Ben Thompson – IBM Integration Bus Architect – bthomps@uk.ibm.com

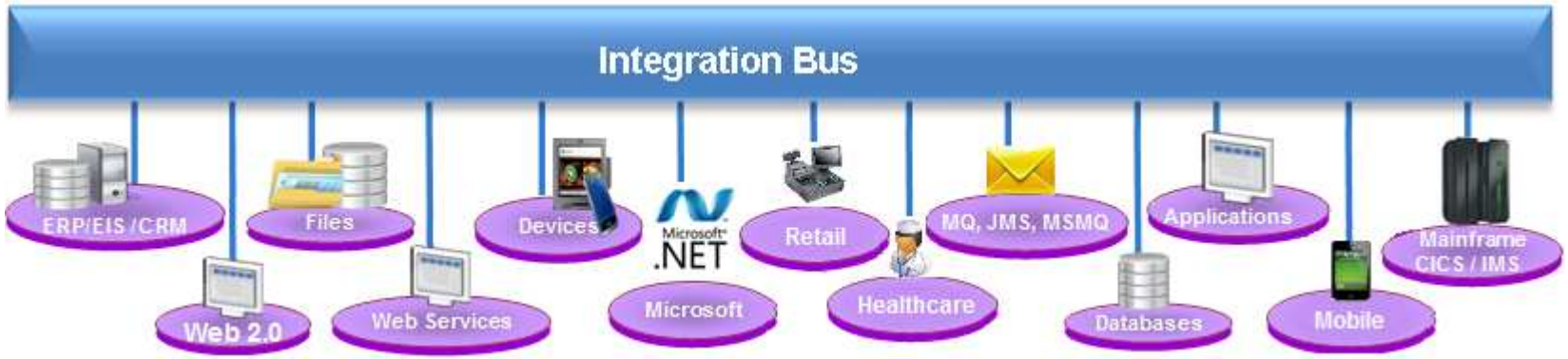
Using Patterns with WMBv8 and IIBv9

Patterns

- ***What is a Pattern, and why do I care?***
- ***Pattern Example – File Record Distribution to WMQ***
- ***Pattern Authoring – Solar Pattern & Map Flow Convert***
- ***Web Patterns***
- ***Associated IIB Technologies***
  - *User Defined nodes*
  - *Subflows as User Defined nodes*
  - *Cloned nodes*

The Patterns Challenge

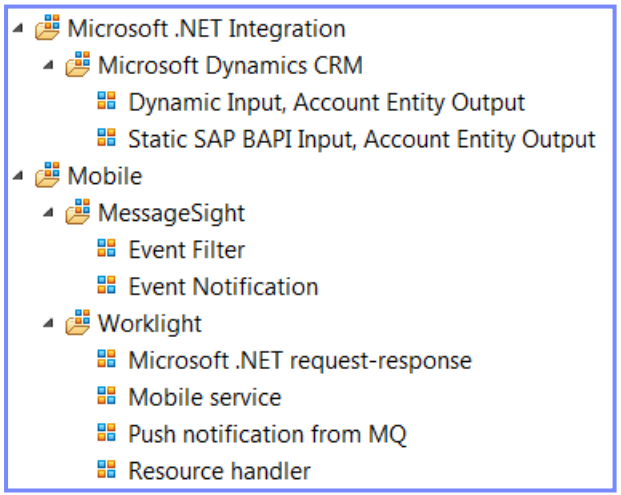
- Connectivity and Integration can be complex
- Increasingly, the mission of IBM Integration Bus is to take difficult integration tasks and make them easy to achieve – intuitive for tooling users, without the need for custom coding
- Patterns can help simplify complex integration challenges by reducing them to the expression of some basic Points of Variability in a set of common reusable solutions.

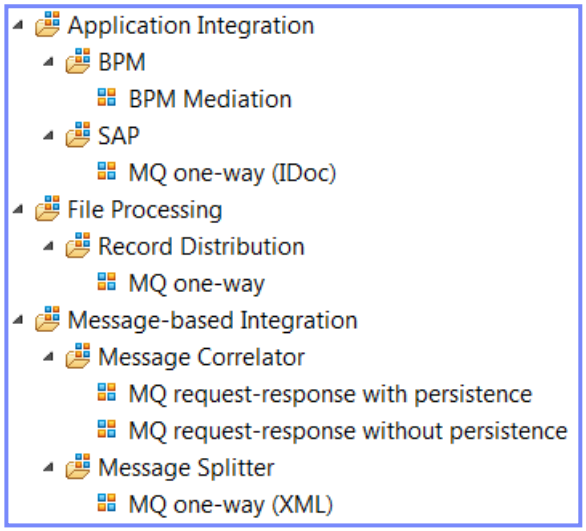


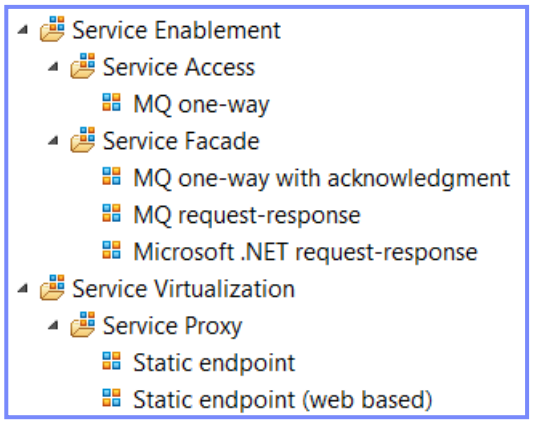
What is a pattern and why should I care?

- **A pattern is a reusable solution that encapsulates a tested approach to solving a common architecture, design, or deployment task in a particular context.**

- **WMB / IIB patterns are used to:**
  - Generate customized solutions to a recurring integration problem in an efficient way
  - Encourage adoption of preferred techniques in message flow design
  - Help guide developers who are new to the product
  - Provide consistency in the generated resources

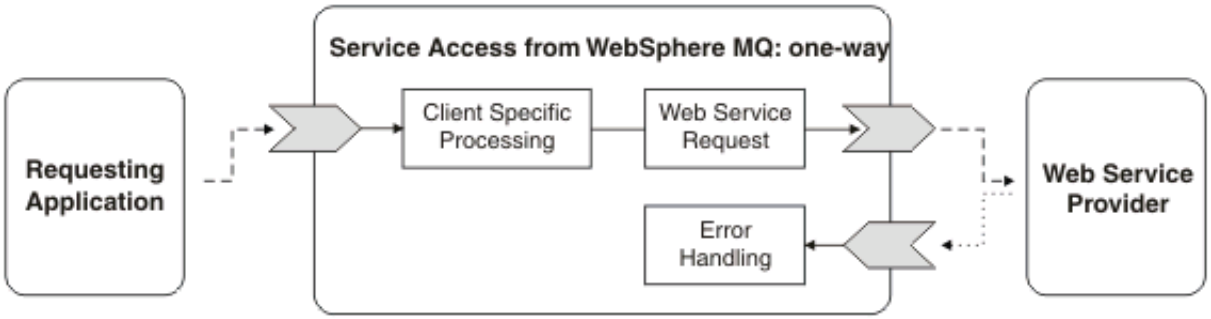






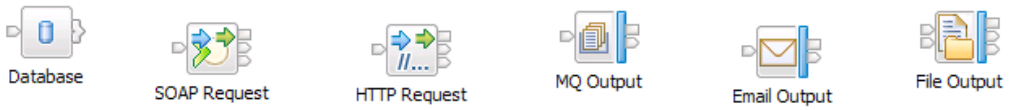
Built-In Patterns

- **Message Broker provides a core set of built-in patterns**
- **These implement a variety of common scenarios**
  - Web service front end to a MQ based application
  - Processing data stored in a file and routing to one or more queues
  - Adding a proxy in front of a web service provider
  - Processing data from an SAP system and routing to MQ
  - Shredding messages and routing to one or more queues
- **Patterns are selected based on client feedback and field experience**
- **This core set of patterns continues to grow with each release**
- **Patterns are an important part of our Industry Pack approach**



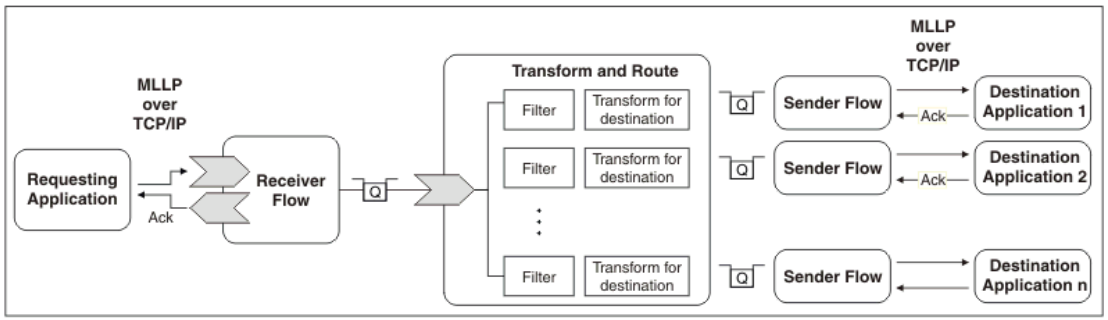
Patterns for Simplified Development

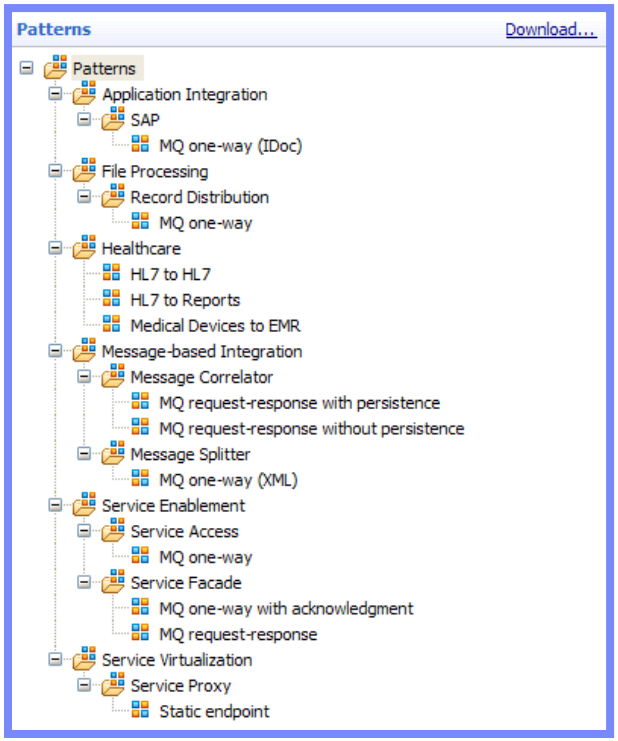
- Create top-down, parameterized connectivity solutions
- Reduce common problems in flow development
- Communicate best practices to the broker community
- Reduce time-to-value for solution development
- Complement regular solution development in broker
- Although sometimes they appear similar, patterns are NOT samples



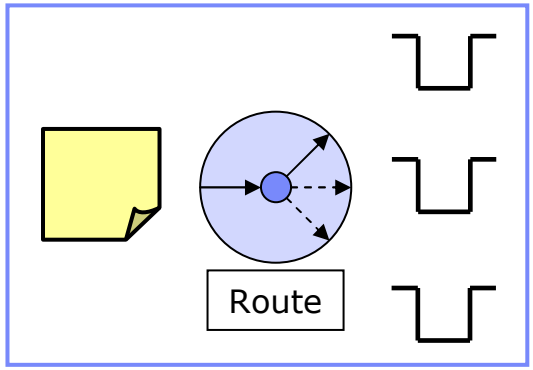
Healthcare: HL7 to HL7 pattern

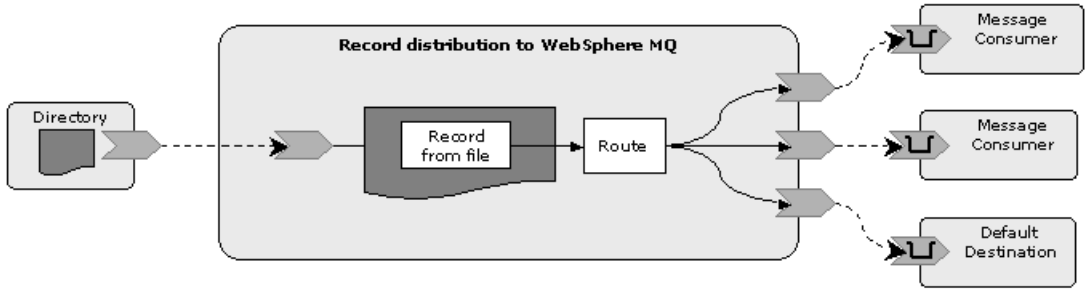
The Healthcare: HL7 to HL7 pattern integrates an application that can send Health Level Seven International (HL7) v2 messages with one or more applications that can receive HL7 messages. The applications must be capable of sending and receiving HL7 messages by using Minimal Lower Layer Protocol (MLLP) over TCP/IP.

