# BTSHOL01: Building your first BizTalk Server solution

Objectives

After completing this lab, you will be able to:

* Create a BizTalk project in Visual Studio 2015
* Configure a BizTalk project for deployment
* Deploy a BizTalk solution and configure it
* Create ports to receive and send messages

You work for Contoso Winery and you are about to begin development on a project utilizing BizTalk Server 2016. In order to get familiar with the development tasks involved, you will build a simple solution, deploy it and send messages through the system.

In this lab you will get a holistic view of much of BizTalk Servers functionality. You will be using many of the concepts and artifacts that later labs will go into more detail on. For this lab much of those artifacts will be pre-prepared for you and your tasks will focus on connecting the dots to make up a while.

The end result is that XML files should be picked up by BizTalk, processed by an orchestration and transformed to a flat-file structure, and sent out to a different folder, being serialized to its flat-file representation along the way by the send pipeline. You will build the project, deploy it, configure, start and test it.

Total estimated time to complete this lab:   
75 minutes  
  
Lab 1a: Exercise 1: 15 minutes  
Lab 1b: Exercise 2-4: 35 minutes  
Lab 1c: Exercise 5-6: 25 minutes

User Name: **Administrator**

Password: **pass@word1**

Lab 1a

Exercise 1  
Configuring messaging ports

The first exercise will focus on the most basic of operations, setting up a pass-through integration flow that takes a file from one folder and delivers it to another. The focus of this exercise it to illustrate the Publish-Subscribe nature of BizTalk Server and how that is configured.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create a receive port and receive location | 1. Open the **BizTalk Server 2016 Administration Console** from **Start menu -> BizTalk Server Administration.** 2. In the **BizTalk Server 2016 Administration Console,** find the **Applications** node and expand to view existing Applications. 3. Right click the **Applications** node and select **New | Application**. Name the new Application **Lab1**. 4. Expand the newly created **Lab1** Application. Right-click the **Receive Ports** node and choose **New | One-way receive port.** 5. In the dialog box that appears, enter **PassThruReceive** as the name and click OK.   *A receive port is a logical entry point into BizTalk server and contains one or more receive locations. This logical entry point allows for several different physical entry points that all appear as one logical entry point. Some settings, such as maps, can be applied at the port level to impact all messages entering through the port.*   1. Highlight the **Receive Ports** node, then right-click on the **PassThruReceive** and choose **New | Receive Location**. 2. For the name, enter **PassThruReceive\_FILE**. 3. In the **Type** dropdown for transport, choose **FILE**. 4. Notice that the default **Receive Pipeline** is **PassThruReceive**.     *A receive location consists of the adapter and the pipeline. The adapter handles receiving the data and converting it to a BizTalk message, while the pipeline pre-processes the message to prepare it for routing.The PassThruReceive pipeline however, does no processing – it just passes the message thru.*   1. Click on the **Configure** button to change settings for the File adapter. 2. For the Receive folder, enter this path: **C:\Labs\Work\Lab 1\Messaging\Receive\** 3. Set the **File mask** property to **\*.txt**     *Notice that the adapter dialog has settings specific to the File adapter. Each adapter has different settings that make sense for how the adapter processes messages.*   1. Click OK on the dialogs until you are returned to the administrative console. |
| 1. Creating a send port   *A Send Port models a physical exit point for BizTalk server and consists of an adapter, pipeline and optionally maps. Send ports are used to send messages to other systems and partners.* | 1. In the **BizTalk Server 2016 Administration Console,** find the **Lab1** Application. 2. Right-click the **Send Ports** node and choose **New | Static One-way send port**. 3. For the **name**, enter **PassThruSendPort**. 4. Choose **FILE** for the transport **type**. 5. Click the **Configure** button to change the output path for the File adapter. 6. Enter **C:\Labs\Work\Lab 1\Messaging\Send\** for the destination folder. 7. Change the **File Name** field to “%datetime%.txt”.   *The %datetime% macro enables you to output a file with a name after the date it was processed.*   1. Click ok to return to the **Send Port Properties** dialog. 2. In the left hand menu, select **Filters**. 3. In the datagrid, select the **Property** column and select **BTS.ReceivePortName**. Make sure the operator is “**==**” and type the name of the receive port: **PassThruReceive**.      1. Click OK on the dialogs to return to the administration console. |
| 1. Start and Test the application. | 1. Right-click the **Lab1** application and choose **Start**. 2. When prompted, click **Start** again. 3. In Windows Explorer, navigate to: 4. **C:\Labs\Work\Lab 1\Messaging\.** 5. Locate the **PassThruFile.txt** file, then **COPY** and paste the file into the **Receive** folder. BizTalk is now monitoring this folder and will pick up your file. 6. Open the **Send** folder and watch for your message to appear. You should see a text file with a datetime for a name get written to the folder.   *It may take a minute for BizTalk to pick up your file and process it for the first time.* |

