B2B Guidance Install and Demo Notes

Tuesday, January 12, 2010
3:02 PM

Prerequisites

- Visual Studio 2008 SP1

- .NET 3.5 SP1

- SQL Server 2008

- BizTalk Server 2009 Environment
    - Including BAM components
    - Including EDI components
    - Including the WCF LOB Adapter for SQL

- ESB Toolkit
    - ESB Core
    - ESB Exception Management Framework
    - ESB Management Portal Sample

- Visual Studio Team System 2008 Team Explorer

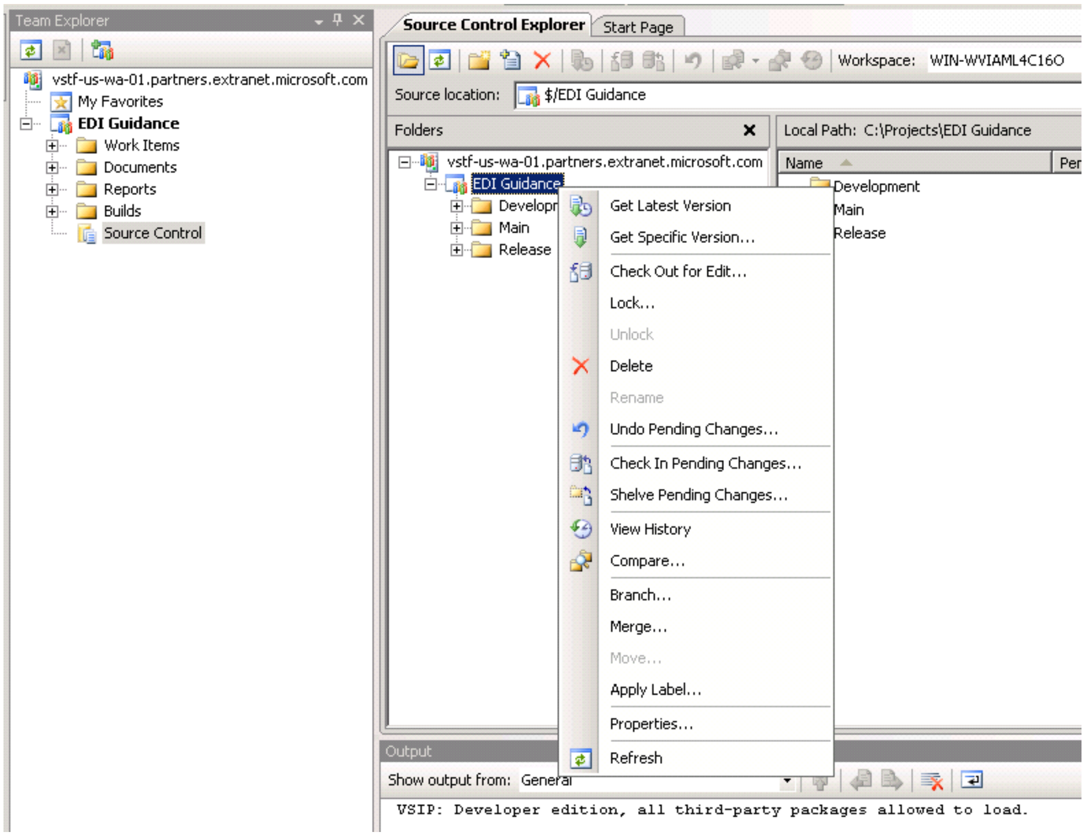
- Obtain access to the EDI Guidance VSTS Project (Contact Jeff King or Karl Rissland)

Install Steps

Pull down the source and setup the environment

- Connect to the VSTS project
  Server Name: vstf-us-wa-01.partners.extranet.microsoft.com
  Port Number: 8443
  Protocol: HTTPS

- Open Team Explorer, double click Source Control, and Get Latest Version



Workspace: WIN-WVIAML4C16O

Source location: $/EDI Guidance

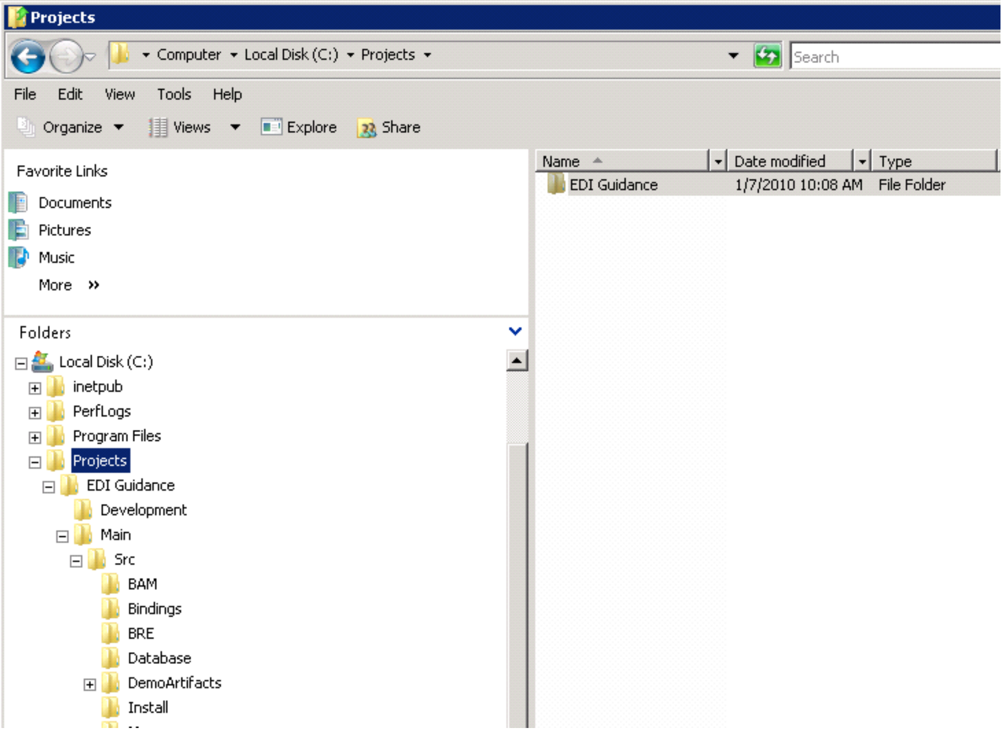
Local Path: C:\Projects\EDI Guidance

Show output from: General

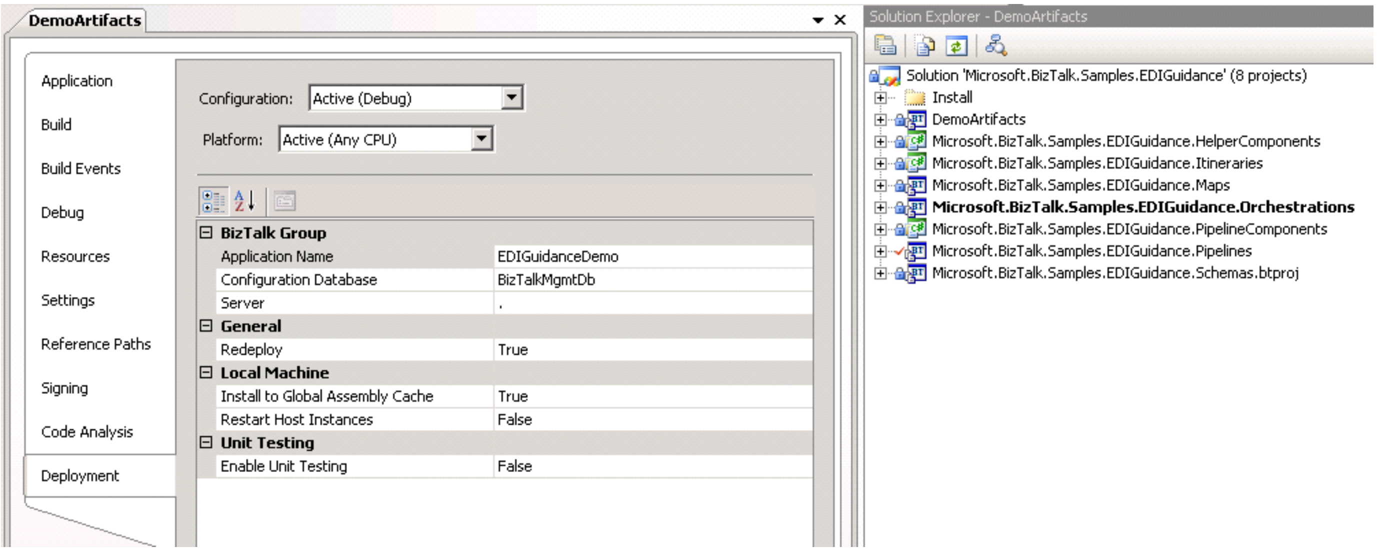
```
VSIP: Developer edition, all third-party packages allowed to load.
```

Put the source in the folder C:\Projects.

