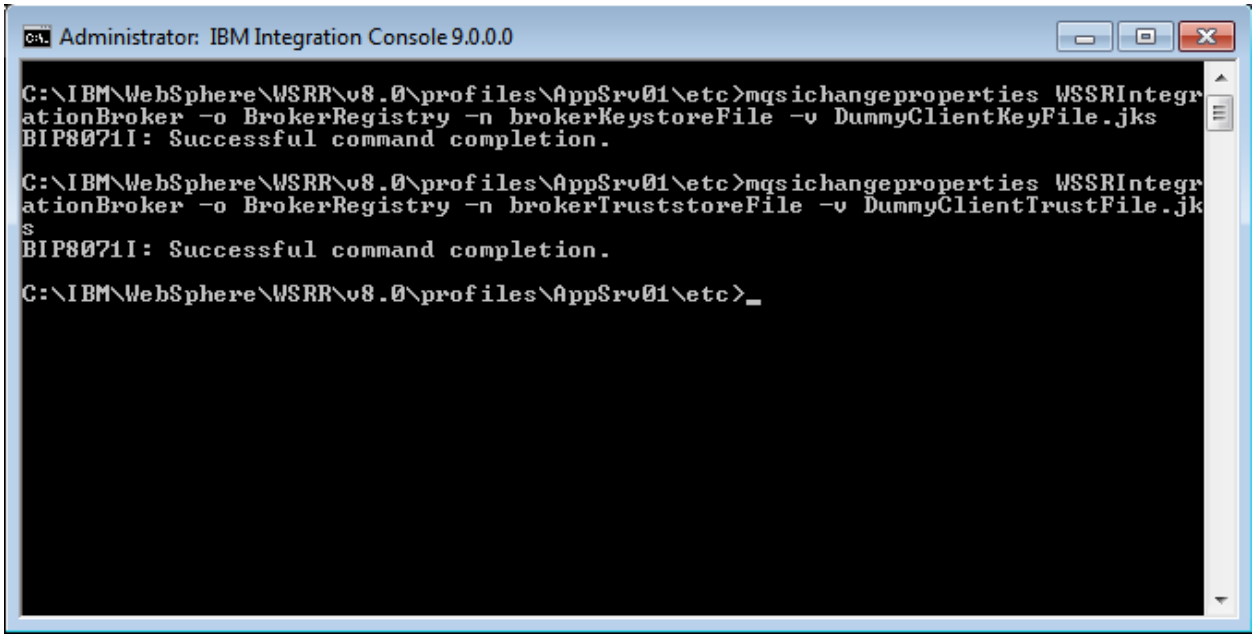


To change the brokerKeystoreFile configuration parameters for the broker, use the following command:

mqsichangeproperties WSRRIntegration -o BrokerRegistry -n brokerKeystoreFile -v C:\IBM\WebSphere\WSRR\v8.0\profiles\WSRRSrv01\etc\DummyClientKeyFile.jks

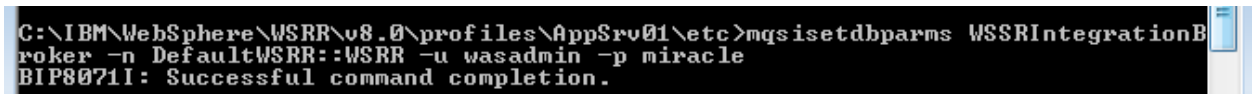
4. Configure the broker truststore to contain signer certificates for your WSRR server. As described previously for the keystore, the broker uses a single truststore, therefore certificates might need to be merged into an existing truststore file. The broker truststore is configured by using the mqsichangeproperties command. To change the brokerTruststoreFile configuration parameters for the broker, use the following command:

mqsichangeproperties WSRRIntegration -o BrokerRegistry -n brokerTruststoreFile -v C:\IBM\WebSphere\WSRR\v8.0\profiles\WSRRSrv01\etc\DummyClientTrustFile.jks



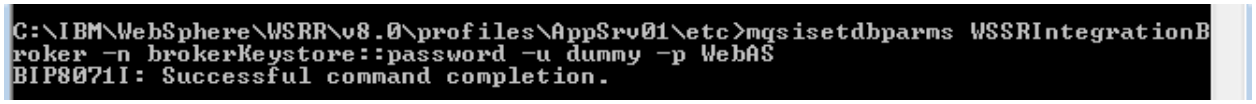
5. Stop the broker by using the mqsistop command. You must stop the broker to complete the following step.
6. Set the WebSphere Application Server user name and password by using the following command:

mqsisetdbparms WSRRIntegration -n DefaultWSRR::WSRR -u wasadmin -p miracle



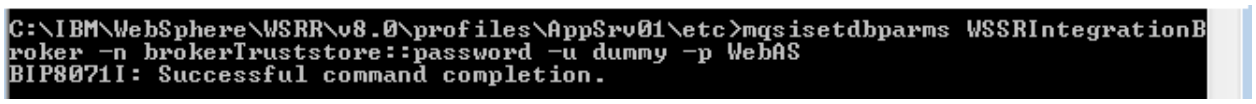
7. Set the brokerKeystore user name and password by using the following command:

mqsisetdbparms WSRRIntegration -n brokerKeystore::password -u dummy -p WebAS



8. Set the brokerTrustStore user name and password by using the following command:

mqsisetdbparms WSRRIntegration -n brokerTruststore::password -u dummy -p WebAS



9. To use cache notification with your secure WSRR server, you must specify a valid user ID and password for the broker to use when connecting to the WSRR JMS cache notification topic. To set the user ID and password that the broker will use to make the JMS cache notification connection, use the following command:

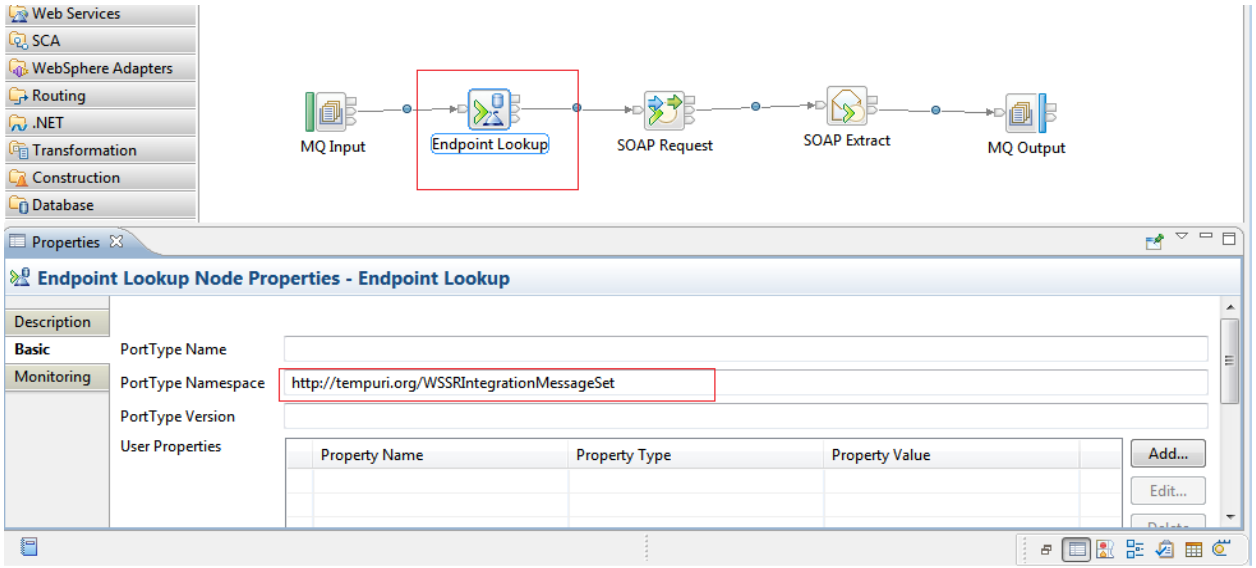
```
C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\etc>mqsisetdbparms WSSRIntegrationB
roker -n jms::DefaultWSRR@jms/SRConnectionFactory -u miracle -p miracle
BIP8071I: Successful command completion.
```

10. Restart the broker by using the mqsistart command.

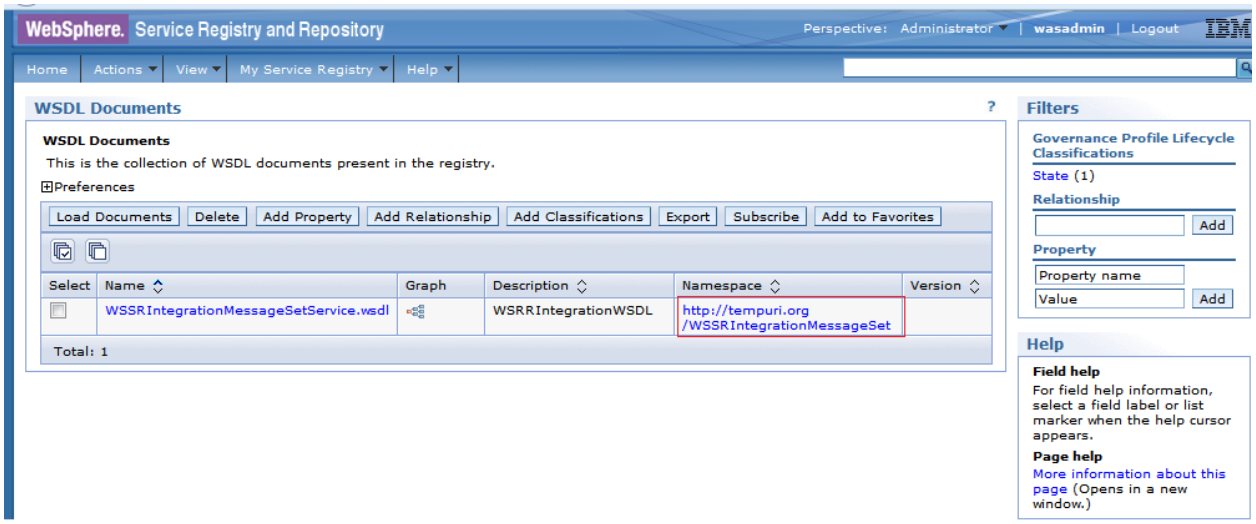
```
C:\IBM\WebSphere\WSRR\v8.0\profiles\AppSrv01\etc>mqsistart WSSRIntegrationBroker

BIP8096I: Successful command initiation, check the system log to ensure that the
 component started without problem and that it continues to run without problem.
```

Message Flow :

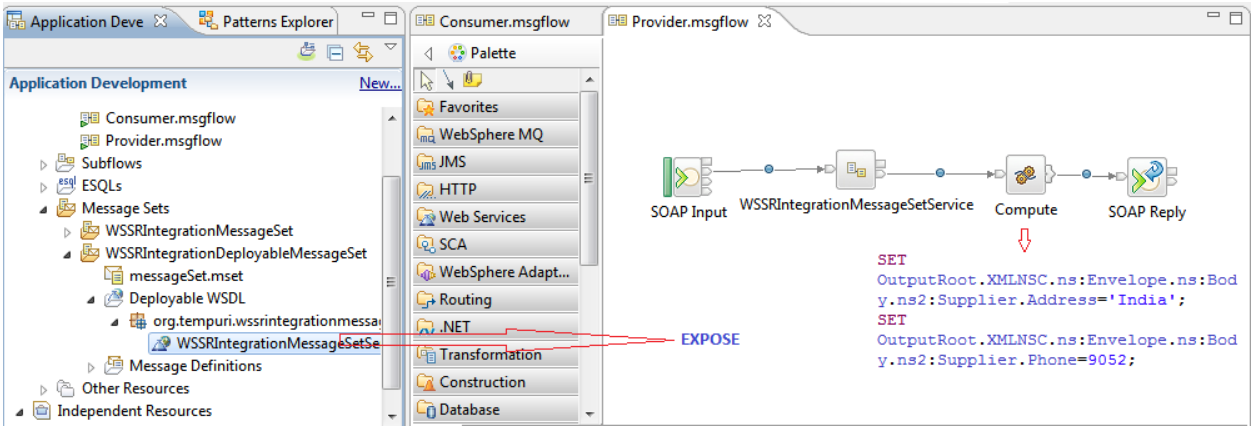
Consumer flow:



Provider ,wsdl lookup from wssr:



provider flow:



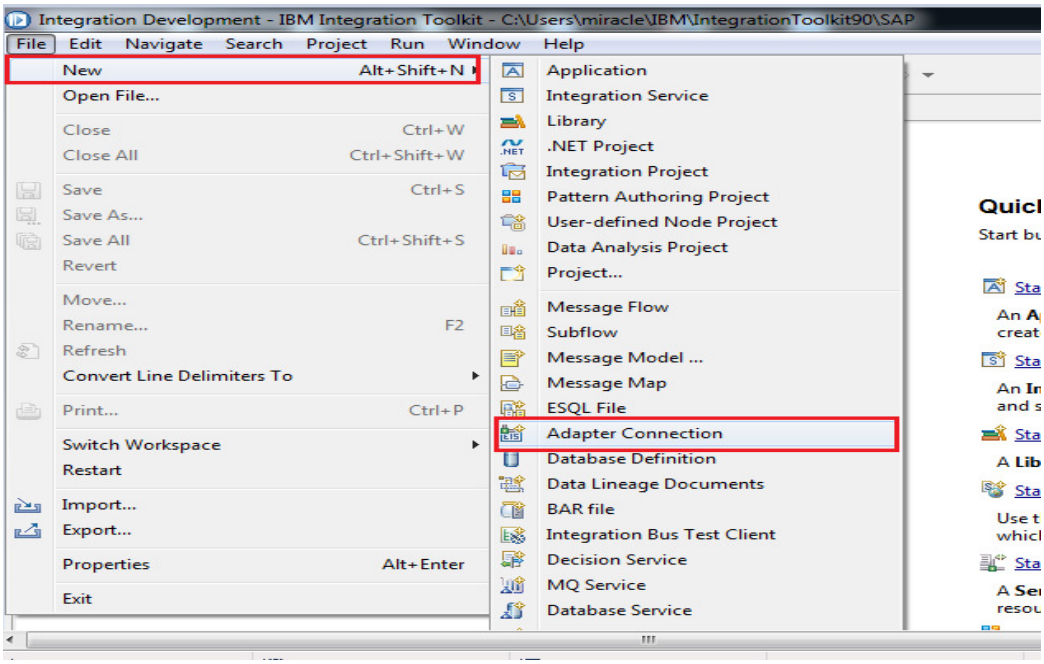
SAP INTEGRATION_ WITH_MB

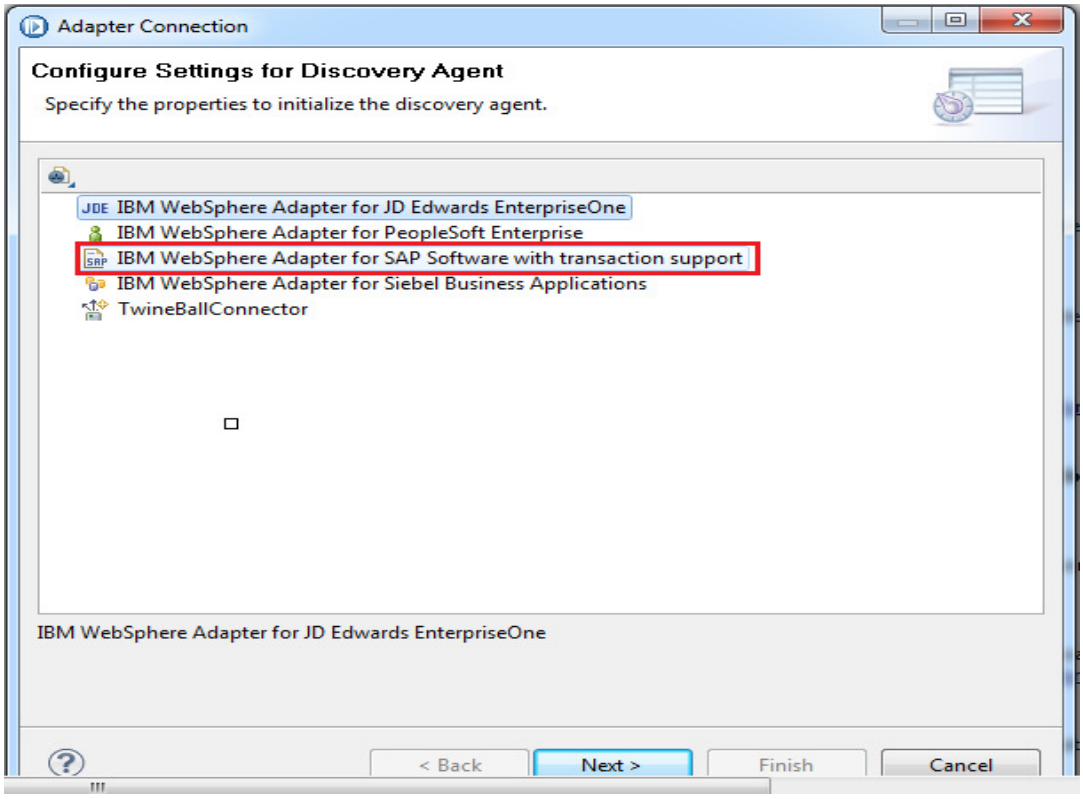
Setting SAP Jars to MB:

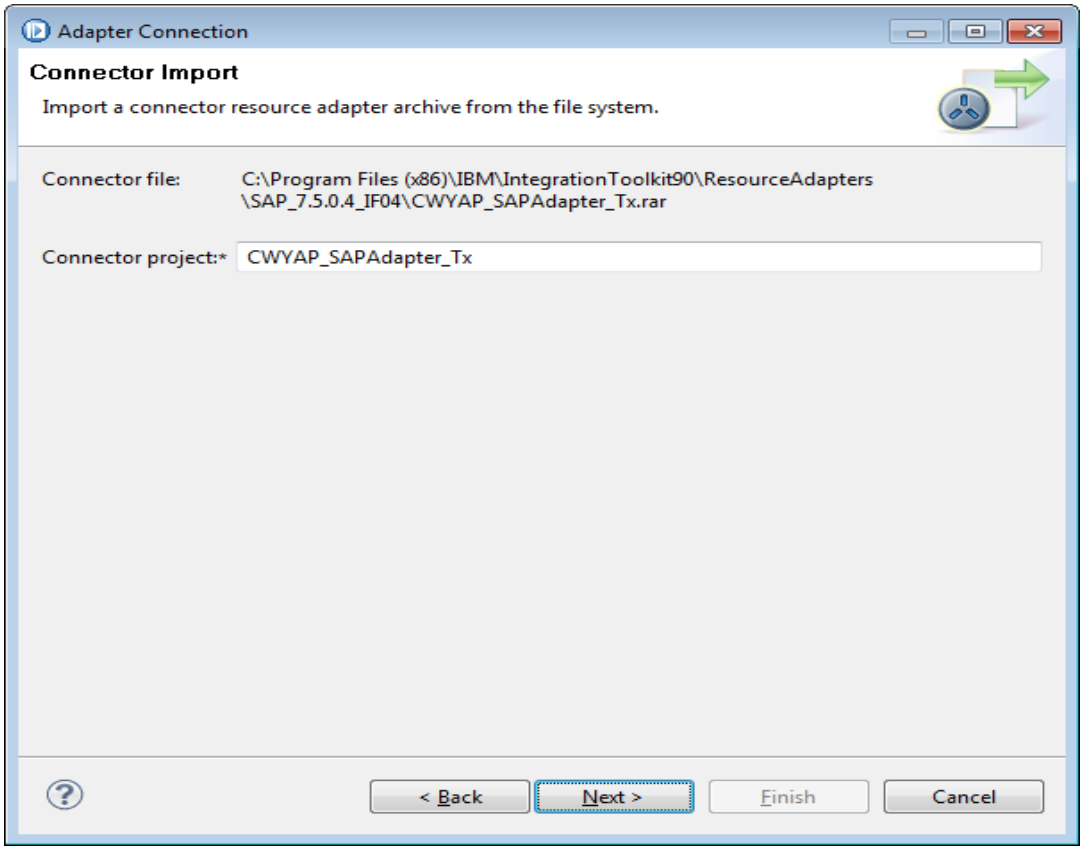
mqsichangeproperties *(BrokerName)* -c EISProviders -o SAP -n nativeLibs -v "C:\Users\miracle\Desktop\WMB Scrap\SAP Jars\SAP64"

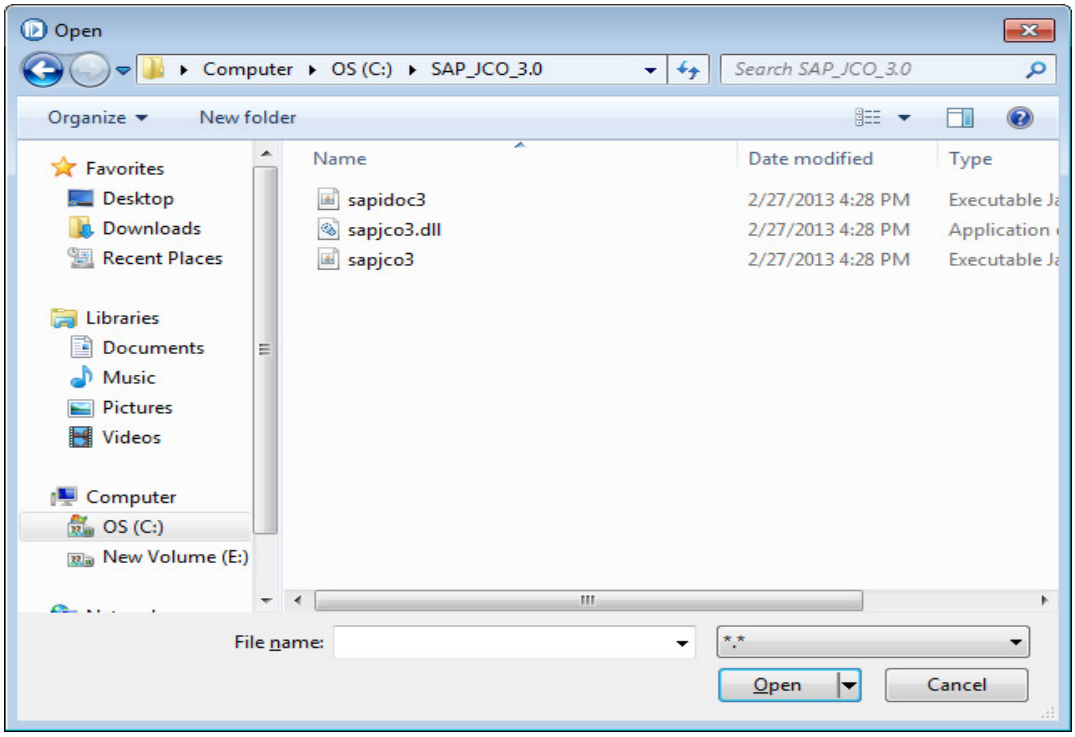
mqsichangeproperties *(BrokerName)* -c EISProviders -o SAP -n jarsURL -v "C:\Users\miracle\Desktop\WMB Scrap\SAP Jars\SAP64"

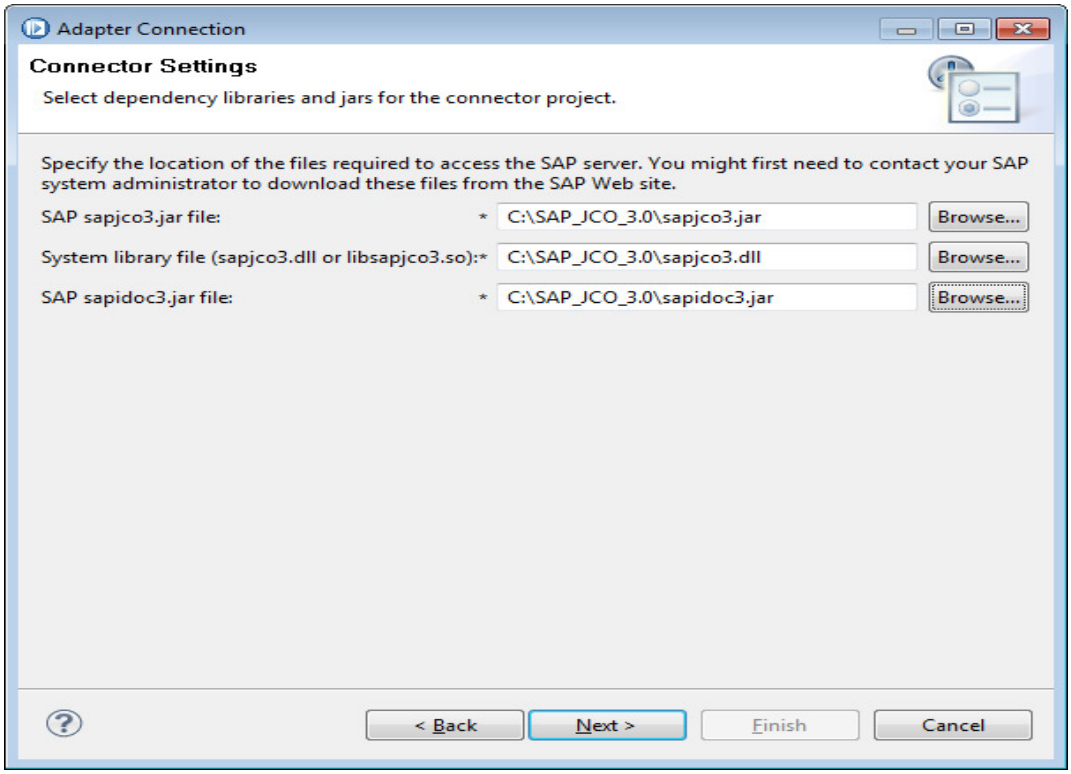
Procedure:

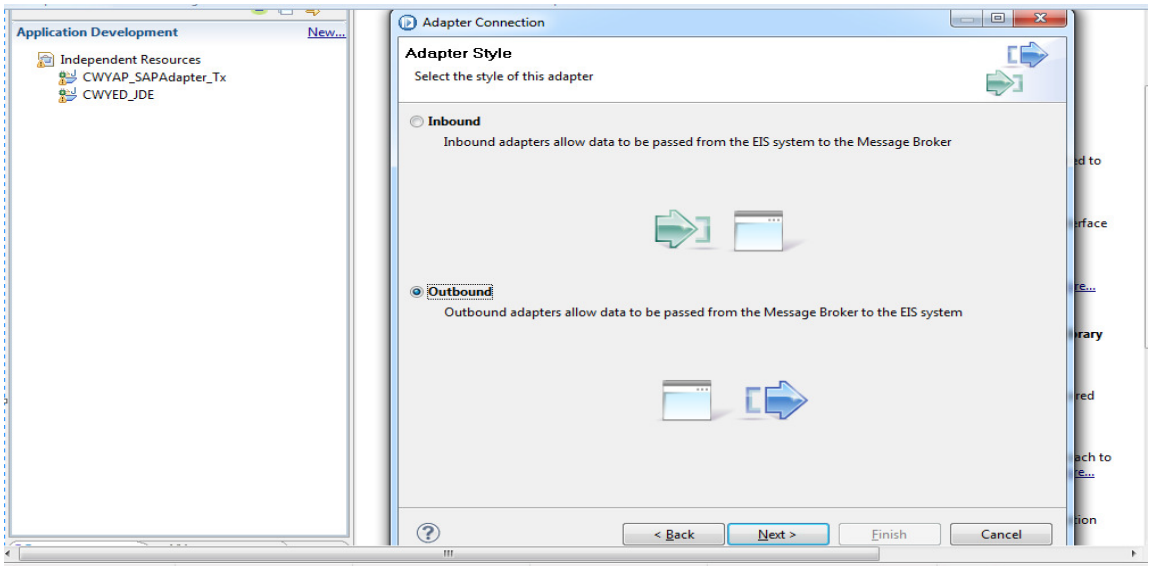


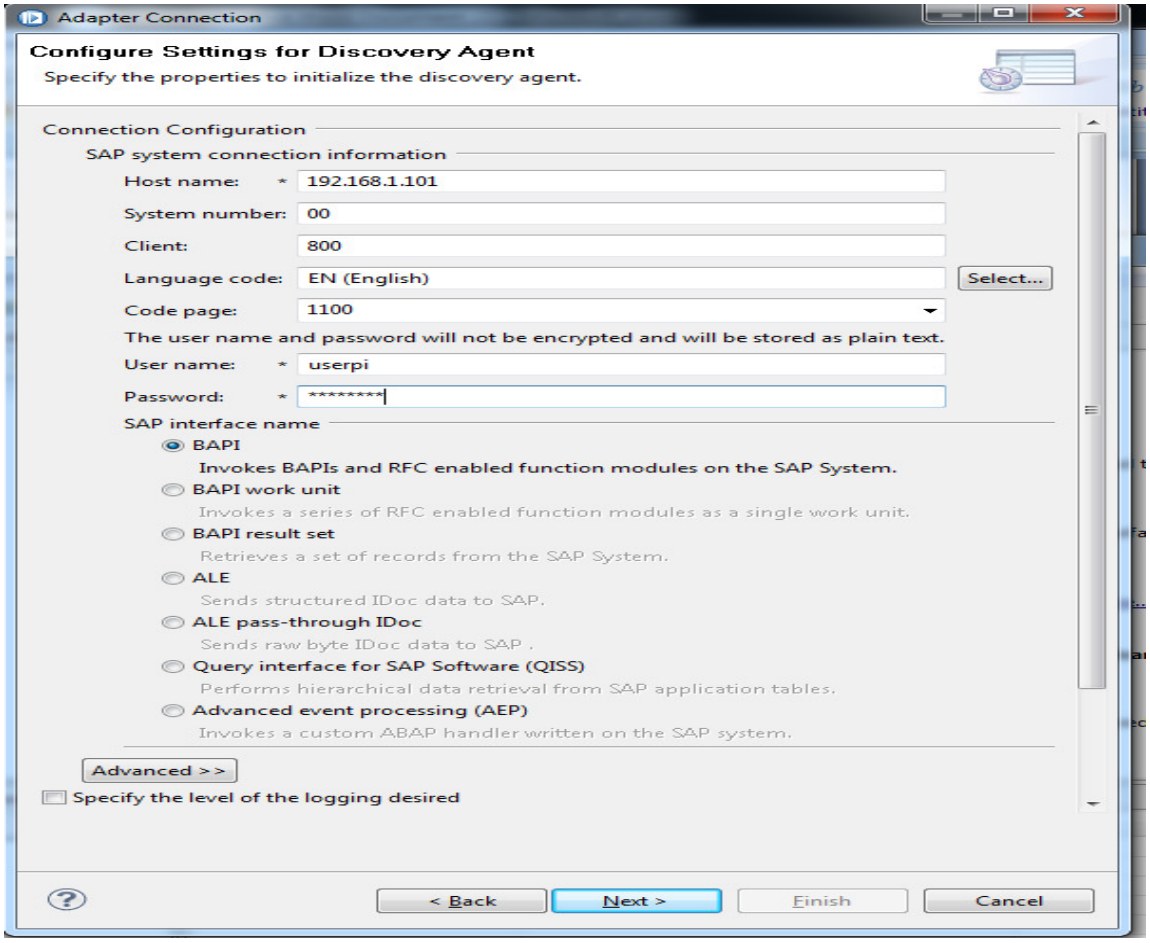

Adapter Connection
Configure Settings for Discovery Agent
Specify the properties to initialize the discovery agent.
IBM WebSphere Adapter for JD Edwards EnterpriseOne
IBM WebSphere Adapter for PeopleSoft Enterprise
IBM WebSphere Adapter for SAP Software with transaction support
IBM WebSphere Adapter for Siebel Business Applications
TwineBallConnector
IBM WebSphere Adapter for JD Edwards EnterpriseOne
< Back
Next >
Finish
Cancel

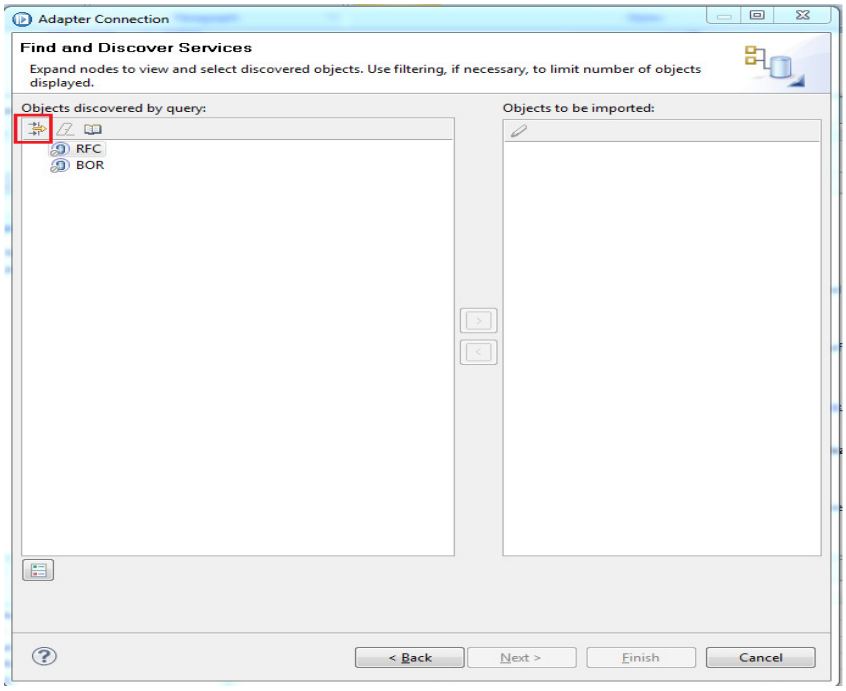

Adapter Connection
Connector Import
Import a connector resource adapter archive from the file system.
Connector file:
C:\Program Files (x86)\IBM\IntegrationToolkit90\ResourceAdapters
\SAP_7.5.0.4_IF04\CWYAP_SAPAdapter_Tx.rar
Connector project:*
CWYAP_SAPAdapter_Tx
< Back
Next >
Finish
Cancel

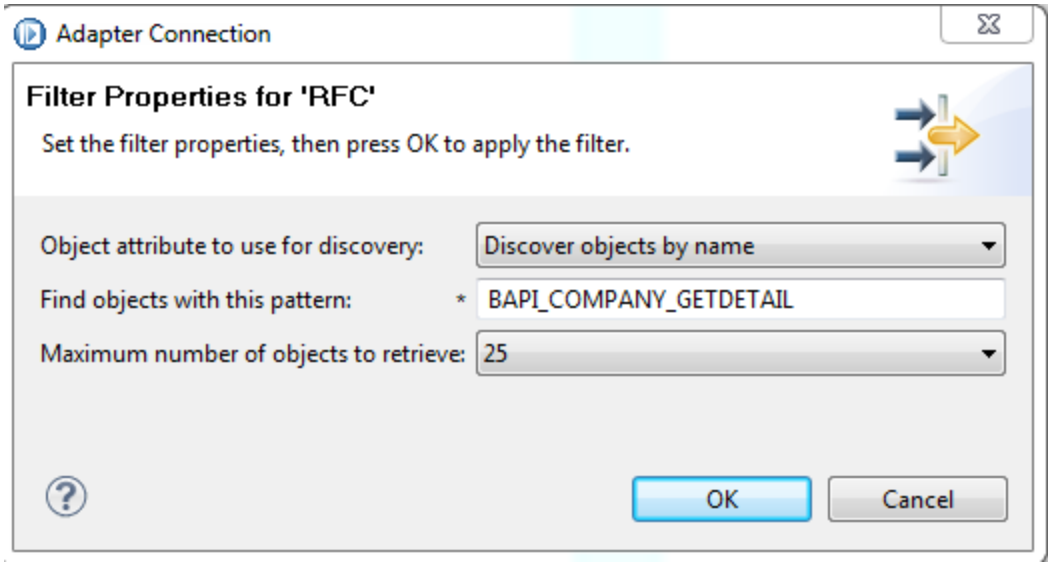

Open
Computer ▸ OS (C:) ▸ SAP_JCO_3.0
Search SAP_JCO_3.0
Organize
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries
Documents
Music
Pictures
Videos
Computer
OS (C:)
New Volume (E:)
Name
Date modified
Type
sapidoc3
2/27/2013 4:28 PM
sapjco3.dll
2/27/2013 4:28 PM
sapjco3
2/27/2013 4:28 PM
File name:
.
Open
Cancel

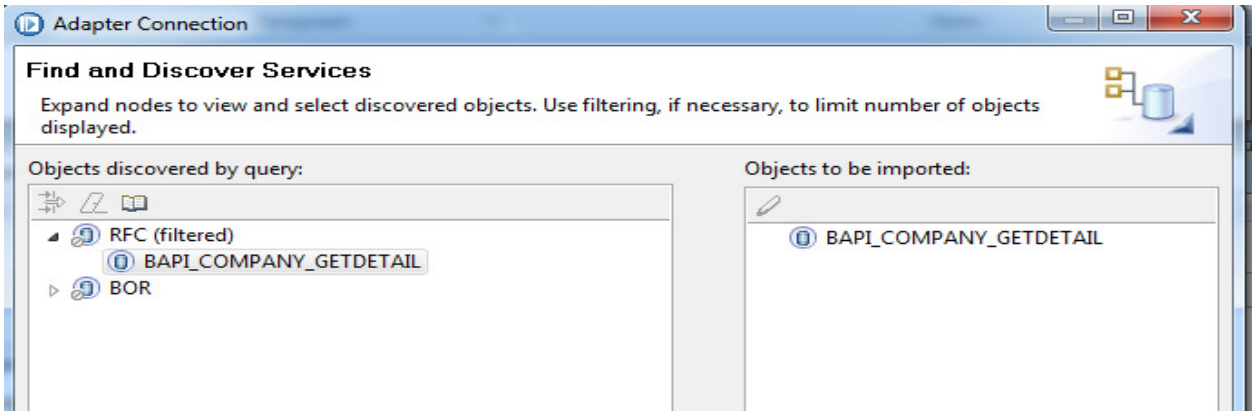

Adapter Connection
Connector Settings
Select dependency libraries and jars for the connector project.
Specify the location of the files required to access the SAP server. You might first need to contact your SAP system administrator to download these files from the SAP Web site.
SAP sapjco3.jar file: * C:\SAP_JCO_3.0\sapjco3.jar Browse...
System library file (sapjco3.dll or libsapjco3.so):* C:\SAP_JCO_3.0\sapjco3.dll Browse...
SAP sapidoc3.jar file: * C:\SAP_JCO_3.0\sapidoc3.jar Browse...
< Back
Next >
Finish
Cancel