Optional: Configuring multiple subscribers

To further illustrate Publish-Subscribe, and the concept that a message is published once, but delivered to many subscribers. You will create a second Send Port that subscribes to the same message.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Creating a second send port | 1. Repeat the steps in Task 2, name the second Send Port **PassThruSendPort2** and when configuring the File Name use %MessageID%.txt as the File Name.   *The %MessageID% macro represents the ID of the message inside the BizTalk messaging infrastructure. It is in the form of a GUID and is guaranteed to be a unique name.* |
| 1. Start and Test the application. | 1. Repeat the steps in Task 3 to start and test the application.. 2. Open the **Send** folder and watch for your **messages** to appear. You should see **two text files**, one with a datetime for a name and one with a GUID get written to the folder.   *The files should appear almost instantaneously this time.* |
| 1. Remove the second send port | 1. Locate the **PassThruSendPort2** Send Port you created in Task 4. 2. Right click it and notice that the option to Delete the port is unavailable (grayed out). *The option to delete a send port is only available when it is not active (Enlisted or Started)..* 3. Right click the port and select to **Unenlist** it. 4. Now right click it and select **Delete**. |

Lab 1b

Exercise 2  
Creating a Schema

In this exercise, you will be creating a BizTalk project in Visual Studio for purposes of deploying the artifacts you’ll need for processing messages. You will also create a schema to represent messages arriving into your system.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Creating the BizTalk Project and adding an existing schema. | 1. Open Visual Studio 2015 from **Start menu -> Microsoft Visual Studio 2015.** 2. Choose **File | New Project** from the menu. 3. In the new project dialog box, choose the **Empty BizTalk Project** template and enter **ContosoMessaging** for the project name. 4. Ensure the path for the solution is **C:\Labs\Lab 1\Start**      1. After the project is created, right-click the ContosoMessaging project and choose **Add | Existing Item**. 2. Browse to the **C:\Labs\Lab 1\Start** folder and select the **CustomerOrderFlatFile.xsd** schema file.   *The schema you have added describes a flat-file message, in this case a message with comma-delimited values. Open the schema and take note of how the message is described using the Xml Schema Definition (XSD) language with additional attributes to describe the flat-file formatting. BizTalk uses this information to translate between XML and the flat file format.* |
| 1. Creating a new schema | * 1. Right-click the ContosoMessaging project in Solution Explorer and choose **Add | New Item**.   2. Select the **Schema** item template and enter **Order.xsd** for the name.   *The schema will open in the BizTalk Schema editor which is the tool you use to work with the schemas that define the structure of your messages.*     * 1. In the treeview, right-click the node labeled “Root” and rename it to “CustomerOrder”.   2. Right-click the **CustomerOrder** node and choose **Insert Schema Node | Child Record**. Name the new record **“CustomerInfo”**.   3. Right-click the **CustomerInfo** node and select **Insert Schema Node | Child Field Element**. Name the new element **“FirstName”**.   4. Repeat the previous step to create two other elements named **LastName** and **Phone**.   5. Right-click the **CustomerOrder** node one more time and choose **Insert Schema Node | Child Record.** Name the record “OrderDetails”.   6. Right-click the **Order.xsd** file in the solution explorer and choose **Generate Instance**.   7. In the output window, hold down the CTRL key and click the link to the generated file to see what an instance of this message looks like.   *You have created a schema to define the structure of an order message from a customer. In this simple example the order details have been left out to save time.* |

