# BTSHOL09: Creating BizTalk Server orchestrations

Objectives

After completing this lab, you will be able to:

* Create a BizTalk orchestration to define message processing logic
* Create logical ports and bind them to physical ports
* Define correlation sets and correlate related messages
* Create messages in orchestrations
* Manipulate context properties and distinguished fields in messages

You work for Contoso Winery and you have been tasked with implementing the business process to handle automated order processing. The process involves receiving order details from customers, and coordinating with the warehouse and accounting systems to ship those items and correctly bill customers. You will use the orchestration capabilities of BizTalk Server 2016 to manage the message exchange between systems.

Estimated time to complete this lab: 60 minutes

User Name: **Administrator**

Password: **pass@word1**

Exercise 1  
Creating the messaging logic

In this exercise, you will be creating a BizTalk orchestration and layout the message processing logic to define the receipt and sending of messages.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Creating the orchestration.   *The solution contains schemas and maps for you to use as you define the business process.* | 1. Open the following solution: **C:\Labs\Lab 9\Start\NWBusinessSolution.sln.** 2. Right-click the **NWBusinessProcesses** project and select **Add | New Item**. 3. Select the BizTalk Orchestration item template and enter “**NWOrderProcessing.odx**” for the name. |
| 1. Add messaging variables   *When defining business processes, the first step is to define variables that can be used to refer to the messages being managed by the orchestration.* | * 1. Make sure the **Orchestration View** window is present and visible. If it is not, choose **View | Other Windows | Orchestration View**.   2. Right-click the **Messages** node in the Orchestration View and select **New Message**.   3. In the properties window, change the identifier property to “OrderMessage”.   4. In the **MessageType** property select **Schemas | Select from referenced assembly**.   Untitled  Select the **NWMessaging.CustomerOrder** schema from the **NWMessaging** project.     * 1. Repeat these steps to create four more message variables with the following names and types.  |  |  | | --- | --- | | **Name** | **Message Type** | | **CustomerReply** | NWMessaging.OrderResponse | | **AccountingRequest** | NWMessaging.AccountingInvoice | | **WarehouseRequest** | NWMessaging.WarehouseShipRequest | | **WarehouseResponse** | NWMessaging.DistResponse | |
| 1. Add messaging activities   *Once message variables are available, you can define the receiving and sending of messages with Send and Receive shapes in the orchestration. Each shape refers to the message variable that it acts upon.* | * 1. Drag a **Receive** shape from the toolbox and drop it on the orchestration design surface.   2. In the properties view, change the **Identifier** for the shape to **Receive\_Order**.   3. Use the drop-down to set the **Message** property to **OrderMessage**.   4. Set the **Activate** property to **True**.   *Setting the Activate property tells BizTalk that the receive shape represents receiving a message that should start a new instance of the orchestration.*   * 1. Add a **Send** shape below the previous receive shape.   2. Change the **Identifier** property for the send shape to **Send\_ShipRequest** and the **Message** to **WarehouseRequest**.   3. Add another **Receive** shape below the send, and change the **Identifier** property to **Receive\_ShipNotification**.   4. Set the **Message** property to **WarehouseResponse**.   5. Add another **Send** shape named **Send\_AccountingRequest** and specify the **AccountingRequest** message.   6. Finally, add one more Send shape named **Send\_ClientNotification** with the message **CustomerReply**.   7. Your orchestration should look like the image below at this point. |
| 1. Add business logic to the orchestration.   *Part of developing orchestrations is defining business processing logic or control flow. There are several shapes in the toolbox for managing the flow of your orchestration.* | * 1. Drag a **Decide** shape from the toolbox and add it to the designer between the **Receive\_ShipNotification** and **Send\_AccountingRequest** shapes.   2. Select the shape named **Rule\_1** and click the ellipses in the properties window to edit the condition expression.   3. Set the condition to:   *WarehouseResponse.Status == "Shipped"*   * 1. Drag the **Send\_AccountingRequest** send shape under the **Rule\_1** shape.   *The decide shape uses a distinguished field in the warehouse reply message to make decisions about whether to notify accounting to invoice the customer. For this process, the customer should only be billed if the item ships. A different process will initiate the accounting work when the item ships later.* |