Your directory structure should look like



- Make sure the deployment properties for the BizTalk projects are correct. Right Click on the DemoArtifacts project and select properties. Make sure the deployment properties are set as follows.
  Application Name: EDIGuidanceDemo
  Configuration Database: BizTalkMgmtDb (Assuming you used this when setting up BizTalk)
  Server: . (a period)


DemoArtifacts
Application
Build
Build Events
Debug
Resources
Settings
Reference Paths
Signing
Code Analysis
Deployment
Solution Explorer - DemoArtifacts
Solution 'Microsoft.BizTalk.Samples.EDIGuidance' (8 projects)
Install
DemoArtifacts
Microsoft.BizTalk.Samples.EDIGuidance.HelperComponents
Microsoft.BizTalk.Samples.EDIGuidance.Itineraries
Microsoft.BizTalk.Samples.EDIGuidance.Maps
Microsoft.BizTalk.Samples.EDIGuidance.Orchestrations
Microsoft.BizTalk.Samples.EDIGuidance.PipelineComponents
Microsoft.BizTalk.Samples.EDIGuidance.Pipelines
Microsoft.BizTalk.Samples.EDIGuidance.Schemas.btproj

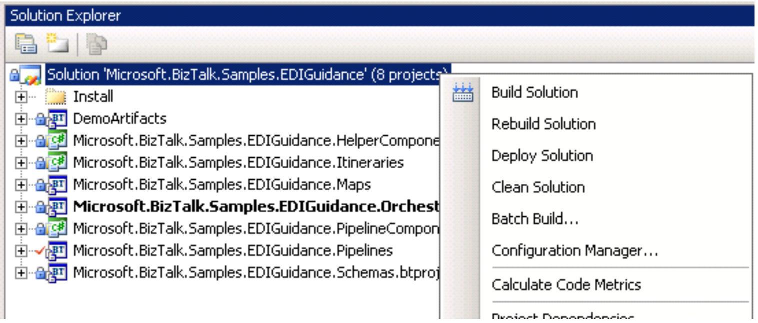
Configuration: Active (Debug)
Platform: Active (Any CPU)

BizTalk Group	
Application Name	EDIGuidanceDemo
Configuration Database	BizTalkMgmtDb
Server	.
General	
Redeploy	True
Local Machine	
Install to Global Assembly Cache	True
Restart Host Instances	False
Unit Testing	
Enable Unit Testing	False

For the BizTalk Projects Microsoft.BizTalk.Samples.EDIGuidance.Maps, Microsoft.BizTalk.Samples.EDIGuidance.Orchestrations, Microsoft.BizTalk.Samples.EDIGuidance.Pipelines, and Microsoft.BizTalk.Samples.EDIGuidance.Schemas are set to
  Application Name: EDIGuidance
  Configuration Database: BizTalkMgmtDb (Assuming you used this when setting up BizTalk)
  Server: . (a period)

- Obtain and install Visual Studio Ref Issue Fix for Knowledge Base Article Number 977428. This is not required for the EDI Guidance to work however this will help keep project references from being lost.

- Rebuild the solution


Solution Explorer
Solution 'Microsoft.BizTalk.Samples.EDIGuidance' (8 projects)
Install
DemoArtifacts
Microsoft.BizTalk.Samples.EDIGuidance.HelperCompone
Microsoft.BizTalk.Samples.EDIGuidance.Itineraries
Microsoft.BizTalk.Samples.EDIGuidance.Maps
Microsoft.BizTalk.Samples.EDIGuidance.Orchest
Microsoft.BizTalk.Samples.EDIGuidance.PipelineCompon
Microsoft.BizTalk.Samples.EDIGuidance.Pipelines
Microsoft.BizTalk.Samples.EDIGuidance.Schemas.btproj
Build Solution
Rebuild Solution
Deploy Solution
Clean Solution
Batch Build...
Configuration Manager...
Calculate Code Metrics

- Verify that the assembly Microsoft.BizTalk.Samples.EDIGuidance.PipelineComponents.dll is installed under C:\Program Files\Microsoft BizTalk Server 2009 Pipeline Components

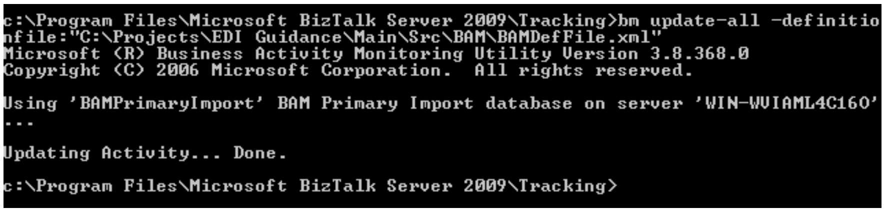
- Click the start menu, type assembly in the search dialog, and hit enter to open the GAC. Verify that Microsoft.BizTalk.Samples.EDIGuidance.HelperComponents and Microsoft.BizTalk.Samples.EDIGuidance.PipelineComponents are in the GAC

- Setup BAM
  To setup BAM you need to extend the EDI tracking profile to include a "picked up" flag.

  First navigate to C:\projects\EDI Guidance\Main\Src\Bam. This contains 4 files. BAMDefFile.xml is the new Observation Model, the other 4 are SQL Queries that help you clear and query the BAM data. This is useful when you are testing.

  Open a command prompt and navigate to C:\Program Files\Microsoft BizTalk Server 2009 \Tracking

  Type bm.exe update-all -DefinitionFile:"C:\projects\EDI Guidance\Main\Src\BAM\BamDefFile.xml"



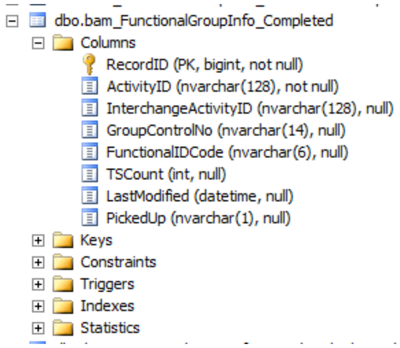
```
c:\Program Files\Microsoft BizTalk Server 2009\Tracking>bm update-all -definitio
nfile:"C:\Projects\EDI Guidance\Main\Src\BAM\BAMDefFile.xml"
Microsoft (R) Business Activity Monitoring Utility Version 3.8.368.0
Copyright (C) 2006 Microsoft Corporation.  All rights reserved.

Using 'BAMPrimaryImport' BAM Primary Import database on server 'WIN-WUIAML4C160'
...

Updating Activity... Done.

c:\Program Files\Microsoft BizTalk Server 2009\Tracking>
```

  To verify that the observation model was updated, open SQL Server management Studio and expand the BAMPrimaryImport database. We are interested in the dbo.bam_FunctionalGroupInfo_Completed table. You could see a PickedUp field.

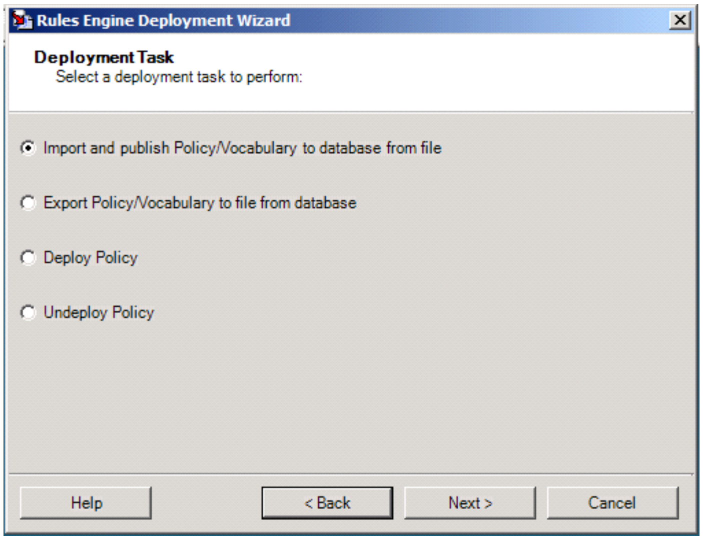

dbo.bam_FunctionalGroupInfo_Completed
Columns
RecordID (PK, bigint, not null)
ActivityID (nvarchar(128), not null)
InterchangeActivityID (nvarchar(128), null)
GroupControlNo (nvarchar(14), null)
FunctionalIDCode (nvarchar(6), null)
TSCount (int, null)
LastModified (datetime, null)
PickedUp (nvarchar(1), null)
Keys
Constraints
Triggers
Indexes
Statistics

- Install the Stored Procedure.
  With SQL Server Management Studio, open the SQL Script Create_EDI_Tools_GetNewInterchanges_SP.sql found at C:\projects\EDI Guidance\Main\src \Database

  Execute the script and then verify that the stored procedure dbo.edi_Tools_GetNewInterchanges was created in the BAMPrimaryImport database.

Deploy the Application

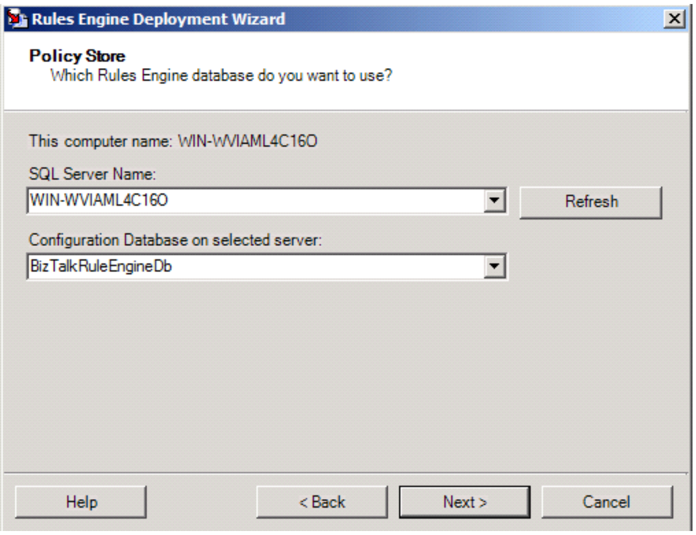
- Deploy the Rules
  Open the Rules Engine Deployment Wizard, click next, choose the import and publish Policy/Vocabulary to database from file, and then click next.



