# BTSHOL10: Implementing Transactions

Objective

After completing this lab, you will be able to:

* Configure orchestration properties and variables for transactions.
* Create and configure business transactions.
* Build and deploy the orchestration transaction project.
* Start and test the orchestration.

Estimated time to complete this lab:   
60 minutes

In the last lab you implemented part of the order processing. In this lab, you will implement the Warehouse process that receives an order, reserves items from stock, reserves distribution resource and creates instruction for the warehouse staff on the items they need to assemble to complete the order. You will update the existing warehouse order-fulfillment orchestration to include transaction handling. This includes providing support for compensating transactions and exception handling. Transactions will be added to each section of a Parallel Action shape and around an entire Parallel Action shape.

User Name: **Administrator**

Password: **pass@word1**

Exercise 1  
Configuring Orchestration Properties and Variables

In this exercise, you will open an existing orchestration and modify that orchestration to support long-running transactions. You will also add an orchestration variable that you will use in a later exercise.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Open the starting solution. | 1. In Microsoft Visual Studio, open the starting **NWBusinessSolution.sln** solution in **C:\Labs\Lab 10\Start\** folder. |
| Review orchestration.  An orchestration must be transactional to contain other transactions. | 1. In Solution Explorer, right-click **WarehouseProcess.odx**, and then click **Open** to open the existing orchestration. 2. Review the orchestration and understand what it is doing,   *On a high level an Order is received for which three things need to be completed (left to right): Reserve the items and quantity requested from stock, request transport from the distributor, send information to warehouse staff of what to gather for this order.*   1. Find the **Order\_toWarehouse** transformation towards the bottom of the orchestration. Double click it, make sure the “When I click OK, launch the BizTalk Mapper” is selected, and press OK.  Review the map.   *Inside an orchestration you can easily map two input documents to one output document.*   1. Close the map and return to the orchestration. |
| Set orchestration properties.  An orchestration must be transactional to contain other transactions. | |  |  | | --- | --- | | Property | Value | | Transaction Type | Long Running | | Transaction Identifier | WarehouseTrans |  1. Right-click an empty area of the **Orchestration Designer** surface, then click **Properties Window**, and modify the following properties: |
| Create an orchestration variable.  In a later exercise, you will set this variable to an exception to demonstrate exception handling. | 1. In Orchestration View, right-click **Variables**, then click **New Variable**, and then create a variable with the following properties:  |  |  | | --- | --- | | Property | Value | | Identifier | varException | | Type | <.NET Class…>  mscorlib  System   Exception | |