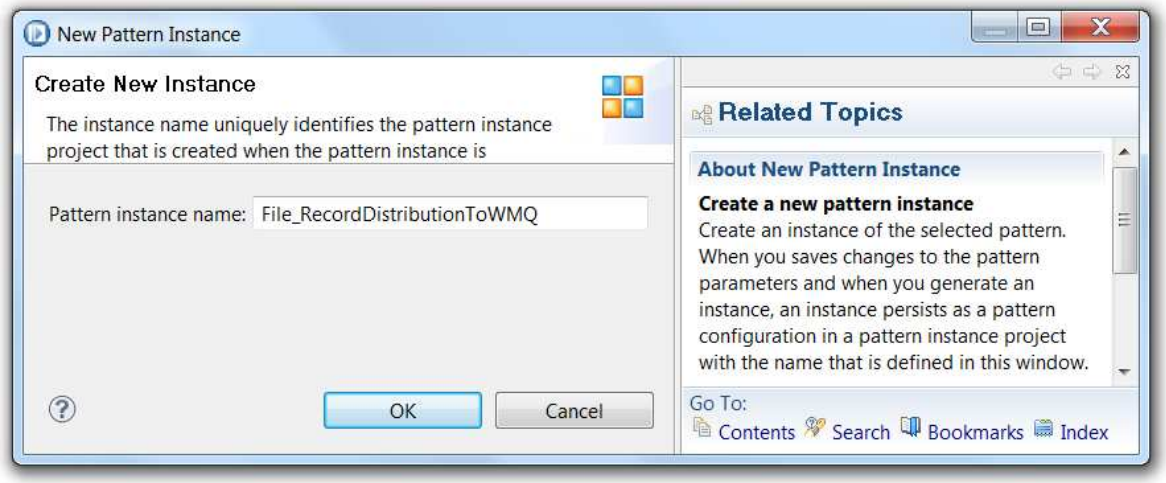


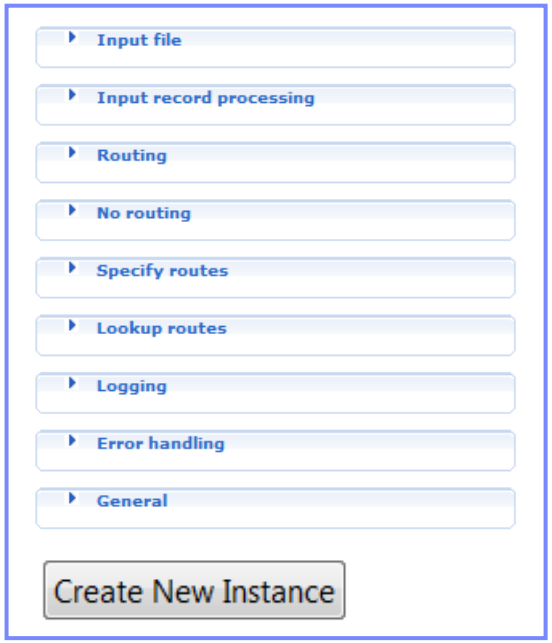


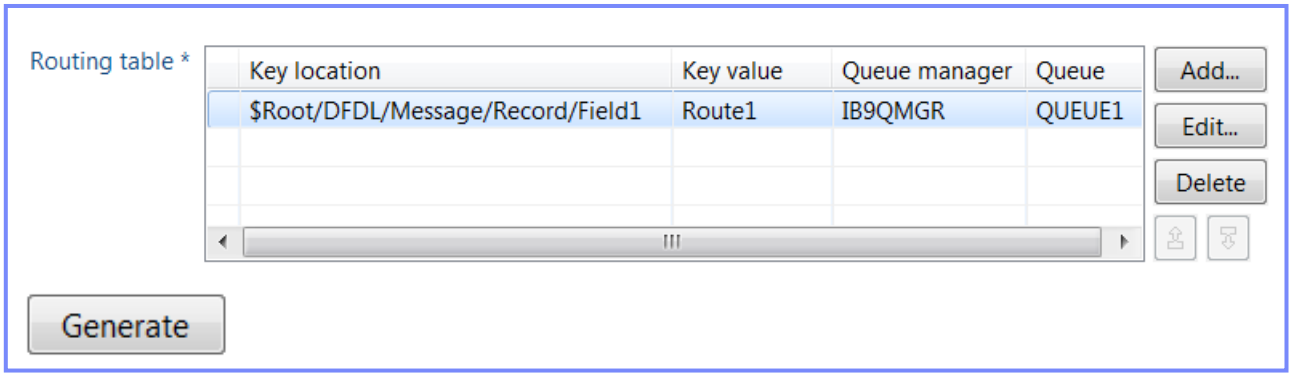
A Simple Example



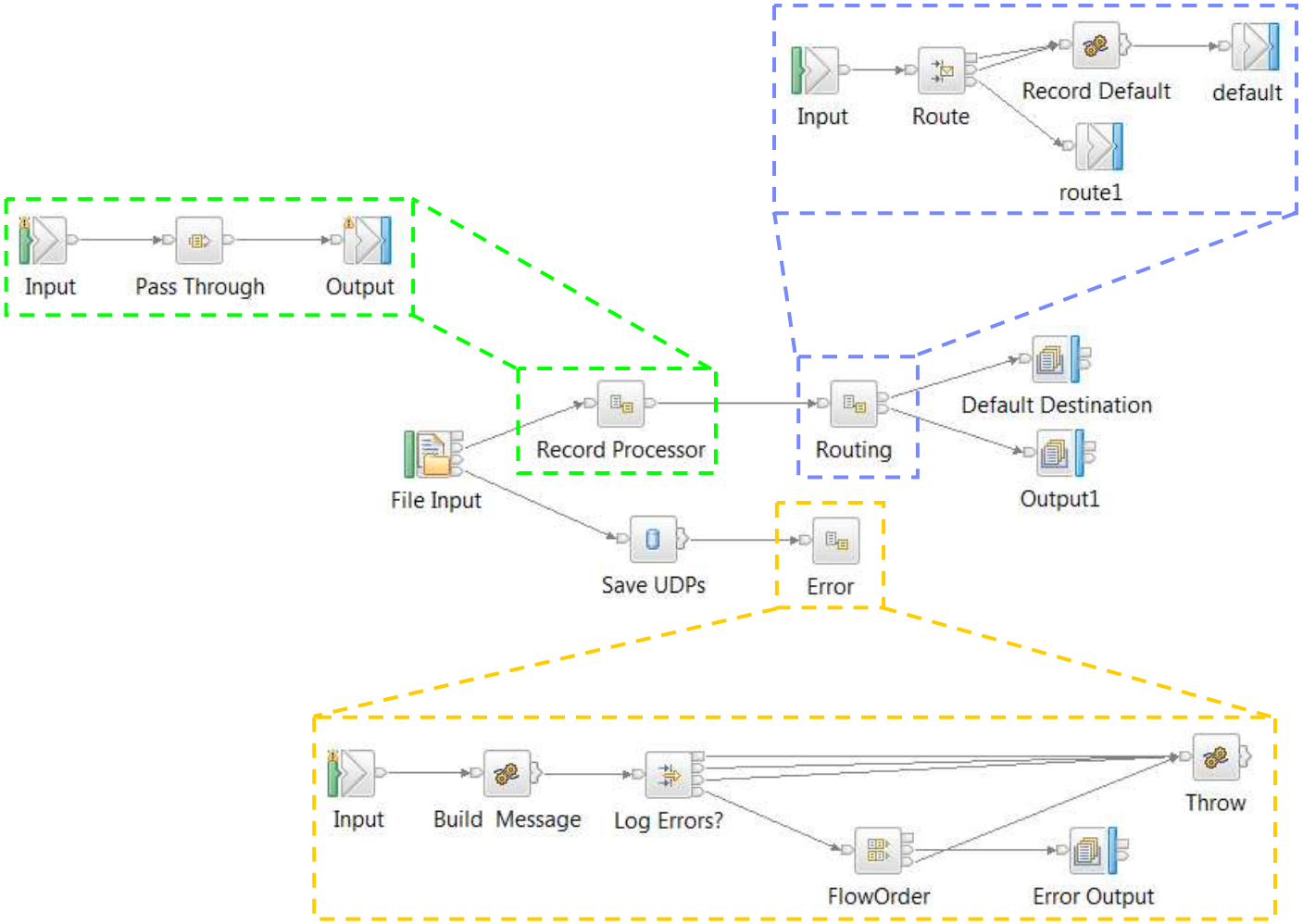








A Simple Example



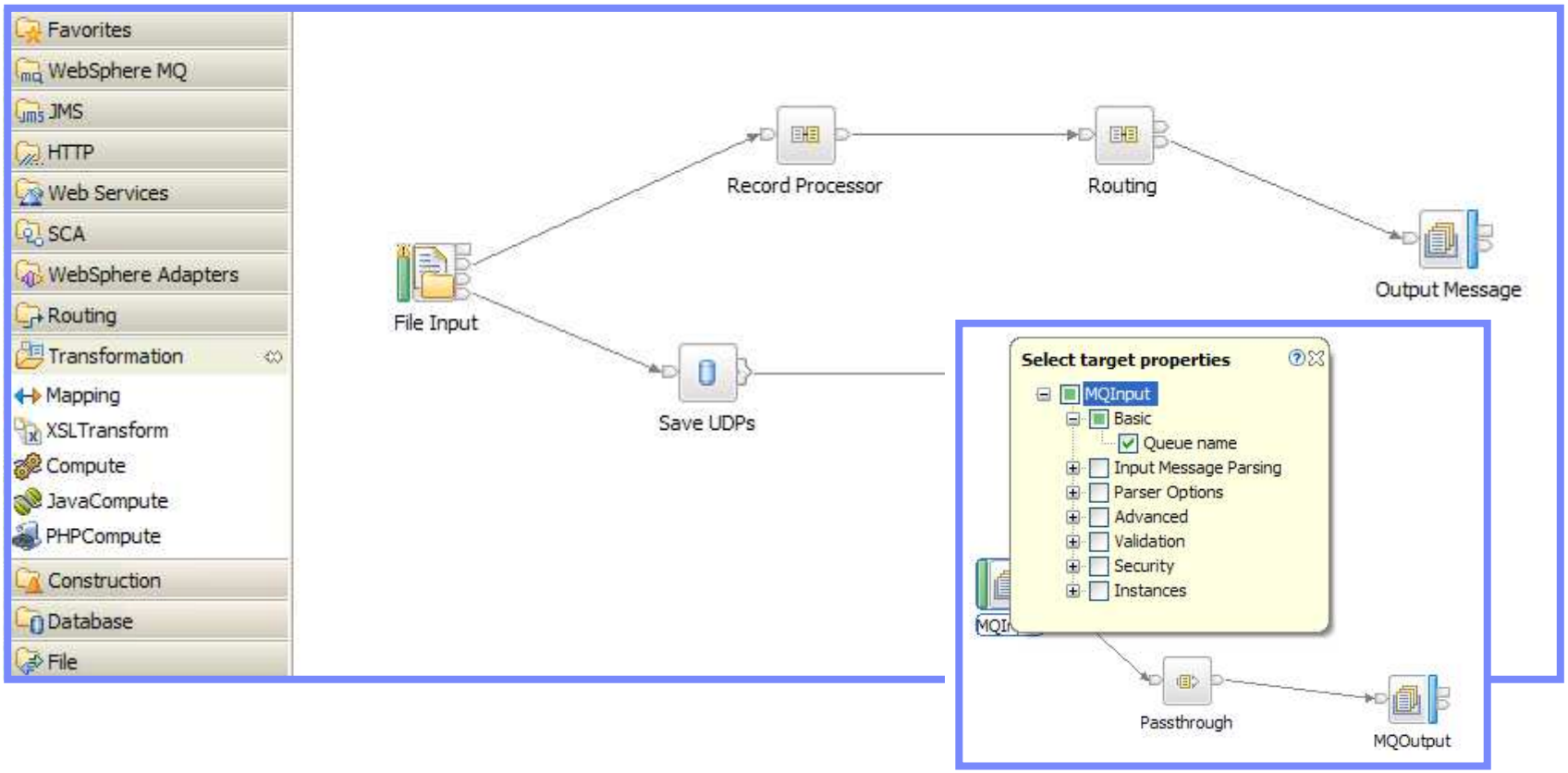
Message Broker Pattern Authoring

Pattern Authoring

- **Patterns becomes even more compelling when you can create your own!**
  - Every organization has their own repeating connectivity patterns!
  - Pattern authoring is the name we give to this technology in Message Broker
- **We recommend you start with a working solution**
  - One or more Message Broker projects
- **Pattern authoring is a *design* activity**
  - It may be long lived
  - It is often not sequential
- **Using patterns is a top-down activity driven by a requirement, but:**
  - Authoring a working solution is (typically) a bottom-up activity
  - So pattern authoring bridges these two different approaches
- **Patterns have their own development cycle**
  - Pattern Authoring editor supports this design activity

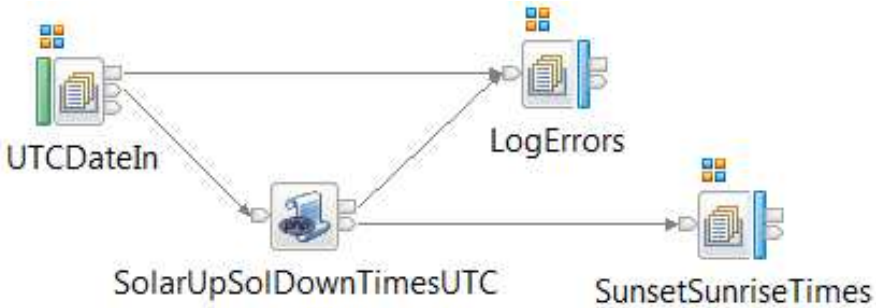
Create Your Working Solution

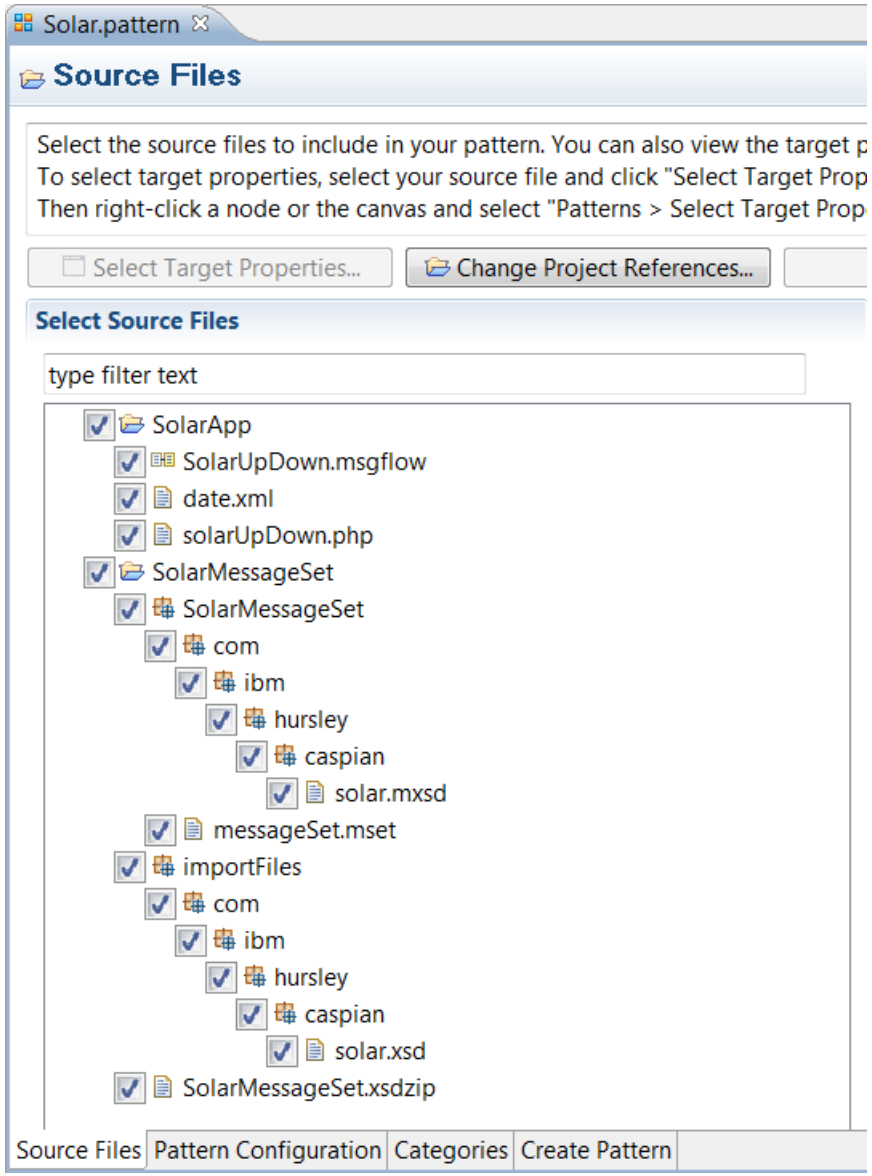
- **No change at all - design your Message Broker solution as you do today**
  - Pattern authoring does not change the tools you use to create solutions
  - The key to a good pattern is to create a good working solution!



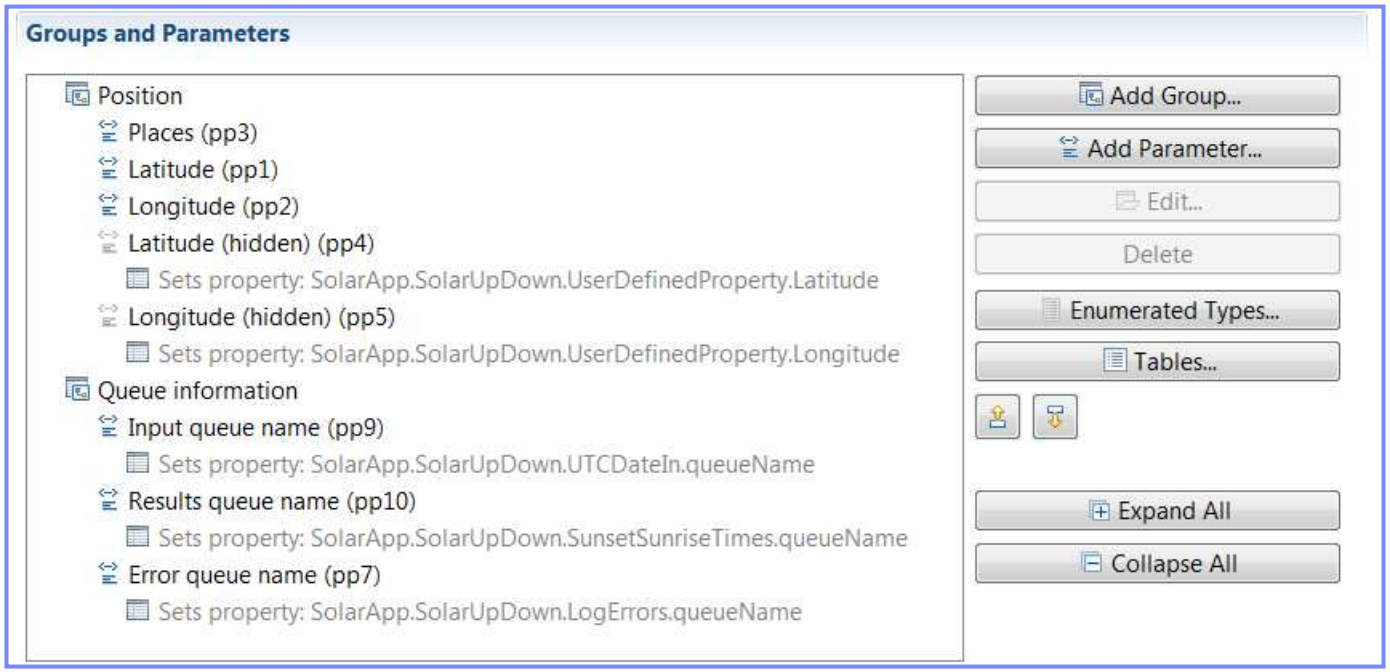
Pattern Authoring

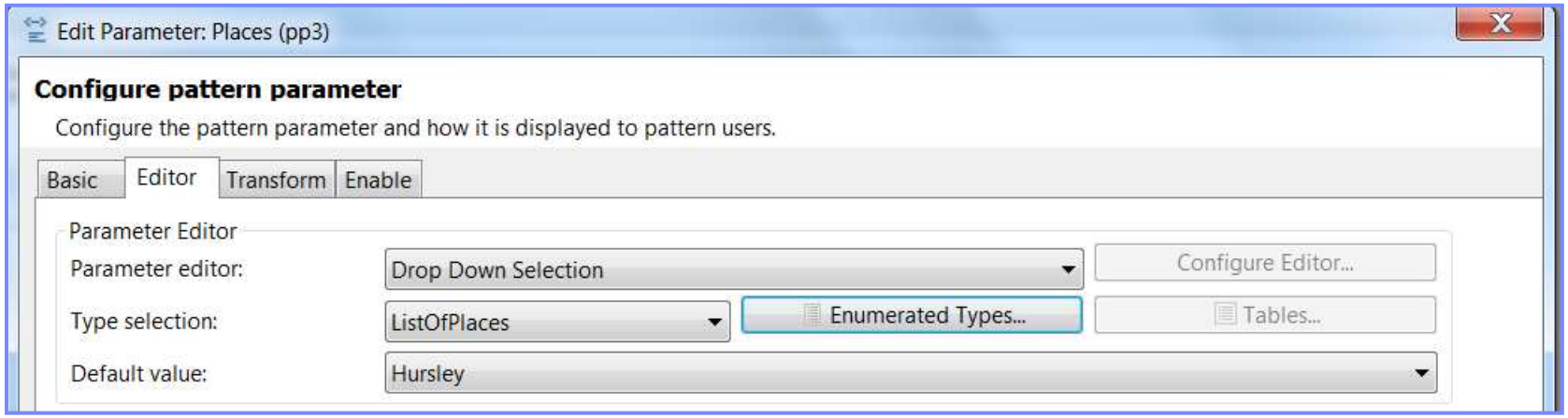
- Create a Template Message Flow
- Create a Pattern Authoring Project
- Add references from the Authoring Project to the projects containing your exemplar(s)
- In the Pattern Authoring Editor, select the Source Files tab and choose the files to be included with the Pattern.
- Define target properties (properties that are changed by a pattern selection)
- A user-defined pattern can change:
  - Promoted node properties
  - Node properties in a message flow
  - User-defined properties





Pattern Authoring – UI Customisation





Pattern Refinement

- **Pattern authoring in Message Broker supports property changes**
  - Node, user-defined properties (UDPs) and promoted node properties
- **Property variability is the most common type of variability that a pattern might need to express - there are many others:**
  - Generate application text files such as ESQL scripts
  - Make structural changes to Message Flows
  - Create administration files such as MQSC scripts
- **It is impossible to try and predict all the possible extensions that a pattern author might wish to implement**
- **In Message Broker we provide two ways to *extend* pattern authoring**
  - Java code that is invoked when pattern instances are generated
  - PHP templates that generate text files in pattern instance projects

```
DEFINE QLOCAL(LOCAL) DEFPSIST(YES) DESCR('<?php echo $_MB['PP']['queueName']; ?>')
```

Packaging a Pattern

Create Pattern

Test your pattern by configuring your pattern plug-in information, click "Create Pattern Plug-ins", and click Test Pattern".