Deployment Task
Select a deployment task to perform:

● Import and publish Policy/Vocabulary to database from file
○ Export Policy/Vocabulary to file from database
○ Deploy Policy
○ Undeploy Policy

Make sure you are connected to a policy store and click next.



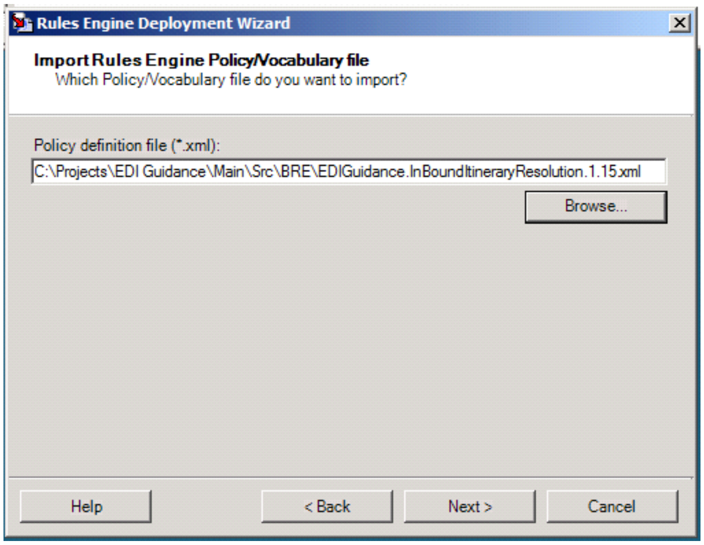
Policy Store
Which Rules Engine database do you want to use?

This computer name: WIN-WVIAML4C16O

SQL Server Name: WIN-WVIAML4C16O

Configuration Database on selected server: BizTalkRuleEngineDb

Select the EDIGuidance.InBoundItineraryResolution.1.15.xml file located at c:\projects\EDI Guidance\Main\Src\BRE\ and click next



Import Rules Engine Policy/Vocabulary file
Which Policy/Vocabulary file do you want to import?

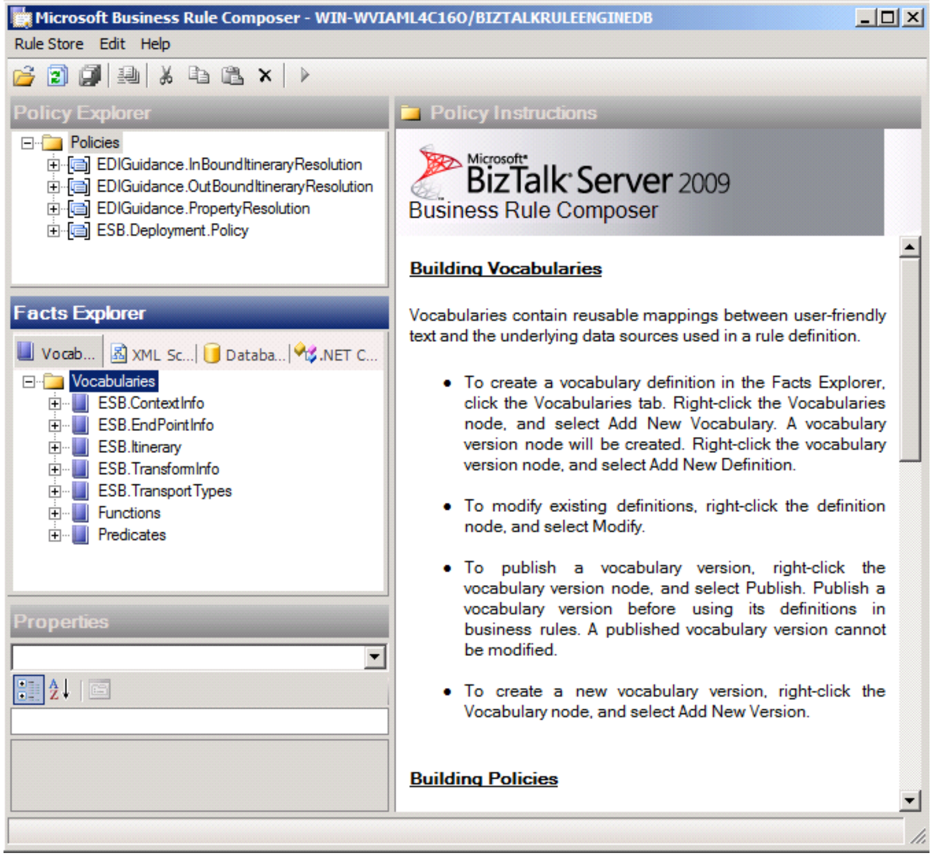
Policy definition file (*.xml): C:\Projects\EDI Guidance\Main\Src\BRE\EDIGuidance.InBoundItineraryResolution.1.15.xml

You will see a summary page, click next and the policy will be imported. Click Next and then finish.

Repeat this process for the following
EDIGuidance.OutBoundItineraryResolution.1.3.xml

EDIGuidance.PropertyResolution.1.2.xml
found in the same directory

You can verify that the policies have been deployed using the Business Rules Composer.



Building Vocabularies

Vocabularies contain reusable mappings between user-friendly text and the underlying data sources used in a rule definition.

- To create a vocabulary definition in the Facts Explorer, click the Vocabularies tab. Right-click the Vocabularies node, and select Add New Vocabulary. A vocabulary version node will be created. Right-click the vocabulary version node, and select Add New Definition.
- To modify existing definitions, right-click the definition node, and select Modify.
- To publish a vocabulary version, right-click the vocabulary version node, and select Publish. Publish a vocabulary version before using its definitions in business rules. A published vocabulary version cannot be modified.
- To create a new vocabulary version, right-click the Vocabulary node, and select Add New Version.

Building Policies

Expand the policies and deploy the rules.

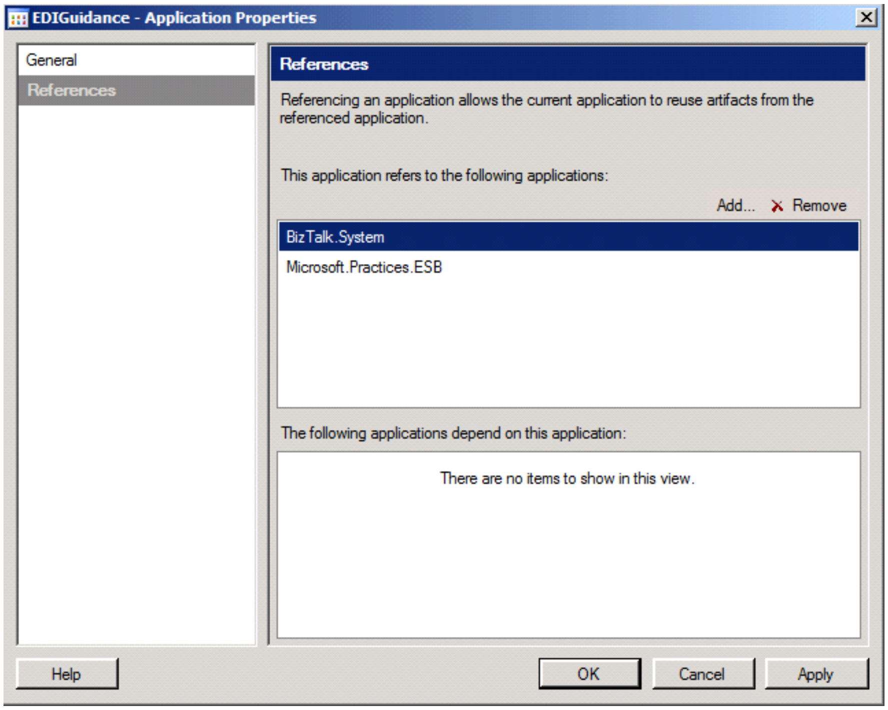
- **Deploy the BizTalk Application**
  Open Visual studio and load the Microsoft.BizTalk.Samples.EDIGuidance solution. In the Solution Explorer window, right click on the solution and select deploy.

  Once the application has finished deploying, open the BizTalk server administration console and verify that two applications have been deployed and that the applicable assemblies are in place. If you already have the administration console open, refresh the view.

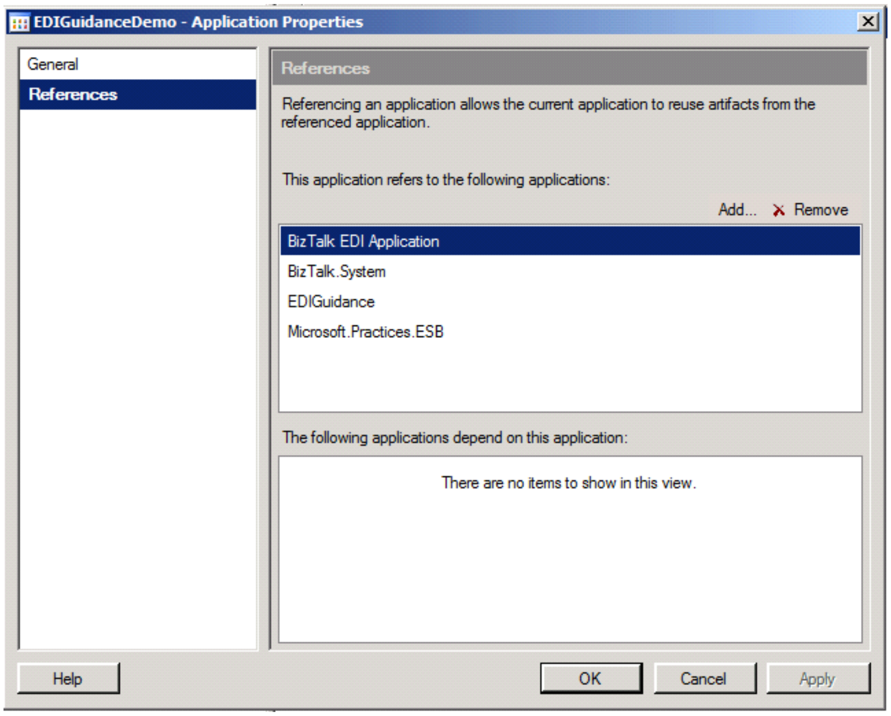
  You should see an application named EDIGuidance and an application named EDIGuidanceDemo.