Exercise 3  
Creating a map and pipeline

In this part of the exercise, you will be creating two components that help in processing messages. First you will create a map which enables you to define transformations of messages from one format or schema into a message that follows a different schema. Then you will define a pipeline that provides staged processing for messages entering or exiting BizTalk Server. Your pipeline will define processing steps to serialize an XML message to its flat-file format and your map will transform a message between the two schemas you worked with in the previous exercise.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Creating the Map | * 1. In **Visual Studio,** right-click the **ContosoMessaging** project in the Solution Explorer and choose **Add | New Item**.   2. Select the Map template and name the file **MapOrderToCustomer.btm**.   3. After the map is added, click the link to **Open Source Schema**.   4. In the type picker dialog, expand the **Schemas** node and select the **ContosoMessage.Order** schema.      * 1. Now click the link in the right pane to **Open Destination Schema** and select the **ContosoMessaging.CustomerOrderFlatFile** schema in the type picker.   *You have selected the two schemas you will be mapping. The map provides instructions for how the runtime transformation of messages should be done, when the type of the incoming message matches the source schema in the map.*   * 1. Right-click on the **<Schema>** node in each pane and select **Expand Tree Node**.   2. In the source schema, under CustomerInfo, click on the **Phone** node and drag it to the **ContactNo** node on the destination schema. There should a line connecting to the two when you release the mouse button.   *This is an example of a simple mapping where data is simply copied.*   * 1. In the toolbox, find the **String Functoids** group and expand it if it is not already visible. Drag the **String Concatenate** functoid onto the grid between the source and destination schemas.   *Functoids provide additional transformation capabilities beyond simple copying of data. Look in the toolbox and you will find many different categories of functoids for different transformation tasks.*   * 1. Drag the **FirstName** and then **LastName** elements from the source schema to the **String Concatenate** functoid on the grid.   2. Now click on the **String Concatenate** functoid on the grid and drag to connect it to the **FullName** element in the destination schema.   3. Double-click the functoid on the grid and click the button to add a new constant value adding a single space as the value (to add space between the first and last names).      * 1. Click OK and save your work. The completed map should look similar to the image below. |
| 1. Creating a send pipeline   *Pipelines define message processing that happens as messages enter or leave BizTalk. They are commonly used to handle manipulation of files such as deserialization into XML, encryption or encoding of messages, and custom components to manipulate messages.* | 1. Right-click the **ContosoMessaging** project and choose **Add | New Item**. 2. Select the **Send Pipeline**  template and name the item **FlatFileSendPipeline.btp** 3. In the designer, drag a **Flat file assembler** from the toolbox and drop it in the **Assemble** stage where indicated.      1. Note in the properties grid that there are various settings for the component that you can modify. For this lab you will leave all settings at their default values.   *Now you have a pipeline that can serialize an xml message into the correct Flat File format using the information found in a schema. In a later exercise you will use this pipeline to do just that.* |