Plug-in Information

Configure the unique identifier for your pattern plug-in.

Pattern name:	MyPattern
Plug-in ID:	com.your.company.domain.MyPattern
Version:	1.0.0.0
Provider:	Your Company Name
Description:	Plug-in created by the Pattern Authoring editor

Create Pattern Plug-ins

Test Pattern...

Debug Pattern...

Translation Options

If you enable this option, the Pattern Authoring editor creates two additional NLS plug-ins. These plug-ins are set up so that you can drop in translated resources, such as Java property files. If you are creating a single language pattern, do not select this check box.

☐ Create translation plug-ins (*.nl1 and *.doc.nl1)

Pattern Distribution

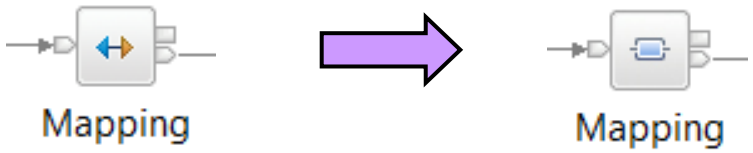
After you have created and tested your pattern plug-ins (see the Plug-in ID above for the plug-in names), package your pattern by clicking "File > Export > General > Archive File" to export these plug-ins.

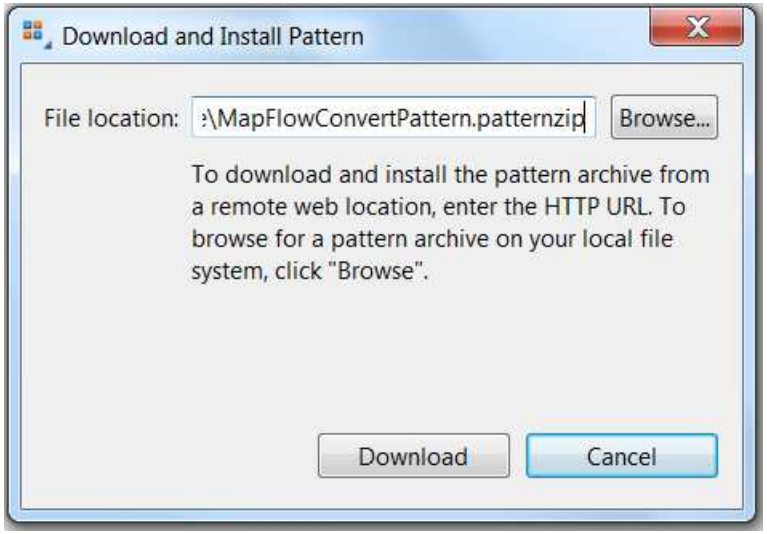
Create Pattern Archive...

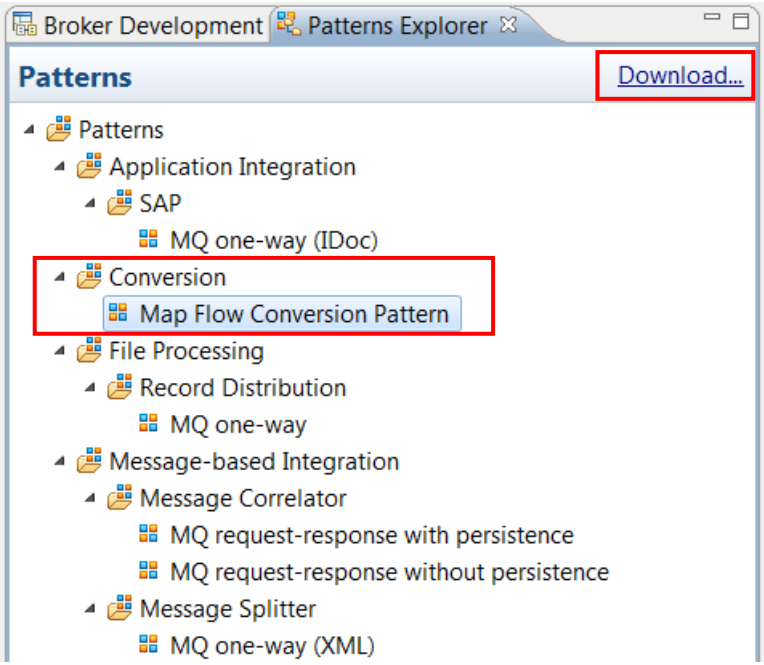
To use the pattern plug-ins, the pattern user must extract the exported archive files into the default WebSphere Message Broker Toolkit plug-ins directory:

Map Flow Conversion Pattern

- The pattern framework can also be used to quickly expose code which uses the WMB / IIB Message Flow API via a User Interface (in some circumstances, avoid needing to write an Eclipse plugin!)
- This pattern provides a message flow conversion utility which changes WMBv7 style mapping nodes into WMBv8 style mapping nodes.
- The pattern can be installed into a WMBv8 Toolkit by launching the supplied installation file MapFlowConvertPattern.patternzip







Pattern Specification

View Pattern Specification

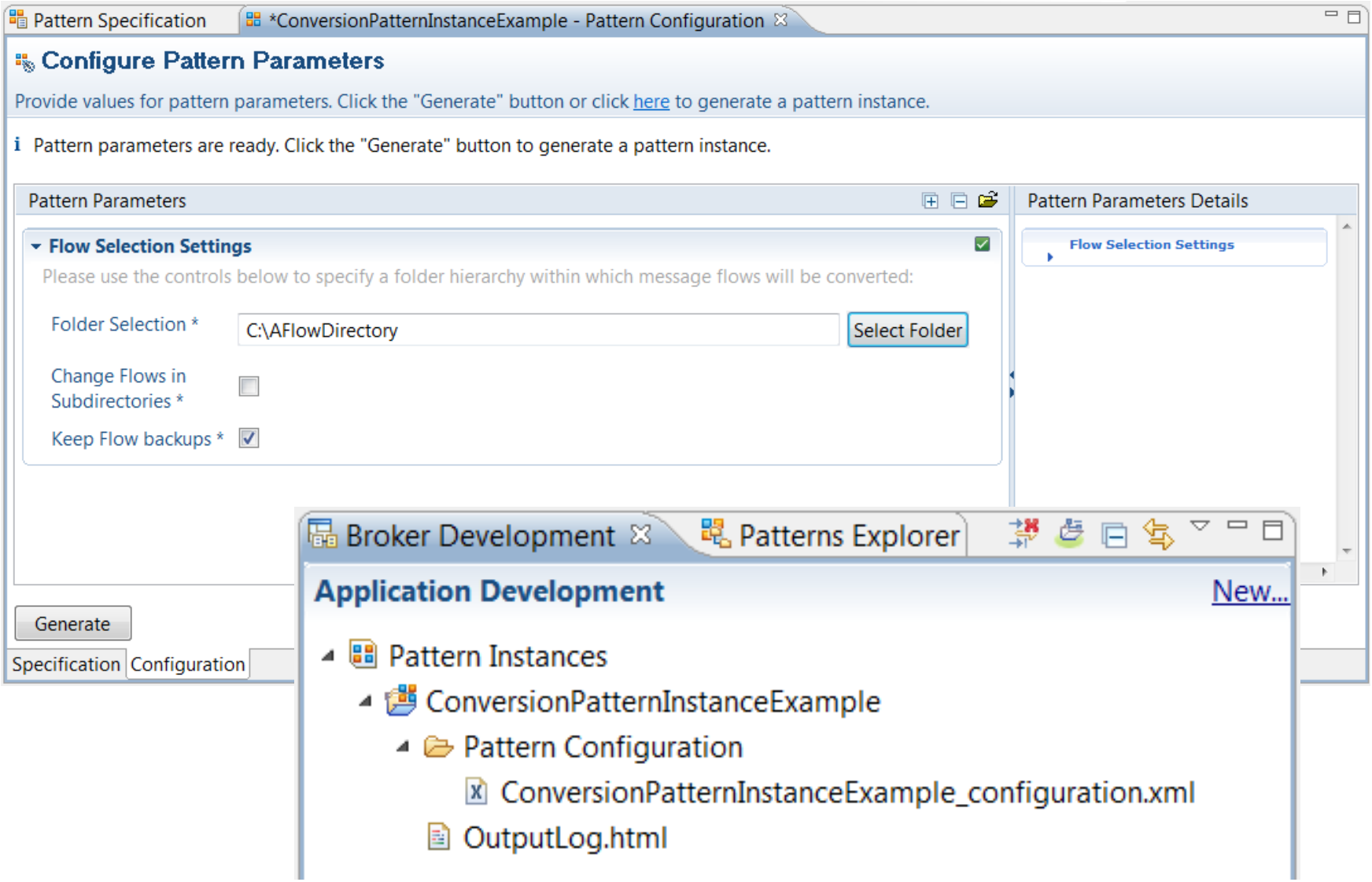
View information about the selected pattern and then click the "Create New Instance" button or click here to start using a pattern.

Parameters for the Map Flow Conversion Pattern pattern

Flow Selection Settings

Pattern parameter	Default	Description
Folder Selection		The *Folder Selection* parameter provides a file path to a directory. This directory is searched for message flows. Any .msgflow file which is found which contains at least one WMBv7 map node will be editted in order to convert the WMBv7 map nodes into WMBv8 map nodes.
Change Flows in Subdirectories	Cleared	The *Change Flows in Subdirectories* parameter specifies whether the pattern should search all subdirectories (as well as the top level directory specified by the *Folder Selection* parameter). All .msgflow files located in all subdirectories will be analysed.
Keep Flow backups	Selected	The *Keep Flow backups* parameter controls whether the pattern will make a backup of the message flow. If selected, this option will cause the pattern to back up the original message flow using its original name, with *.backup* inserted before the file extension, in the same directory in which the original flow resides. For example, a flow named *MessageFlow1.msgflow* would be backed up using the name *MessageFlow1.backup.msgflow*.

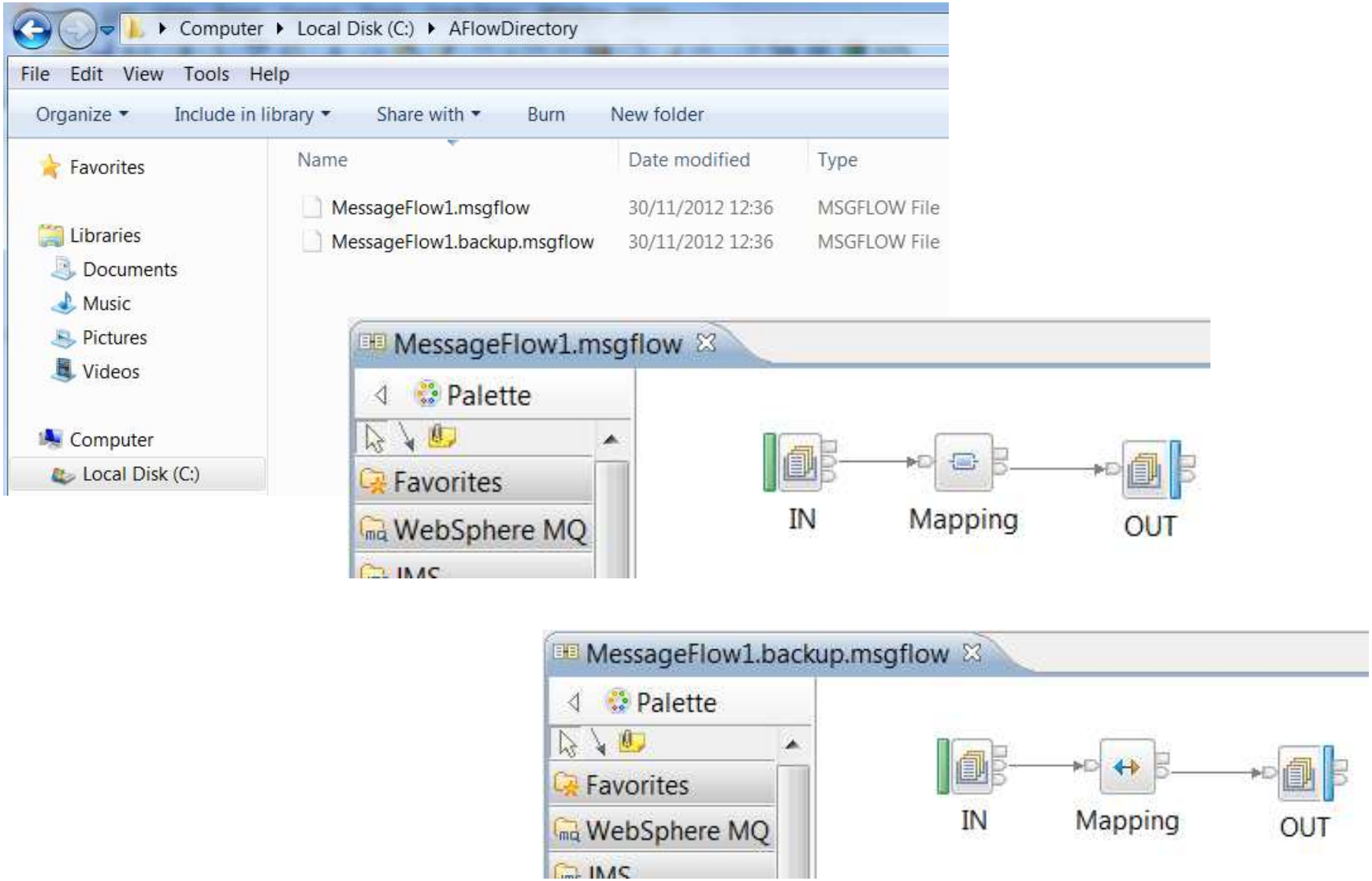
Create New Instance



C:\Map_Migration\workspacePatternTest1\ConversionPatternInstanceExample\OutputLog.html