- **Import the Bindings**

  Right Click the EDIGuidance application and select properties. Click on the references option and add Microsoft.Practices.ESB as a reference and click OK.


EDIGuidance - Application Properties
General
References
References
Referencing an application allows the current application to reuse artifacts from the referenced application.
This application refers to the following applications:
Add...
Remove
BizTalk.System
Microsoft.Practices.ESB
The following applications depend on this application:
There are no items to show in this view.
Help
OK
Cancel
Apply

Repeat this step for the EDIGuidanceDemo application and add the EDIGuidance application and the BizTalk EDI Application as well as the microsoft.practices.ESB application.


EDIGuidanceDemo - Application Properties
General
References
References
Referencing an application allows the current application to reuse artifacts from the referenced application.
This application refers to the following applications:
Add...
Remove
BizTalk EDI Application
BizTalk.System
EDIGuidance
Microsoft.Practices.ESB
The following applications depend on this application:
There are no items to show in this view.
Help
OK
Cancel
Apply

Right Click the EDIGuidance application and select Import Bindings. Select the EDIGuidance.Bindings.XML file found in c:\projects\EDI Guidance\Main\Src\Bindings.

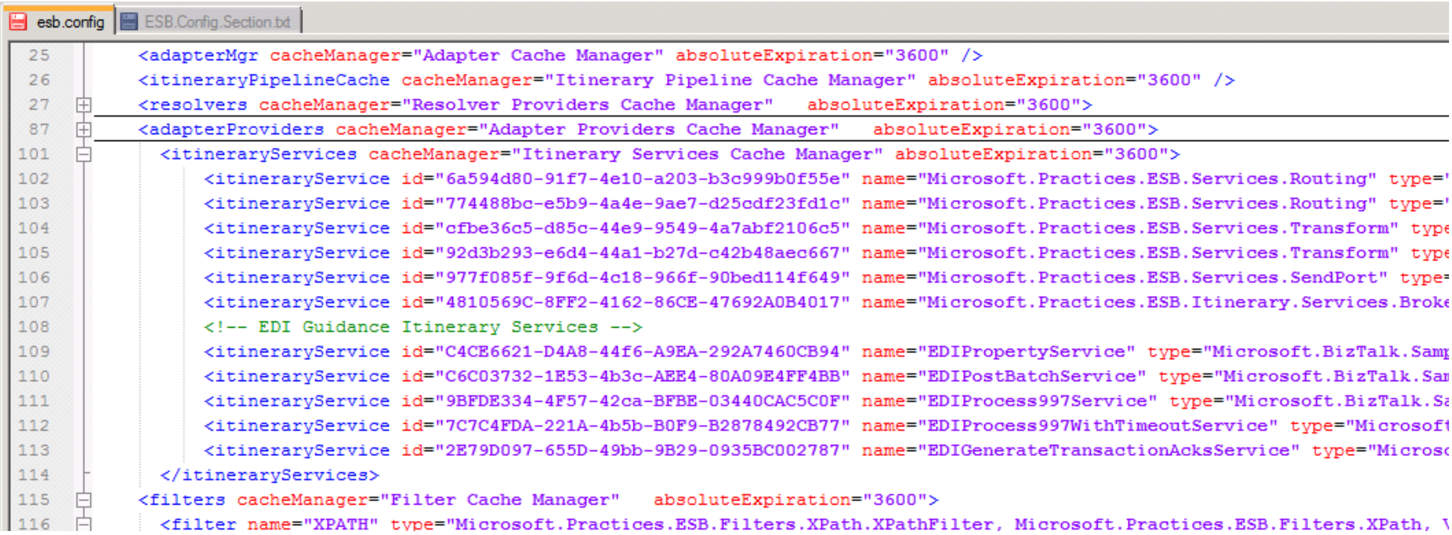
Right Click the EDIGuidanceDemo and select import Bindings. Select the EDIGuidance.DemoArtifacts.Bindings.XML file found in c:\projects\EDI Guidance\Main\Src \Bindings.

- **Install Custom Itinerary Services**
  We have a few custom orchestration services that are being used to drive the scenarios. You need to register the custom services so they can be used from the itinerary designer.

  To start, open the ESB.Config file located in C:\program files\Microsoft BizTalk ESB Toolkit 2.0 in your favorite text editor.

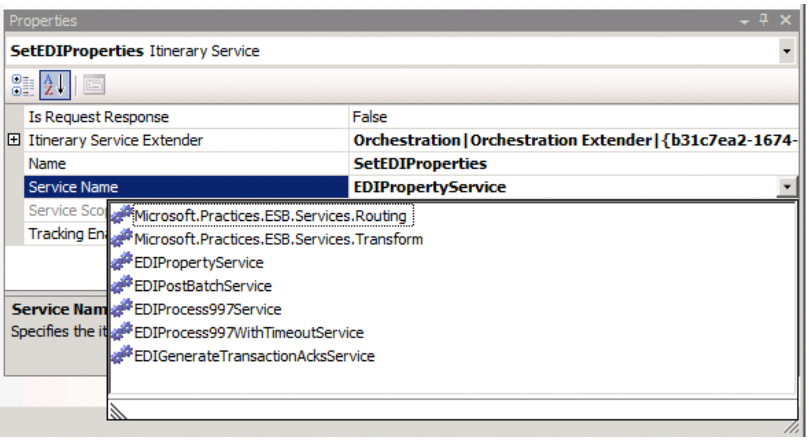
  In the <itineraryServices> section you need to add several entries. The entries can be found in

ESB.Config.Section.txt located at C:\Projects\EDI Guidance\Main\Src\Install. When you are finished, your config file should look similar to the following;



```
    <adapterMgr cacheManager="Adapter Cache Manager" absoluteExpiration="3600" />
    <itineraryPipelineCache cacheManager="Itinerary Pipeline Cache Manager" absoluteExpiration="3600" />
    <resolvers cacheManager="Resolver Providers Cache Manager"   absoluteExpiration="3600">
    <adapterProviders cacheManager="Adapter Providers Cache Manager"   absoluteExpiration="3600">
      <itineraryServices cacheManager="Itinerary Services Cache Manager" absoluteExpiration="3600">
        <itineraryService id="6a594d80-91f7-4e10-a203-b3c999b0f55e" name="Microsoft.Practices.ESB.Services.Routing" type='
        <itineraryService id="774488bc-e5b9-4a4e-9ae7-d25cdf23fd1c" name="Microsoft.Practices.ESB.Services.Routing" type='
        <itineraryService id="cfbe36c5-d85c-44e9-9549-4a7abf2106c5" name="Microsoft.Practices.ESB.Services.Transform" type
        <itineraryService id="92d3b293-e6d4-44a1-b27d-c42b48aec667" name="Microsoft.Practices.ESB.Services.Transform" type
        <itineraryService id="977f085f-9f6d-4c18-966f-90bed114f649" name="Microsoft.Practices.ESB.Services.SendPort" type=
        <itineraryService id="4810569C-8FF2-4162-86CE-47692A0B4017" name="Microsoft.Practices.ESB.Itinerary.Services.Broke
        <!-- EDI Guidance Itinerary Services -->
        <itineraryService id="C4CE6621-D4A8-44f6-A9EA-292A7460CB94" name="EDIPropertyService" type="Microsoft.BizTalk.Samp
        <itineraryService id="C6C03732-1E53-4b3c-AEE4-80A09E4FF4BB" name="EDIPostBatchService" type="Microsoft.BizTalk.Sa
        <itineraryService id="9BFDE334-4F57-42ca-BFBE-03440CAC5C0F" name="EDIProcess997Service" type="Microsoft.BizTalk.S
        <itineraryService id="7C7C4FDA-221A-4b5b-B0F9-B2878492CB77" name="EDIProcess997WithTimeoutService" type="Microsof
        <itineraryService id="2E79D097-655D-49bb-9B29-0935BC002787" name="EDIGenerateTransactionAcksService" type="Micros
      </itineraryServices>
    <filters cacheManager="Filter Cache Manager"   absoluteExpiration="3600">
      <filter name="XPATH" type="Microsoft.Practices.ESB.Filters.XPath.XPathFilter, Microsoft.Practices.ESB.Filters.XPath,
```

If you have the itinerary designer open, you will need to close and reopen the itinerary for the changes to take affect. To verify the changes, add an orchestration extender and drop down the service name list. You should see the 5 Itinerary Services we added to the config file.



SetEDIProperties Itinerary Service

Is Request Response: False
Itinerary Service Extender: **Orchestration|Orchestration Extender|{b31c7ea2-1674-**
Name: **SetEDIProperties**
Service Name: **EDIPropertyService**
Service Sco
Tracking En

- Microsoft.Practices.ESB.Services.Routing
- Microsoft.Practices.ESB.Services.Transform
- EDIPropertyService
- EDIPostBatchService
- EDIProcess997Service
- EDIProcess997WithTimeoutService
- EDIGenerateTransactionAcksService

Service Nam
Specifies the it

NOTE: the itineraries are not encrypted with a certificate. To disable the validation errors you will receive if you do not associate a certificate you need to update the following registry entry; HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\BizTalk ESB Toolkit\2.0\Designer and set the RequireX509Certificate property to false. If you are running a 64bit OS the key is HKEY_LOCAL_MACHINE\SOFTWARE\SysWOW64\Microsoft\BizTalk ESB Toolkit\2.0\Designer.