Exercise 2  
Creating messages and ports

In this exercise, you will be defining the steps necessary to create new messages and define logical messaging ports in the orchestration. Orchestration is primarily about managing messages, which means the bulk of the work you do in orchestrations is receiving, sending, and creating messages.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Define the message creation activities.   *Before you can send a message, you must create it. Messages that are received are created for you.* | * 1. In Visual Studio 2015, make sure the orchestration is still visible.   2. Drag a **Transform** shape from the toolbox onto the design surface and drop it after the **Receive\_Order** shape.   *Notice that the Transform shape is automatically wrapped with a Construct Message shape. All message construction must happen in a Construct Message shape and can use a combination of Transform shapes to create messages from maps, or the Message Assignment shape to use expressions to manipulate and create messages.*   * 1. Click the ellipses to edit the **Map Name** property.   2. In the Transform Configuration dialog, select the **Existing Map** radio button and then navigate to select the **MapOrderToShipRequest** map through “<Select from referenced assembly…>”.   3. For the **Source**, choose **OrderMessage** and for the **Destination** choose the **WarehouseRequest**.   Untitled   * 1. Highlight the **ConstructMessage\_1** shape and examine the properties. Notice the **Messages Constructed** property which indicates you are constructing the WarehouseRequest message. This helps enforce the rules around message creation.   2. Add two more Transform shapes to the orchestration; one **preceding** the **Send\_AccountingRequest** and one preceding the **Send\_ClientNotification**.   3. For the transform before the Send\_AccountingRequest, select the **MapCustomerOrderToAccountingInvoice** map using the **OrderMessage** as the source and **AccountingRequest** as the destination.   4. For the last transform, select the **MapWarehouseResponseToOrderResponse** map and use **WarehouseResponse** as the source and **CustomerReply** as the destination. |
| 1. Create the logical messaging ports.   *In an orchestration, the message shapes (receive and send) are paired with logical port definitions that define operations and message exchange patterns. Later, these logical ports will be “bound” to physical ports in the system.* | 1. From the toolbox, drag a **Port** shape and drop it on the leftmost pane of the design surface labeled **Port Surface**.   *A wizard will start to aid you in configuring both a port and a port type. A port type can be used by more than one port instance as long as they share the same logical operations.*   1. In the first page of the wizard, click **Next**. 2. Enter “**OrderReceivePort**” for the **Name** property and click **Next**. 3. On the **Select Port Type** page of the wizard, enter “**OrderReceivePortType**” for the **Port Type Name** and leave all the other settings with their default values.   Untitled   1. On the **Port Binding** page of the wizard, leave the **default** values to indicate you are receiving a message and you will provide the binding details later. 2. **Finish** the wizard and notice the configured port now visible on the left port surface. 3. Drag the green connector from the new port to the connector on the **Receive\_Order** shape.   *If you review the properties on the Receive\_Order shape, you will notice the operation is now set to the Operation\_1 operation on the OrderReceivePort.*   1. Repeat this process to add another port to the left port surface with the following settings (any setting not specified can be left at its default value).  |  |  |  | | --- | --- | --- | | **Wizard Page** | **Property** | **Value** | | Port Properties | Name | CustomerAcknowledgementPort | | Port type | Port Type Name | CustomerAckPortType | | Port Binding | Port direction | I’ll always be sending . . . | | Port Binding | Port Binding | Dynamic | | Port Binding | Send Pipeline | Microsoft.BizTalk.DefaultPipelines.PassThruTransmit |   *Notice that for this activity, the port binding will be done dynamically. This means that instead of statically defining the physical port information, it will be defined during the execution of each orchestration instance.*   1. Drag the connector from this new port to the **Send\_ClientNotification** shape in the designer. 2. Now add three ports to the right port surface using the following data to complete the wizard.  |  |  |  | | --- | --- | --- | | **Wizard Page** | **Property** | **Value** | | Port Properties | Name | WarehouseSendPort | | Port type | Port Type Name | WarehouseSendPortType | | Port Binding | Port direction | I’ll always be sending . . . |  |  |  |  | | --- | --- | --- | | **Wizard Page** | **Property** | **Value** | | Port Properties | Name | WarehouseReceivePort | | Port type | Port Type Name | WarehouseReceivePortType | | Port Binding | Port direction | I’ll always be receiving . . . |  |  |  |  | | --- | --- | --- | | **Wizard Page** | **Property** | **Value** | | Port Properties | Name | AccountingPort | | Port type | Port Type Name | AccountingPortType | | Port Binding | Port direction | I’ll always be sending . . . |  1. Connect the **WarehouseSendPort** to the **Send\_ShipRequest** shape. 2. Connect the **WarehouseReceivePort** to the **Receive\_ShipNotification** shape. 3. Connect the **AccountingPort** to the **Send\_AccountingRequest** shape. 4. Drag an **expression** shape from the toolbox and add it to the designer just above/before the **Send\_ClientNotification** shape.   *Because the customer port is dynamic, you need to add logic to your orchestration to set the address that should be used. This address will indicate the adapter through the scheme (e.g. FILE, HTTP) and the address to use. The pipeline was specified on the port when you completed the wizard. For this example, we will assume the client provides their reply address as part of the incoming message. An expression shape can be used to set the address on the port using a distinguished field from the OrderMessage.*   1. Double-click the expression shape to edit the expression and add the following code:   *CustomerAcknowledgementPort(Microsoft.XLANGs.BaseTypes.Address) = OrderMessage.ReplyAddress;*   1. Save all the files. |