Exercise 4  
Creating an orchestration

In this exercise, you will be creating an orchestration to define a business process around your messages. Orchestrations are meant to coordinate, or orchestrate, message exchanges, primarily for longer running processes. This example will create a very simple orchestration to receive and send a message. You would not necessarily use an orchestration for this purpose as you can achieve the same thing without orchestration; it is used here simply to help you understand how all of the pieces fit together.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Defining the orchestration messages   *Orchestrations primarily manage message interactions. The first step is to create some message variable that represent the messages you will be processing in the orchestration.* | 1. Right-click the **ContosoMessaging** project in Solution Explorer and choose **Add | New Item**. 2. Select **BizTalk Orchestration** and name the file **OrderProcessing.odx** 3. After the orchestration designer appears, make sure the **Toolbox** and the **Orchestration View** windows are visible. You can make them visible by selecting them from the **View** menu. 4. In the **Orchestration View** tool window, right click the **Messages** node and select **New Message**. 5. Once the new message appears (named Message\_1), ensure it is selected and then change the **Identifier** property in the property grid to “**OrderMessage**”. 6. In the property grid, expand the **Message Type** values to select **Schemas | ContosoMessaging.Order**.      1. Repeat steps d-f to create another message with the identifier of “**CustomerMessage**” and the message type of **ContosoMessaging.CustomerOrderFlatFile**. |
| 1. Define the messaging activities.   *In the orchestration you will need activities or shapes that model the receipt and sending of messages. You will also need shapes to create message that are not received.* | 1. From the toolbox, drag a **Receive** shape onto the design surface and drop it near the top. 2. Follow this with a **transform** shape and then a **send** shape so that you orchestration looks like the following image. 3. Select the **Receive\_1** shape and set the **Activate** property to **True**. 4. Select the **OrderMessage** you created in the previous step for the **Message** property on the **Receive\_1** shape.   *In the last two steps, you have told BizTalk that you expect to receive an order message and that it should activate, or start, a new instance of this orchestration when that messages arrives. When the orchestration is started, the OrderMessage variable will hold a reference to the received message.*   1. Select the **Send\_1** shape and set the **Message** property to **CustomerMessage**.   *In order to send the CustomerMessage you first have to create it. In the following steps you will use the map created in an earlier exercise to create the CustomerMessage by transforming the OrderMessage.*   1. Select the **Transform\_1** shape and click the ellipses to edit the **MapName** property. 2. When the Transform Configuration dialog appears, select the Radio Button next to **Existing Map**. 3. From the drop down, choose the **ContosoMessaging.MapOrderToCustomer** map you created earlier. 4. Highlight **Source** in the Transform tree on the left and select the **OrderMessage** under the Variable Name column. 5. Highlight **Destination** in the Transform tree on the left and select the **CustomerMessage** under the Variable Name column. 6. Click OK to close the transform configuration. |
| 1. Defining the orchestration ports   *Ports allow you to define logical entry and exit points into the business process. In a later step you will “bind” these logical ports to physical ports.* | 1. Drag a **port** from the toolbox and drop it on the leftmost area of the designer labeled **Port Surface**. 2. When the wizard appears, click **Next**. 3. Enter **OrderPort** for the **Name** and click **Next**. 4. In the **Select a Port Type** wizard page, enter **OrderPortType** for the **Port type name** and leave the other values at their defaults. 5. On the **Port Binding** page, leave the **defaults** to indicate you will be receiving messages through this port and will specify the binding information later. 6. Finish the wizard. 7. Drag another port to the rightmost area of the designer and drop it under the label Port Surface. 8. Complete the wizard as before, with the following values.  |  |  |  | | --- | --- | --- | | **Wizard Page** | **Property** | **Value** | | Port Properties | Name | CustomerPort | | Select a port type | Port Type Name | CustomerPortType | | Port binding | Port direction of communication | I’ll always be sending message on this port. |  1. Now that the ports are defined, drag the green connector from the **OrderPort** and drop it on the green connector on the **Receive\_1** shape. 2. Drag the connector from the **CustomerPort** and drop it on the connector for the **Send\_1** shape. 3. Build the solution to ensure that all steps have been completed correctly. |

Optional: Deploy and Test

There is no optional exercises for Lab 1b. If you are done ahead of time you can start with Lab 1c.

Lab 1c

Exercise 5  
Deploying your solution

In this exercise, you will be configuring and then deploying your solution into BizTalk Server so it can be managed and started.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Configuring the solution for deployment | 1. Right-click the ContosoMessaging project in Solution Explorer and choose **Properties**. 2. In the properties page, select the **Signing** tab. 3. Click the box next to **Sign the assembly** and then choose **Browse** from the drop down to choose a strong name key file. Select the key.snk file from C:\Labs\Lab 1\Start\key.snk     *All BizTalk assemblies must be strong-named and deployed into the Global Assembly Cache (GAC). This enables numerous features, not the least of which is side-by-side versioning of your solution components.*   1. Now click the **Deployment** tab on the properties page. 2. Enter **Lab1** for the **Application Name** 3. Change the **Server** property to **“.”** (period without the quotes)      1. Save all files.   *By using “.” for the server name, the project file becomes more portable and can be used for local development on any development server.* |
| 1. Deploy the solution | 1. Right-click the **ContosoMessaging** Solution in Solution Explorer and choose **Deploy Solution**. 2. Make sure the **Output** window is visible and monitor the progress of the deployment. When the deployment is completed, the status bar in Visual Studio should read **Deploy succeeded**.   *At this point you should be able to navigate to C:\Windows\Microsoft.NET\assembly\GAC\_MSIL and find the ContosoMessaging assembly deployed into the Global Assembly Cache (GAC) where BizTalk can find it (this location is different from the pre-.NET 4 GAC that is located at c:\windows\assembly).*  *Additionally, metadata about your assembly and the components within it, has been deployed to the BizTalk management database.* |

