# BTSHOL08: Using the WCF Adapters in BizTalk Server 2016

Overview

In this lab, you will learn how to use the WCF Adapter in **BizTalk Server 2016**. After completing this lab, you will be able to:

* Configure BizTalk Server 2016 to use the WCF Adapter
* Create an Orchestration that consumes a WCF Service
* Expose a BizTalk Orchestration as a WCF service.

Scenario

You work for a travel booking company, who receives flight requests. You need to build a service that correlates these requests to airline companies your company has partnerships with.

**Part 2 (If you have time)**

There have also been requests for booking hotels at the same time as booking the flights. Your job is to update the service to handle requests for hotel bookings along with the already existing flight bookings.

Estimated time to complete this lab: 60 minutes

User Name: **Administrator**

Password: **pass@word1**

Exercise 1

Create a TravelService Orchestration

You will develop an orchestration that will be exposed as a web service. The orchestration will in turn call to an external Flight Booking Service, representing the airline company.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Open the solution, and examine the contracts | 1. Open the following solution: C:\Labs\Lab 8\Start\Lab8.sln 2. In the Solution explorer, open TravelRequest.xsd.   The TravelRequest represents the incoming request to your service.   1. In the Solution explorer, open TravelResponse.xsd*.*   The TravelResponse represents the outgoing response from your service. |
| 1. Create an orchestration | 1. In the Solution explorer, right-click the TravelService project and select *Add -> New Item*. 2. In the *Add New Item* dialog, select *BizTalk Orchestration.* Set the name to *TravelServiceOrchestration* |
| 1. Create a receive port.   *This will be the port exposing your orchestration as a web service* | 1. In the orchestration designer, right-click the left port surface and select *New Configured Port*. 2. Click *Next* to set the name of the port. Set the name to *TravelBookingService.* Click *Next.*   The name of the port will become the name of the service.   1. In the *Select Port Type* dialog, set the *Port Type* name to *TravelServicePortType*. 2. Set the *Communication Pattern* to *Request-Response*, and the Access Restrictions to *Public.*   The Access Restrictions must be set to public when exposing a port as a web service   1. Click *Next,* and leave the Port direction to “I’ll be **receiving**…” 2. Rename the *Operation\_1* operation on the *TravelService* to *SubmitTravelRequest.* |
| 1. Start the external Flight Booking Service.   *This is the actual service representing your partner.* | 1. Open an Explorer window and browse to **C:\Labs\Lab 8\Services.** 2. Double-click the *FlightBookingService.exe* file to start up the service. |
| 1. Create a send port | 1. Go back to the Solution Explorer, and right-click the *TravelService* project. Select *Add->Add Generated Item.* 2. In the *Add Generated Item* dialog, select *Consume WCF Service,* and click *Add.* 3. Click the *Next* button until you get to select the source of the metadata. 4. In the *Metadata source* dialog, *Metadata Exchange (MEX) endpoint.* Click *Next.* 5. In the *Metadata Address (URL)* field enter [**http://localhost:8080/FlightBookingService**](http://localhost:8080/FlightBookingService)**.** Andclick the *Get* button. 6. Click *Next*and *Next* again to complete the WCF Service Consuming Wizard.   This step has created a number of artifacts for you. First of all it has created request- and response schemas that we’ll be using in the next step when creating messages. It has also created an orchestration named “FlightBookingService”. Although we won’t be using this orchestration, we will leave it for now as it contains a port type necessary for the next step.  Lastly, it has also created a bindings files that you will be using to eventually set up your physical send port.   1. Right-click the right port surface and select *New Configured Port.* 2. Click *Next* to set the port name. Set the name to *FlightBookingService*, and click *Next*. 3. Select *Use an existing Port Type,* and select *IFlightBookingService*. 4. Click *Next*, and set the port direction to “I’ll be **sending**…” 5. Click *Next* and *Finish* to complete the wizard. |
| 1. Add Receive and Send shapes. | So far you have created an orchestration with two ports. –One receiving and one sending.   1. In the *Orchestration Designer,* add four shapes in sequential order :  |  |  | | --- | --- | | Name | Type | | Receive TravelRequest | Receive Shape | | Send TravelResponse | Send Shape | | Receive FlightRequest | Receive Shape | | Send FlightResponse | Send Shape |  1. Set ***Activate* to *true***for the initial Receive Shape (*Receive TravelRequest*) |
| 1. Create the messages | 1. In the *Orchestration Viewer,* create two messages as below:  |  |  | | --- | --- | | Name | Type | | msgTravelRequest | TravelService.TravelRequest | | msgTravelResponse | TravelService.TravelResponse |  1. Drag the *Request* arrow from the **FlightBookingService** to the *Send TravelResponse* shape. 2. Drag the *Response* arrow from the **FlightBookingService** to the *Receive FlightRequest* shape. 3. Rename the *Message\_1* message to *msgFlightRequest*, and the Message\_2 to *msgFlightResponse* 4. Right-click the *Receive TravelRequest* shape and set the *Message* to *msgTravelRequest.* 5. Right-click the *Send FlightResponse* shape and set the *Message* to *msgTravelResponse.* 6. Drag the *Request* arrow from the *TravelService* to the *Receive TravelRequest* shape. 7. Drag the *Response* arrow from the *TravelService* to the *Send FlightResponse* shape |
| 1. Create Flight Booking request message and Travel Booking Response message. | You are almost done, but you have to make a couple of transformations before you can deploy your solution.   1. Add a *Transform* shape after the first *Receive* shape. 2. Rename the *ConstructMessage\_1*  to *Construct Flight Request.* Set the *Messages Constructed* to *msgFlightRequest.* 3. Double-click the *Transform\_1* shape. Set the name to *TravelService.TravelRequest\_to\_FlightRequest*. 4. Set the *Source* to *msgTravelRequest* and the *Destination* to *msgFlightRequest.parameters.* Click *Ok*  to launch the BizTalk Mapper. 5. Keep the left Shift key down on your keyboard, while you drag the *TravelRequest* node from the source schema to the *FlightBookingRequest* node in the destination schema. 6. Select *Link By Name,* from the context menu. 7. Add a *Add Days* functoid from the tool box the mapper surface. 8. Drag the *DepartureDate* node from the source schema to the *Add Days* functoid. Do the same with the *NumberOfNights* node. 9. Drag the *ReturnDate* node from the destination schema to the *Add Days* functoid. 10. Save and close the *TravelRequest\_to\_FlightRequest* map.   Before you’re done, you also have to create the Travel response message…   1. Add a *Transform* shape after the *Receive FlightRequest* shape 2. Rename the *ConstructMessage\_1* shape to *Construct Travel Response.* Set the *Messages Constructed* to *msgTravelResponse.* 3. Double-click the *Transform\_2* shape. Set the name to *TravelService.FlightResponse\_to\_TravelResponse* 4. Set the *Source* to *msgFlightResponse.parameter* and the *Destination* to *msgTravelResponse.* 5. Keep the left Shift key down on your keyboard, while you drag the *SendFlightBookingRequestResult* node from the source schema to the *TravelResponse* node in the destination schema. 6. Select *Link By Name,* from the context menu. 7. Save and close the *FlightResponse\_to\_TravelResponse* map. 8. Your orchestration should look something like this: |
| 1. Build and deploy your solution… | 1. Right click TravelService and select properties -> Deployment and set application name to “*TravelService*”. Save the change. 2. Right-click the TravelService project and select *Deploy.*   The project has already been configured with an Application and a key… |