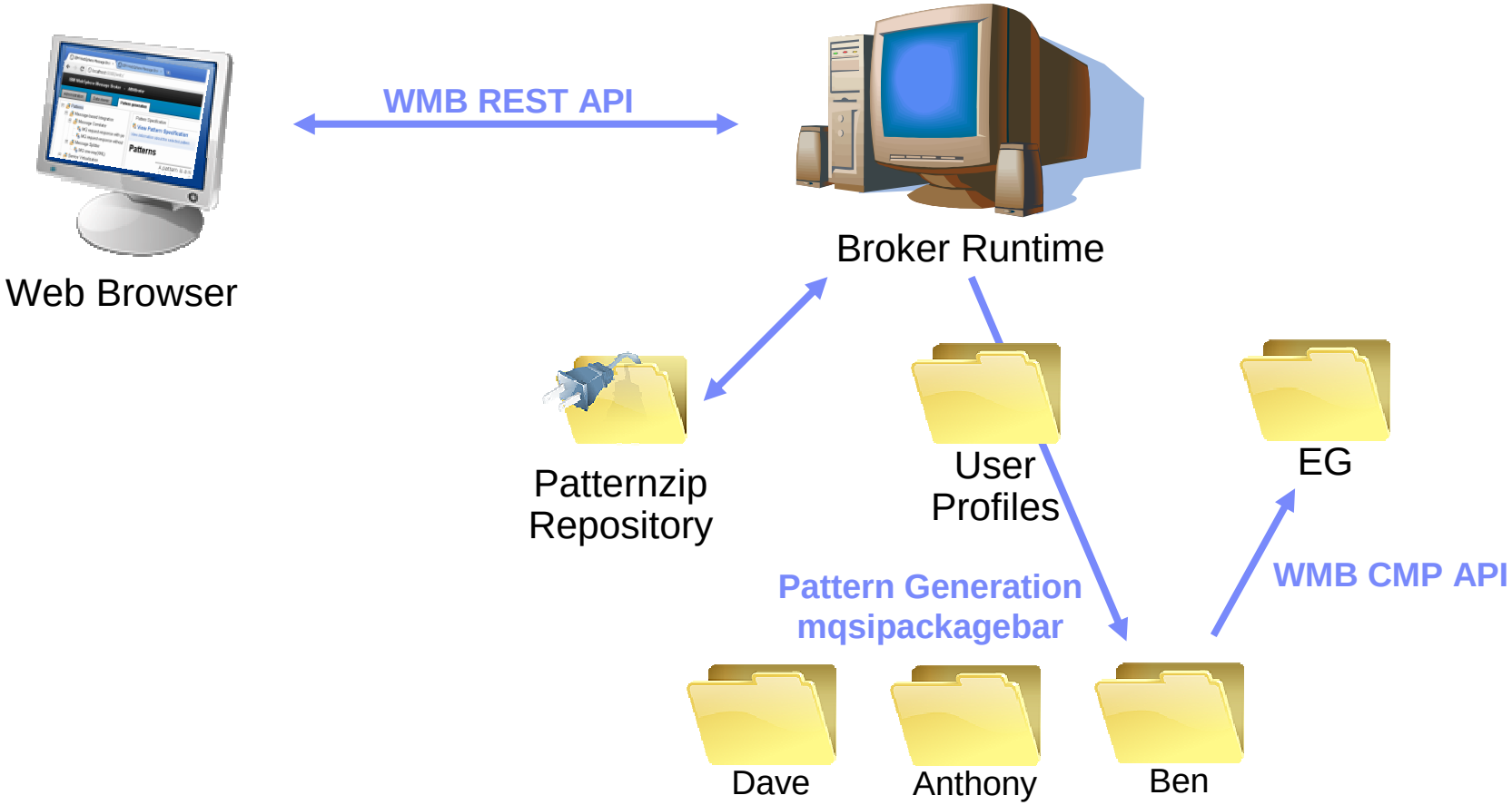
MessageFlow1.msgflow

- 2012-11-30 12:36:35 A message flow file has been located at C:\AFlowDirectory\MessageFlow1.msgflow
- The message flow contains a WMBv7 style mapping node named Mapping
- The WMBv7 style mapping node named Mapping had a Long Description of *BLANK*
- The WMBv7 style mapping node named Mapping had a Short Description of *BLANK*
- The WMBv7 style mapping node named Mapping had a display rotation value of LEFT_TO_RIGHT
- The WMBv7 style mapping node named Mapping was located at the X coordinate 148
- The WMBv7 style mapping node named Mapping was located at the Y coordinate 35
- Disregarded properties: The following properties of a WMBv7 style mapping node have no equivalents on the WMBv8 style mapping node, so have been ignored but they are recorded below for logging purposes ...
- The WMBv7 style mapping node named Mapping had a Property of dataSource with the value *BLANK*
- The WMBv7 style mapping node named Mapping had a Property of connectDatasourceBeforeFlowStarts with the value false
- The WMBv7 style mapping node named Mapping had a Property of transaction with the value automatic. WMBv8 style mapping nodes always have Transaction=Automatic
- The WMBv7 style mapping node named Mapping had a Property of mappingMode with the value message
- The WMBv7 style mapping node named Mapping had a Property of treatWarningsAsErrors with the value false
- The WMBv7 style mapping node named Mapping had a Property of throwExceptionOnDatabaseError with the value true
- The flow has had its mapping node(s) upgraded and is now resaved to the same location: C:\AFlowDirectory\MessageFlow1.msgflow
- The original flow has been backed up to the location: C:\AFlowDirectory\MessageFlow1.backup.msgflow



Web Based Patterns

WMB Web Pattern Workflow



IBM Integration

Welcome, Default ▼

IB9NODE
- Servers
- Patterns
  - Samples
    - Solar
  - Service Virtualization
    - Service Proxy
      - Static endpoint (web based)
- Policy
- Data
- Security
- Monitoring
- My Workspace
  - Pattern Instances
  - Resources
    - WSDLs

Patterns

Specification

View Pattern Specification

Create new instance

Service Proxy: static endpoint pattern

Use this pattern to provide decoupling between web service requesters and web service providers by routing through a virtual service that is bound directly to the target service provider.

Use this patte

- When yo use the address ecause it must be
- To suppo
- When yo r when you want to es to the service p

Specify New Pattern Instance Name

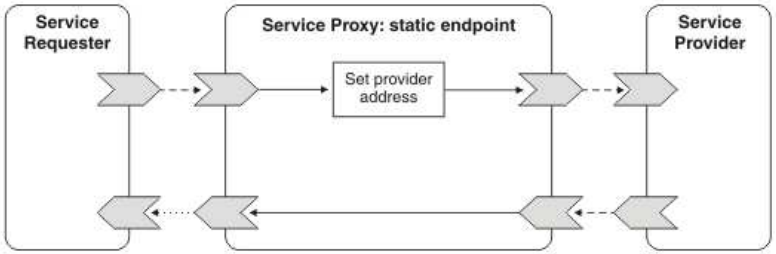
Enter the name of the pattern configuration.

Pattern Instance Name: StaticEndpointInstance01

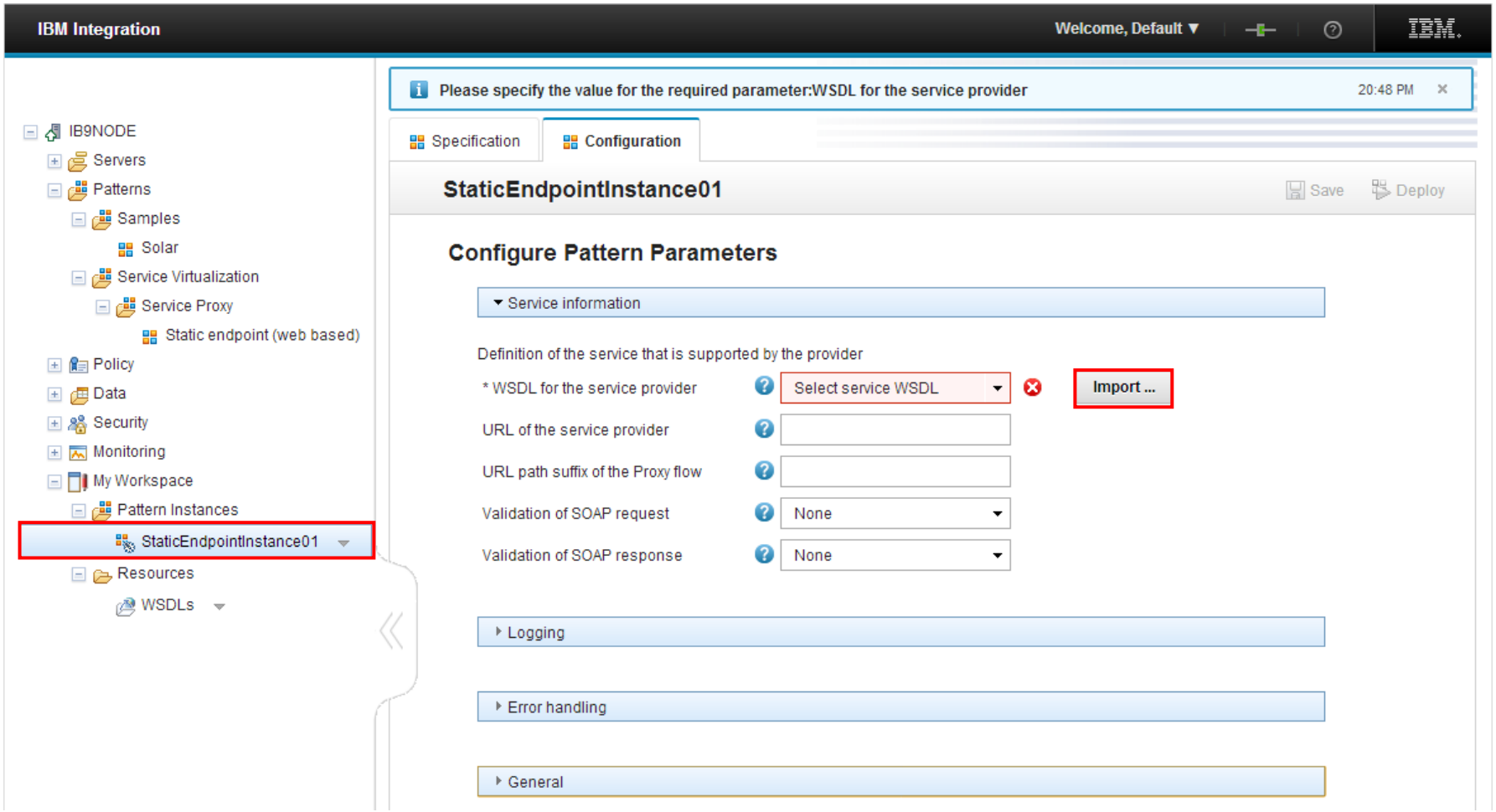
OK Cancel

Solution

The solution is to implement a message flow that provides a proxy for the provider web service and can also include logging and error handling capabilities.