Exercise 6  
Configure and Test your solution

In this exercise, you will be importing a *binding file*. The binding file does the work that you did yourself in exercise 1 - create the physical receive and send ports that define the actual entry and exit points for BizTalk Server. The end result is that the XML file should be picked up by BizTalk, processed by your orchestration and transformed to the flat-file structure, and sent out to a different folder, being serialized to its flat-file representation along the way by the send pipeline.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Import bindings | 1. Open the **BizTalk Server 2016 Administration Console** from **Start menu -> BizTalk Server Administration.** 2. In the **BizTalk Server 2016 Administration Console,** find the **Applications** node and expand it to find the **Lab1** node representing your application. 3. Expand **Lab1** and notice the various artifacts that make up your application. You can navigate to the Maps and Pipelines nodes to see several of the components you have built and deployed. 4. Right-click **Lab1** and Select **Import | Bindings…** 5. Browse to **C:\Labs\Lab 1\Start** and select **bindings.xml**.   *You have now imported settings that creates and configures receive ports, send ports and orchestrations. In the next exrcise you will review them.*  *Note:From the history of delivering this course, this may very well be your first run in with naming collisions in binding files. If you have not named everything exactly as instructed, this binding file will not work for you.* |
| 1. Review Receive Location properties | 1. In the **Lab1** application highlight the **Receive Locations** node, then in the right hand side, right-click on the **CustomerOrder\_FILE** and choose **Properties**. 2. Review the properties, notice that we are using the XMLReceive pipeline. to enable us to receive and interpret Xml files.   *The XMLReceive pipeline enables us to receive and interpret xml files.*     1. Click on the **Configure** button to change settings for the File adapter. 2. Notice that the **Receive folder** is set to **C:\Labs\Work\Lab 1\Messaging\Receive\** and the **File mask** to **\*.xml**.      1. Click OK on the dialogs until you are returned to the administrative console. |
| 1. Review Send Port settings   *A Send Port models a physical exit point for BizTalk server and consists of an adapter, pipeline and optionally maps. Send ports are used to send messages to other systems and partners.* | 1. In the **BizTalk Server 2016 Administration Console,** find the **Lab1** Application. 2. Right-click the **Send Ports** node and in the right hand side, right-click **CustomerSendPort** and select **Properties**. 3. Notice that the **FlatFileSendPipeline** is the selected **Send pipeline**.     *The pipeline being used is the one you created earlier and will serialize the customer message into its flat-file format.*   1. Click the **Configure** button to view the output path for the File adapter. Notice that it the Destination folder is **C:\Labs\Work\Lab 1\Messaging\Send\** and that the **File Name** field is set to “%MessageID%.txt”. 2. Click OK on the dialogs to return to the administration console. |
| 1. Bind the orchestration ports to the physical ports.   *In the orchestration, there are logical ports representing message exchanges. In the previous steps you created physical ports with adapters, pipelines and addresses. Now you need to bind those two together to provide concrete implementations that the orchestration can use.* | 1. Right-click the **Lab1** application in the **BizTalk Server Administration Console** and choose **Configure**. 2. In the Configure Application dialog, select the **OrderProcessing** orchestration. Notice that the orchestrations logical ports are *bound* to physical ports.      1. Click OK to close the dialog. 2. Right-click the **Lab1** application and choose **Start**. 3. When prompted, click **Start** again. |
| 1. Test the application. | 1. In Windows Explorer, navigate to:   **C:\Labs\Work\Lab 1\Messaging\.**   1. Open the **CustomerOrder.xml** file to view the contents. This file matches the schema you created in the first exercise. 2. Close the file, then **COPY** and paste the file into the **Receive** folder. BizTalk is now monitoring this folder and will pick up your file. 3. Open the **Send** folder and watch for your message to appear. You should see a text file with a GUID for a name get written to the folder. Open the file in Notepad and examine the contents of the comma-delimited flat-file.   *It may take a minute for BizTalk to pick up your file and process it for the first time.* |
| 1. Delete the application   ***Note****: If you have time enough to do the Optional Exercise –* ***DO NOT*** *delete the application yet.* | *If you intend to do the optional excerises below, do not delete the BizTalk application at this time.*   1. When you are done testing the application, open the **BizTalk Server Administration** console. 2. Right-click the **Lab1** application and choose **Stop**. 3. In the dialog, select **Full Stop** and click the **Stop** button. 4. Once the application has stopped, right-click the application and choose **Delete**. |