Exercise 2

Create physical ports and bind the TravelService Orchestration

So far you have created the orchestration. But before you can try it out you need to:

1. Create physical ports
2. Bind the orchestration
3. Start the application
4. Publish Metadata (WSDL).

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create the Send port | As you might recall, when you created the orchestration send port, two bindings files were created, -one using WCF-BasicHttp binding and one using WCF-Custom binding. These files will come to good use now, although you’ll only be using one of them.   1. In your *TravelService* project, open the *FlightBookingService.BindingInfo.xml* file, and examine the context. 2. Right-click to tab on the top of the window and select *Copy full path:*   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML14cc466f.PNG   1. Open the BizTalk Server Administration Console, and browse the *TravelService* application. 2. Right-click the *TraveService* application and select *Import -> Bindings.* Paste the path from the clipboard. Click *Open*. |
| 1. Create the Receive port | 1. Open the *TravelService* application and right-click the *Receive Ports.* Select *New->Request Response Receive Port.* 2. Set the name to *ReceiveTravelBooking,* andclick *Receive Locations.* 3. Click *New* and set the name to *ReceiveTravelBooking\_WCF-NetTcp*. 4. In the *Type* drop-box select *WCF-NetTcp*. Click *Configure*. 5. In the *Address (URI)* field, type net.tcp://localhost:888/travelbookingservice and click *Ok*. 6. Set the Receive Pipeline to *XmlReceive.* 7. Click *Ok* and *Ok* again to close the dialog. |
| 1. Bind the orchestration and start the application. | 1. In the BizTalk Server Administration Console under the TravelService, select Orchestrations. 2. Double-click the *TravelService.TravelServiceOrchestration* and select *Bindings* from the left pane. 3. Set the host to BizTalkServerApplication and bind the receive- and send ports to the once you’ve just created. 4. Click *Ok* to close the dialog. 5. Start the Send port, the Receive location and the orchestration (leave the *FlightBookingServiceClient* in *Unenlisted* state). |
| 1. Publish metadata (WSDL) | Your service is now ready to receive incoming calls. However, at this point there is no way for the consumer of your service to know what your service looks like. For this you need to publish your Receive Locations through a WSDL.   1. Go back to your Visual Studio Solution. In the file menu select *Tools->BizTalk WCF Service Publishing Wizard.*      1. Click *Next* on the Welcome page, and proceed with selecting *Metadata only endpoint (MEX).*   Since you have already created the Receive port, you only need to create a MEX endpoint.   1. Select the *ReceiveTravelBooking\_WCF-NetTcp* from the available *Receive Locations.* 2. Click *Next* and *Next* again to browse to the BizTalk assembly. Click the *Browse* button. 3. Browe to C:\Labs\Lab 8\Start\TravelService\bin\Debug\TravelService.dll and click *Ok.* 4. Click *Next* until you get to the “WCF Service IIS Location”. 5. Select the “Allow anonymous access…” checkbox. 6. Click *Next* and *Create* to complete the wizard. 7. Test your MEX endpoint by opening <http://localhost/TravelService/TravelService_TravelServiceOrchestration_TravelBookingService.svc?wsdl> in your browser. |

