# BTSHOL08B: REST AND JSON

Overview

In this lab, you will learn how to call out to use REST and JSON with BizTalk Server:

* Install Node.js
* Create a Simple RESTful Web Service in Node.js
* Install Postman as a tool to submit REST and JSON requests to the service
* Create schemas in BizTalk for the JSON request and response
* Deploy solution
* Configure the solution and create receive and send ports
* Test the solution
* Optional: Work more with REST and JSON on the receive side

Scenario

In your company a system is being updated. This updated version will have a new API based on REST and JSON. You need to integrate with this new API on the send side from BizTalk and supply an XML and SOAP based façade on the receive side.

Estimated time to complete this lab: 60 minutes

User Name: **Administrator**

Password: **pass@word1**

Exercise 1

Install Node.js

The runtime we are going to use to create a JSON formatted REST web service for you to consume is the Javascript runtime Node.js. Javascript is a language that has gotten an increased popularity over the last few years and Node.js is a popular runtime.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Download Node.js | 1. Open Internet Explorer and go to <https://nodejs.org> 2. Download the version of Node “Recommended for most users”. |
| 1. Install Node.js | 1. Run the downloaded installation. 2. Accept the license agreement and accept all default installation options, including path and which components to install. |
| 1. Verify Node.js installation | 1. Open a Command prompt 2. In the prompt, type: **node --version**   The response should be the version of Node.js you installed.  If it was not, review previous install steps. |

Exercise 2

Use the Node Package Manager (npm) to download packages

Node.js comes with a package manager, much like NuGet for .Net. The Node Package Manager, or npm is an easy way to extend Node.js with additional functionality.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Download and install packages | 1. Open Internet Explorer and go to <https://npmjs.org> 2. In the “f*ind packages*” search box in the top, type “*express*”.   This should take you to a result list where express by dougwilson described as “Fast, unopinionated, minimalist web framework” is listed as the first hit.   1. Press express. 2. This will take you to a page with some more information about the express package. 3. One of the things you can learn is the command to install the package: **npm install express** 4. In the command prompt window, navigate to **Labs\Lab 8\Services** folder, and type in the command to install the express package.   The npm command has been added to your path with the installation of Node.js and a short diagnostics for the download and installation procedure should follow.   1. Once it is installed, you should see the file package-lock.json and the folder node\_modules added to the folder. 2. You have now successfully downloaded a javascript based web framework to serve your content and are ready to create your service. |

Exercise 3

Create your Javascript service

Next we will create a simple web service that allows you to request customer info.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create a data file | 1. Using Notepad or Visual Studio 2. Enter the following text in a new text document   {  "customer1":  {"name":"Microsoft","value":"Gold","id":1},  "customer2":  {"name":"AddSkills","value":"Gold","id":2},  "customer3":  {"name":"Contoso","value":"Silver","id":3},  "customer4":  {"name":"Northwind","value":"Bronze","id":4}  }   1. Save the file as customers.json in the **Labs\Lab 8\Services** folder. |
| 1. Create the service | 1. Using Notepad or Visual Studio 2. Enter the following text in a new text document   var express = require('express');  var app = express();  var fs = require("fs");  app.get('/:id', function (req, res) {  fs.readFile( \_\_dirname + "/" + "customers.json", 'utf8', function (err, data) {  var customers = JSON.parse( data );  var customer = customers["customer" + req.params.id]  console.log( customer );  res.type('json')  res.end( JSON.stringify(customer));  });  })  var server = app.listen(8081, function () {  var host = server.address().address  var port = server.address().port  console.log("Customers service started at http://%s:%s", host, port)  })   1. Save the file as server.js in the **Labs\Lab 8\Services** folder.   The file you just created sets up a simple webservice listening on port 8081 that reacts to HTTP GET requests with an id. When it gets a request it reads the customers.json file, looks for customer entry that matches the id sent in and returns it. |