Open the project Microsoft.BizTalk.Samples.EDIGuidance.Itineraries. Open each itinerary, right click the design surface and export the itinerary to the itinerary DB.

- Add registry entries necessary for the BRE to support Static components
  Double click and run C:\Projects\EDI Guidance\Main\Src\Install\BRE_StaticSupport.reg

Bugs

- The off ramp information stored in the rules is incorrect. The rules have the path to the test data as "c:\projects\EDIGuidance\Main\src\TestData" while the actual path is "c:\projects\EDI Guidance\Main\src\TestData" You will need update the static resolver within the outbound itineraries.

- The on ramp information for rcv Terminate and LOBOnRamp are incorrect. Change the path from "c:\projects\EDIGuidance\Main\src\TestData" to" c:\projects\EDI Guidance\Main\src\TestData"

- EDIItinerarySelectReceiveXML pipeline is not configured for EDIOnRamp_THEM2_File. Select the pipeline set the ItineraryFactKey to Resolver.Itinerary and the ResolverConnectionstring to BRI:\\policy=EDIGuidance.InBoundItineraryResolution;useMsg=true;recognizeMessageFormat=true;

- In the EDIGuidanceDemo application, the LOBOnRamp_File receive location file mask is incorrect. It needs to be changed to *LOB*copy.xml

Demonstration Steps

Before starting, make sure that the ESB Toolkit application and the EDIGuidance applications are started and running. Make sure that batch processing has been started for both trading parties. Lastly, make sure that all the EDI Guidance BRE Policies have been deployed.

As with most BizTalk demos, this isn't very exciting, you literally see a file dropped in one folder, the file disappears, and a new file appears with a GUID in the file name. For this reason, it is recommended that you use SysInternals DebugView to help your audience understand what is occurring within the system (see Configure Diagnostics and Trace in the Troubleshooting and Tips section below)

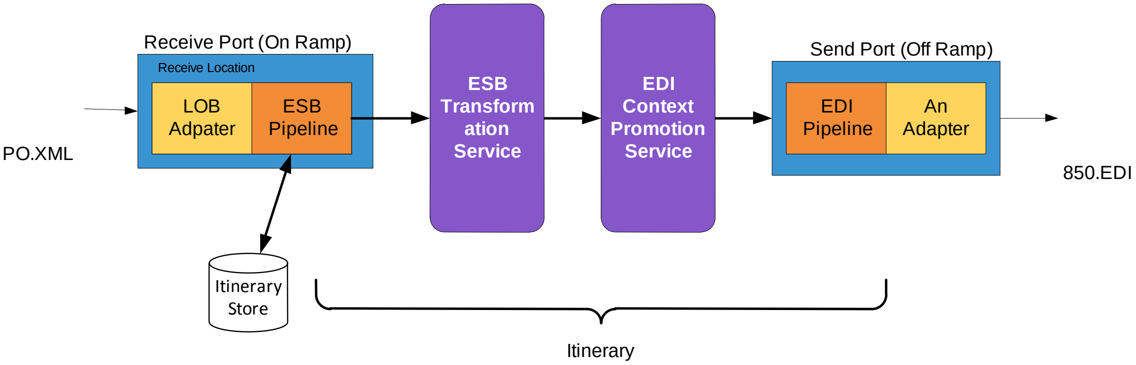
The receive locations are configured to pickup messages based on a particular file mask. This allows us to leverage one folder for all the demos as opposed to having multiple windows open and constantly dragging and dropping files.

Outbound EDI

In this scenario we will pickup a LOB PO message and deliver the message to a trading partner. The routing and trading party destination will be defined within an itinerary. The outbound itineraries are selected based on rules. Essentially we select the itineraries based on the content of the messages. The rules are located in the EDIGuidance.OutBoundItineraryResolution policy.

To keep things simple, we are using Static Resolvers for our Destination Addresses and our Maps. We will also demonstrate how you can handle message differences between two trading parties without having to create two full fidelity PO to 850 maps.

Overview



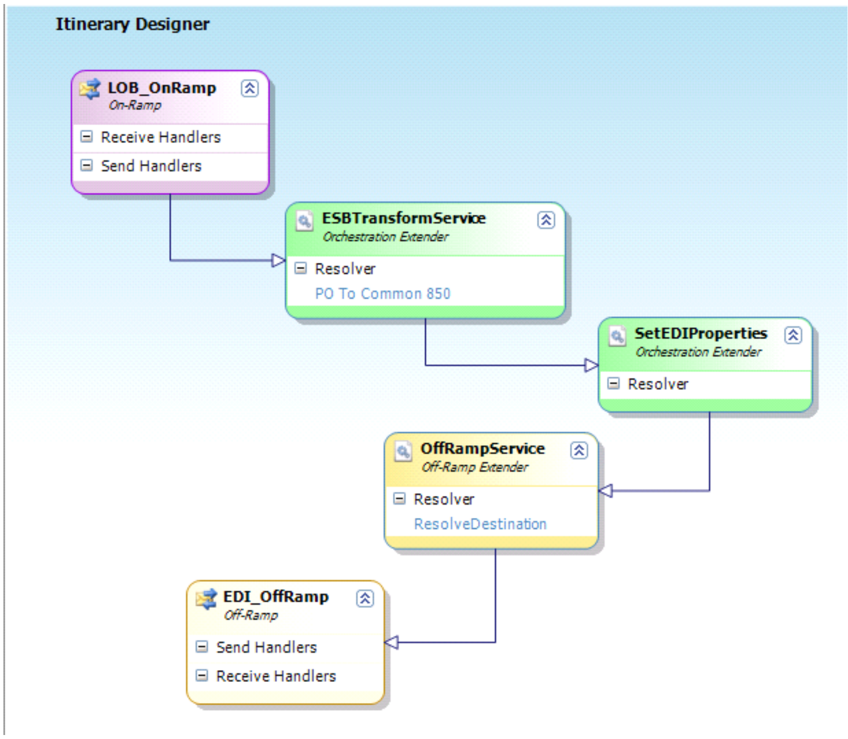
When we receive the PO, we examine the message with an ESB pipeline. The pipeline will be used to resolve which itinerary should be attached to the message and then attaches the message prior to publishing the message.

The first step in the itinerary is to transform the message from the ERP PO Format to an EDI 850 format. This is done with a transformation service. The map is resolved with the static resolver, i.e. hardcoded into the itinerary. The service also advances the itinerary to the next step.

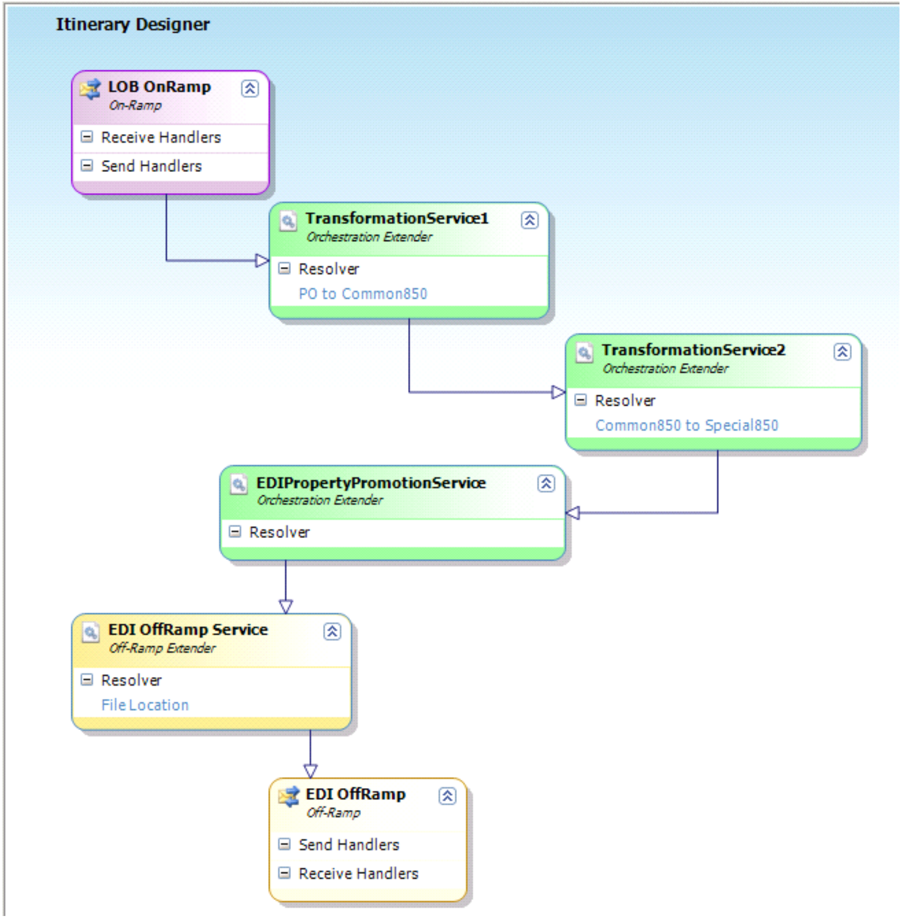
Next, the message makes its way to the EDI Context promotion itinerary service. This is a custom service which is part of the EDIGuidance application. This service provides a way to lookup information within the EDI document, determine which trading party the document needs to be sent to, set the appropriate EDI context properties, and advance the itinerary to the next step.