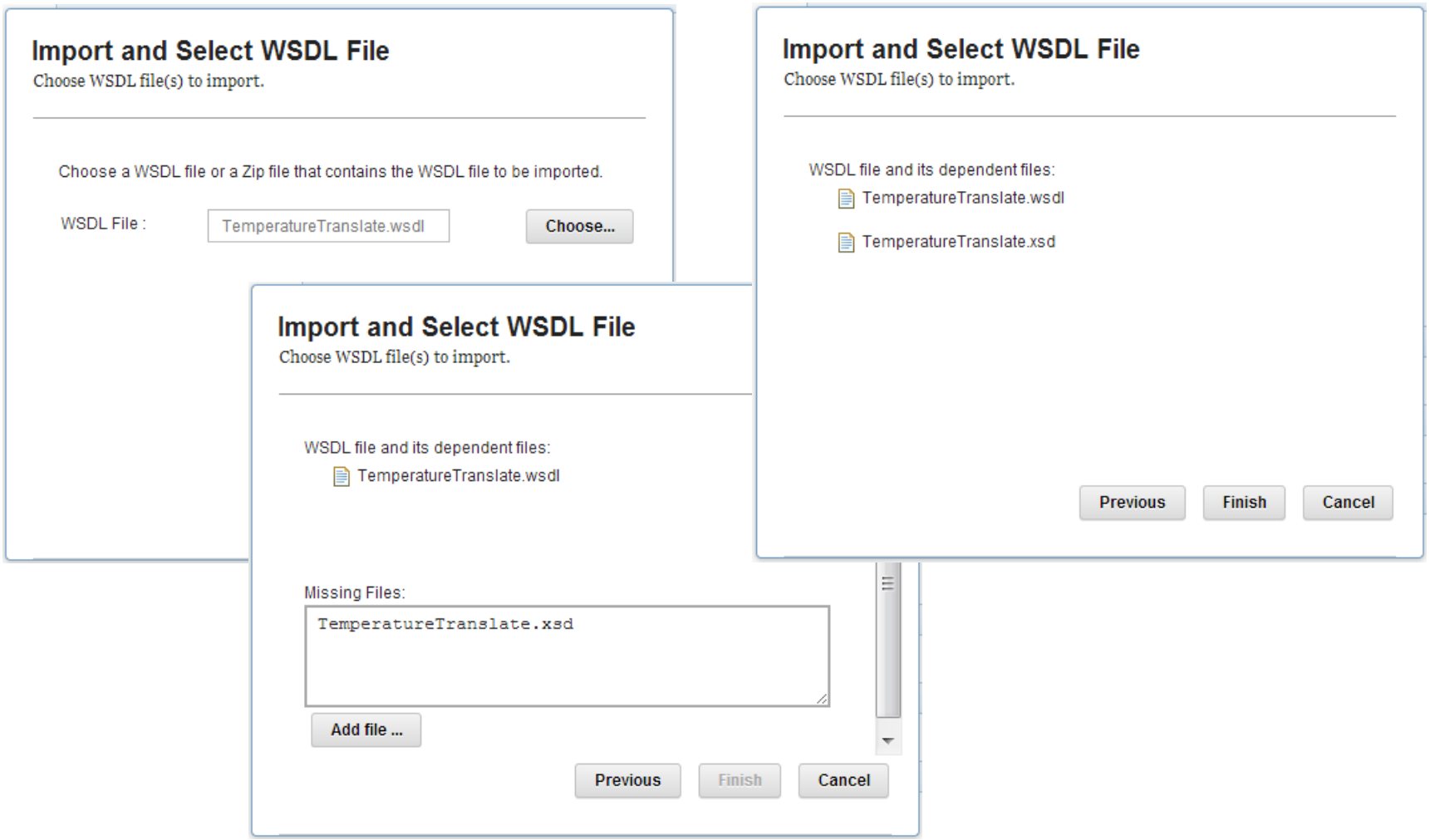


Configure Web Pattern Parameters

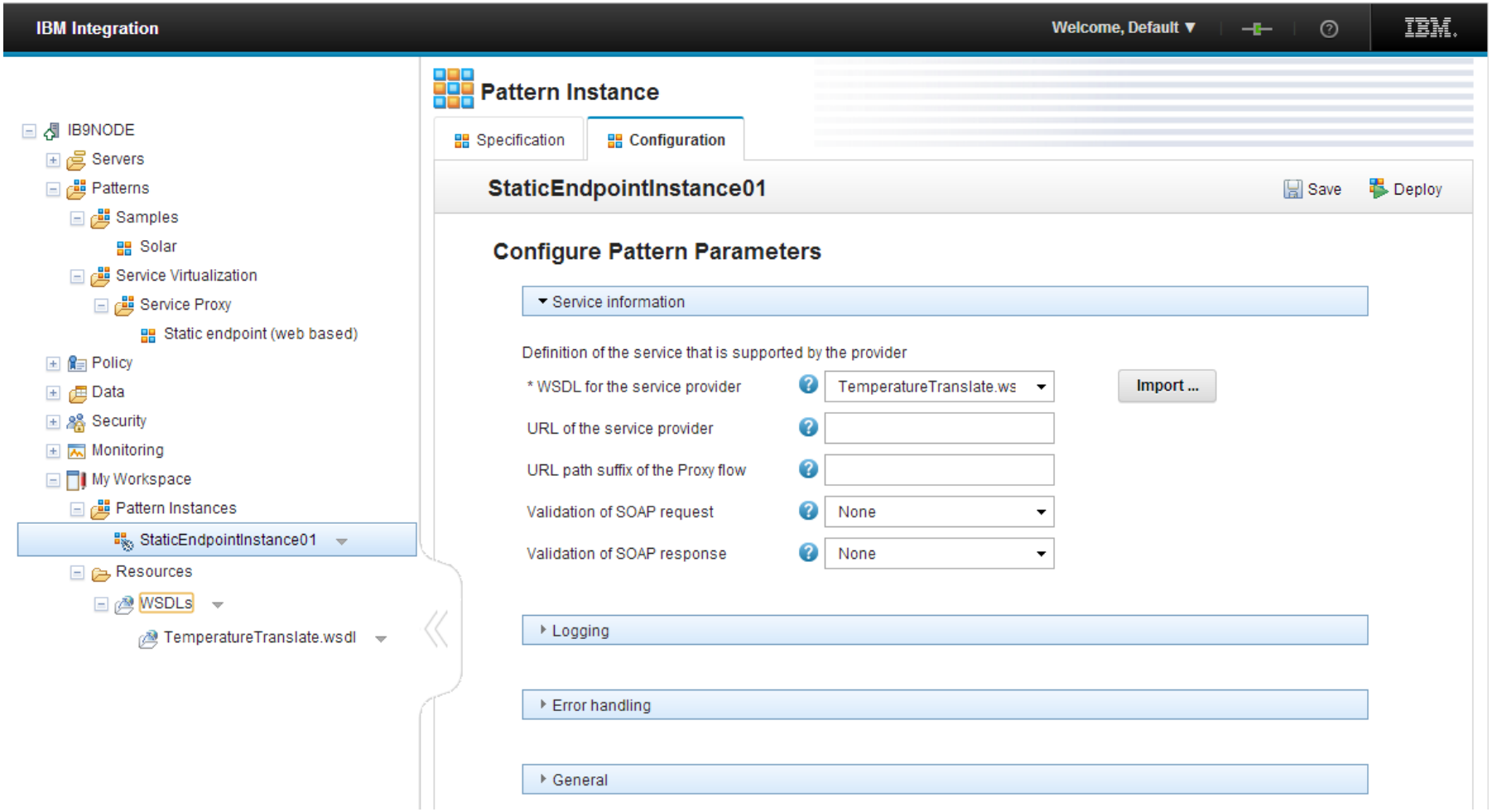


Import and Select a WSDL file

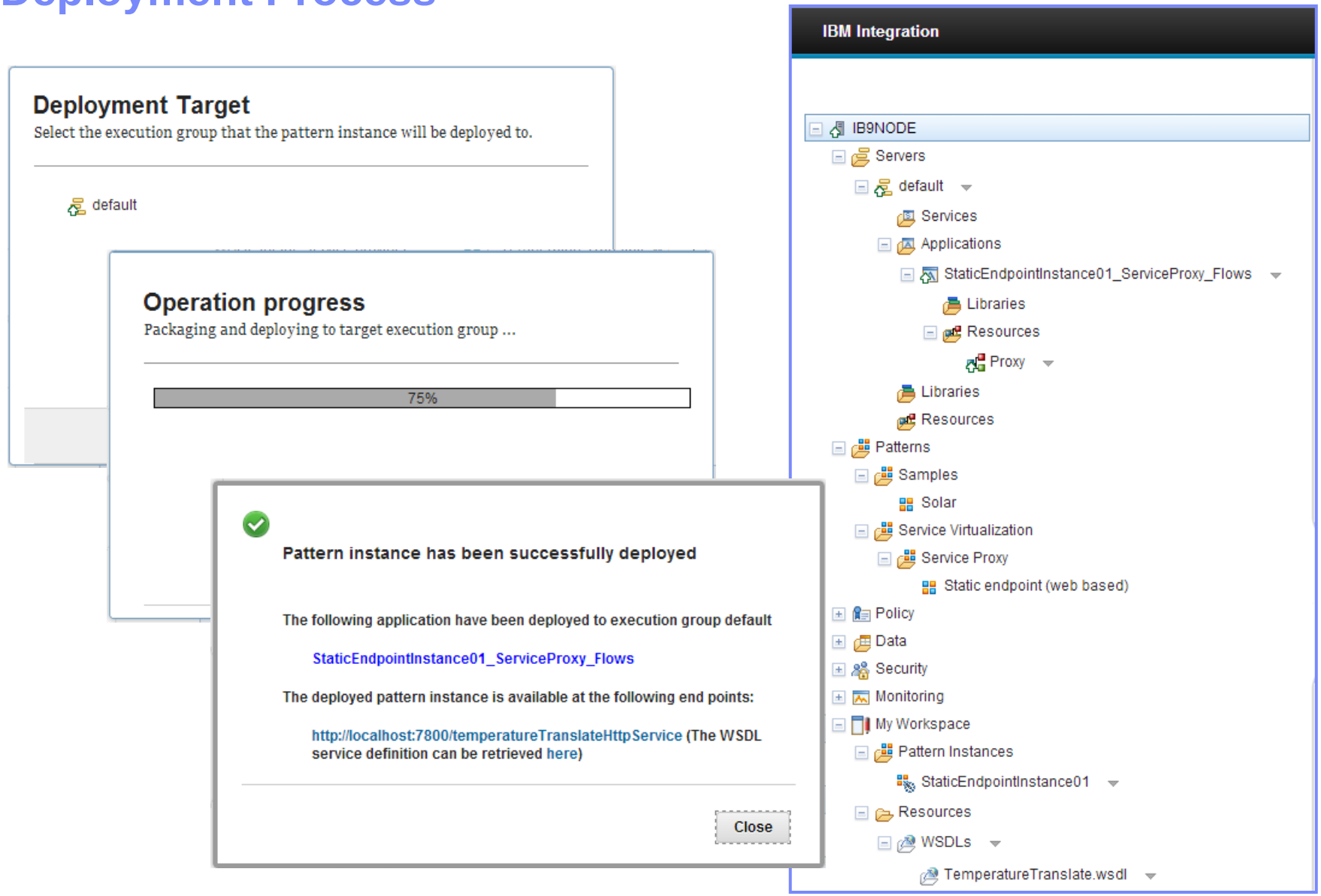
- **The only mandatory pattern parameter is a WSDL interface for the Static Endpoint**



Deploy the pattern



Deployment Process



Results

Name	Date modified	Type	Size
.uploaded	15/07/2013 21:12	File folder	
predeploy	15/07/2013 21:15	File folder	
StaticEndpointInstance01_ServiceProxy_Flows	15/07/2013 21:15	File folder	
StaticEndpointInstance01.bar	15/07/2013 21:15	Toolbar Setup File	9 KB
StaticEndpointInstance01_configuration.xml	15/07/2013 21:13	XML Document	1 KB

Welcome, Default ▼

Admin Log

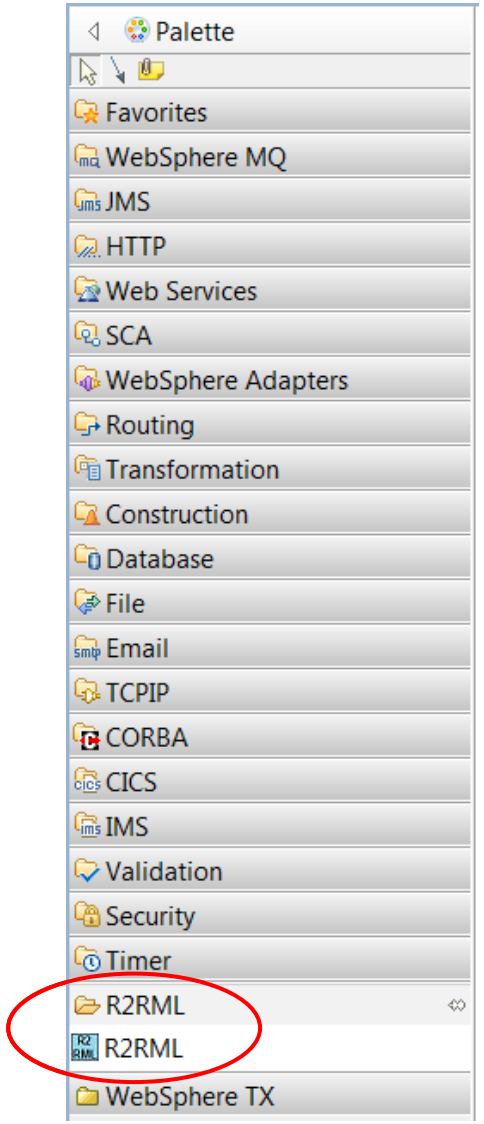
Overview

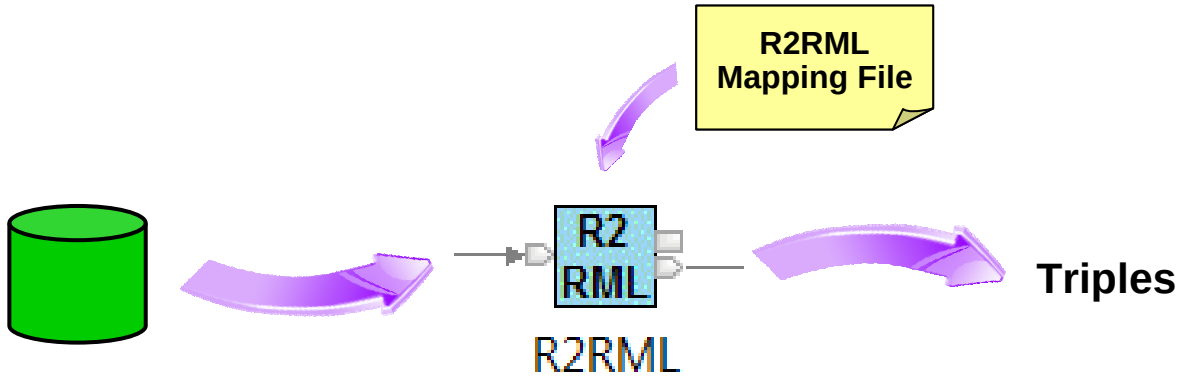
C:\ProgramData\IBM\MQSI\registry\IB9NODE\CurrentVersion\WebAdmin\user\Default\workspace

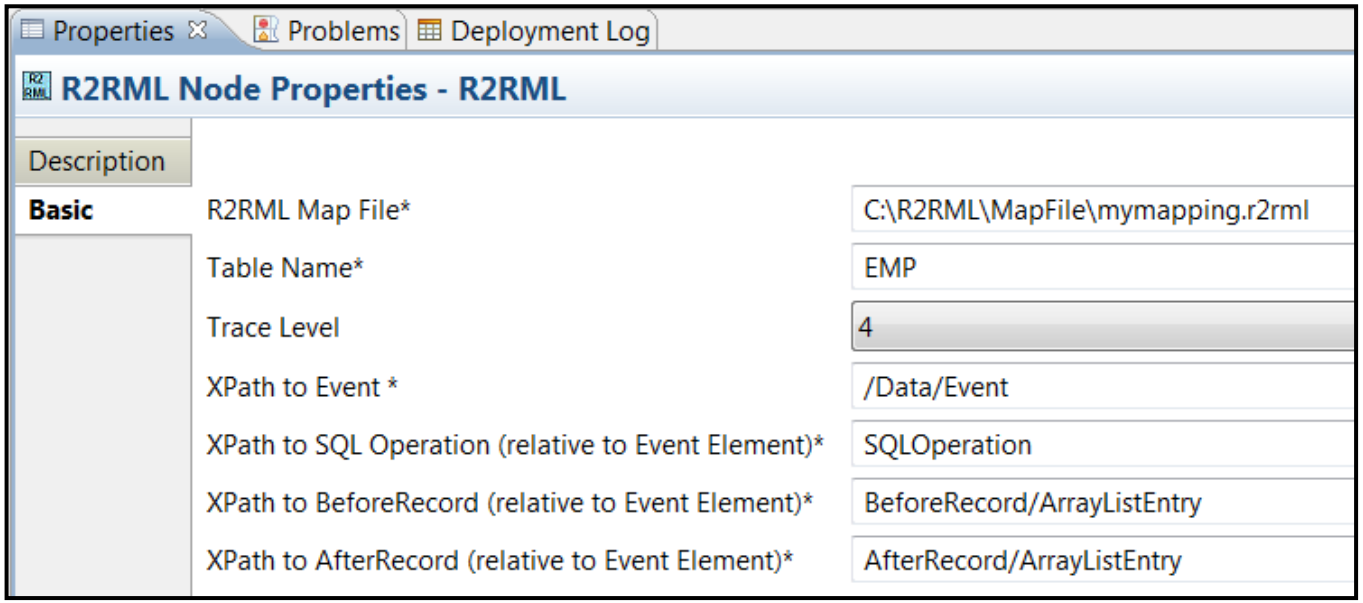
Message	Source	Timestamp	Message Detail
BIP2881I	Change Notification	2013-07-15 21:15:35.292	The resource 'StaticEndpointInstance01_ServiceProxy_Flows' of type '
BIP2871I	Administration Result	2013-07-15 21:15:35.004 GMT Daylight Time	The request made by user 'bthomps[Default]' to 'deploy' the resource 'C:\ProgramData\Application Data\IBM\MQSI\registry\IB9NODE\CurrentVersion\WebAdmin\user\Default\workspace\StaticEndpointInstance01.bar' of type 'BAR' on parent 'default' of type 'ExecutionGroup' has the status of 'COMPLETE'.
BIP2873I	Administration Request	2013-07-15 21:15:26.434	The user 'bthomps[Default]' has requested a deploy of the BAR file 'C:\
BIP2871I	Administration Request	2013-07-15 21:15:08.603	The request made by user 'bthomps[Default]' to 'execute' the resource

User Defined Nodes

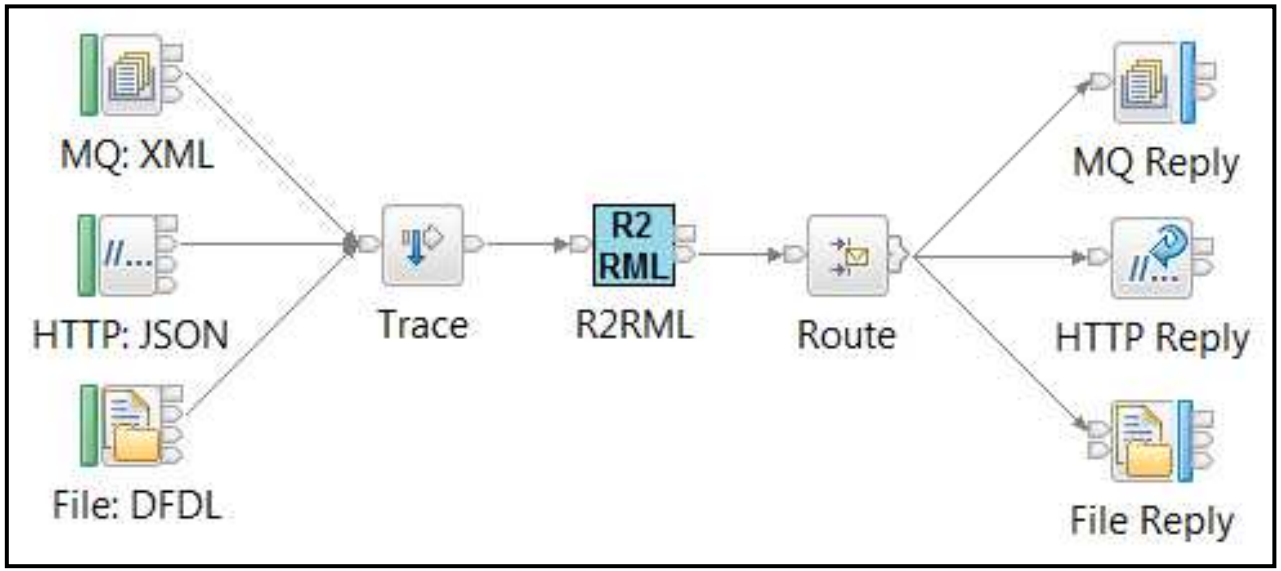
An example UDN … Introducing the R2RML Node …