Optional: Messaging only – no orchestration

In this exercise you will reconfigure the solution to eliminate the need for the orchestration project by configuring the map to execute on the send port instead.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Unenlist the orchestration | 1. In the **BizTalk Server 2016 Administration Console** locate the **OrderProcessing** orchestration. 2. Right-Click and select to **Unenlist** it. |
| 1. Configure the Send Port | 1. Locate the **CustomerSendPort** Send Port. 2. Right-click and choose **Properties…** to configure it (or double-click it). 3. On the left hand side, select **Outbound Maps**. 4. Click the first row of the **Map column** in the Outbound Maps grid. 5. Select the **ContosoMessaging.MapOrderToCustomer** map that you have created earlier. 6. On the left hand side, select **Filters**. 7. Add a filter for **BTS.ReceivePortName = OrderReceivePort**. 8. Close the dialog by clicking Ok. |
| 1. Start, Test and Delete the application  ***Note****: If you have time enough to do the Optional Exercise –* ***DO NOT*** *delete the application yet.* | 1. Repeat Step 11 of Lab1c Exercise 8 to test the solution.   *By configuring the map to run on the send port and configuring the send port to subscribe directly to the receive port you have eliminated the need for an orchestration in this flow.*   1. **Delete** the application as described in Step 9 of Lab1c Exercise 6 **or, if you plan to continue**, **remove** the **Map** and **Filter** on the **Send Port** you used changed. |

Optional: Direct Bound Orchestrations

In this exercise you will be changing the orchestrations **Port Binding**. That is - how the orchestrations **Logical Ports** are connected to the **Physical Ports** created in the **Administration Console**.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Update Orchestration | 1. In the **Visual Studio Solution**, open the Orchestration **OrderProcessing.odx**. 2. For the **OrderPort** and **CustomerPort**, update the **Binding** property to be **Direct**. Image below shows example for OrderPort. |
| 1. Re-deploy orchestration | 1. **Build** and **Deploy** the Orchestration. Refer to previous steps in previous tasks for details if you are unsure of the process. 2. In the **Administration Console**, expand **Platform Settings**, expand **Host Instances**, select the running BizTalkServerApplication and choose restart.   *We do this because BizTalk Server is a .NET application. .NET applications, with the exception of IIS, do not automatically reload assemblies that are already loaded in memory when a newer version is placed in the GAC.* |
| 1. Re-configure ports | 1. In the **Administration Console**, find the orchestration. 2. Select it and choose **Properties**. 3. Go to **Bindings**. Note that there are now no logical ports that can be connected to physical ports. 4. Close the dialog. 5. In the **Administration Console**, to to **Send Ports**. 6. Open the **CustomerSendPort** 7. Go to **Filters** and add a filter for **BTS.MessageType =** [**http://NWMessaging.CustomerOrderFlatFile#Customers**](http://NWMessaging.CustomerOrderFlatFile#Customers) 8. Close the dialog by clicking OK. |
| 1. Test the solution | 1. Repeat the steps in Exercise 6 – Task 8 and note the result. |
| 1. Delete the application | 1. Repeat the steps in Exercise 6 – Task 9. |
| 1. Read about binding options | 1. Read about binding options on the SDK help page <http://msdn.microsoft.com/en-us/library/aa578247.aspx>. |