Lastly, the message is routed to the Dynamic send port where the XML representation of EDI is de normalized into actual EDI. The destination URI for the trading party is determined by the Static Resolver, i.e. hardcoded into the itinerary.

This is how the itinerary looks within Visual Studio



There are times when a new trading partner will need to be added but that trading partner may have slightly different message requirements. Since we are using the ESB Toolkit, this is fairly easy to implement. Just build a second map which articulates the differences between the current specification and the partner specific specification, create a new itinerary, and add a new rule to assign the itinerary. Sounds a little complex, but it is much simpler than building a full fidelity PO to Partner Specific Map. Here is the new itinerary.



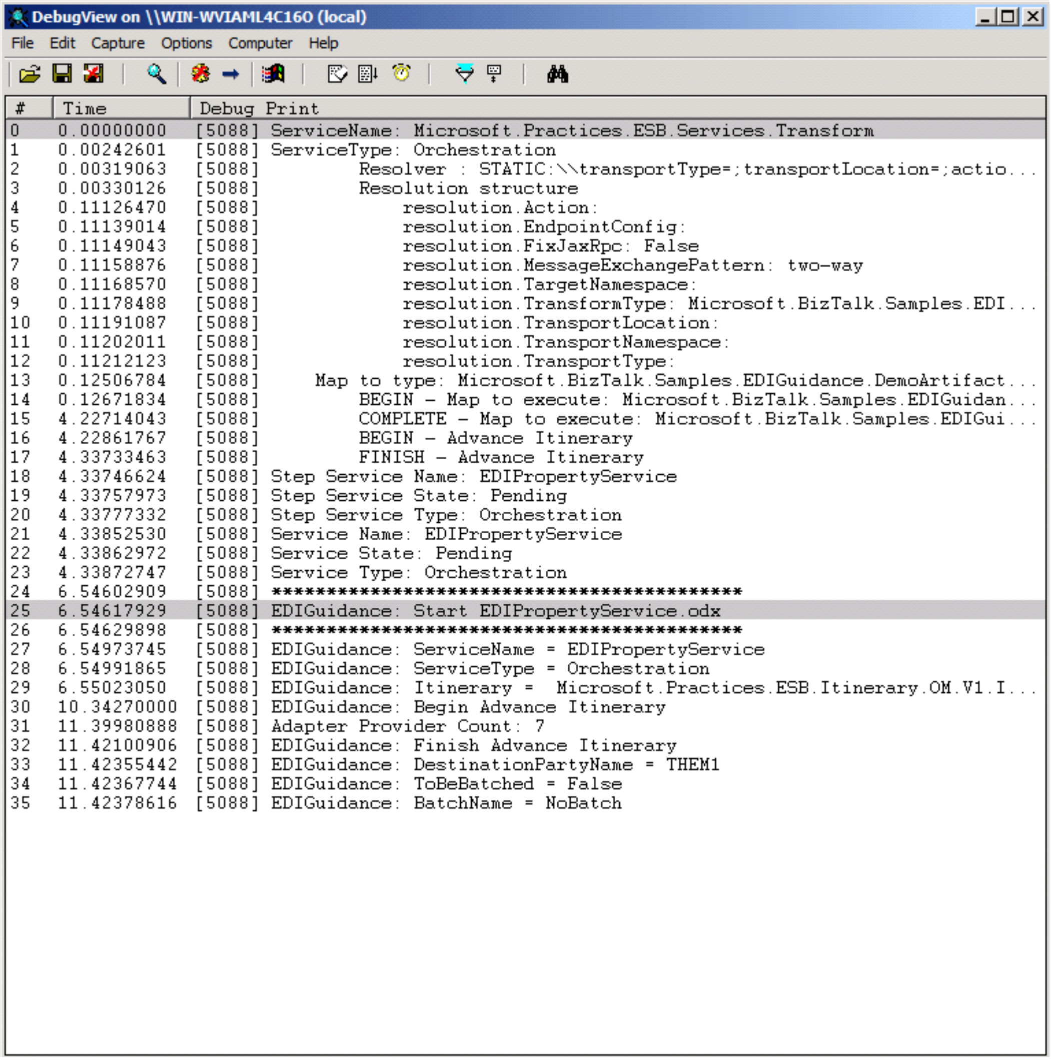
Not that we are transforming the message twice. The first transformation is the same transformation we used earlier, the second transformation is where we map the standard EDI message to the partner specific EDI message.

Step 1 - Open the TestData Folder
The test data is located at C:\Projects\EDI Guidance\Main\Src\TestData

Step 2 - Send a sample message
In the folder we are interested in THEM1_LOB_PO_1.xml. Simply copy and past the file into the same folder. You will end up with THEM1_LOB_PO_1 - Copy.xml. The file mask on the receive location is *LOB*Copy.xml so the file will be picked up and processed.

Step 3 - Receive an EDI Message
After a few moments (it may take a little while on the first run) an edi file 850-{GUID}.txt is created in the same folder.

Step 4 - Review Debug Messages (Optional)
If you are using SysInternals DebugView you can review the debug statements that are written as the PO is transformed into an EDI message. This will help visualize what is happening after your PO disappears and before your EDI message appears. I would highly recommend you use this tool when demonstrating to a technical audience, business audiences probably won't care how it happened, just that it was transformed to EDI. Below is a sample trace with highlights.



```
#   Time        Debug Print
0   0.00000000  [5088] ServiceName: Microsoft.Practices.ESB.Services.Transform
1   0.00242601  [5088] ServiceType: Orchestration
2   0.00319063  [5088]         Resolver : STATIC:\\transportType=;transportLocation=;actio...
3   0.00330126  [5088]         Resolution structure
4   0.11126470  [5088]             resolution.Action:
5   0.11139014  [5088]             resolution.EndpointConfig:
6   0.11149043  [5088]             resolution.FixJaxRpc: False
7   0.11158876  [5088]             resolution.MessageExchangePattern: two-way
8   0.11168570  [5088]             resolution.TargetNamespace:
9   0.11178488  [5088]             resolution.TransformType: Microsoft.BizTalk.Samples.EDI...
10  0.11191087  [5088]             resolution.TransportLocation:
11  0.11202011  [5088]             resolution.TransportNamespace:
12  0.11212123  [5088]             resolution.TransportType:
13  0.12506784  [5088]     Map to type: Microsoft.BizTalk.Samples.EDIGuidance.DemoArtifact...
14  0.12671834  [5088]         BEGIN - Map to execute: Microsoft.BizTalk.Samples.EDIGuidan...
15  4.22714043  [5088]         COMPLETE - Map to execute: Microsoft.BizTalk.Samples.EDIGui...
16  4.22861767  [5088]         BEGIN - Advance Itinerary
17  4.33733463  [5088]         FINISH - Advance Itinerary
18  4.33746624  [5088] Step Service Name: EDIPropertyService
19  4.33757973  [5088] Step Service State: Pending
20  4.33777332  [5088] Step Service Type: Orchestration
21  4.33852530  [5088] Service Name: EDIPropertyService
22  4.33862972  [5088] Service State: Pending
23  4.33872747  [5088] Service Type: Orchestration
24  6.54602909  [5088] *******************************************
25  6.54617929  [5088] EDIGuidance: Start EDIPropertyService.odx
26  6.54629898  [5088] *******************************************
27  6.54973745  [5088] EDIGuidance: ServiceName = EDIPropertyService
28  6.54991865  [5088] EDIGuidance: ServiceType = Orchestration
29  6.55023050  [5088] EDIGuidance: Itinerary =  Microsoft.Practices.ESB.Itinerary.OM.V1.I...
30  10.34270000 [5088] EDIGuidance: Begin Advance Itinerary
31  11.39980888 [5088] Adapter Provider Count: 7
32  11.42100906 [5088] EDIGuidance: Finish Advance Itinerary
33  11.42355442 [5088] EDIGuidance: DestinationPartyName = THEM1
34  11.42367744 [5088] EDIGuidance: ToBeBatched = False
35  11.42378616 [5088] EDIGuidance: BatchName = NoBatch
```

Step 5 - Testing partner specific EDI
Copy and past THEM2_LOB_PO_1.xml into the TestData directory. The receive location is using a file mask to pickup the copied PO. After a few moments, the file will disappear.