Exercise 4

Start and test the service

In this exercise you will start and test the service.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Start the service | 1. In the Command prompt, still navigated to the **Labs\Lab 8\Services** folder, type **node server.js** 2. This will start your service and the application should respond with: *Customers service started at http://:::8081* |
| 1. Download tools and test the service   *To test the service you will use one of the de facto standard tools in a developers tool box - Postman* | 1. Open Internet Explorer and go to <https://www.getpostman.com/>. 2. Download and run the 64-bit installation. 3. When you start the application you can either choose to sign up or login (if you already have an account), or at the bottom you can simply choose to go straight to the app. 4. Close the Create New window that opens as you open the app. 5. In the middle of the screen, you will have a *New Tab* and a window to *Enter request URL* next to a drop down with *GET* selected. As the request URL enter **http://127.0.0.1:8081/2**   The service reply should be the AddSkills entry. |
| 1. Save the response | 1. Save the AddSkills response in a text file called **customer.json** in the Services folder. The contents of the response and the file should be as below.   {"name":"AddSkills","value":"Gold","id":2} |

Exercise 5

Create BizTalk artifacts

In this exercise you will start and test the service.

|  |  |  |  |
| --- | --- | --- | --- |
| Tasks | | Detailed steps | |
| 1. Create project | | 1. Open Visual Studio 2. Choose to create a new *Empty BizTalk Server Project*, place it in the **Labs\Lab 8** folder, make sure *Create directory for solution* is selected and write **RESTJSON** as *Name*. | |
| 1. Create the customer response schema.  This creates the XSD schema that describes the XML structure that flat files will be converted to.   The schema will not contain any metadata that describes the JSON format. Instead, the resulting schema looks like a regular XML Schema. | | 1. In Solution Explorer, right-click the RESTJSON project, point to Add, and then click New Item. 2. In the Add New Item dialog box, in the Templates pane, select **JSON Schema Wizard**. 3. In the Name box, type Customer.xsd, and then click **Add**. 4. In the **BizTalk JSON Schema Wizard**, click **Next**. 5. On the **JSON Schema Information** page, in the **Instance file** box, browse to or type **C:\Labs\Lab 8\Services\customer.json.** 6. In the **Record name** box, type **Customer**, and then click **Finish.**      1. Review the created XML Schema, noting that it looks like a regular XML Schema without any added metadata. 2. After the id property of the schema, add a new element called **timestamp**. Leave it as type xs:string. | |
| 1. Create the customer request schema | | 1. In Solution Explorer, right-click the RESTJSON project, point to Add, and then click New Item. 2. In the Add New Item dialog box, in the Templates pane, select **Schema**. 3. Name the schema **CustomerRequest.xsd** 4. Rename the root node to **CustomerRequest** 5. Create a new attribute under the root node called **Id** 6. Right click the **Id** attribute and select *Promote – Quick Promotion* 7. For the property schema **PropertySchema.xsd** that gets created you can (but do not have to) remove the Property1 property. The important thing is that you now have the property Id setup as a promoted property. | |
| 1. Create a map | | 1. In Solution Explorer, right-click the RESTJSON project, point to Add, and then click New Item. 2. In the Add New Item dialog box, in the Templates pane, select **Map**. 3. Name the map **AddTimestamp.btm** 4. Select the **Customer** schema as the source schema. 5. Select the **Customer** schema as the destination schema. 6. Map all elements across except for **timestamp**.   The timestamp element will never be created by the service which you can see if you return to Postman. We will use it as our proof that the message has been processed by BizTalk   1. Use a *Date and Time* functiod and connect it to the **timestamp** element in the destination. | |