Exercise 2  
Create and Configure Transaction Shapes

In this exercise, you will create and configure shapes required to support business transactions, including compensation and exceptions.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Create a scope for the parallel actions.  This scope indicates that all three parallel actions should complete successfully or none of them should. | 1. From the Toolbox, drag a **Scope** shape directly below the the Initializing Receive (located at the top of the Orchestration Designer), and configure the scope with the following properties:  |  |  | | --- | --- | | Property | Value | | Name | Transaction Shipping | | Transaction Type | Long Running | | Transaction Identifier | ShippingTrans |  1. Drag the **OrderHandling** parallel shape into the **Transaction Shipping** scope shape. |
| Create a scope for each of the parallel branches.  Each of the parallel actions is encompassed in a transaction itself. | |  |  | | --- | --- | | Property | Value | | Name | Trans Reserve Stock | | Transaction Type | Long Running | | Transaction Identifier | StockTrans |  1. Drag a **Scope** shape to the top of the first parallel branch (directly above the ConstructStockReservationRequest shape) and configure the scope with the following properties:  |  |  | | --- | --- | | Property | Value | | Name | Trans Ensure Distribution | | Transaction Type | Long Running | | Transaction Identifier | DistTrans |  1. Drag a **Scope** shape to the top of the second parallel branch and configure the scope with the following properties:  |  |  | | --- | --- | | Property | Value | | Name | Trans Notify Staff | | Transaction Type | Long Running | | Transaction Identifier | StaffTrans |  1. Drag a **Scope** shape to the top of the third parallel branch and configure the scope with the following properties: |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Move Scopes | 1. For each of the parallel actions, drag all of the shapes below the scopes into the corresponding scope. |
| Add a Decide shape to the first parallel branch.  This procedure is a simple test to see if the returned message from the stock system indicates the order was accepted and the items were in stock. | 1. Drag a **Decide** shape directly below the **RcvStock** Receive shape in the first parallel branch.  |  |  | | --- | --- | | Property | Value | | Name | Succeed? | | Expression | StockOutputMessage.parameters.ReserveStockResult.Success == true |  1. Click the **Rule\_1** shape and modify the properties as follows: |
| Add an Exception shape.  Normally, it would be inefficient to throw an exception to do flow control. We use it here to demonstrate exception handling and compensation. | 1. Drag a **Throw Exception** shape directly below the **Else** rule.   Modify the properties for the **Throw Exception** shape as follows:  |  |  | | --- | --- | | Property | Value | | Description | Out of stock | | Exception Object | varException | | Name | OutOfStock | |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Add compensation blocks.  Compensation blocks are called when an outer transaction fails after a nested transaction has already committed. Compensation is called *only* on committed transactions. | 1. Right-click the **Trans Reserve Stock** scope shape, and then click **New Compensation Block**. 2. Click the **Compensation Block** shape, and then modify the properties to change the name to **StockCompensation**.  |  |  | | --- | --- | | Property | Value | | Message | StockInputMessage | | Name | Recall Stock Reservation |  1. Drag a **Send** shape into the Compensation Block and modify the properties of the **Send** shape as follows: 2. Right-click the **Trans Ensure Distribution** scope shape, and then click **New Compensation Block**. 3. Click the **Compensation Block** shape, and then modify the properties to change the name to **DistCompensation**.  |  |  | | --- | --- | | Property | Value | | Message | DistributorInputMessage | | Name | Recall Dist Planning |  1. Drag a **Send** shape into the compensation block and modify the properties of the **Send** shape as follows: 2. Right-click the **Trans Notify Staff** scope, and then click **New Compensation Block**. 3. Click the **Compensation Block** shape, and then modify the properties to change the name to **StaffCompensation**.  |  |  | | --- | --- | | Property | Value | | Message | StaffInputMessage | | Name | Recall Staff Notification |  1. Drag a **Send** shape into the compensation block and modify the properties of the **Send** shape as follows: |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Create orchestration ports.  Logical ports are created here and will be bound at deployment. | |  |  | | --- | --- | | Parameter | Value | | Name | StockCompensateSend | | Create… Port Type Name | StockCompensateTypeSend | | Communication Pattern | One-Way | | Access Restrictions | Internal | | Port direction of communication | ...Sending… | | Port binding | Specify Later |  1. Right-click the **Port Surface** (left side), click **New Configured Port**, and then create a send port using the following information:  |  |  | | --- | --- | | Parameter | Value | | Name | DistCompensateSend | | Create… Port Type Name | DistCompensateTypeSend | | Communication Pattern | One-Way | | Access Restrictions | Internal | | Port direction of communication | ...Sending… | | Port binding | Specify Later |  1. Right-click the **Port Surface** (left side), click **New Configured Port**, and then create a send port using the following information:  |  |  | | --- | --- | | Parameter | Value | | Name | StaffCompensateSend | | Create… Port Type Name | StaffCompensateTypeSend | | Communication Pattern | One-Way | | Access Restrictions | Internal | | Port direction of communication | ...Sending… | | Port binding | Specify Later |  1. Right-click the **Port Surface** (right side), click **New Configured Port**, and then create a send port using the following information: |
| Connect Orchestration shapes to orchestration ports. | |  |  | | --- | --- | | Shape | Port | | Recall Stock Reservation | StockCompensateSend | | Recall Dist Planning | DistCompensateSend | | Recall Staff Notification | StaffCompensateSend |  1. Drag a connection between the **Orchestration Send** shape to the corresponding send port as follows: |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Create an exception handler.  The exception handler will be called when any exception occurs within its scope. | 1. Right-click the **Transaction Shipping** Scope shape, and then select **New Exception Handler**.  |  |  | | --- | --- | | Property | Value | | Exception Object Type | General Exception | | Name | ShippingException |  1. Right-click the **Exception Handler** (below the scope) and set the properties as follows:  |  |  | | --- | --- | | Name | Compensation | | CompensateStock | StockTrans | | CompensateDist | DistTrans | | CompensateStaff | StaffTrans |  1. Drag three **Compensate** shapes into the exception handler and set their properties as follows:  |  |  | | --- | --- | | Property | Value | | Description | Could not complete process | | Exception Object | General Exception | | Name | RethrowException |  1. Drag a **Throw Exception** shape below the compensation shapes and set the properties as follows:   Tip: Go to the BizTalk menu – select “Zoom” and select “50%” to be able to see the whole orchestration at once. Go back to “100%” when you are done. |

Exercise 3  
Building and Deploying the Orchestration Project

In this exercise, you will build the project to generate an assembly that contains the resources (the orchestration) that you have just created. This process also ensures that there are no compile-time errors in the work you have completed so far. You will then deploy the orchestration.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Build and deploy the orchestration project.  . | 1. On the File menu, click Save All. 2. In Solution Explorer, right-click NWMessaging, and then click Build. 3. Right-click NWMessaging, and then click Deploy. 4. Right-click NWBusinessProcesses, and then click Build. 5. Right-click NWBusinessProcesses, and then click Deploy.   The projects are compiled and displayed under Resources in the BizTalk Administration Console (you may have to refresh the view). |