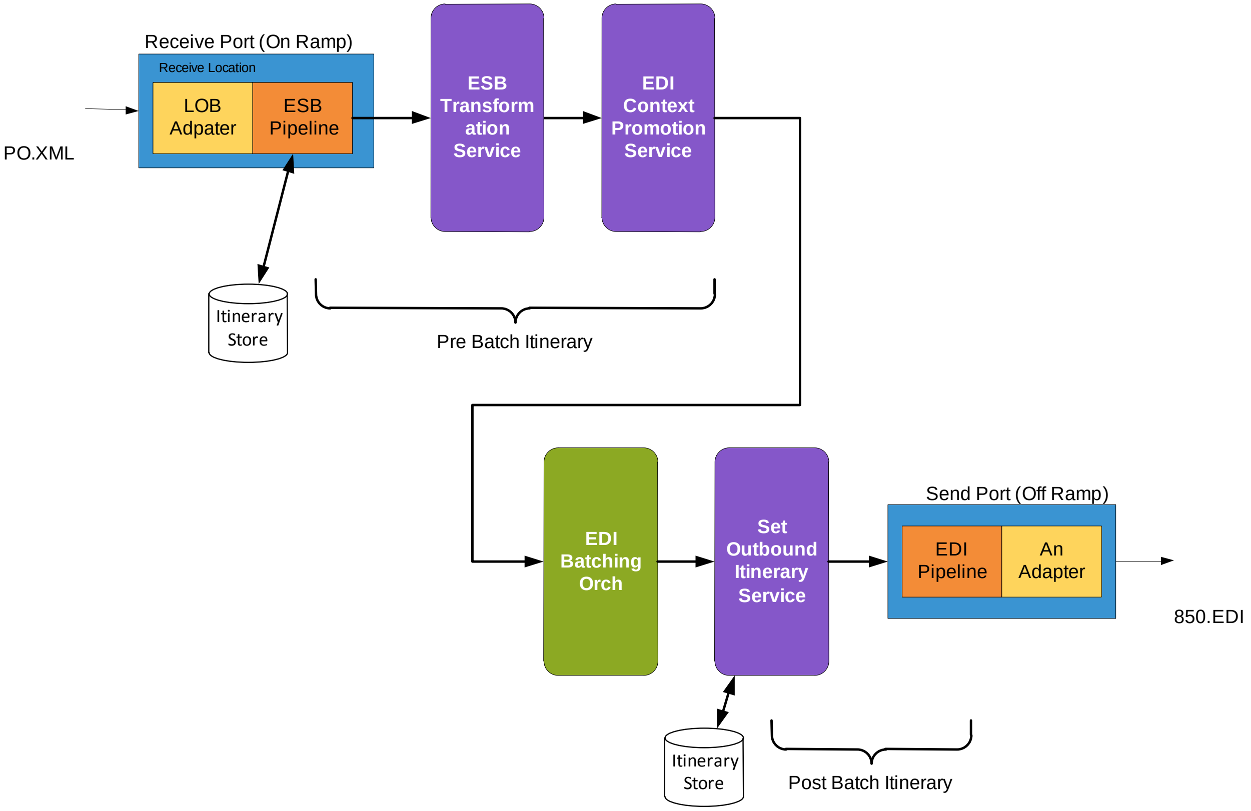
Step 6 - Receiving the EDI message
After a few moments, you will see the file special850 - {GUID}.txt created in the same folder.

Step 7 - Review the debug information (optional)
You can view the debug information shown in DebugView and verify that the transformation service was called twice and which maps were called.

Outbound Batched EDI
This scenario is the same as the Outbound EDI scenario except that we will batch the outbound EDI messages using the EDI Batching functionality. The batch is configured to release when the maximum number of transaction sets in the interchange is 3, or three PO messages. Note that an EDI batch can

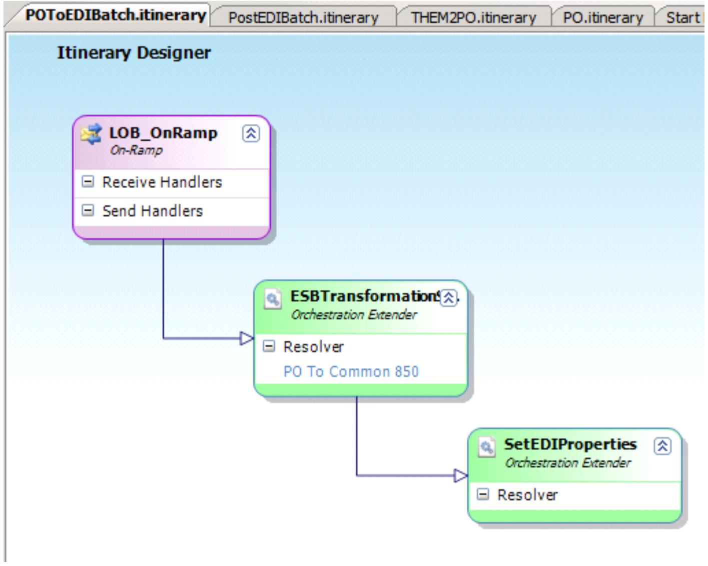
contain multiple different types of transactions, for this demo we are just sending PO messages.

Overview



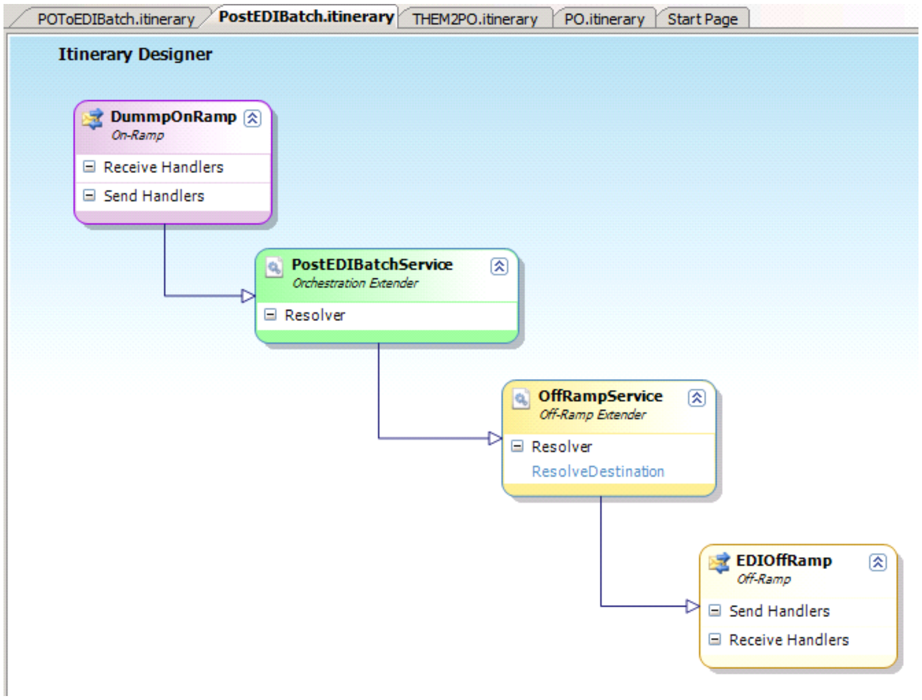
This process is very different from the Outbound EDI scenario due to our requiring the EDI messages to be batched. As soon as we require Batching, we have to interoperate with the EDI Batch orchestration, which is not ESB aware. Also, since an EDI Batch may contain many different types of EDI messages and may have different release criteria, we have to have two itineraries. The first itinerary gets the PO converted to EDI and then sends the PO to the batch orchestration. The second itinerary picks up the released batch and sends it to the trading party.

If we take a look at POToEDIBatch.itinerary we can see how the first itinerary was created



Fairly simple. Pickup the message, transform it to the XML representation of EDI, then resolve the EDI properties and the itinerary is complete. In this case, the SetEDIProperties service will set the ToBeBatched context property and the BatchName context property via a BRE Policy. This will route the message to the appropriate EDI Batch.

The itinerary that will be assigned to a batch message is the PostEDIBatch.Itinerary



The itinerary is assigned to the message via the PostEDIBatchService. This service is an orchestration which will pickup a batch message from the EDI Batch Orchestration and assign an itinerary. Note, the DummyOnRamp shape, this is not needed except to pass itinerary validation.

<u>Step 1 - Open the TestData Folder</u>

The test data is located at C:\Projects\EDI Guidance\Main\Src\TestData

<u>Step 2 - Send the sample messages</u>

In the folder we are interested in 3 files;

THEM1_LOB_PO_ForBatch_1.xml
THEM1_LOB_PO_ForBatch_2.xml
THEM1_LOB_PO_ForBatch_2.xml

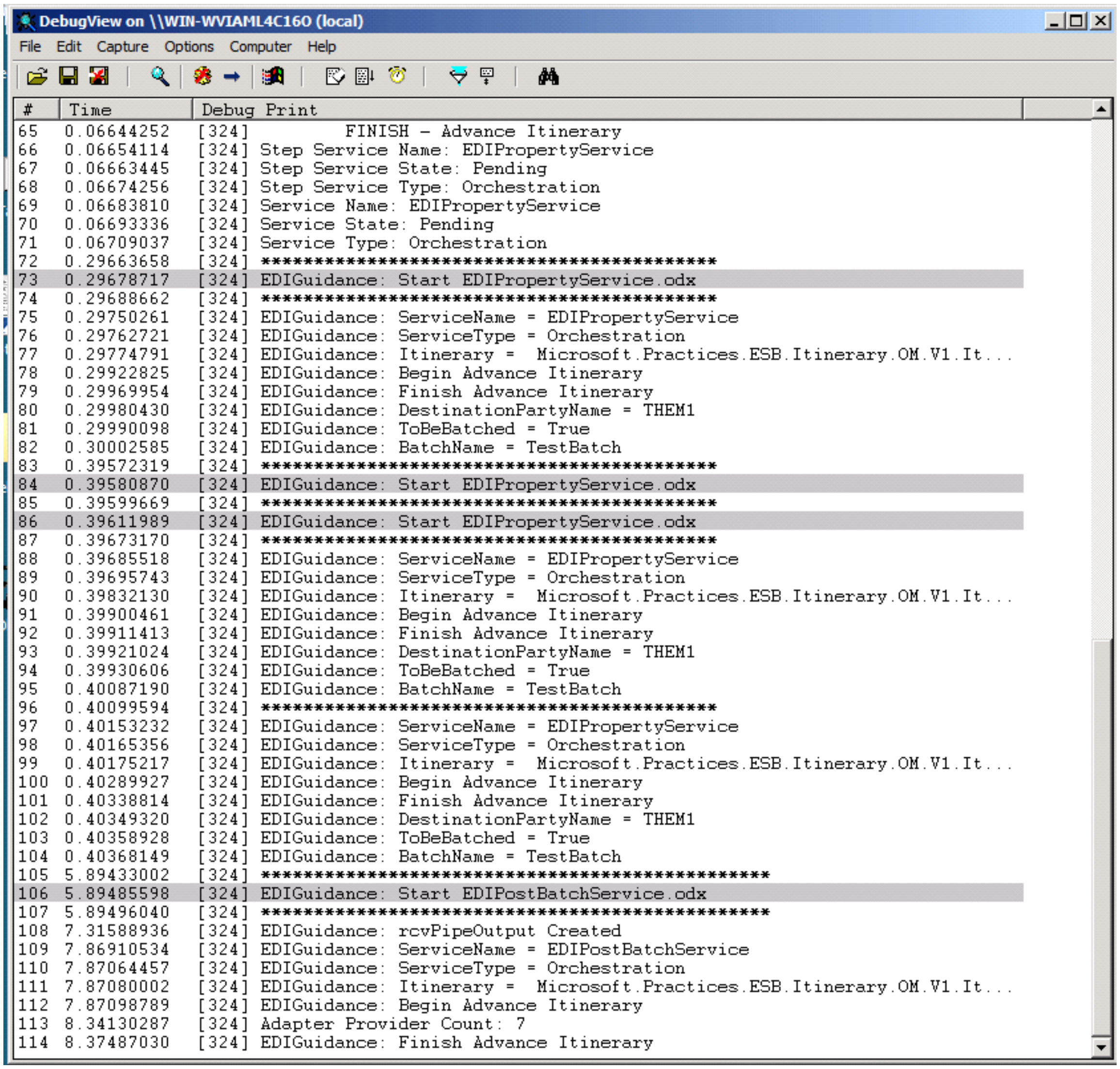
Simply copy and past these files into the same directory. The receive port will pickup the copied files based on a file mask.