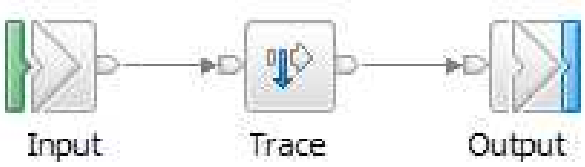


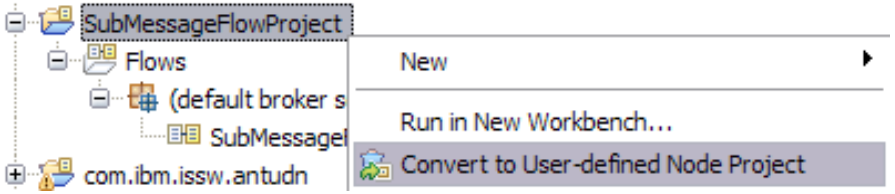
Example Message Flow

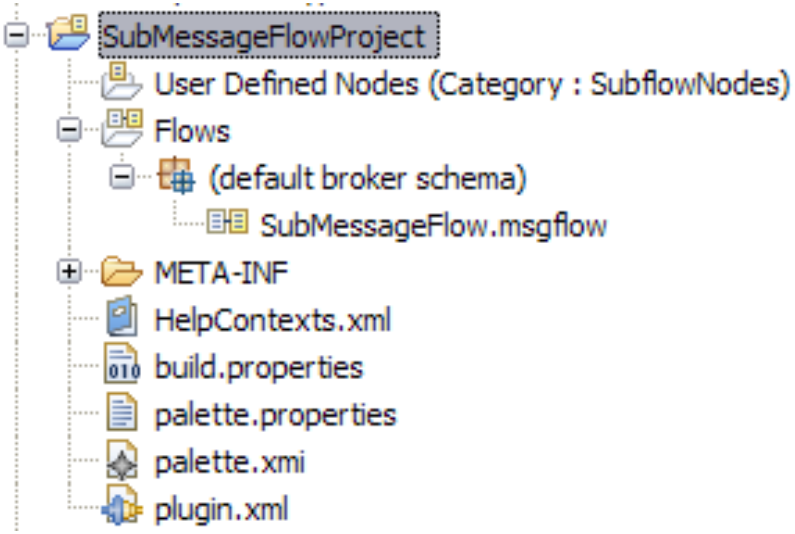


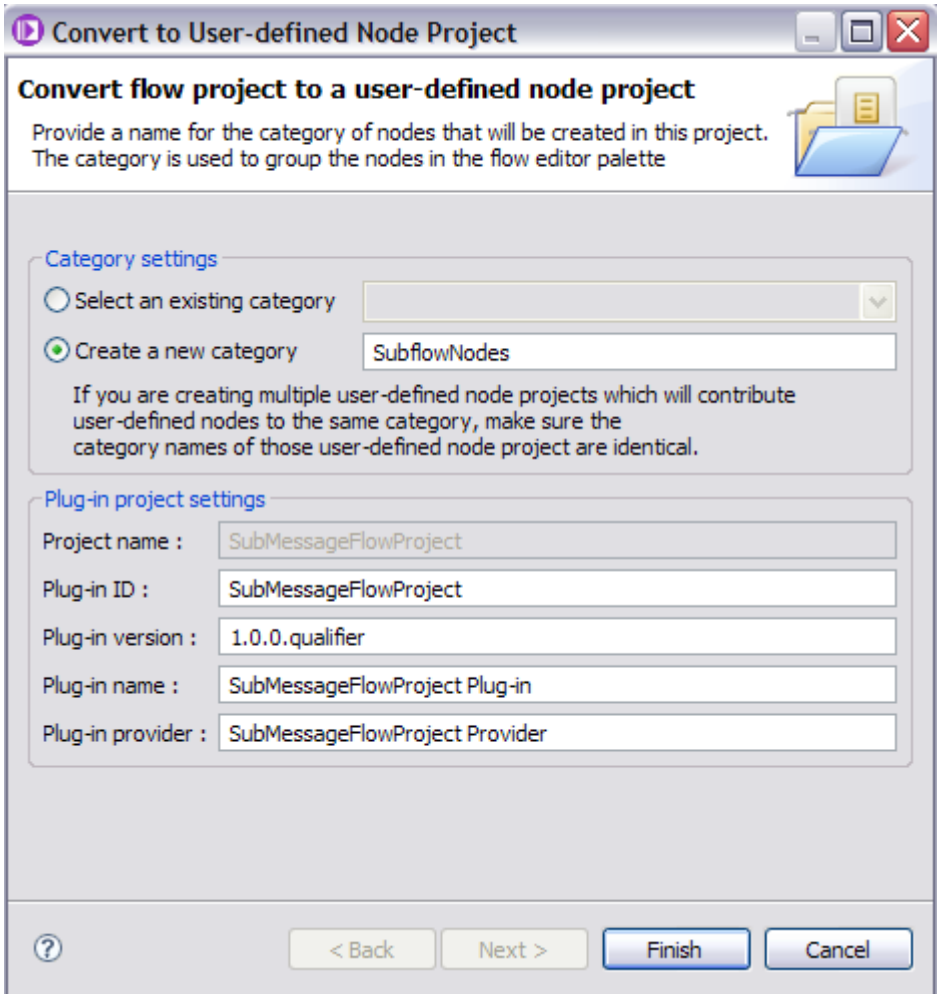
Subflow User Defined Nodes

Converting a Subflow into a UDN

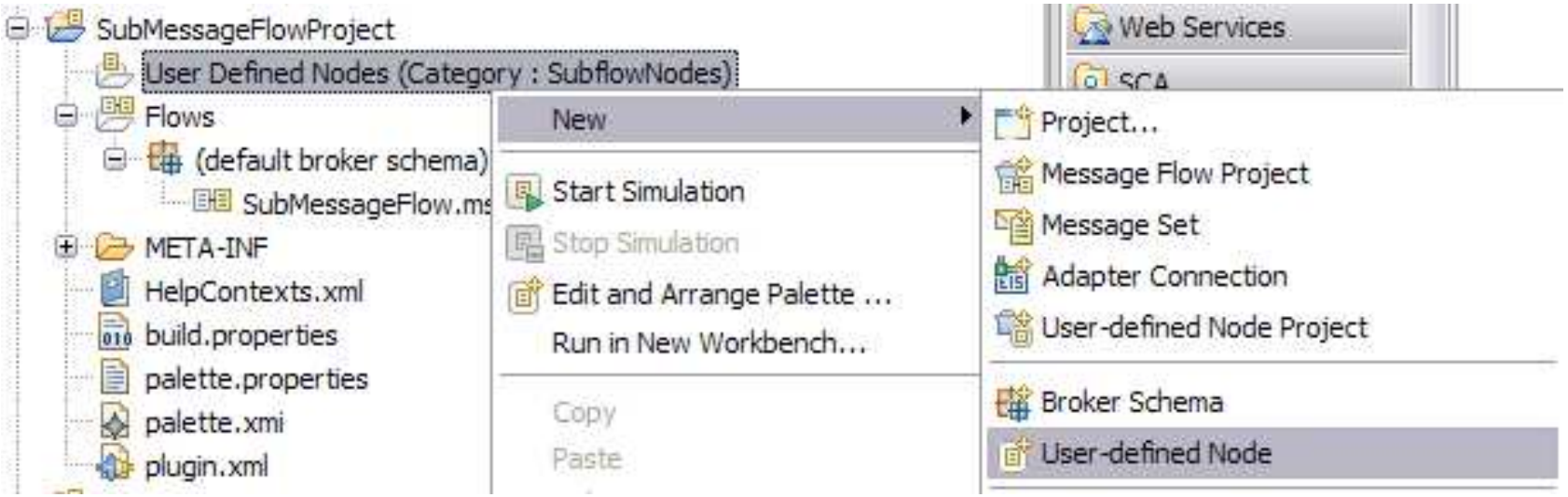




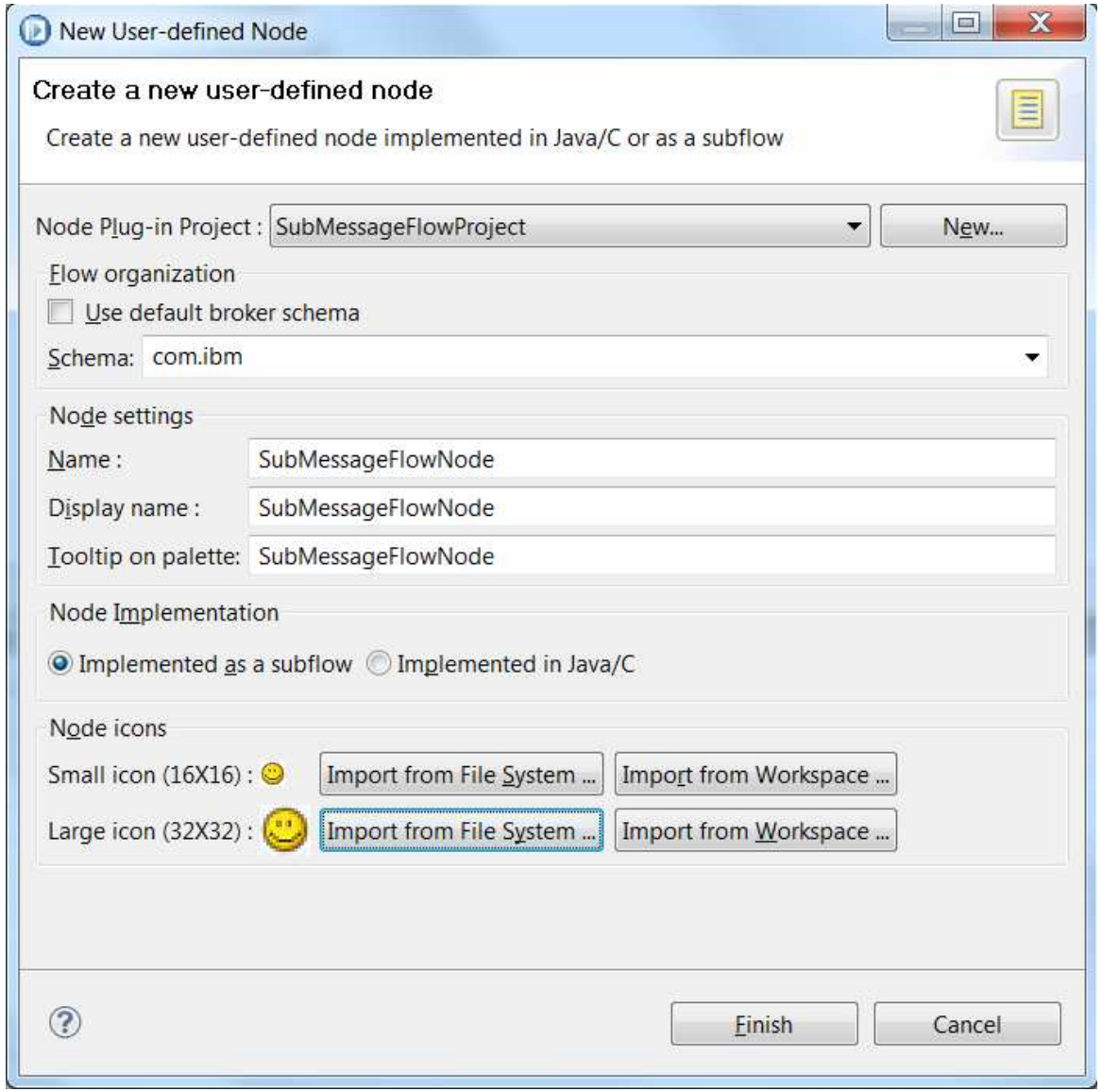




Add User Defined Node File

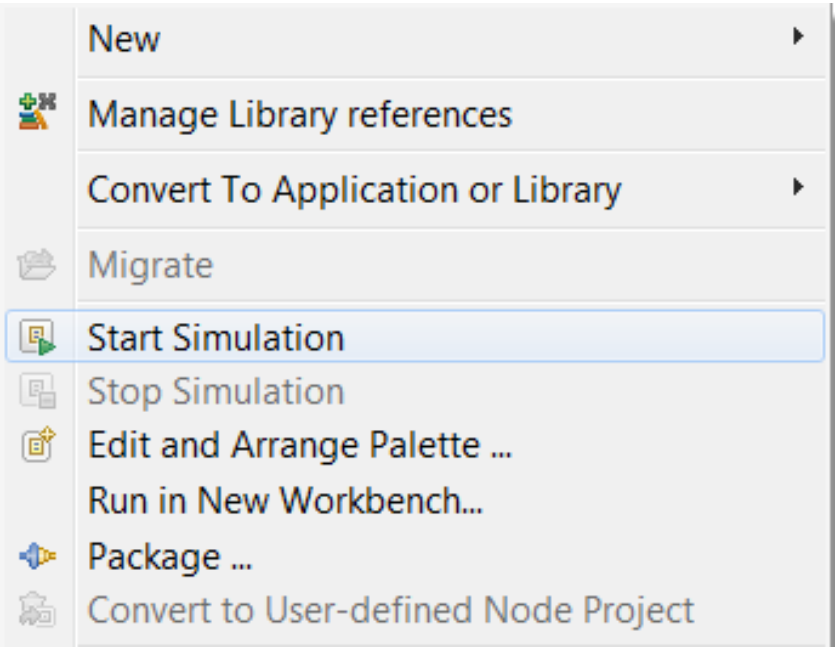


New User Defined Node

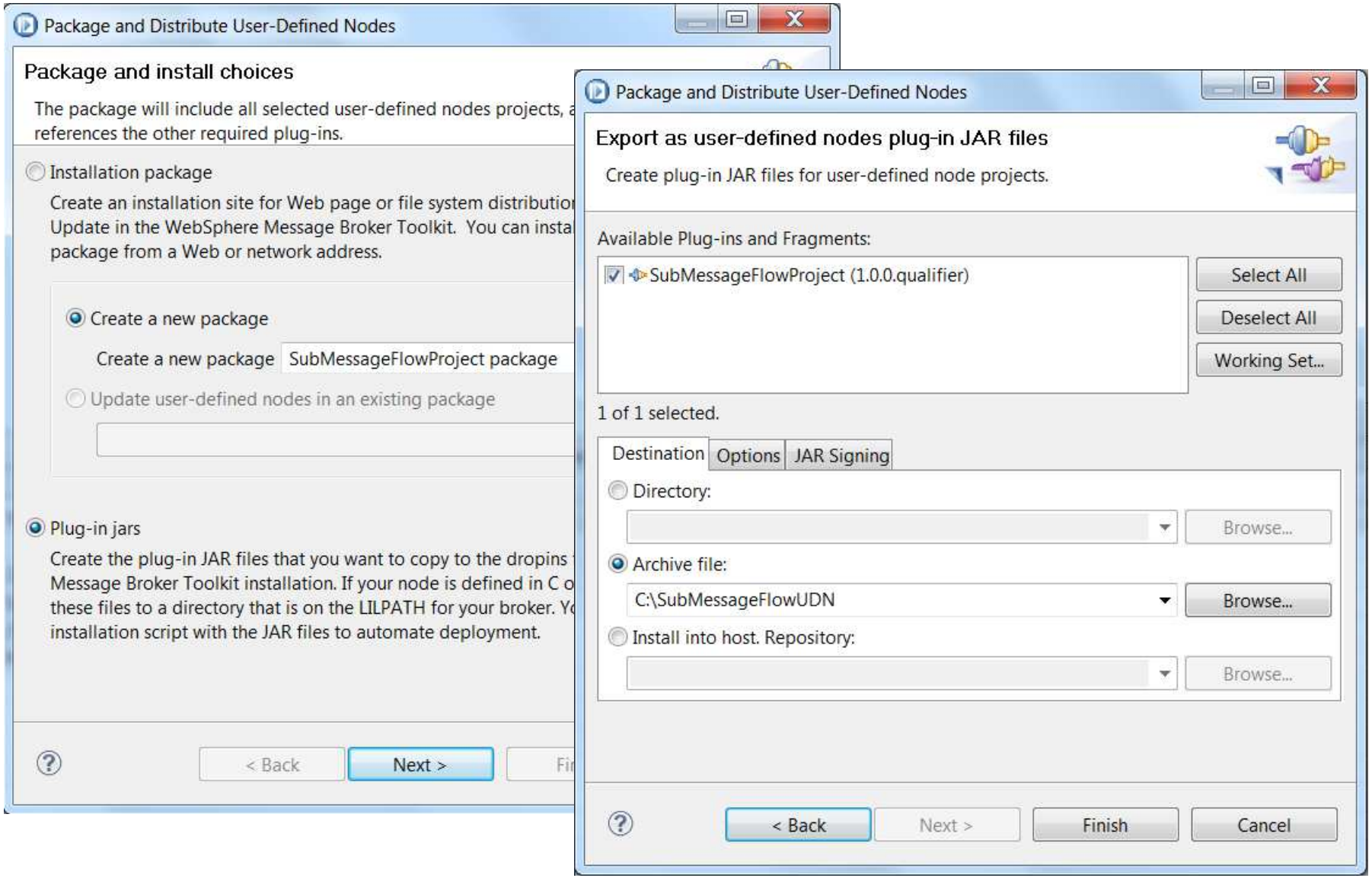


Resulting Project – Right click UDN Project and Simulate to Test

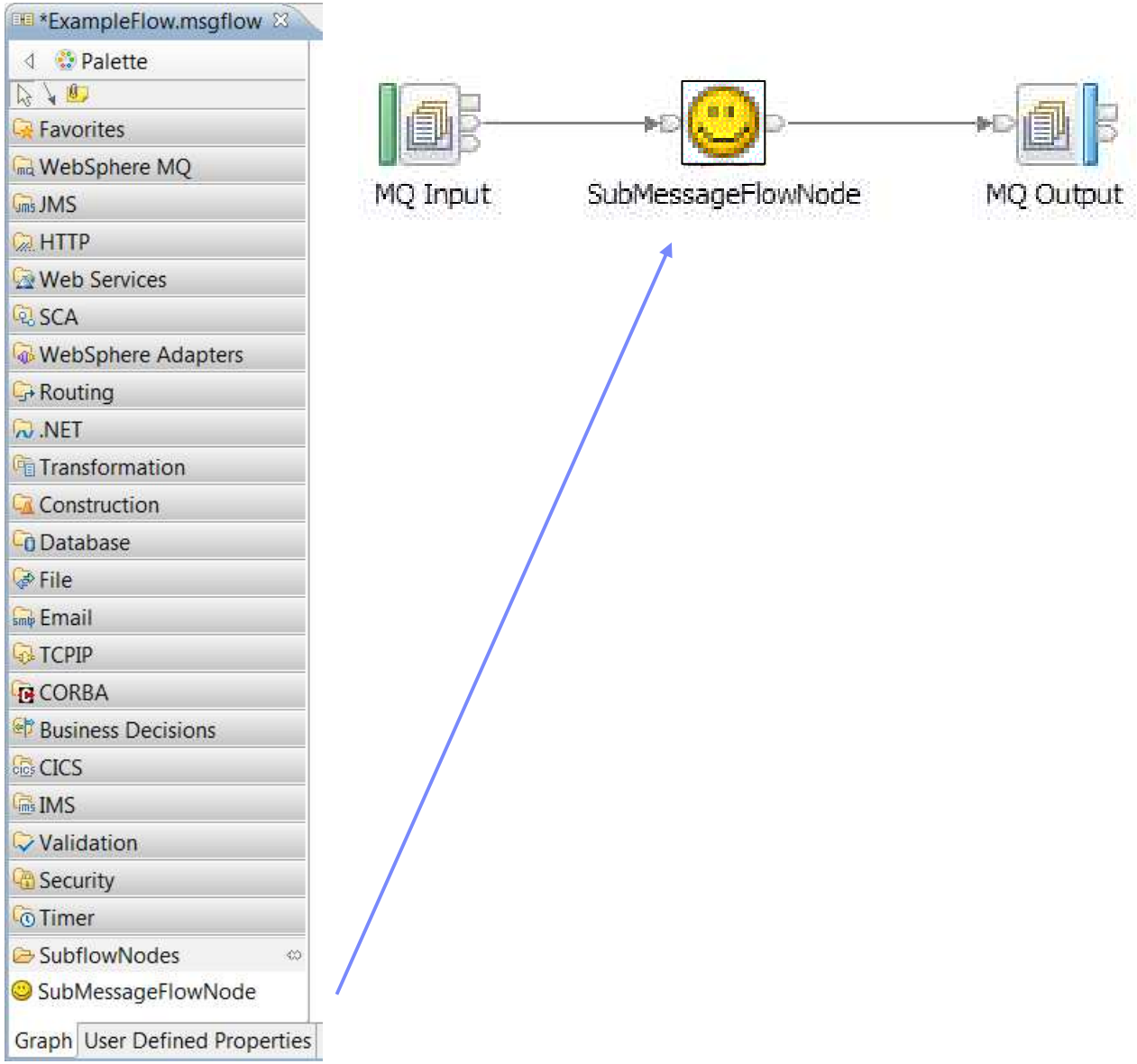
- SubMessageFlowProject
  - User Defined Nodes (Category : SubflowNodes)
    - com.ibm
      - SubMessageFlowNode.msgflow (SubMessageFlowNode)
      - SubMessageFlowNode.properties
  - Other Resources
    - icons/full/clcl16/com/ibm
    - icons/full/obj16/com/ibm
    - icons/full/obj32/com/ibm
    - META-INF
    - build.properties
    - HelpContexts.xml
    - palette.properties
    - palette.xmi
    - plugin.xml