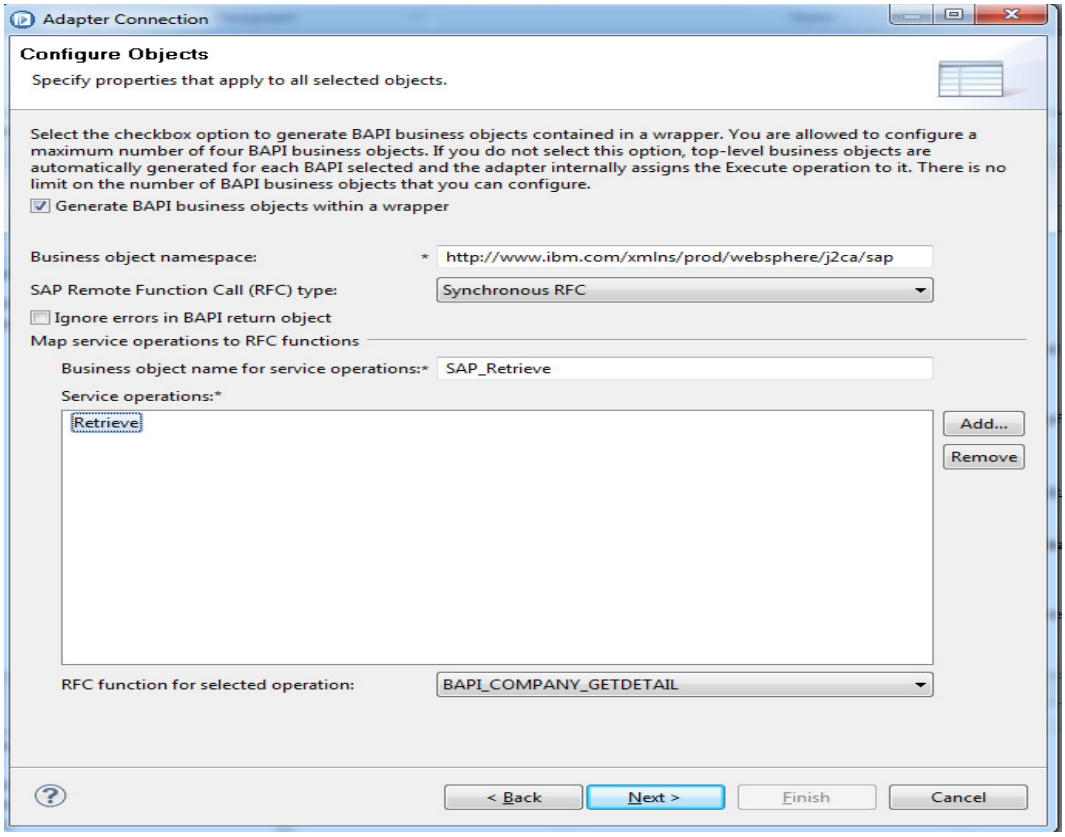

Application Development
New...
Independent Resources
CWYAP_SAPAdapter_Tx
CWYED_JDE
Adapter Connection
Adapter Style
Select the style of this adapter
Inbound
Inbound adapters allow data to be passed from the EIS system to the Message Broker
Outbound
Outbound adapters allow data to be passed from the Message Broker to the EIS system
< Back
Next >
Finish
Cancel

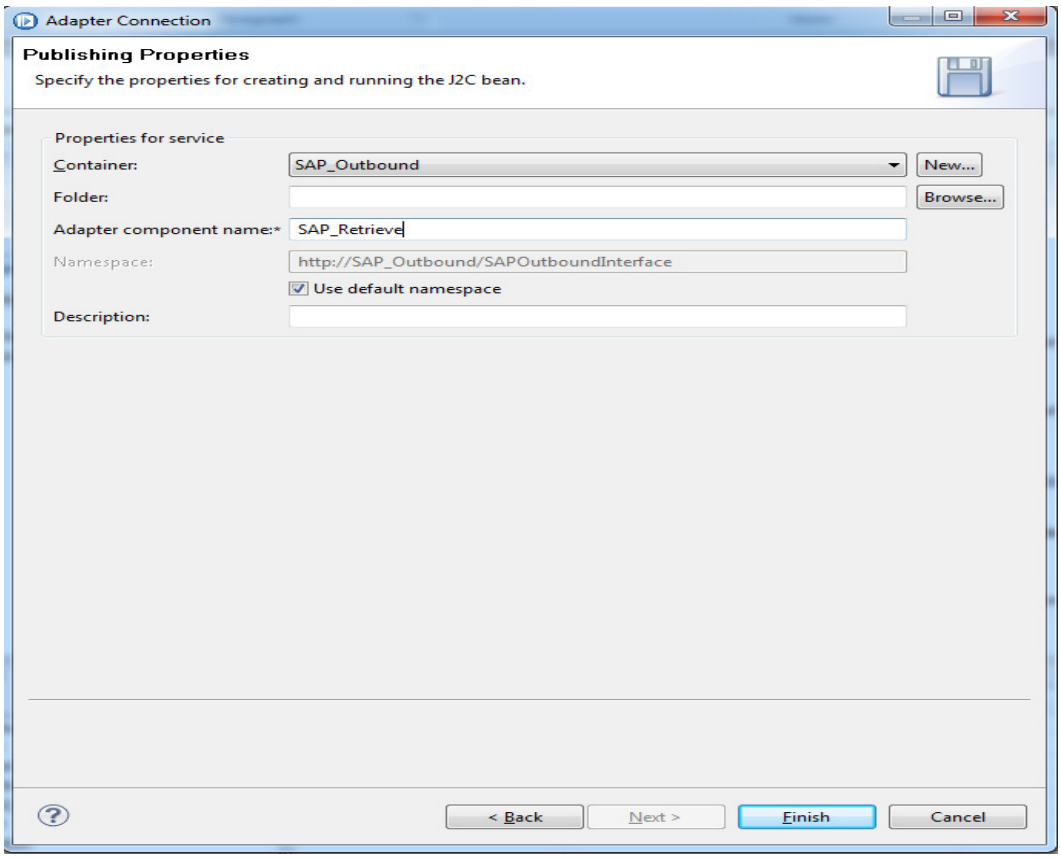

Adapter Connection
Configure Settings for Discovery Agent
Specify the properties to initialize the discovery agent.
Connection Configuration
SAP system connection information
Host name: * 192.168.1.101
System number: 00
Client: 800
Language code: EN (English)
Select...
Code page: 1100
The user name and password will not be encrypted and will be stored as plain text.
User name: * userpi
Password: * ********
SAP interface name
BAPI
Invokes BAPIs and RFC enabled function modules on the SAP System.
BAPI work unit
Invokes a series of RFC enabled function modules as a single work unit.
BAPI result set
Retrieves a set of records from the SAP System.
ALE
Sends structured IDoc data to SAP.
ALE pass-through IDoc
Sends raw byte IDoc data to SAP .
Query interface for SAP Software (QISS)
Performs hierarchical data retrieval from SAP application tables.
Advanced event processing (AEP)
Invokes a custom ABAP handler written on the SAP system.
Advanced >>
Specify the level of the logging desired
< Back
Next >
Finish
Cancel

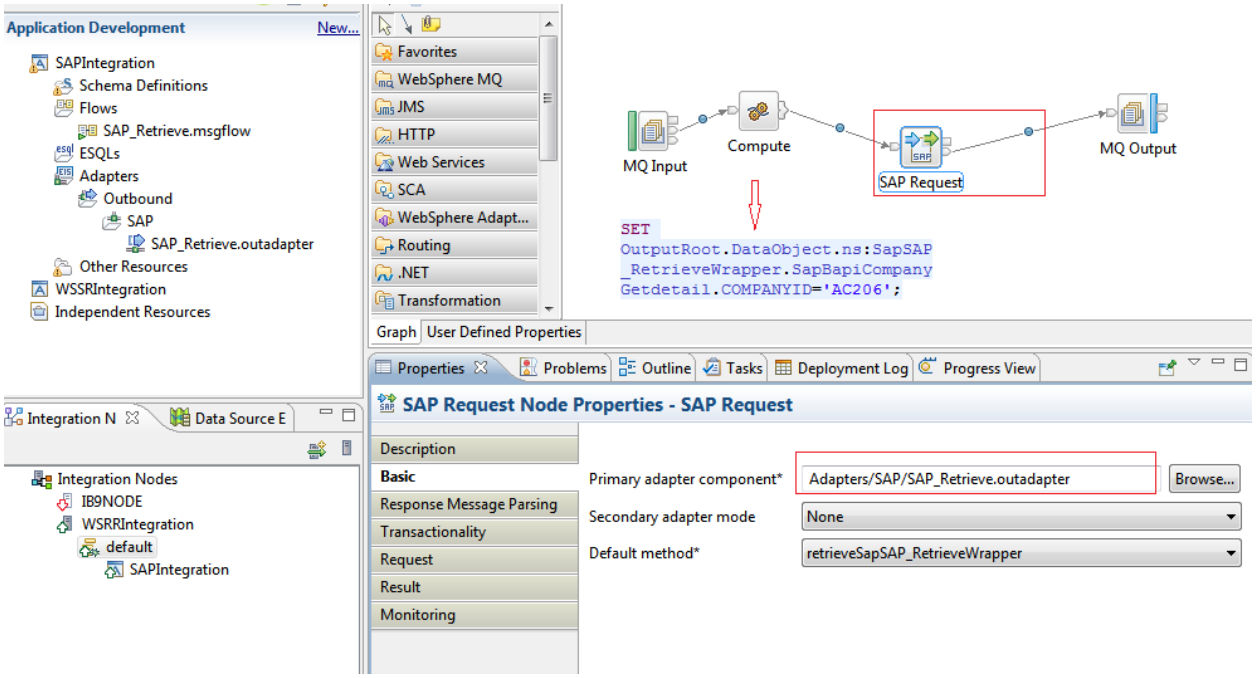

Adapter Connection
Find and Discover Services
Expand nodes to view and select discovered objects. Use filtering, if necessary, to limit number of objects displayed.
Objects discovered by query:
Objects to be imported:
RFC
BOR
< Back
Next >
Finish
Cancel


Adapter Connection
Filter Properties for 'RFC'
Set the filter properties, then press OK to apply the filter.
Object attribute to use for discovery:
Discover objects by name
Find objects with this pattern:
*
BAPI_COMPANY_GETDETAIL
Maximum number of objects to retrieve:
25
OK
Cancel


Adapter Connection
Find and Discover Services
Expand nodes to view and select discovered objects. Use filtering, if necessary, to limit number of objects displayed.
Objects discovered by query:
Objects to be imported:
RFC (filtered)
BAPI_COMPANY_GETDETAIL
BOR
BAPI_COMPANY_GETDETAIL

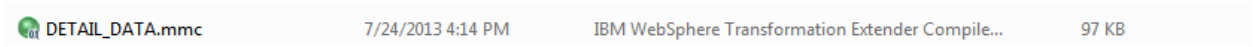

Adapter Connection
Configure Objects
Specify properties that apply to all selected objects.
Select the checkbox option to generate BAPI business objects contained in a wrapper. You are allowed to configure a maximum number of four BAPI business objects. If you do not select this option, top-level business objects are automatically generated for each BAPI selected and the adapter internally assigns the Execute operation to it. There is no limit on the number of BAPI business objects that you can configure.
Generate BAPI business objects within a wrapper
Business object namespace: * http://www.ibm.com/xmlns/prod/websphere/j2ca/sap
SAP Remote Function Call (RFC) type: Synchronous RFC
Ignore errors in BAPI return object
Map service operations to RFC functions
Business object name for service operations:* SAP_Retrieve
Service operations:*
Retrieve
Add...
Remove
RFC function for selected operation: BAPI_COMPANY_GETDETAIL
< Back
Next >
Finish
Cancel


Adapter Connection
Publishing Properties
Specify the properties for creating and running the J2C bean.
Properties for service
Container: SAP_Outbound
New...
Folder:
Browse...
Adapter component name:* SAP_Retrieve
Namespace: http://SAP_Outbound/SAPOutboundInterface
Use default namespace
Description:
< Back
Next >
Finish
Cancel

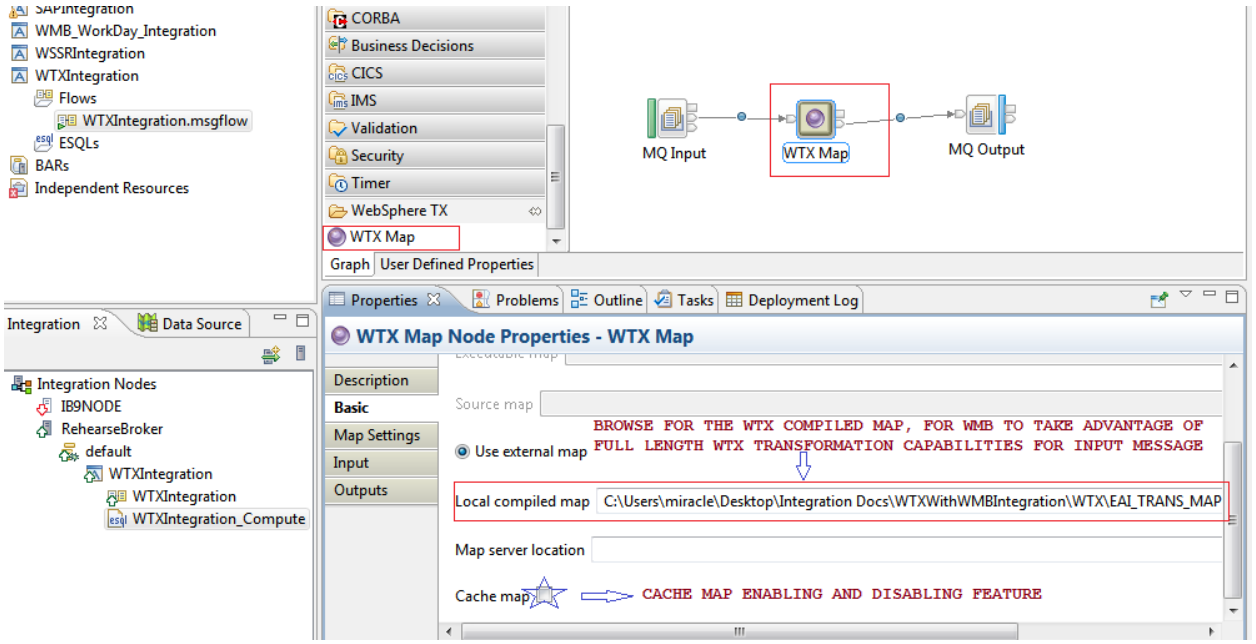


WTX_INTEGRATION_WITH_MB

WTX COMPILED MAP PROVIDED BY WTX TEAM:

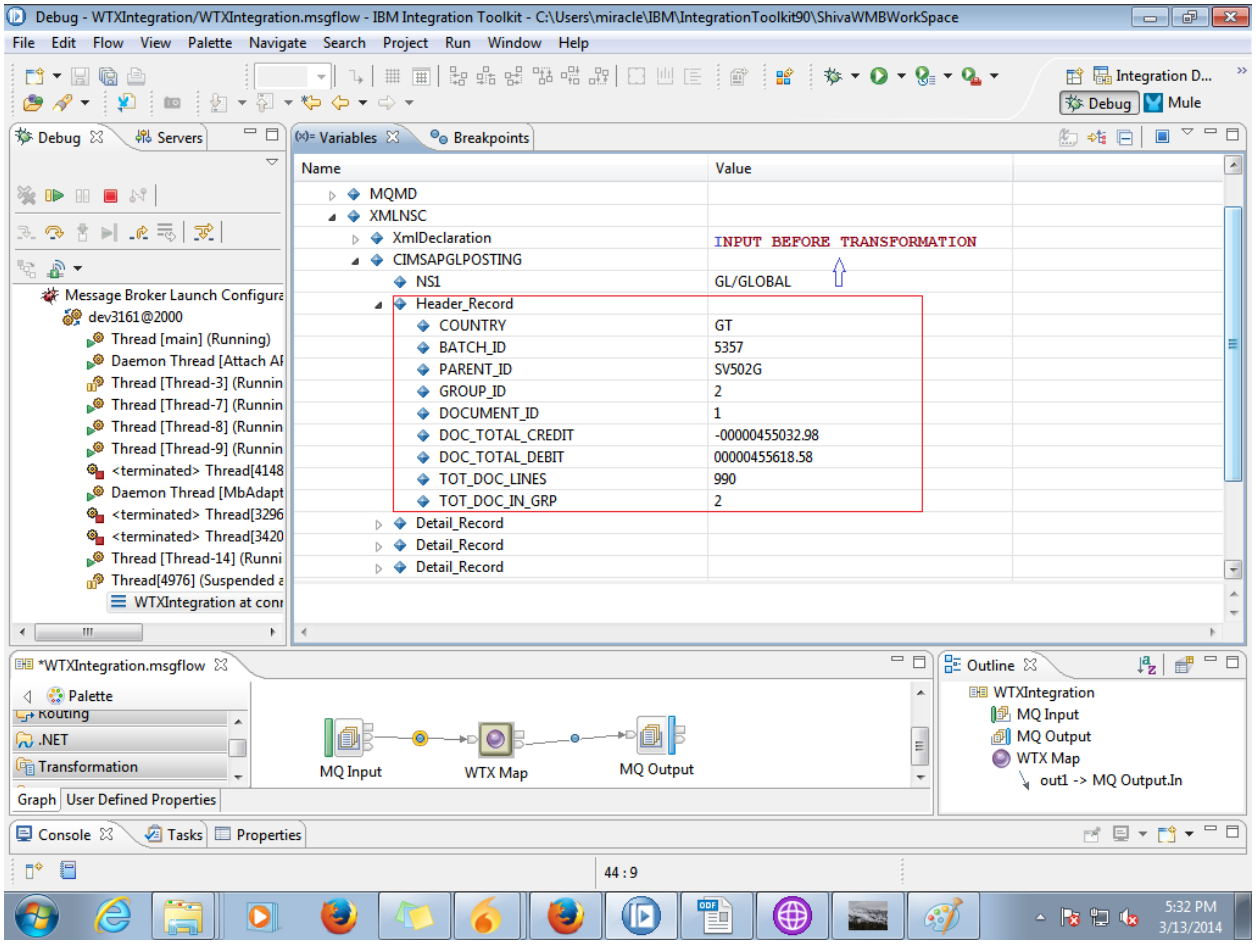


BROWSE :

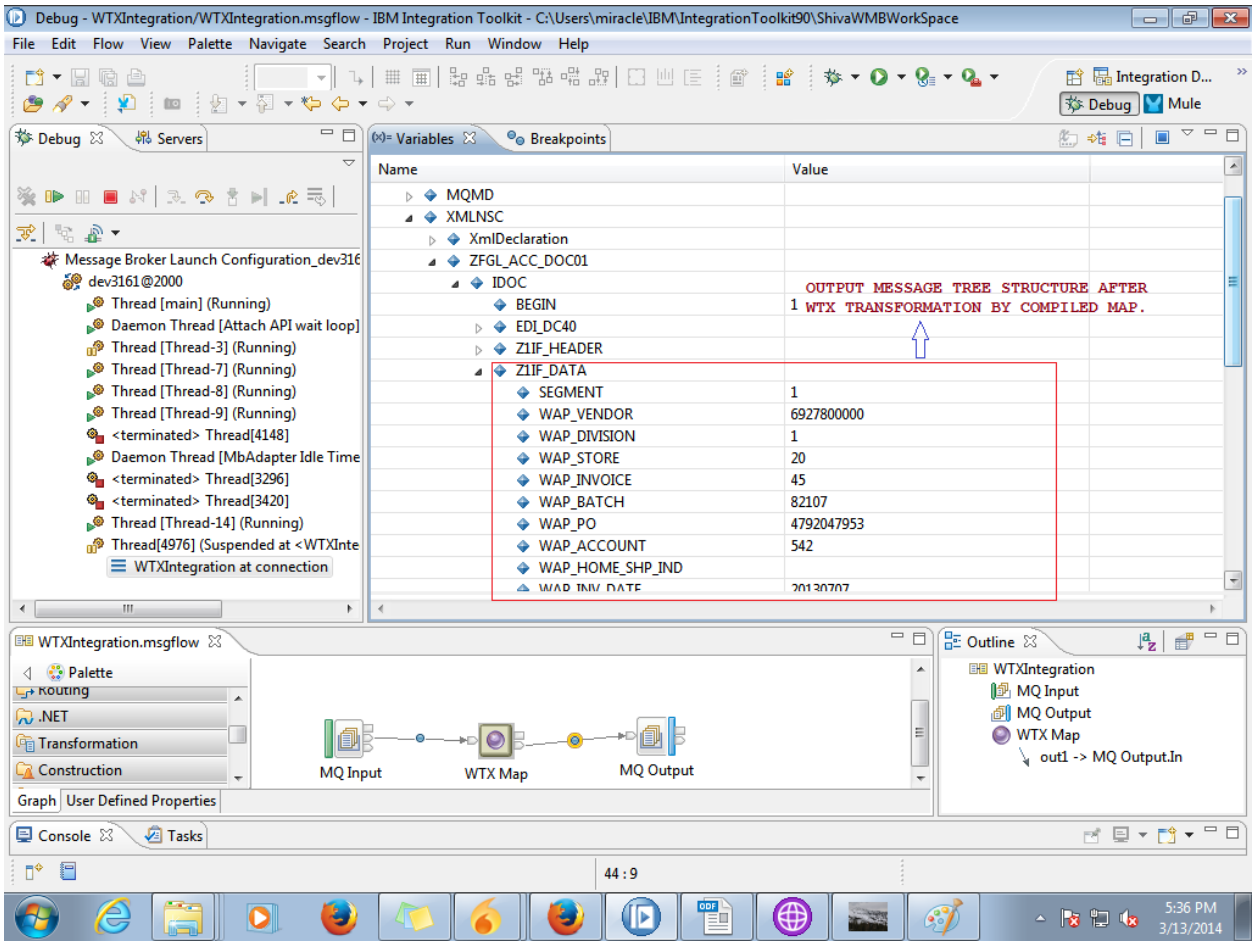


INPUT PROVIDED BY WTX TEAM AND BEFORE WTX TRANSFORMATION:

```
<?xml version="1.0" encoding="utf-8"?>
<NS1:CIMSAPGLPOSTING xmlns:NS1="GL/GLOBAL">
 <Header_Record>
  <NS1:COUNTRY>GT</NS1:COUNTRY>
  <BATCH_ID>5357</BATCH_ID>
  <PARENT_ID>SV502G  </PARENT_ID>
  <GROUP_ID>2</GROUP_ID>
  <DOCUMENT_ID>1</DOCUMENT_ID>
  <DOC_TOTAL_CREDIT>-00000455032.98</DOC_TOTAL_CREDIT>
  <DOC_TOTAL_DEBIT>00000455618.58</DOC_TOTAL_DEBIT>
  <TOT_DOC_LINES>990</TOT_DOC_LINES>
  <TOT_DOC_IN_GRP>2</TOT_DOC_IN_GRP>
 </Header_Record>
 <Detail_Record>
  <VENDOR>6927800000</VENDOR>
  <DIVISION>1</DIVISION>
  <STORE>20</STORE>
  <INVOICE>45</INVOICE>
  <BATCH>82107</BATCH>
  <PO>4792047953</PO>
  <ACCOUNT>542</ACCOUNT>
  <HOME_SHOP_IND> </HOME_SHOP_IND>
  <INV_DATE>20130707</INV_DATE>
  <POST_DATE>20130707</POST_DATE>
  <INV_COST> 00000000197.20</INV_COST>
  <RETL_AMT> 00000000000.00</RETL_AMT>
  <DEPT>0</DEPT>
  <OSI_TR_ID>SV502G  </OSI_TR_ID>
  <COUNTRY>GT</COUNTRY>
  <CURRENCY>GTQ</CURRENCY>
```

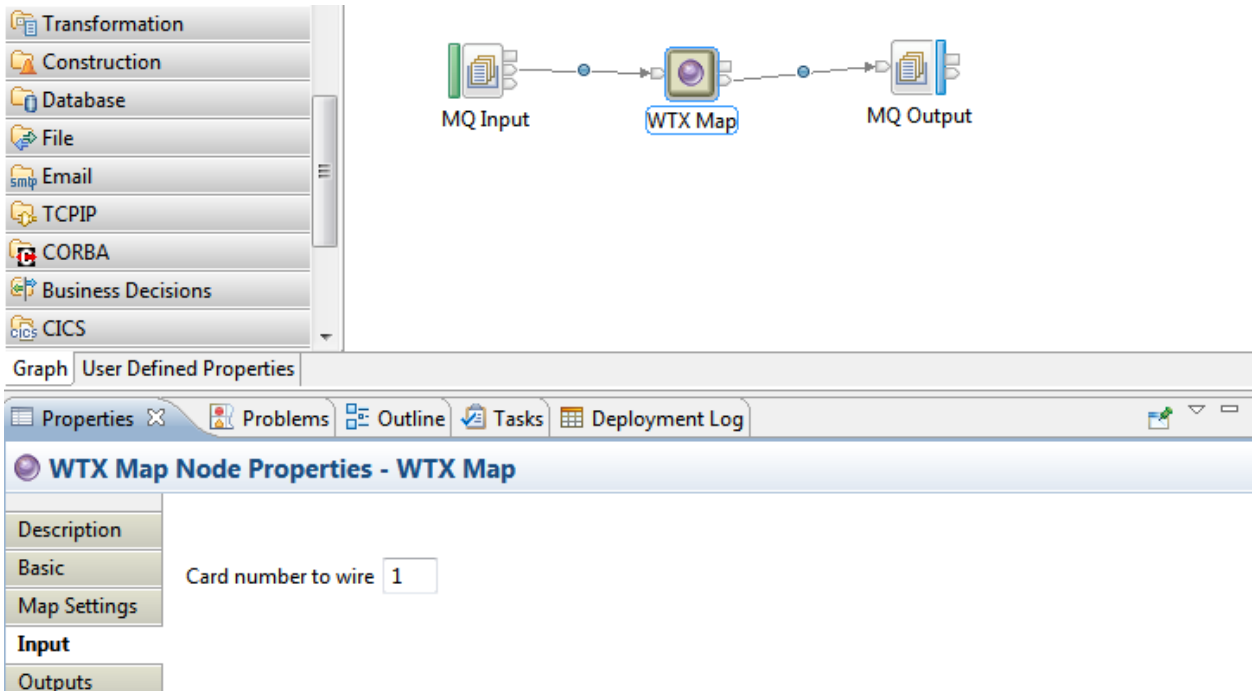


OUTPUT TREE STRUCTURE AFTER WTX TRANSFORMATION:

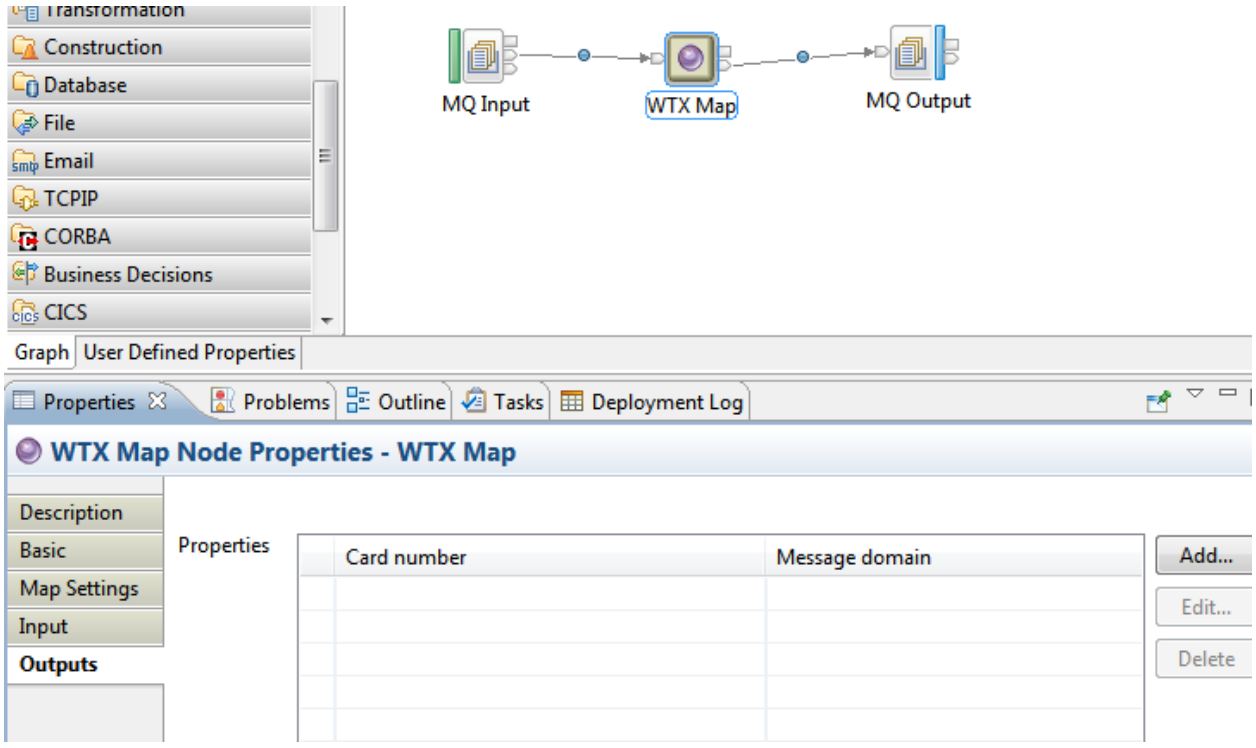



```
  <RCVPOR></RCVPOR>
  <RCVPRN></RCVPRN>
 </EDI_DC40>
 <Z1IF_HEADER SEGMENT="1">
  <CNTRY>GT</CNTRY>
  <BTCH_ID>5357</BTCH_ID>
  <GRP_ID>2</GRP_ID>
  <DOC_ID>1</DOC_ID>
  <PARNT_ID>SV502G</PARNT_ID>
  <DOC_CR>-00000455032.98</DOC_CR>
  <DOC_DR>00000455618.58</DOC_DR>
  <GRP_DOC_CNT>2</GRP_DOC_CNT>
  <DOC_REC_CNT>990</DOC_REC_CNT>
 </Z1IF_HEADER>
 <Z1IF_DATA SEGMENT="1">
  <WAP_VENDOR>6927800000</WAP_VENDOR>
  <WAP_DIVISION>1</WAP_DIVISION>
  <WAP_STORE>20</WAP_STORE>
  <WAP_INVOICE>45</WAP_INVOICE>
  <WAP_BATCH>82107</WAP_BATCH>
  <WAP_PO>4792047953</WAP_PO>
  <WAP_ACCOUNT>542</WAP_ACCOUNT>
  <WAP_HOME_SHP_IND></WAP_HOME_SHP_IND>
  <WAP_INV_DATE>20130707</WAP_INV_DATE>
  <WAP_POST_DATE>20130707</WAP_POST_DATE>
  <WAP_INV_COST> 00000000197.20</WAP_INV_COST>
  <WAP_RETL_AMT> 00000000000.00</WAP_RETL_AMT>
  <WAP_DEPT>0</WAP_DEPT>
  <WAP_OSI_TR_ID>SV502G</WAP_OSI_TR_ID>
  <WAP_COUNTRY>GT</WAP_COUNTRY>
```

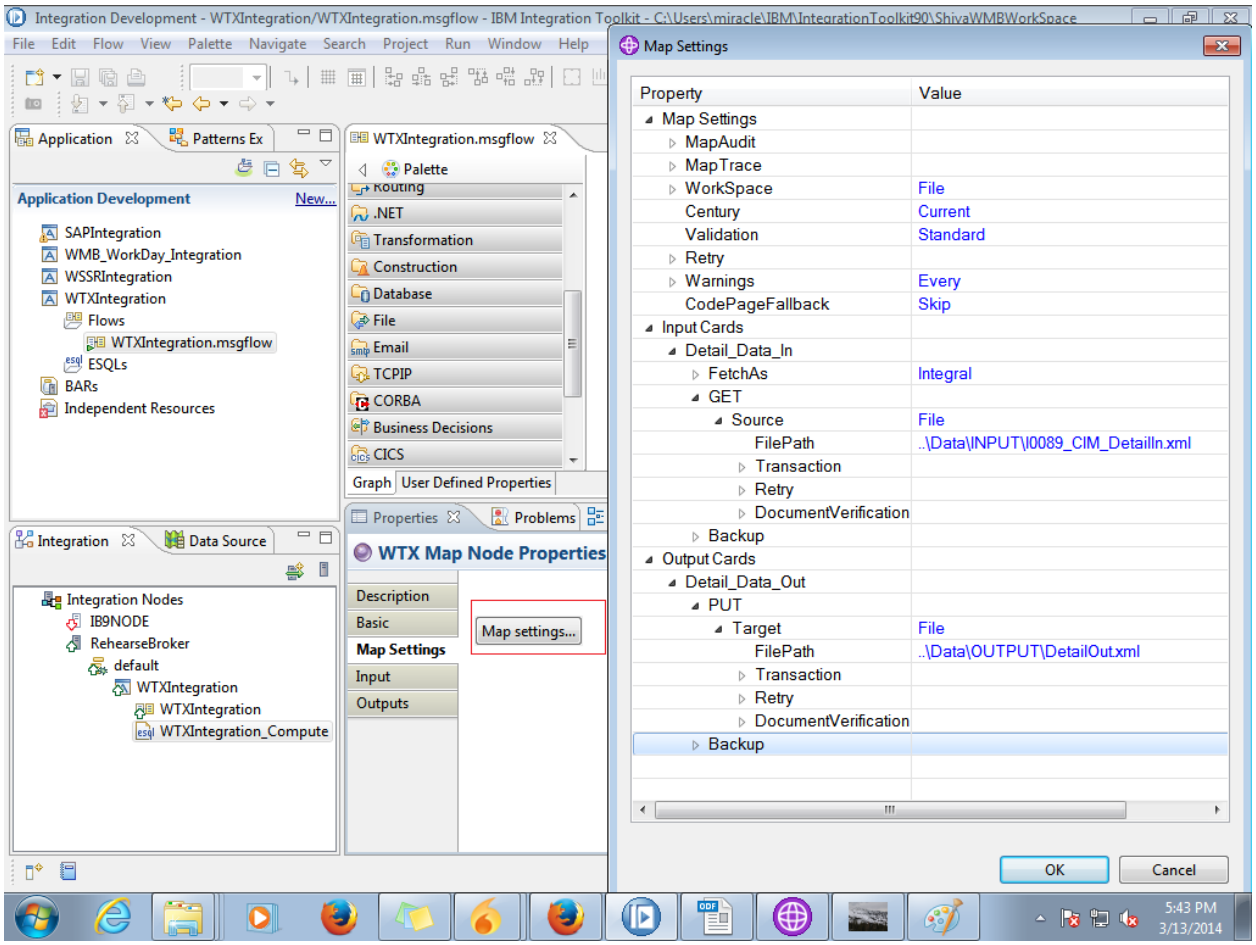
WTX MAP NODE INPUT PROPERTIES:



WTX MAP NODE OUTPUT PROPERTIES:



WTX MAP SETTINGS SPECIFIED:



Card Number To Wire

This property specifies which input card of the map should be wired. The result of this property is that, specified map input card receives its data from the prior node in the message flow.

Outputs

On the Outputs tab of the WTX Map node, configure the properties to parse the output from the map. Here you can specify Message set, Message type, Message format.

CacheMap:

At design time, set map caching to optimize runtime performance. Map caching optimizes the runtime performance because the process does not load the compiled map for every iteration of the running of the map. The compiled map is loaded only on the first invocation of the map. When map caching is set on, any changes that you make to the compiled map are used by the message after you stop and restart the execution group. When map caching is set off, any changes that you make to the compiled map are used by

the next message that passes through the message flow. Set up map caching for each WTX Map node to enable individual control for each map node instance. Set map caching on in the Basic tab under Properties in the WebSphere Message Broker Toolkit.

Note: This caching should be enabled when you are sure that, map location will not be changed frequently.

Dynamically overriding the properties of next node using compute node:

* select the compute mode as local environment and message . This property enables us to propagate local environment to the next node

```
If InputRoot.XMLNSC.Employee_Details.employee.address = 'MI' THEN
              SET OutputLocalEnvironment.WTX.MapServerLocation =
'C:\WTX_MAPS\xml_to_fixed.mmc';
            ELSE
              SET OutputLocalEnvironment.WTX.MapServerLocation =
'C:\WTX_MAPS\xml_to_csv.mmc';
            END IF;
```

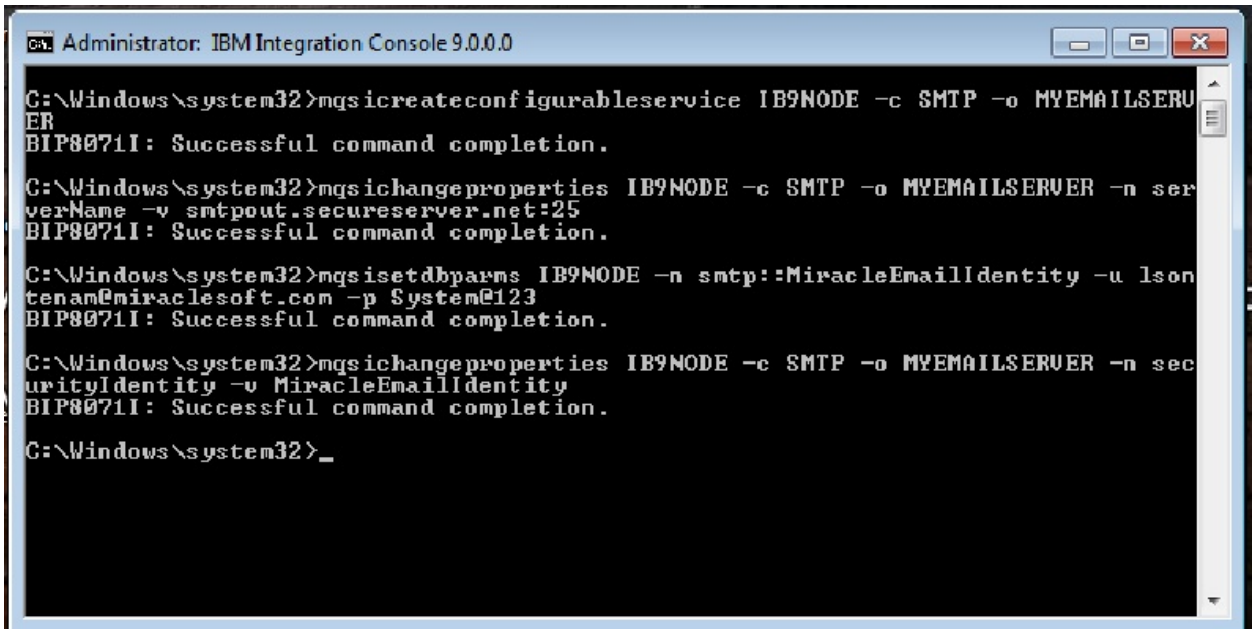
Email Output & File Output Node Scenario

Objective: Whenever there is an Exception/Error in processing a message flow,we suppose to email the error details and also place the error file in FTP.

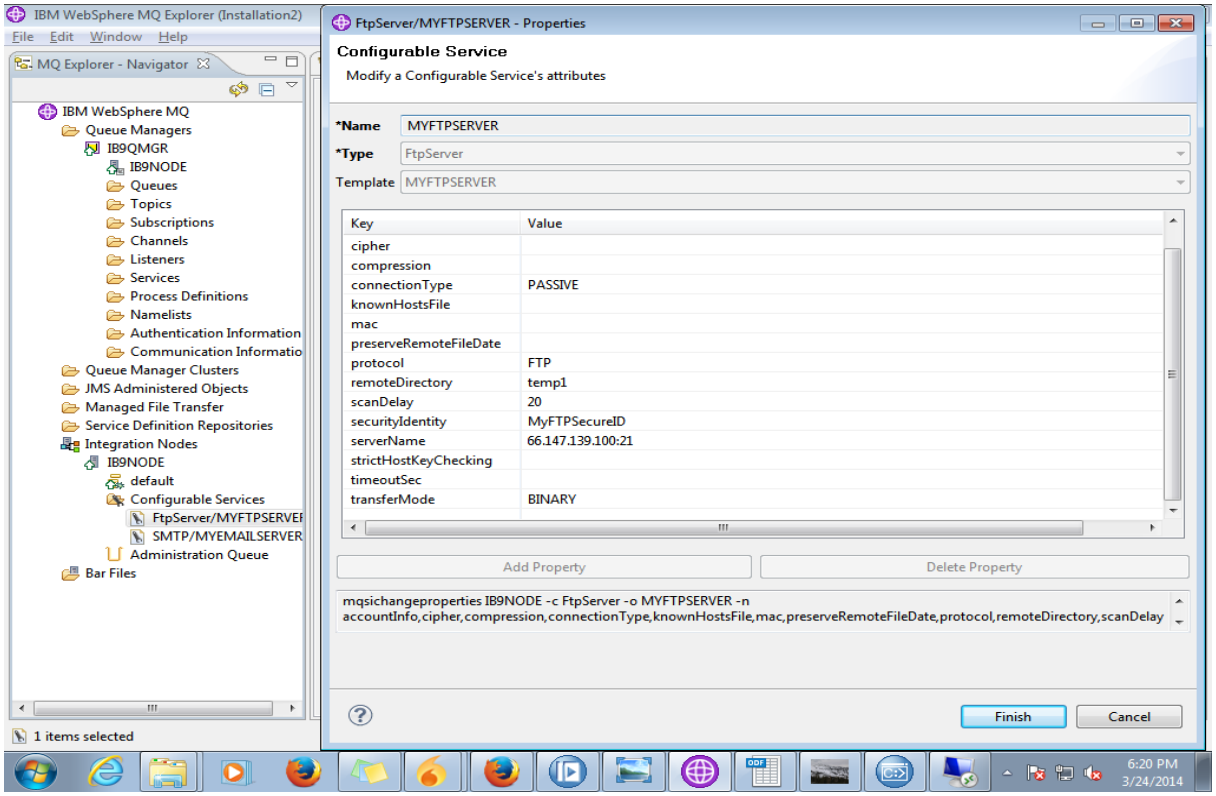
Note: Create two Configurable Services as below, for setting the security and location properties of your Email SMTP and FTP servers.