| 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 RESTJSON project in Solution Explorer and choose **Add | New Item**. 2. Select **BizTalk Orchestration** and name the file **GetCustomerProcess.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 “**RequestMessage**”. 6. In the property grid, expand the **Message Type** values to select **Schemas | RESTJSON.CustomerRequest**. 7. Repeat steps d-f to create another message with the identifier of “**ServiceResponse**” and the message type of **RESTJSON.Customer**. 8. Repeat steps d-f to create another message with the identifier of “**BizTalkResponse**” and the message type of **RESTJSON.Customer**. | |
| 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 **Send** shape, a second **Receive** shape, a **Transform** shape and then a second **Send** shape so that your orchestration looks like the following image. 3. Select the **Receive\_1** shape and set the **Activate** property to **True**. 4. Select the **RequestMessage** message you created in the previous step for the **Message** property on the **Receive\_1** shape. 5. Select the **Send\_1** shape and set the **Message** property to the same **RequestMessage** message. 6. Select the **Receive\_2** shape and set the **Message** property to the **ServiceResponse** message. 7. Select the **Send\_2** shape and set the **Message** property to the **BizTalkResponse** message.   In order to send the BizTalkResponse message you first have to create it. In the following steps you will use the map created in an earlier exercise to create the BizTalkResponse message by transforming the ServiceResponse message.   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 **RESTJSON.AddTimestamp** map you created earlier. 4. Select **ServiceResponse** as *Source*. 5. Select **BizTalkResponse** as *Destination*. 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. Add a configured port on the left hand side port surface. Name it **RcvPort**. Give it a Port Type name of  **RcvPortType** and select **Request-Response** as **Communication Pattern** and leave the other values at their defaults. 2. 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.      1. Finish the wizard. 2. Repeat steps a-f to create one more port and place it on the right hand side with the help of the following: Name: SndPort Port Type Name: SndPortType Communication Pattern: Solicit-Reponse Port Direction: Sending a request and receiving a response. 3. Drag the connector from the **RcvPort Request** and drop it on the connector for the **Receive\_1** shape. 4. Drag the connector from the **SndPort Request** and drop it on the connector for the **Send\_1** shape. 5. Drag the connector from the **SndPort Response** and drop it on the connector for the **Receive\_2** shape. 6. Drag the connector from the **RcvPort Response** and drop it on the connector for the **Send\_2** shape. 7. Build the solution to ensure that all steps have been completed correctly. | |
| 1. Create a JSON to XML receive pipeline | | 1. Add a new **Receive Pipeline**. 2. In the Name box, type JSONtoXMLReceive, and then click **Add**. 3. From the ***Toolbox***, drag a **JSON decoder** component to the **Decode stage** of the pipeline. 4. Configure the JSON decoders properties so that *Root Node* matches the root node of the Customer schema, **Customer**, and the *Root Node Namespace* is the namespace of the Customer schema, **http://RESTJSON.Customer**. 5. From the *Toolbox*, drag a **XML disassembler** component to the **Disassemble stage** of the pipeline. 6. The XML disassembler needs no further configuration. | |

Exercise 6  
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 **RESTJSON** 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 **New** from the drop down to create a new strong name key file. Do not use a password. Use whatever name you like. 4. Now click the **Deployment** tab on the properties page. 5. Enter **RESTJSON** for the **Application Name** 6. Change the **Server** property to **“.”** (period without the quotes) 7. 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 **RESTJSON** Solution in Solution Explorer and choose Deploy Solution. |