Exercise 3

Testing the service

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Test the service using the *WcfTestClient.* | To test your service, you will be using a generic tool called *WcfTestClient.* This is a toolyou’ll likely grow familiar with when developing web services, as it doesn’t require you to actually build a test client.   1. In Visual Studio, under the tools menu, click the *WcfTestClient.* 2. Right-click the *My Service Projects* node and select *Add Service.* 3. Set the endpoint address to: <http://localhost/TravelService/TravelService_TravelServiceOrchestration_TravelBookingService.svc> 4. Click *Ok*, and disregard the warning dialog. 5. Double-click the *SubmitTravelRequest* operation in the treeview. 6. Continue filling in the travel information, and click the *Invoke* button to call the TravelBookingService.   C:\Users\ADMINI~1\AppData\Local\Temp\SNAGHTML1506ddbf.PNG   1. Examine the result (you can hover the BookingNotes to see the complete message). |

Exercise 4 (*If you have time*)

Complement the service

In this exercise you’ll add another service call to your orchestration. This will leverage the customer to make hotel arrangements along with booking the flight.

As this is an optional exercise, you’ll not get as much help.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Start the external Hotel Booking Service. | 1. Open an Explorer window and browse to **C:\Labs\Lab 8\** **Services.** 2. Double-click the *HotelBookingService.exe* file to start up the service. |
| 1. Add another send port in the orchestration | 1. Add the HotelBookingService to your already existing orchestration the same way you did in Exercise 1.e. The address to the hotel service is: <http://localhost:8080/HotelBookingService> |
| 1. Add shapes | 1. As you are about to make an additional service call from your orchestration, it might be a good idea to call the two services in parallel. 2. Add a Parallel Action shape to your orchestration. Place it after the first Receive shape. 3. Move the Construct Flight Request, Send TravelResponse and the Receive FlightRequest shape to one of the branches of the Parallel shape. 4. Add necessary shapes on the other branch to make the call to the HotelBookingService. |
| 1. Create a Travel Response Message. | The current transformation to create the msgTravelResponse message only uses the response from the flight service. You need to create a new map, using both the flight booking response and the hotel booking response to create the outgoing response from your orchestration.   1. Remove the *Transform\_2* shape, and replace it with a new one. 2. Double-click the new Transformation shape, and add both the *msgFlightResponse.parameters* message **AND** the *msgHotelResponse.parameters* to thesource messages. 3. Set the msgTravelResponse as the Destination. Click *Ok* to launch the BizTalk Mapper. 4. Expand both schema node on both sides of the mapper. As you can see, there are two source schemas now. 5. Proceed with the mapping as you see fit. 6. Your complete orchestration might look like this: |
| 1. Build and Deploy | 1. Remember to import the binding for the HotelBookingService. 2. As your service contract has not changed, you don’t have to publish your orchestrations as a WCF service again. However, don’t forget to restart the host instance… |
| 1. Test the service | 1. Repeat Exercise 3 to test your service. |

Optional:

“Weather Service” Challenge!

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

In this exercise you will extend the solution above to include additional service calls. You will do this with even less information about the steps needed to complete the exercise. Tip! If in doubt, refer to the steps described for the previous exercises.

The travel booking service that you work for needs to distinguish itself from competitors. One way to do this is by supplying additional services to its customers. One such service is adding information about the current weather at the location requested.

The weather service that you will use for this task is located at <http://webservicex.net/ws/WSDetails.aspx?WSID=56&CATID=12> (this is an informational page to open in a browser window, not a service or wsdl endpoint).

The steps to complete include:

* + Use the Add… - Add Generated Item… Consume WCF Service wizard.
  + Consume service metadata to get the schema, orchestration with porttypes and binding of the service.
  + Create a new logical port in the TravelServiceOrchestration.
  + Create the needed message variables and maps to create the request and receive the response.
  + Create a new map to merge the responses from the Flight, Hotel and Weather services into the Travel response where you add the GetWeatherResponse message from the weather service to the BookingNotes field of the Travel Response.
  + Configure the send port by importing the generated binding file.
  + Deploy and test the solution. Note! This will require you to enter valid Destination information when sending the booking request, for example entering “Stockholm, Sweden” and extracting that in the map to the weather request to form the parameters City and Country.

Feel free to discuss the exercise with your instructor.