-You can create configurable services either using GUI in MQ explorer or through command console.Both are illustrated below:

SMTP:

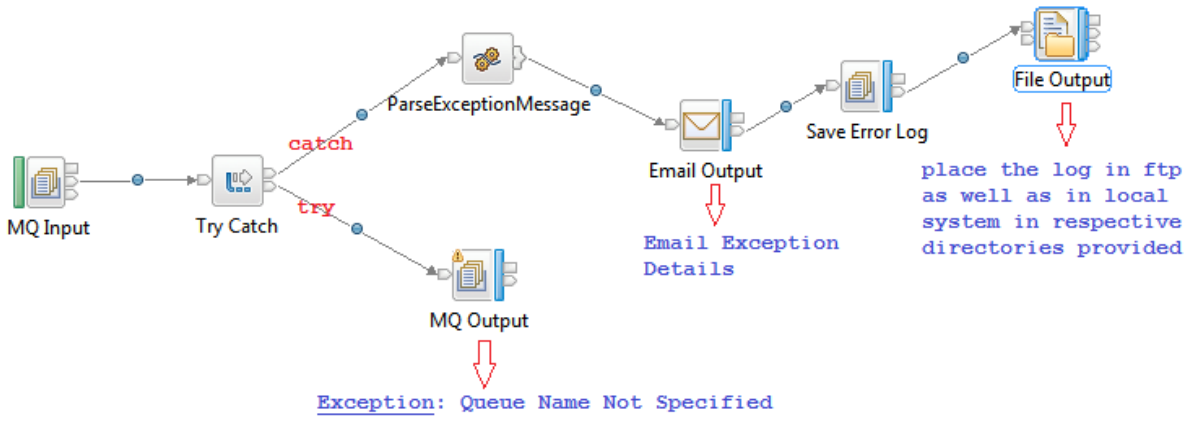

Administrator: IBM Integration Console 9.0.0.0
C:\Windows\system32>mqsicreateconfigurableservice IB9NODE -c SMTP -o MYEMAILSERVER
BIP8071I: Successful command completion.
C:\Windows\system32>mqsichangeproperties IB9NODE -c SMTP -o MYEMAILSERVER -n serverName -v smtpout.secureserver.net:25
BIP8071I: Successful command completion.
C:\Windows\system32>mqsisetdbparms IB9NODE -n smtp::MiracleEmailIdentity -u lsontenam@miraclesoft.com -p System@123
BIP8071I: Successful command completion.
C:\Windows\system32>mqsichangeproperties IB9NODE -c SMTP -o MYEMAILSERVER -n securityIdentity -v MiracleEmailIdentity
BIP8071I: Successful command completion.
C:\Windows\system32>_

FTP:

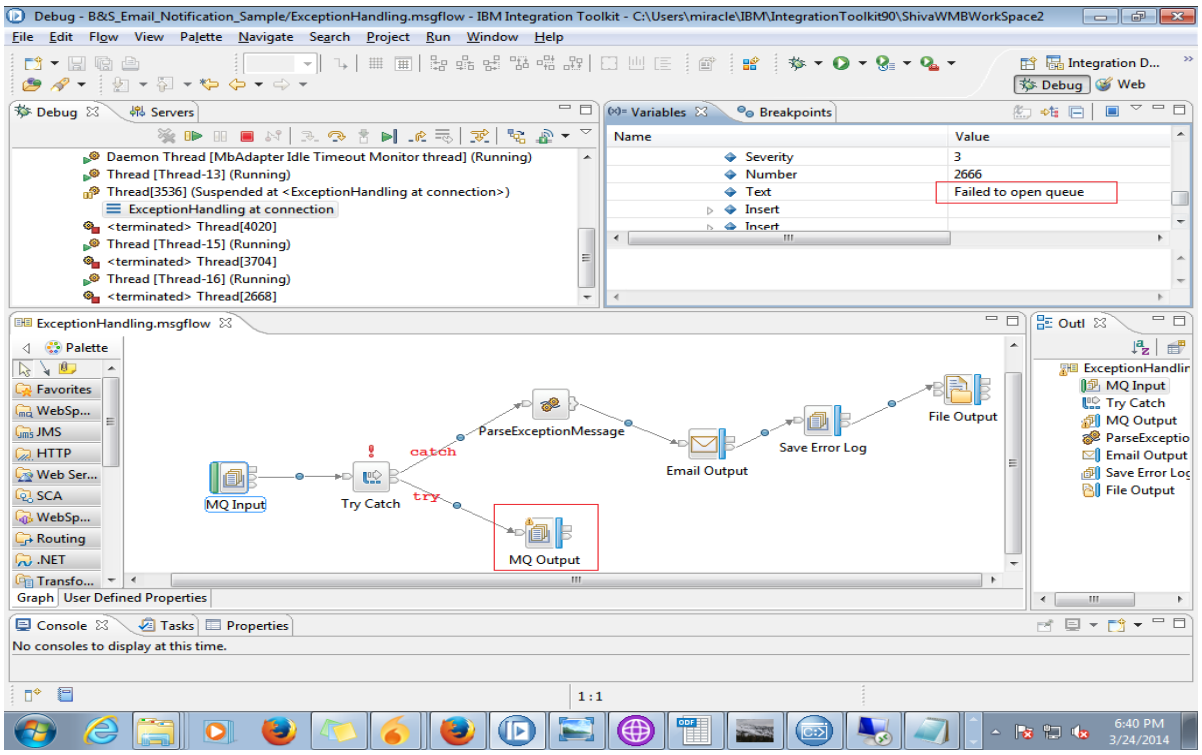

IBM WebSphere MQ Explorer (Installation2)
File Edit Window Help
MQ Explorer - Navigator
IBM WebSphere MQ
Queue Managers
IB9QMGR
IB9NODE
Queues
Topics
Subscriptions
Channels
Listeners
Services
Process Definitions
Namelists
Authentication Information
Communication Informatio
Queue Manager Clusters
JMS Administered Objects
Managed File Transfer
Service Definition Repositories
Integration Nodes
IB9NODE
default
Configurable Services
FtpServer/MYFTPSERVER
SMTP/MYEMAILSERVER
Administration Queue
Bar Files
1 items selected
FtpServer/MYFTPSERVER - Properties
Configurable Service
Modify a Configurable Service's attributes
*Name MYFTPSERVER
*Type FtpServer
Template MYFTPSERVER
Key Value
cipher
compression
connectionType PASSIVE
knownHostsFile
mac
preserveRemoteFileDate
protocol FTP
remoteDirectory temp1
scanDelay 20
securityIdentity MyFTPSecureID
serverName 66.147.139.100:21
strictHostKeyChecking
timeoutSec
transferMode BINARY
Add Property
Delete Property
mqsichangeproperties IB9NODE -c FtpServer -o MYFTPSERVER -n accountInfo,cipher,compression,connectionType,knownHostsFile,mac,preserveRemoteFileDate,protocol,remoteDirectory,scanDelay
Finish
Cancel
6:20 PM
3/24/2014

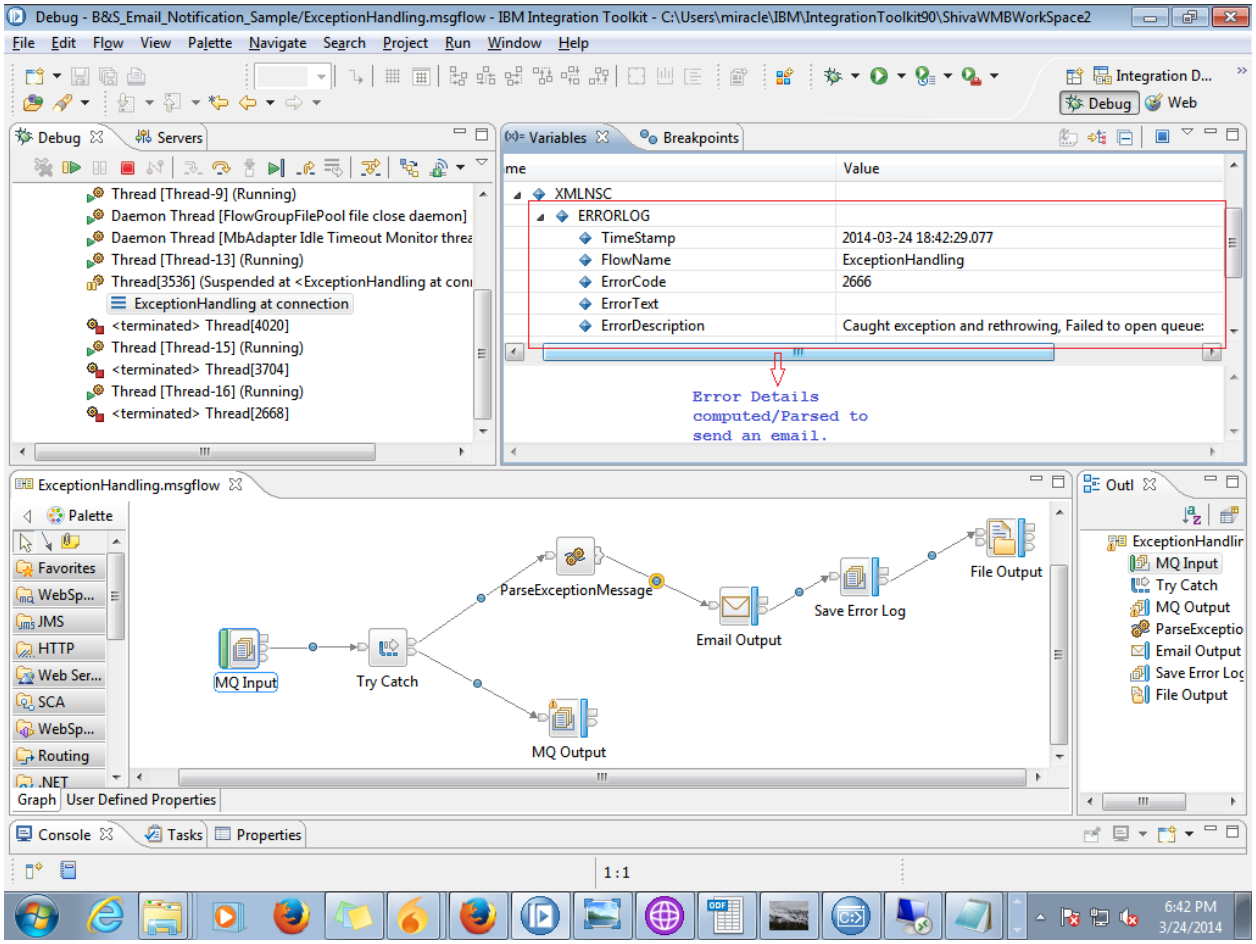
```
C:\Windows\system32>mqsisetdbparms IB9NODE -n ftp::MyFTPSecureID -u miracle-ftp
-p [illegible]ss001
BIP8071I: Successful command completion.

C:\Windows\system32>_
```

Message Flow:

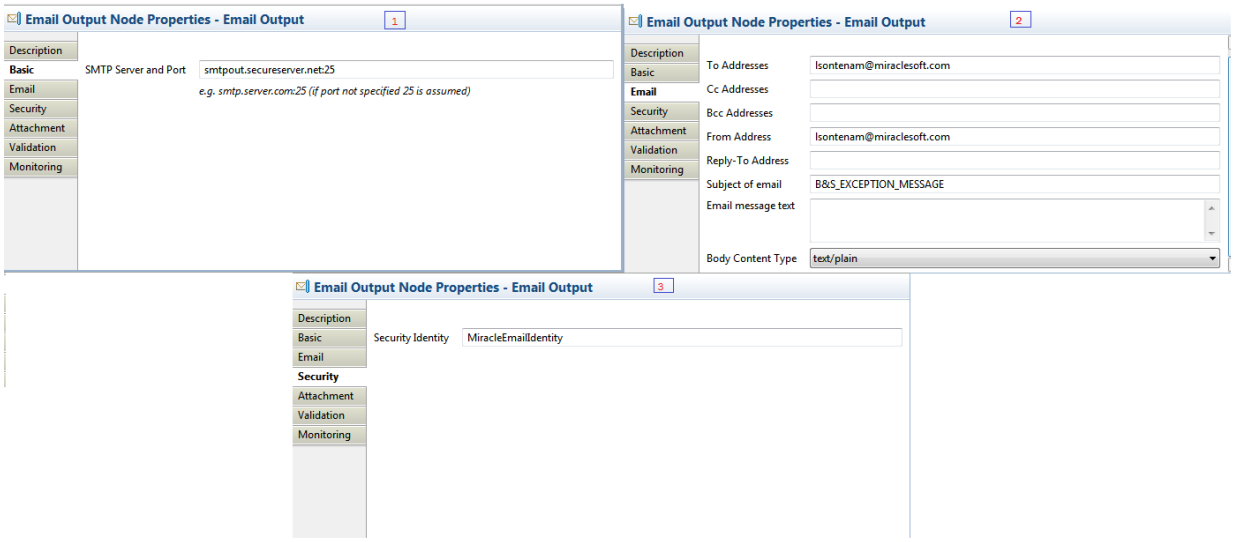


Processing for Email:

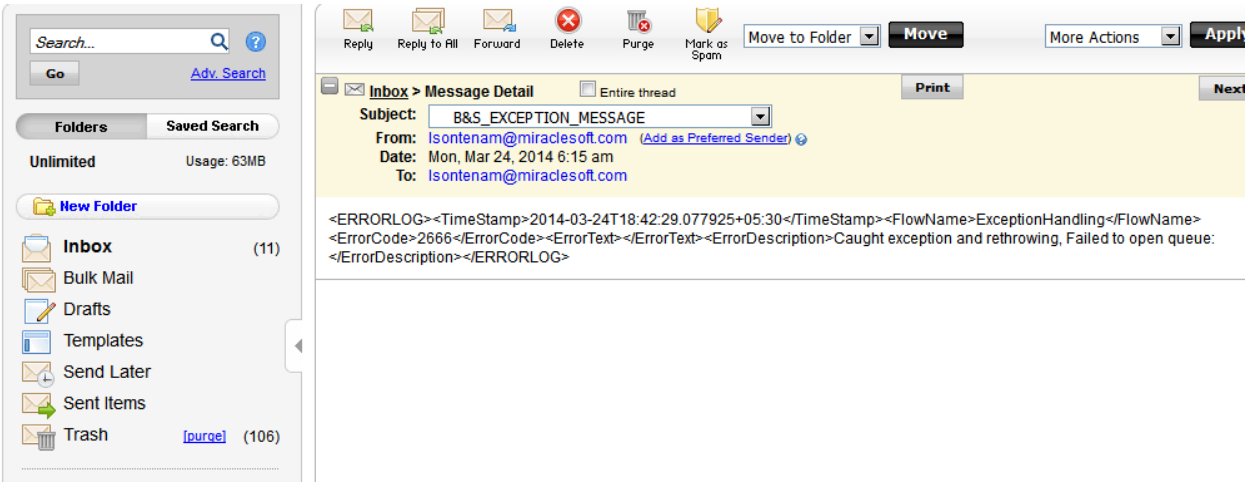




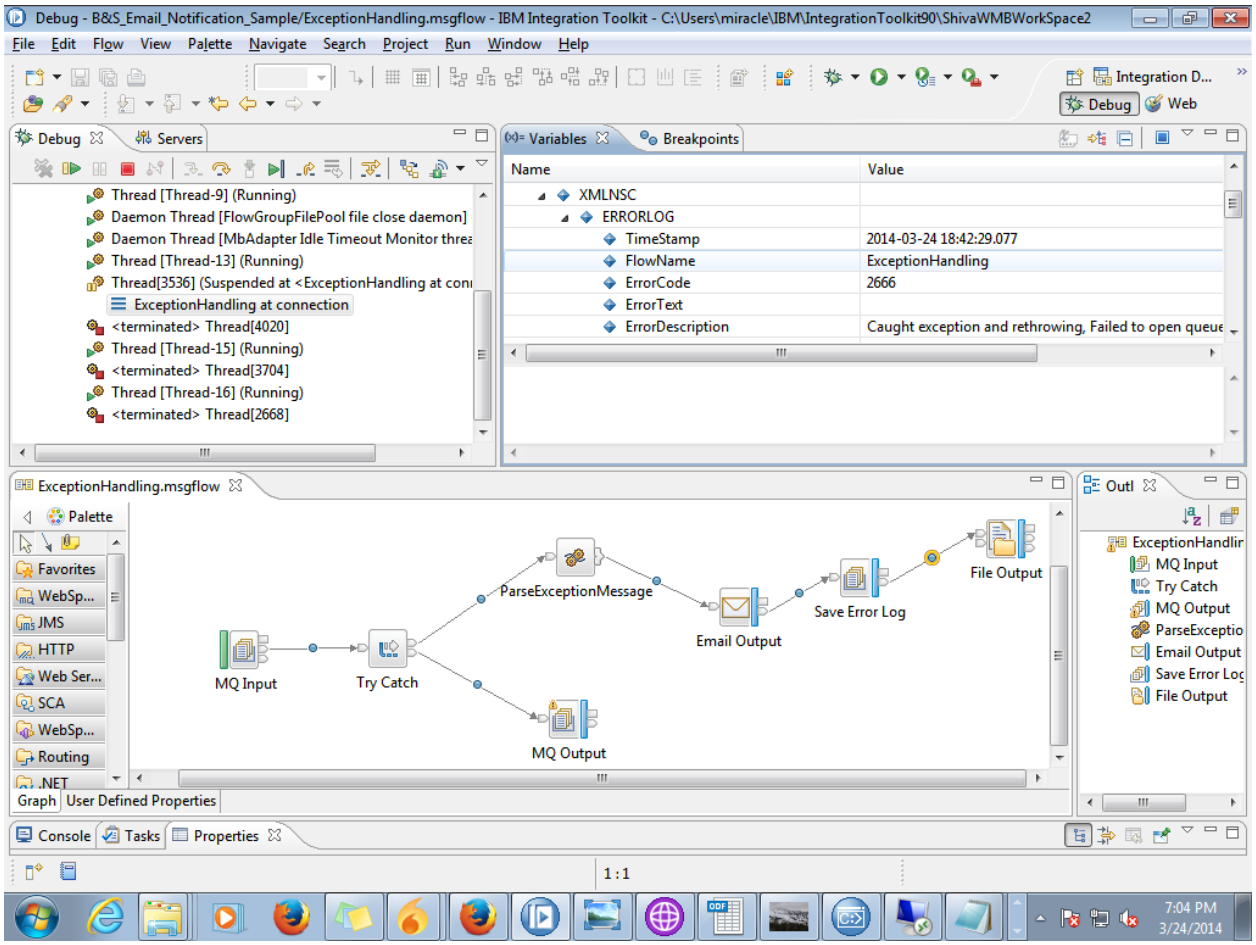
EmailOutput Node Properties:



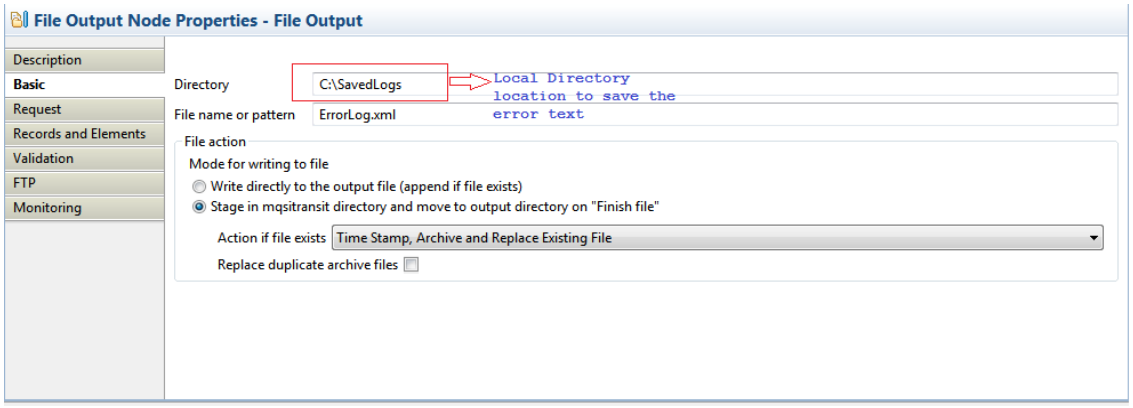
Email received:

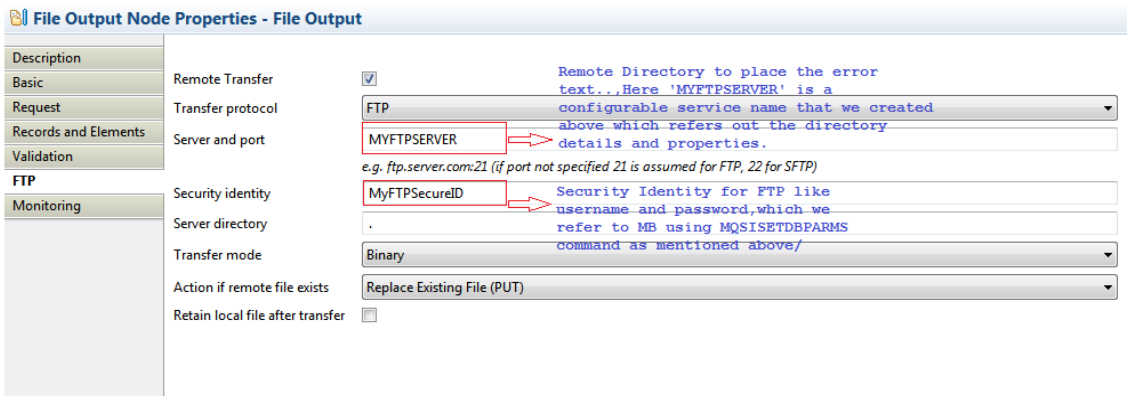


Processing for FTP:

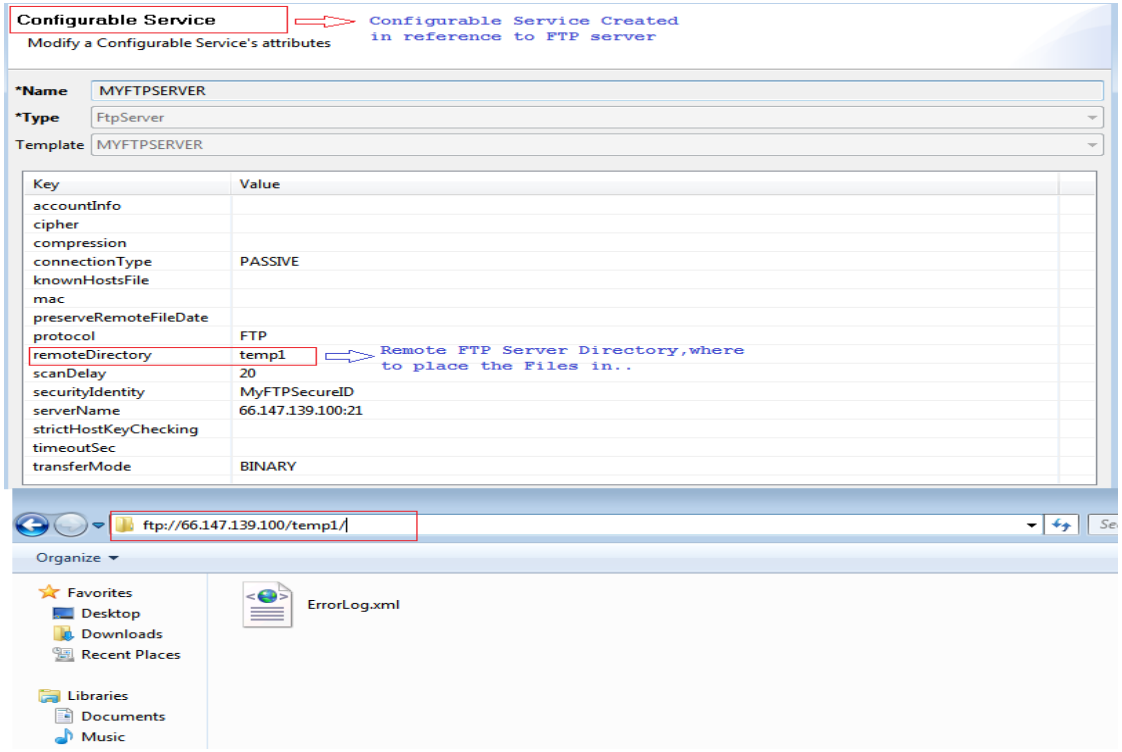


FileOutput Node Properties:





File Received:



DatabaseInput Node Scenario

Overview

Purpose

Use a Database Input node to respond to events in a database. For example, the broker can keep an external system synchronized with a database by sending updates to the target system whenever data is changed in the database.

What this lab is about

In this Lab you will know, how to create a Message Flow Project that interact with Oracle Database - Using Database Input Node.

-You will also be deploying the application using IIB Toolkit.

What you will do in this lab

You will go through each step in the following sections. Each step might contain many subsets

Key Steps are

- Setup Database Tables, Triggers.
- How to create a Data Definition file.
- Configuring the Database input node properties.
- How to deploy and test the message flow .

Exercise

The steps involved are:

1. Setup Database Tables and Triggers.
2. Creating a Data Definition file.
3. Creating an Application and adding this Data Definition in the Project References .
4. Creating a Message Set and Message Definition from Data Definition File Created and finally adding them to application references.
5. Create a Message Flow.

6. Data Source Name Creation.
7. Configure DatabaseInput Node
8. Deploy the Message flow and testing the DatabaseInput node.