<u>Step 3 - Receive an EDI Message</u>

After a few moments (it may take a little while on the first run) an edi file EDIBatch-{GUID}.txt is created in the same folder. **Note: If you don't see the batch message, make sure that EDI Batch orchestration has been started.**

<u>Step 4 - Review Debug Messages (Optional)</u>

If you are using SysInternals DebugView you can review the debug statements that are written as the PO s are transformed into an EDI message, sent to the batching orch, and then sent to the trading party. This will help visualize what is happening after your PO disappears and before your EDI message appears. I would highly recommend you use this tool when demonstrating to a technical audience, business audiences probably won't care how it happened, just that it was transformed to EDI. Below is a sample trace with highlights.


DebugView on \\WIN-WVIAML4C160 (local)
File Edit Capture Options Computer Help
#
Time
Debug Print

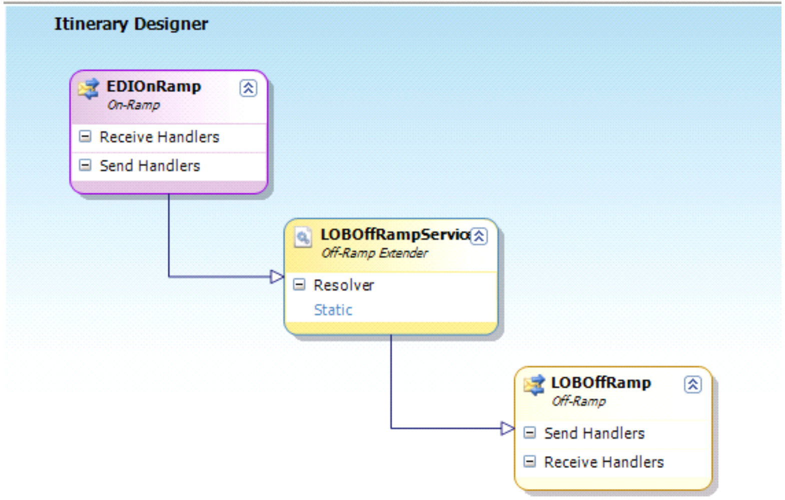
```
65  0.06644252  [324]         FINISH - Advance Itinerary
66  0.06654114  [324] Step Service Name: EDIPropertyService
67  0.06663445  [324] Step Service State: Pending
68  0.06674256  [324] Step Service Type: Orchestration
69  0.06683810  [324] Service Name: EDIPropertyService
70  0.06693336  [324] Service State: Pending
71  0.06709037  [324] Service Type: Orchestration
72  0.29663658  [324] *******************************************
73  0.29678717  [324] EDIGuidance: Start EDIPropertyService.odx
74  0.29688662  [324] *******************************************
75  0.29750261  [324] EDIGuidance: ServiceName = EDIPropertyService
76  0.29762721  [324] EDIGuidance: ServiceType = Orchestration
77  0.29774791  [324] EDIGuidance: Itinerary =  Microsoft.Practices.ESB.Itinerary.OM.V1.It...
78  0.29922825  [324] EDIGuidance: Begin Advance Itinerary
79  0.29969954  [324] EDIGuidance: Finish Advance Itinerary
80  0.29980430  [324] EDIGuidance: DestinationPartyName = THEM1
81  0.29990098  [324] EDIGuidance: ToBeBatched = True
82  0.30002585  [324] EDIGuidance: BatchName = TestBatch
83  0.39572319  [324] *******************************************
84  0.39580870  [324] EDIGuidance: Start EDIPropertyService.odx
85  0.39599669  [324] *******************************************
86  0.39611989  [324] EDIGuidance: Start EDIPropertyService.odx
87  0.39673170  [324] *******************************************
88  0.39685518  [324] EDIGuidance: ServiceName = EDIPropertyService
89  0.39695743  [324] EDIGuidance: ServiceType = Orchestration
90  0.39832130  [324] EDIGuidance: Itinerary =  Microsoft.Practices.ESB.Itinerary.OM.V1.It...
91  0.39900461  [324] EDIGuidance: Begin Advance Itinerary
92  0.39911413  [324] EDIGuidance: Finish Advance Itinerary
93  0.39921024  [324] EDIGuidance: DestinationPartyName = THEM1
94  0.39930606  [324] EDIGuidance: ToBeBatched = True
95  0.40087190  [324] EDIGuidance: BatchName = TestBatch
96  0.40099594  [324] *******************************************
97  0.40153232  [324] EDIGuidance: ServiceName = EDIPropertyService
98  0.40165356  [324] EDIGuidance: ServiceType = Orchestration
99  0.40175217  [324] EDIGuidance: Itinerary =  Microsoft.Practices.ESB.Itinerary.OM.V1.It...
100 0.40289927  [324] EDIGuidance: Begin Advance Itinerary
101 0.40338814  [324] EDIGuidance: Finish Advance Itinerary
102 0.40349320  [324] EDIGuidance: DestinationPartyName = THEM1
103 0.40358928  [324] EDIGuidance: ToBeBatched = True
104 0.40368149  [324] EDIGuidance: BatchName = TestBatch
105 5.89433002  [324] ************************************************
106 5.89485598  [324] EDIGuidance: Start EDIPostBatchService.odx
107 5.89496040  [324] ************************************************
108 7.31588936  [324] EDIGuidance: rcvPipeOutput Created
109 7.86910534  [324] EDIGuidance: ServiceName = EDIPostBatchService
110 7.87064457  [324] EDIGuidance: ServiceType = Orchestration
111 7.87080002  [324] EDIGuidance: Itinerary =  Microsoft.Practices.ESB.Itinerary.OM.V1.It...
112 7.87098789  [324] EDIGuidance: Begin Advance Itinerary
113 8.34130287  [324] Adapter Provider Count: 7
114 8.37487030  [324] EDIGuidance: Finish Advance Itinerary
```

Inbound EDI

The inbound scenario is rather simple. For the most part we are just passing in an EDI message, validating it, and then passing the XML representation of the message to a folder. For the most part, this is a setup for the error handling demo. This shows that we can correctly receive an EDI message.

Overview

In this scenario we are setup to receive an 855 message. Once we receive the message we assign an itinerary by calling a set of rules. The itinerary is very simple, we are not transforming the message, just passing the message through the system. The itinerary is pictured below.



Note: while we are not transforming the message, if you want to add a PO Ack schema and create a map, it would be trivial to modify the itinerary to use the new map/schema.

Step 1 - Open the TestData Folder
The test data is located at C:\Projects\EDI Guidance\Main\Src\TestData

Step 2 - Send the sample messages
Copy and past the THEM1_ISA855.txt file into the TestData folder. In a few moments the folder will disappear.

Step 3 - Receive an EDI Message
After a few moments,

Step 4 - Review Debug Messages (Optional)
If you are using SysInternals DebugView you can review the debug statements that are written as the PO s are transformed into an EDI message, sent to the batching orch, and then sent to the trading party. This will help visualize what is happening after your PO disappears and before your EDI message appears. I would highly recommend you use this tool when demonstrating to a technical audience, business audiences probably won't care how it happened, just that it was transformed to EDI. Below is a sample trace with highlights.

Inbound Batched EDI

997 Processing

Inbound Fix and Resubmit

Troubleshooting and Tips

- Configure Diagnostics and Trace
  The EDIGuidance application is instrumented with debug statements. If you download and use DebugView by Windows SysInternals, you can see these debug statements as messages are processed. This is useful since a file disappearing and reappearing doesn't give the functionality justice. You can download the tool at http://technet.microsoft.com/en-us/sysinternals/bb896647.aspx. Make sure you have Capture Global Win32 events selected under the capture menu. One more tip, if you use the highlight/filter option under the edit menu and use the string 'ServiceName: Microsoft.Practices.ESB;EDIGuidance: Start' you will highlight key processing events which will make it simpler to explain what is happening to an audience.