Packaging a User Defined Node

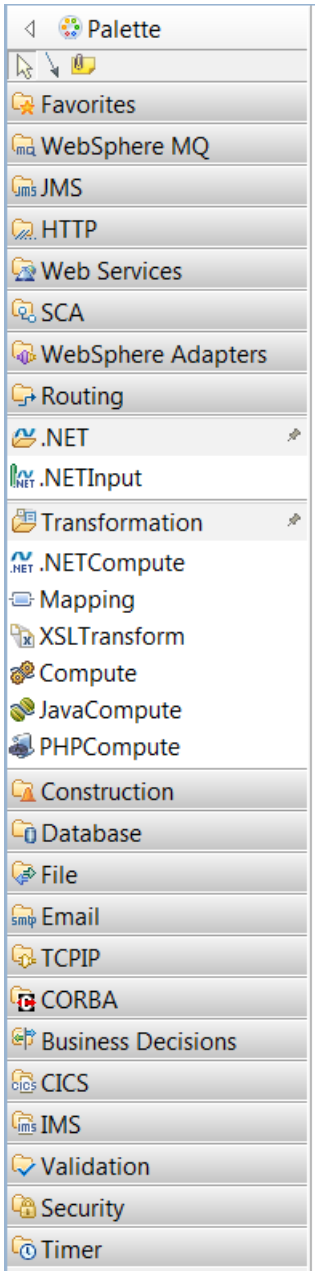


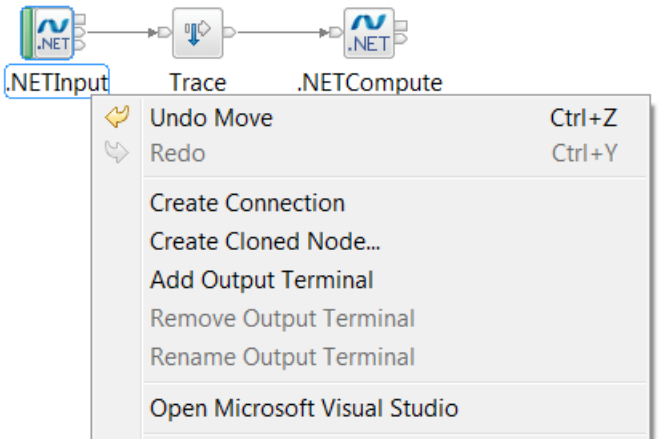
SubFlow added to the node palette

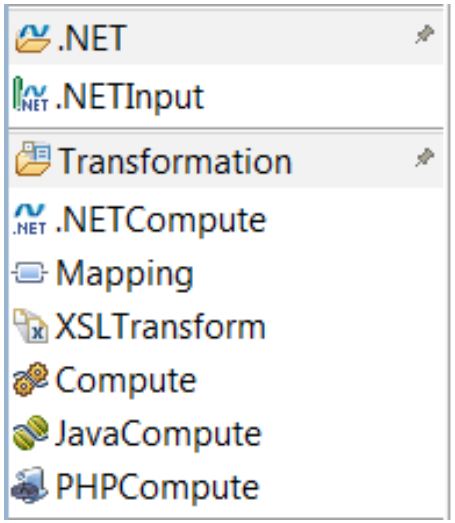


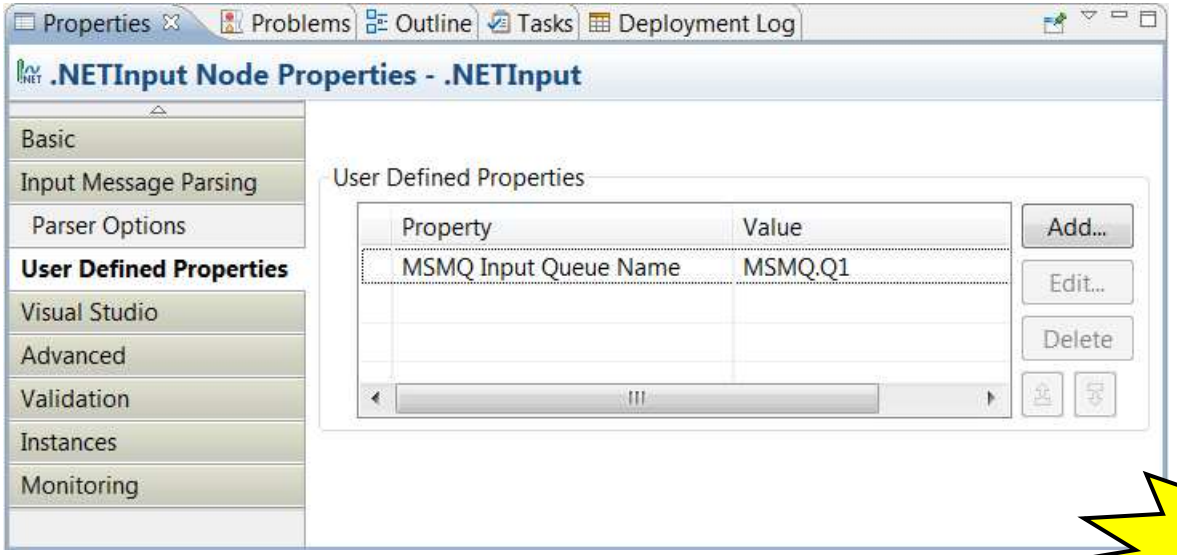
Clone Nodes

.NET Cloned Nodes



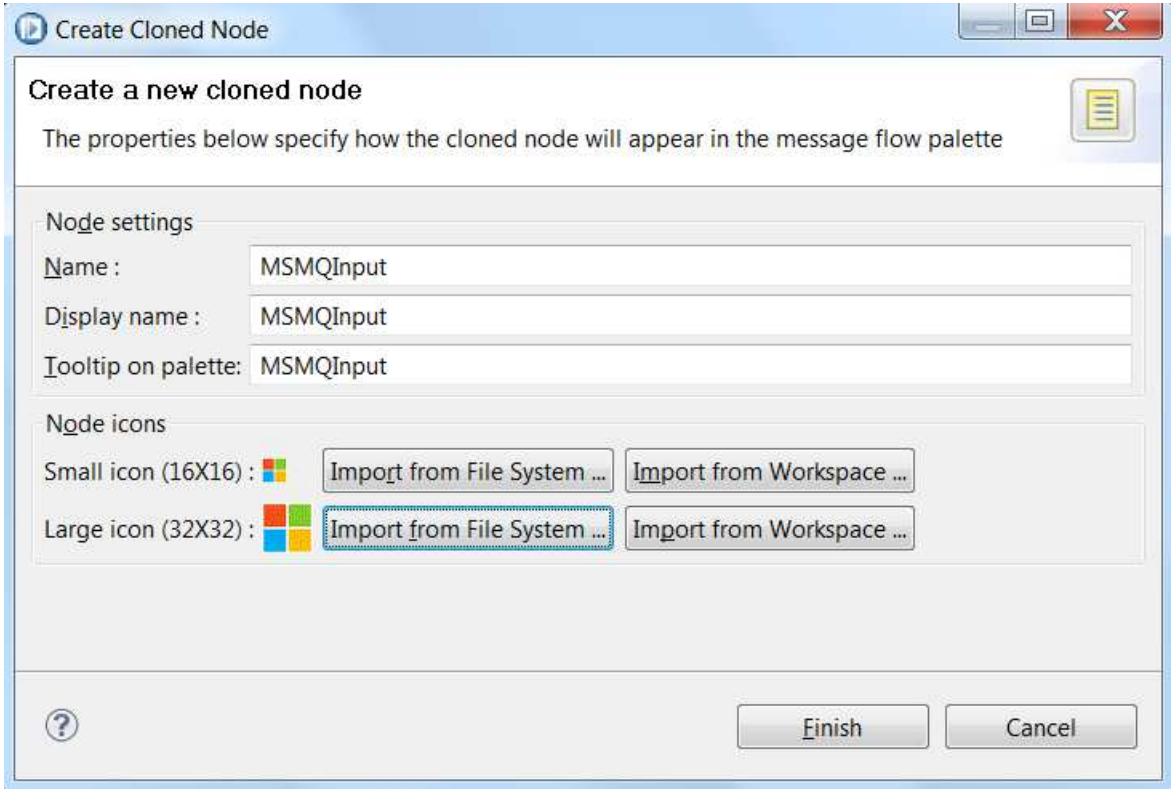


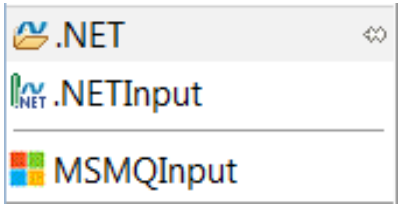




New in IIBv9

Create Cloned node process



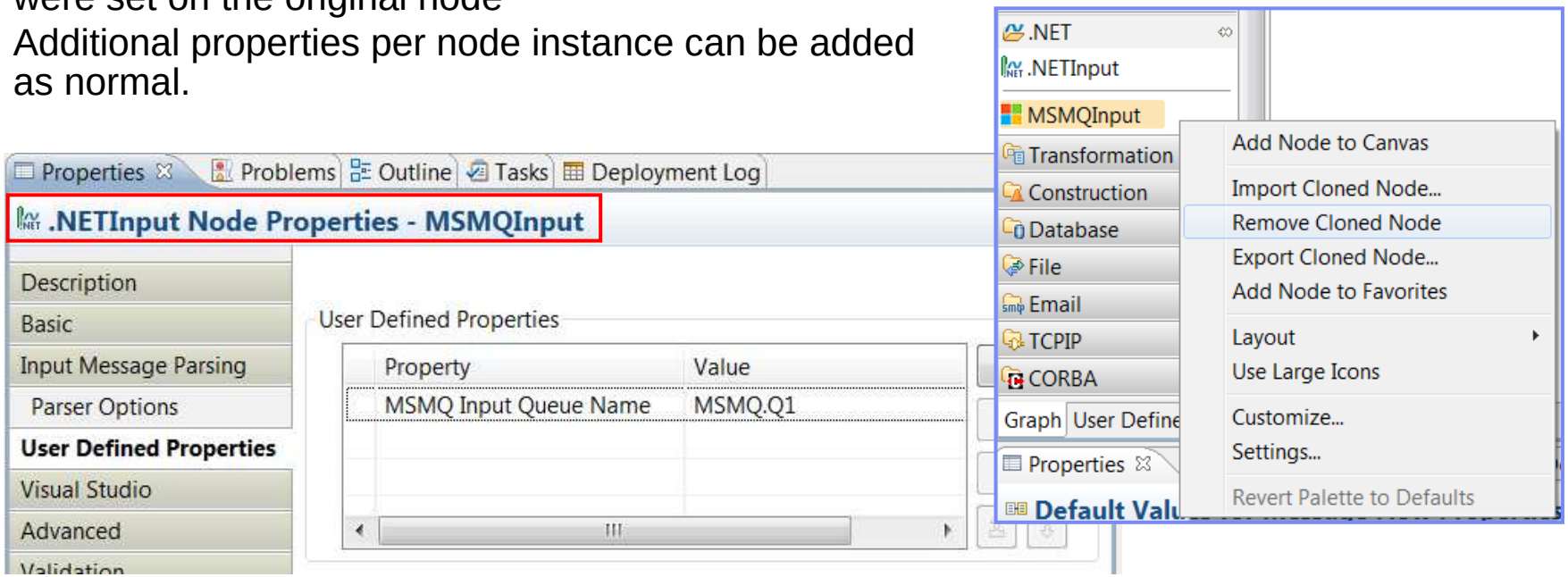


New in IIBv9

Cloned node example

New in IIBv9

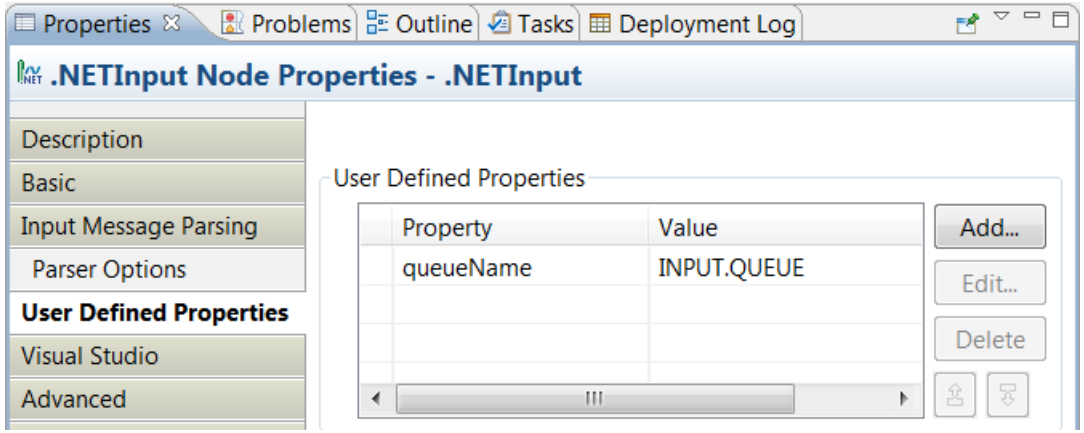
- The cloned node is added to the palette
- The cloned node carries with it the properties which were set on the original node
- Additional properties per node instance can be added as normal.




```
<nodes xmi:type="ComIbmDotNetInput.msgnode:FCMComposite_1" xmi:id="FCMComposite_1_1" location="107,127">
  <translation xmi:type="utility:ConstantString" string=".NETInput"/>
  <userTable userProperty="MSMQ Input Queue Name" userValue="MSMQ.Q1"/>
</nodes>
<nodes xmi:type="ComIbmDotNetInput.msgnode:FCMComposite_1" xmi:id="FCMComposite_1_2" location="100,254"
       derivedType="MSMQInput.templatemsgnode">
  <translation xmi:type="utility:ConstantString" string="MSMQInput"/>
  <userTable userProperty="MSMQ Input Queue Name" userValue="MSMQ.Q1"/>
</nodes>
```

Accessing .NETInput node properties

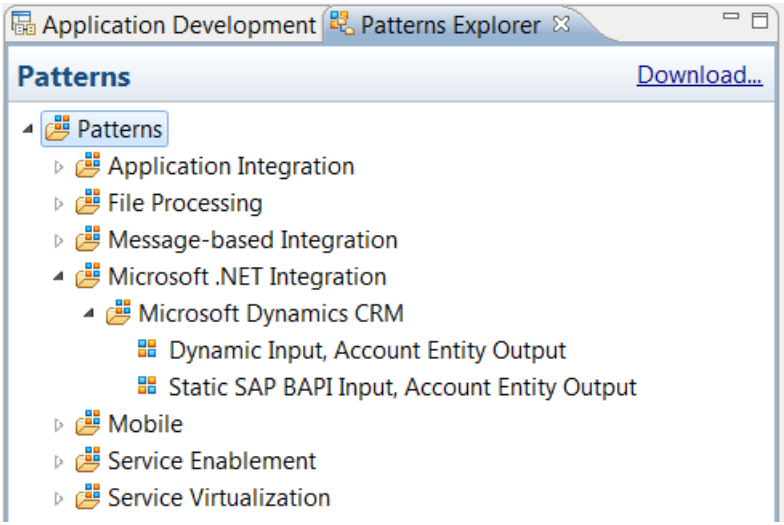
- Use the User Defined Properties tab of a .NETInput node to specify properties which can be accessed by your .NET connector code.
- The `EventInputConnector` or `PollingInputConnector` class, both define a properties parameter, which is of the data type Dictionary<string,string>. This parameter carries user-defined properties for the node, and any flow-level user-defined properties.