Exercise 7  
Configure your solution in BizTalk

In this exercise, you will be configuring your solution in the BizTalk Administration Console creating the ports you need and connecting them to the orchestration.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Create a Receive Port | 1. Open the BizTalk Administration Console 2. Expand the **RESTJSON** Application. 3. Right click Receive Ports and create a new Request-Response port called **RcvCustomer**. 4. Create the Receive Location with the help of the following information *Name*: **RcvCustomer\_WCF-http** *Type*: **WCF-Custom**  On the Configure dialog for the WCF-Custom adapter  *Binding*: **BasicHttp**  *Address*: **http://localhost:8082/customer.svc** *Receive pipeline*: **XMLReceive**   By using WCF-Custom we are enabling hosting http services inside the BizTalk host process.  **e.** Close the dialog by clicking ok to save the changes. |
| 1. Create a Send Port | 1. Right click Send Ports and create a new Solicit-Response port called **SndCustomer**. 2. Configure the Send Port with the help of the following information *Type*: **WCF-WebHttp**  On the Configure dialog for the WCF-WebHttp adapter  *Address*: **http://localhost:8081/** *HTTP Method and URL Mapping*:  **<BtsHttpUrlMapping>  <Operation Name="Operation\_1" Method="GET" Url="/{id}" />  </BtsHttpUrlMapping>**  *Variable Mapping*:  *Variable*: ID  *Property Name*: **Id**  *Property Namespace*: **https://RESTJSON.PropertySchema**  *Messages – Outbound Message*: **GET** Receive pipeline: **JSONtoXMLReceive**   By using our recently configured pipeline we are making sure that the JSON response of the service is decoded and sent into BizTalk as XML.   1. Close the dialog by clicking ok to save the changes. |
| 1. Configure orchestration bindings | 1. Under the BizTalk application **JSONREST**, go to *Orchestrations* and select the **GetCustomerProcess** orchestration. 2. **Double click** it to open the *Orchestration properties* dialog. 3. Go to the **Bindings** option. 4. Configure the bindings so that a host is selected and the recently created physical receive and send ports are selected and connected to the appropriate logical orchestration ports. 5. Close the dialog by clicking ok to save the changes. |
| 1. Start the application | 1. Right click the RESTJSON BizTalk application and select Start… 2. Click Start. |

Exercise 8  
Test your solution

In this exercise, you will test and verify that your BizTalk solution works as expected.

|  |  |
| --- | --- |
| Tasks | Detailed steps |
| 1. Download tools and test the service   *To test the service you will use one of the de facto standard tools in a developers tool box - Postman* | 1. Go back to Postman, or re-open it if you have close it. 2. Create a new tab by click the + to the right hand side of the open tabs. 3. Enter *Enter request URL* as **http://localhost:8082/customer.svc** 4. Change *GET* to **POST** 5. Click *Body*, select **raw** and select **XML (text/xml)** as the content type. 6. Paste the following as the body.   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" >  <soapenv:Body>  <ns0:CustomerRequest xmlns:ns0="http://RESTJSON.CustomerRequest" Id="2"/>  </soapenv:Body>  </soapenv:Envelope>   1. Click Send. 2. The response should be as below   <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">  <s:Body>  <ns0:Customer xmlns:ns0="http://RESTJSON.Customer">  <name>AddSkills</name>  <value>Gold</value>  <id>2</id>  <timestamp>2018-01-31T10:03:22</timestamp>  </ns0:Customer>  </s:Body>  </s:Envelope>   1. Change the request to ask for another customer id and try again. The reply this time will depend on the id you asked for. |

Challenge 1  
Add a REST receive location

A receive port can have more than one receive. Add another receive location using the knowledge you have received by doing this lab and create a second receive location that receives XML through REST instead of SOAP.

Hints: You could use the WCF-WebHttp adapter and the WCF Publishing Wizard to enable this. If you publish to <http://localhost/GetCustomer> you can then access the service from Postman at <http://localhost/GetCustomer/Service1.svc> where you then send an xml using POST with only the xml (no SOAP envelope, headers or body).

For even more help, check the screenshots on the next few pages (don’t do it unless you are stuck!).

Challenge 2  
Use JSON format

A further challenge for this lab is to change format on the incoming request from XML to JSON.

Hints: To do this you need to transform JSON to XML in the receive pipeline and XML to JSON in the send pipeline on the receive location. You have the schemas you need, and you have the receive pipeline you need, but you are missing a send pipeline that encodes from XML to JSON. Add it and configure a request-response send port to use it.

No additional help is available for this challenge.