Exercise 3  
Defining correlation

In this exercise, you will configure the correlation of messages so that all messages related to a particular order get processed by the same orchestration instance. For this example, the messages will be correlated based on the order id, which will be present in each message. The warehouse response will be sent back to BizTalk and must get routed to the correct instance of the orchestration using data within the message itself.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Defining the correlation type.   *Correlation types define the collection of promoted properties that will be used to correlate incoming message to an instance.* | 1. In **Orchestration View** window, right-click the **Correlation Types** node under the Types header and click **New Correlation Type**. 2. In the dialog that appears, select the **NWMessaging.CustomerPONumber** property definition and click the **Add** button to move it into the correlation type definition.   Untitled   1. In the property grid, change the **identifier** for the correlation type to **CustomerPOCorrType**. |
| 1. Defining the correlation set   *A correlation set is a specific instance of a correlation type. You can have multiple sets within an orchestration all sharing the same type.* | 1. In **the Orchestration View**, right-click the **Correlation Sets** node and select **New Correlation Set**. 2. In the property grid, select **CustomerPOCorrType** for the **Correlation Type** property and enter **CustomerOrderCorrelation** for the **identifier** property. |
| 1. Configuring the messaging activities to use correlation   *Once the correlation sets and types are defined, the messaging activities need to be configured to Initialize a correlation set, which means to store the values of the properties at the time the activity executes, or to Follow a correlation set, which usually means to build a subscription based on the stored values.* | 1. Highlight the **Send\_ShipRequest** shape in the designer. 2. In the property grid, set the **Initializing Correlation Sets** property to **CustomerOrderCorrelation**.   *Now when a message is sent with this shape, the value of the promoted property in the message will be saved in the correlation set. Any shapes following this same correlation set can build a specific subscription based on this value.*   1. Highlight the **Receive\_ShipNotification** shape. 2. In the property grid, set the **Following Correlation Sets** property to **CustomerOrderCorrelation**.   *Now this activity will build a subscription once the correlation set is initialized with a specific value so that it is only subscribed for messages with the specific value in the promoted property.*   1. Right click the solution in Solution Explorer and choose **Deploy** to package up your solution and deploy it to BizTalk. |

Exercise 4  
Configuring messaging ports

In this exercise, you will import the configuration of the physical receive and send ports that define the actual entry and exit points for BizTalk Server. You will then bind the physical ports to the logical ports you defined in the orchestration. The end result is that the XML file should be picked up by BizTalk, processed by your orchestration and sent out to the warehouse. You will also have BizTalk monitor another folder for the warehouse response and send results to accounting, when appropriate, and a reply to the customer.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Import bindings to define physical ports.   *Binding files can be used to store information about physical ports to more easily redeploy application configuration.* | 1. Open the **BizTalk Server Administration Console,** from **Start | All Programs | Microsoft BizTalk Server 2016**. 2. Expand the nodes until you find the **Applications | Lab9** node. 3. Right-click the **Lab9** application node and choose **Import | Bindings**. 4. Browse to the binding file at: **C:\Labs\Lab 9\Start\BindingFile.xml** 5. Once the import is complete, examine the **Send Ports** and **Receive Locations** created as a result of the binding exercise. There should be three Send Ports and two Receive Ports (each with a single Receive Location).   *The physical locations are defined, but need to be bound to the logical ports in the orchestration.* |
| 1. Binding the orchestration ports to physical ports. | 1. In the **BizTalk Server 2016 Administration Console,** find the **Lab9 Application**. 2. Right-click the **Lab9** application and choose **Configure**. 3. In the configuration dialog, select the **NWOrderProcessing** orchestration to configure the ports.   Untitled   1. Choose the following values for the bindings to associate the ports and choose the host for the orchestration.  |  |  | | --- | --- | | **Property** | **Value** | | Host | BizTalk Server Application | | OrderReceivePort | CustomerOrderReceivePort | | WarehouseReceivePort | WarehouseReceivePort | | AccountingPort | AccountingSendPort | | WarehouseSendPort | WarehouseSendPort |   *Notice that you do not need to configure the dynamic customer port as a dynamic send port was created automatically when the orchestration was deployed.*   1. Right-click the **Lab9** application node and choose **Start**, then click **Start** in the dialog box that appears. |