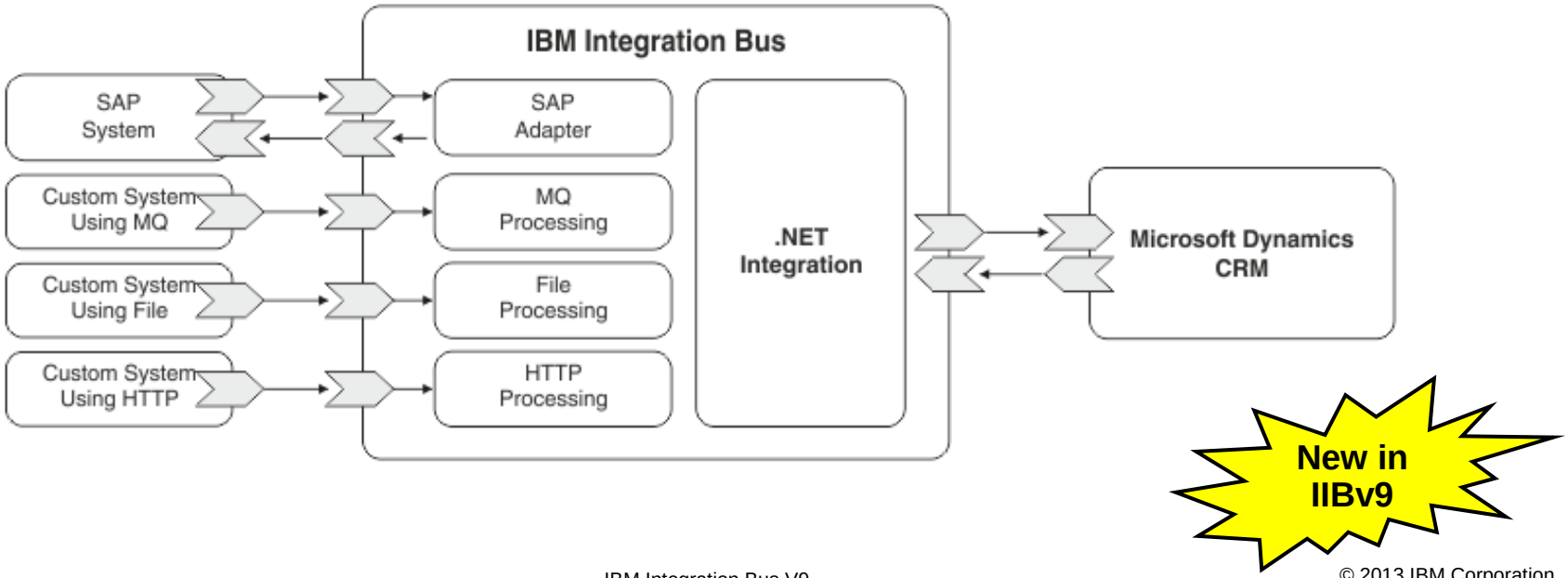

```
public PollingInputConnector(NBConnectorFactory connectorFactory, string name,
     Dictionary<string, string> properties) : base(connectorFactory, name, properties)
{
  // Check the Dictionary of user defined properties defined on the node.
  // If there is a property named queueName then take its value for use later:
  if (properties.ContainsKey("queueName"))
  {
     queueName = properties["queueName"];
  }
}
```

New in IIBv9

New .NET Patterns

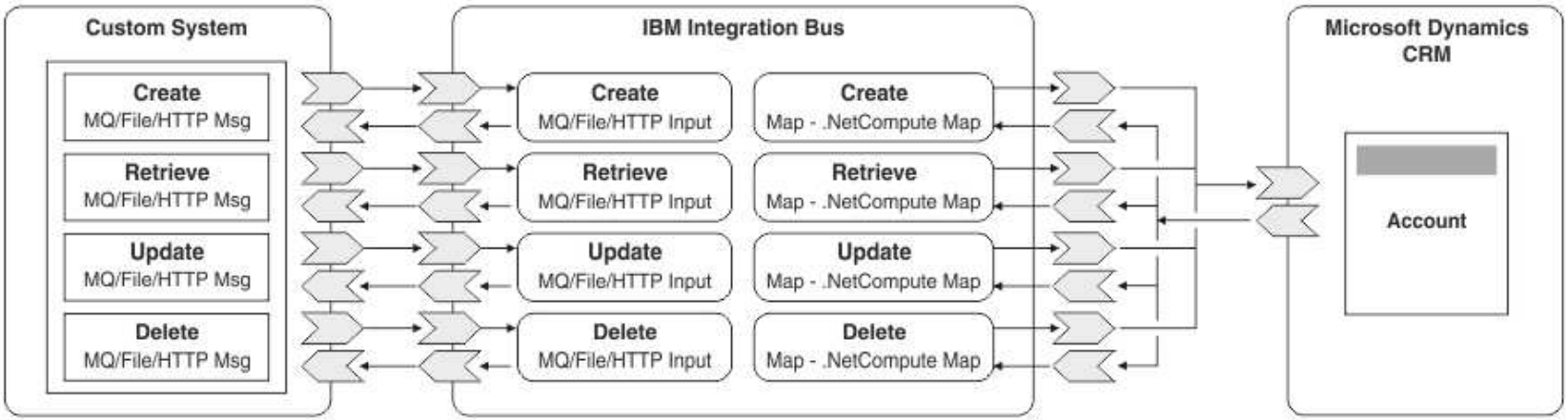


- Two new patterns for integrating with Microsoft Dynamics CRM
- SAP, or “raw” input protocols
- Top level Source / Target maps created
- Early bound C# code generated for the Account Entity Microsoft Dynamics CRM object

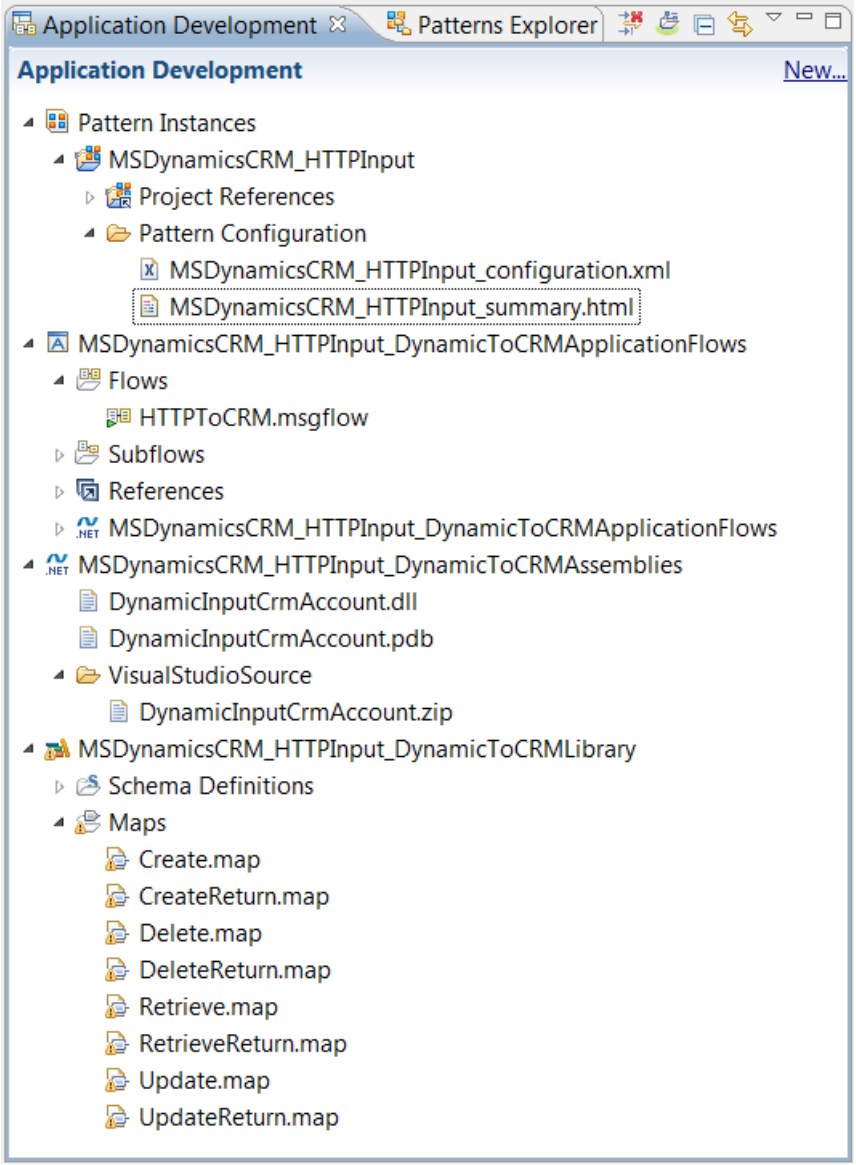


Microsoft Dynamics CRM – Dynamic Input, Account Entity Output

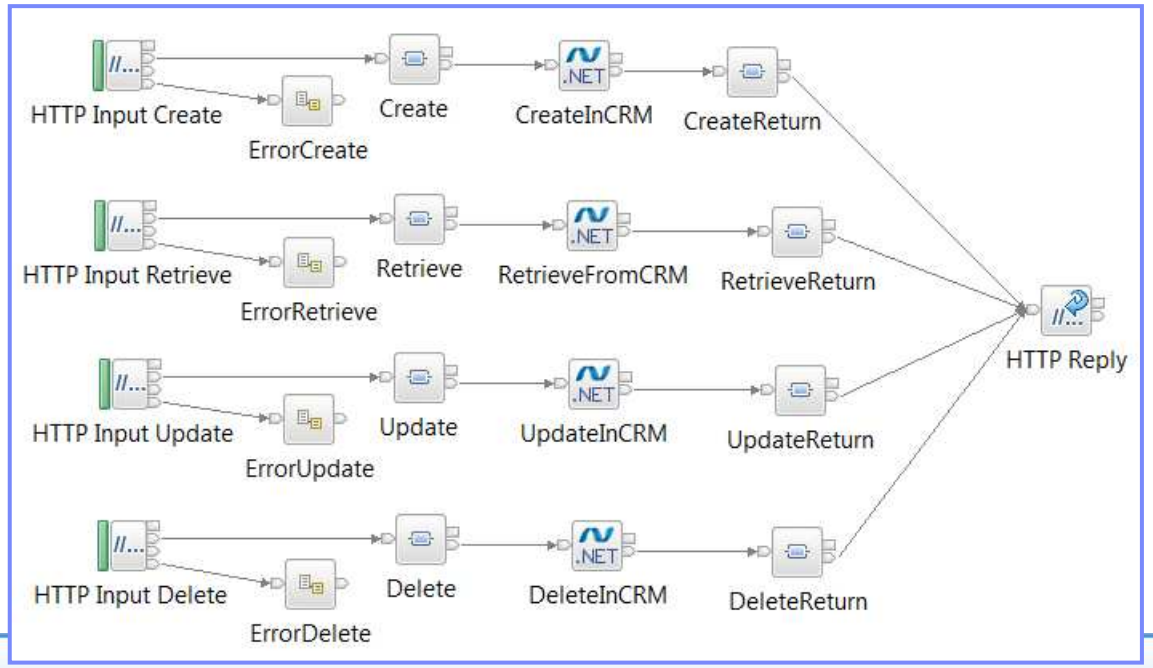
- The pattern creates a message flow that provides either MQ, File, or HTTP input nodes to receive one or more of four event types: Create, Retrieve, Update, and Delete events
- Received events are mapped into the relative Microsoft Dynamics CRM Account Entity data using a graphical data map. The input object for the map is specified at pattern creation time.
- An output (response) object is also specified.
- Flow invokes the .NETCompute node that connects to Microsoft Dynamics CRM, and implements the early binding programming style to transfer the Account Entity data from the Mapping node.
- The CRM response data received by the .NETCompute node is passed to an additional Mapping node which converts to the output object.
- The .NET project contains the assembly files, and relative configuration files, used by the .NETCompute node in the application.

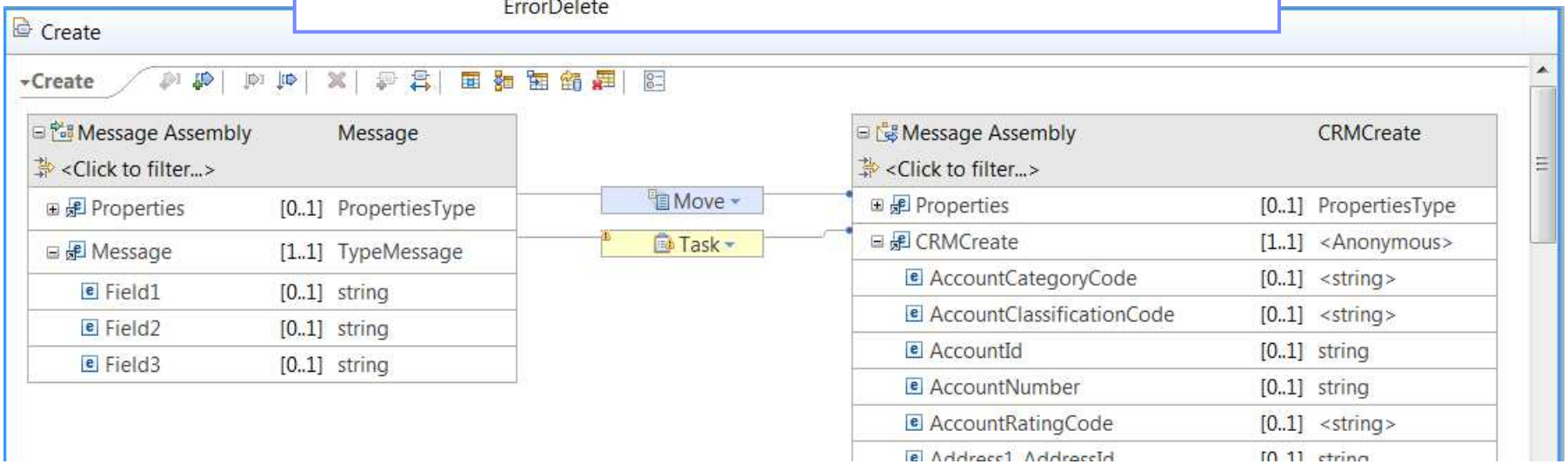


Generated Resources



Pattern Example





Pattern Instance Results Summary

Summary for pattern instance MSDynamicsCRM_HTTPInput

To complete pattern instance MSDynamicsCRM_HTTPInput_DynamicToCRMApplicationFlows, please review the actions in this file

Tasks to complete

The created flow instance HTTPToCRM will not run successfully until the Graphical Data Maps on the mapping nodes listed below are completed to transform data from the input message to the required fields of the CRM Account action data message. The maps are generated with a "Task" transform.

Below new maps must be edited:

- Create.map
- CreateReturn.map
- Retrieve.map
- RetrieveReturn.map
- Update.map
- UpdateReturn.map
- Delete.map
- DeleteReturn.map

Further optional tasks

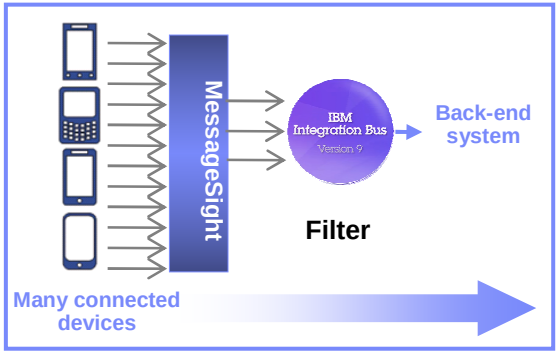
There are further optional tasks that you might consider.

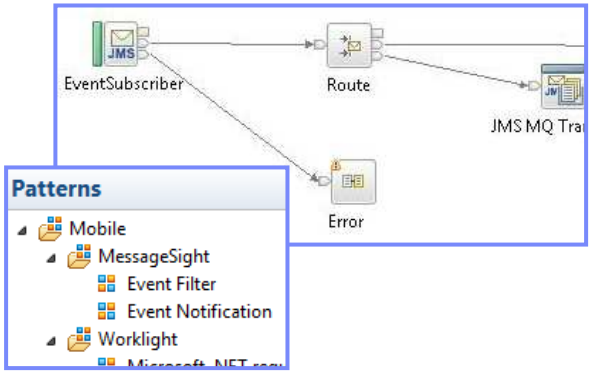
- The flow HTTPToCRM currently uses different URL suffixes to distinguish different operations, you can modify the flow by using only one input node for each transport, and add a filter or compute node after the input node to decide which operation the input message want to invoke in CRM system to meet your business needs.
- The flow instance HTTPToCRM created by the pattern will only process the standard attributes on the Microsoft CRM account entity. If you have customized your CRM system to add custom attributes to the CRM account entity, and you want to include these in the Create, Retrieve or Update operations, you will need to extend both the Graphical Data Mapping and the .NETCompute nodes. To do this:
  1. Extend the CRM account data models to include the custom attributes. These are defined by the XML schema installed in the created application.
  2. Extend the .NETCompute node C# code to set and get the custom attributes. A file named CRMSourceCode.zip containing the Microsoft Visual Studio project holding the source code for the .NETCompute nodes is created in the .Net project **MSDynamicsCRM_HTTPInput_DynamicToCRMAssemblies** , in the directory VisualStudioSource.
  3. Extend the Graphical Data Map on the mapping nodes to transform the input data to the additional custom attributes now modeled in the updated schema for the Microsoft CRM account entity data format.

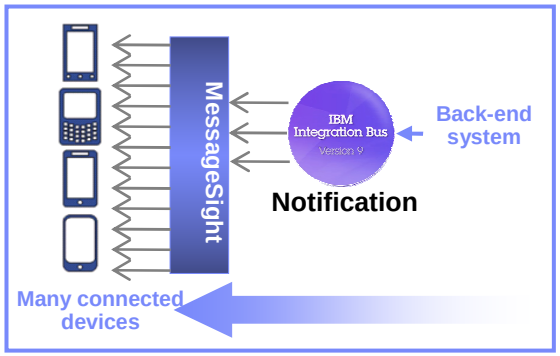
Easily Integrate with Appliance-based Messaging

- **Introducing IBM MessageSight**
  - An appliance-based messaging server built on special purpose hardware
  - Supports very large numbers of connected clients and devices, and high volumes of messages
  - Secures the edge of the enterprise and enables use cases like mobile and telemetry

- **Two new patterns for integrating IBM MessageSight with backend systems**
  - Covers common use cases for bi-directional connectivity
  - Use of JMS enables standards-based appliance connectivity that is also extensible to other providers
  - Pattern design allows for future selection of high performance, standard MQTT as protocol







- **1) Event Filter Pattern**
  - Messaging appliance routes inbound events into the broker via JMS
  - The broker narrows down events using decision service and inserts the subset into backend systems

- **2) Event Notification Pattern**
  - The broker detects an event from a backend system (e.g. message queue, database trigger)
  - Broker fans out event via JMS to the appliance to interested connected clients

© 2013 IBM Corporation

Summary

- Pattern Development Concepts
- Subflow User Defined Nodes
- Web Patterns
- Clone nodes and new IIBv9 Patterns