Exercise 4  
Starting and Testing the Orchestration

In this exercise, you will start the BizTalk run-time process on the server. You will then enlist the service to associate the business process that you designed in the orchestration with the physical environment in which the orchestration will run. Finally, you will start the processing of the orchestration so that you can test your application.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Start the BizTalk run-time process.  The run-time process runs under the context of a BizTalk Server host instance. The host instance defines a security boundary for sending, receiving, and processing messages. | 1. Open the **BizTalk Server 2016 Administration Console**. 2. In the left pane, expand Microsoft BizTalk Server 2016 Administration, expand **BizTalk Group**, expand **Platform Settings**, and then click Host Instances. 3. In the right pane, right-click BizTalkServerApplication, and then click Restart.   Even though the status may be Running, a Host Instance must sometimes be restarted when a project is deployed. |
| Import Bindings to create ports | 1. In BizTalk Administration Console, right-click **BizTalk Group**, and then click **Refresh**. 2. Under **Applications**, right click **Lab10** and select Import > Bindings… 3. Browse to the file C:\Labs\Lab 10\Start\Lab10Bindings.xml and press Open. |
| Enlist and start the orchestration.  Performing this step configures the physical environment in which your solution runs to start processing messages.  The orchestration will fail if the receive locations have not been configured properly. | 1. Under **Lab10**, expand Orchestrations, double-click NWBusinessProcess.WarehouseProcess. 2. In the **Orchestration Properties** dialog box, select **Binding** and then configure port-bindings as follows:  |  |  | | --- | --- | | Orchestration Ports | Binding | | OrderPort | ReceiveOrder | | StockServicePort | WcfSendPort\_StockService\_WSHttpBinding\_IStockService | | StockCompensateSend | CompensateStock | | DistributorServiceSend | WcfSendPort\_DistributorService\_WSHttpBinding\_IDistributorService | | DistCompensateSend | CompensateDist | | StaffServicePort | WcfSendPort\_StaffService\_WSHttpBinding\_IStaffService | | StaffCompensatePort | CompensateStaff | | WarehouseResponsePort | WarehouseProcessCompleted |  1. In the **Port Bindings Properties** dialog box, configure the **Host** to use **BizTalkServerApplication**, and click **OK**. 2. In BizTalk Administration Console, right-click Lab10, click **Start**, and thenclick **Start** on the **Start** dialog box. 3. Open Windows Explorer and navigate to **C:\Labs\Lab 10**. 4. Click **StartServices.bat** – this will start three services responsible for replying to the messages sent to Stock, Distribution and Staff systems.   Your solution is now ready for testing. |

(*continued*)

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| Run the scenario.  This step is to test the orchestration process and business policy execution. | 1. Using Window Explorer, copy  **C:\Labs\Work\Lab 10\ Messaging\Warehouse\CustomerOrder1.xml** to C:\Labs\Work\Lab 10\ Messaging\Warehouse\Receive.   Do not move the file because BizTalk Server processes it and then removes it from the folder.   1. **Browse to C:\Labs\Work\Lab 10\Messaging\Warehouse\Send.** There will be an output file there because the process completes successfully. You may have to wait a few seconds for the process to complete. Check the Group Hub page. |
| Run the alternate outcome scenario. | 1. Using Windows Explorer, copy  **C:\Labs\Work\Lab 10\Messaging\Warehouse\CustomerOrderNOT\_IN\_STOCK.xml**, and then paste it to  C:\Labs\Work\Lab 10\ Messaging\Warehouse\Receive. 2. **Browse to C:\Labs\Work\Lab 10\Messaging\Warehouse\Send.**   You should not see a new message because the order was rejected by the StockService.   1. Browse to **C:\Labs\Work\Lab 10** \Messaging\Warehouse\CompensateDist to view the contents of the folder. 2. Browse to **C:\Labs\Work\Lab 10** \Messaging\Warehouse\CompensateStaff to view the contents of the folder. 3. Browse to **C:\Labs\Work\Lab 10** \Messaging\Warehouse\CompensateStock to view the contents of the folder.   You should not see a new messagein this folder since only transactions that have completed are compensated. Sincethe StockTrans caused an exception to be thrown, it did not complete and will not be compensated. |
| Run the alternate outcome scenario.  This scenario can be tried given time to try the default behavior of compensation. | 1. Go back to the orchestration and remove all compensate shapes. 2. Remove all suspended service instances. 3. Deploy the orchestration. 4. Restart the host instance. 5. Drop the **CustomerOrderNOT\_IN\_STOCK.xml** into the C:\Labs\Work\Lab 10\ Messaging\Warehouse\Receive folder. 6. Review the output folders. |
| Delete the application | 1. In the **BizTalk Server Administration Console**, right-click the **Lab10** application and choose **Stop**. 2. When prompted, select **Full Stop** and click the **Stop** button. 3. After the application stops, right-click **Lab10** again, and select **Delete**. |