Exercise 5  
Test the solution

In this exercise, you will test the solution by submitting an order, then monitoring the warehouse data and submitting a response from the warehouse. Based on the response you choose from the warehouse, you will see different output for the accounting messages.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Test normal flow with shipped items. | 1. In Windows Explorer, navigate to **C:\Labs\Work\Lab 9\Messaging\** 2. Open the **CustomerOrder1.xml** file and notice the **CustomerOrderPONumber**, and the **reply address** (near the bottom of the message) and then close the file. 3. **Copy** the **CustomerOrder1.xml** file into the **Receive** folder. 4. Navigate to **the Warehouse\Send** folder and after a short wait, you should see the warehouse message that was sent from the orchestration. 5. Open the message to examine it and see that it is a ship request and contains the same customer number, then **delete** the message. 6. Go back to the **messaging** folder and review the **WarehouseResponse.xml** file. It is a response from the warehouse with the correct customer number in it. This will enable the message to get routed to the correct orchestration instance. 7. **Copy** the **WarehouseResponse.xml** file into the **Warehouse\Receive** folder. 8. You should now find two messages from the orchestration, one in the **Accounting** folder which is the accounting invoice request, and one in the **SendOrder** folder which is the customer reply. 9. Delete each of these files after reviewing their contents. |
| 1. Test with a delayed shipment. | 1. Repeat steps **a - h** above with these changes. At steps **f-g** use the **Warehouse\_Delayed.xml** file to represent a response from the warehouse indicating the shipment is delayed. 2. In step **h**, when you check the folders, you should only see output in the SendOrder folder. Your orchestration logic dictates that only responses of Shipped will send message to accounting. |
| 1. Delete the application | 1. In the **BizTalk Server Administration Console**, right-click the **Lab9** application and choose **Stop**. 2. When prompted, select **Full Stop** and click the **Stop** button. 3. After the application stops, right-click **Lab9** again, and select **Delete**. |

Optional:

“Exchange Office” Challenge!

If you are done well ahead of the rest of the group, then here is a challenge for you!

This exercise will require skills in schema building, mapping, Orchestrations, consuming web services, port configuration and routing.

I want to exchange money between different currencies. I want to do this by sending a message containing a sum of money and specifying a currency and I want to get back a sum of money in a different currency.

A requirement is that you must get the currency conversion rate from the web service available at <http://www.webservicex.net/ws/WSDetails.aspx?CATID=2&WSID=10> (this is an informational page to open in a browser window, not a service or wsdl endpoint).

[*In case this freely available online service is unavailable your course instructor can supply you with a local substitute*]

You can have the local initiating request and final local response look however you want, but to give you an idea there is a psuedo request message example below

<Conversion>  
 <SourceAmount>100</SourceAmount>  
 <SourceCurrency>SEK</SourceCurrency>  
 <TargetCurrency>USD<</TargetCurrency>  
</Conversion>

Psuedo response below

<Conversion>  
 <SourceAmount>100</SourceAmount>  
 <SourceCurrency>SEK</SourceCurrency>  
 <TargetAmount>6,53</TargetAmount>  
 <TargetCurrency>USD<</TargetCurrency>  
</Conversion>

Note that because you are calling a Two-Way / Solicit-Response Send Port that does not mean that you need to have a Two-Way Receive Port. You can receive the request in a One-Way File Receive Port and return the response in a One-Way File Send Port.

Feel free to discuss the exercise with your instructor. This is (for this course) advanced material and similar to the type of tasks a BizTalk developer might perform!