Labs for:

Publishing and consuming APIs from extensions

Peik Bech-Andersen

Lab 1: Enable SOAP, OData and API in the Administration tool

Prerequisite to this lab:

In order to complete this lab, there are a number of prerequisites:

- A Dynamics 365 BC must be available. Both the cloud solution and on-premise version will work.
- There must be access to the Dynamics 365 BC as **super user**
- Check that the "**Enable Develop Service Endpoint**" and "**Enable loading application symbol references at server startup**" settings must be activated in the Dynamics 365 BC service tier.

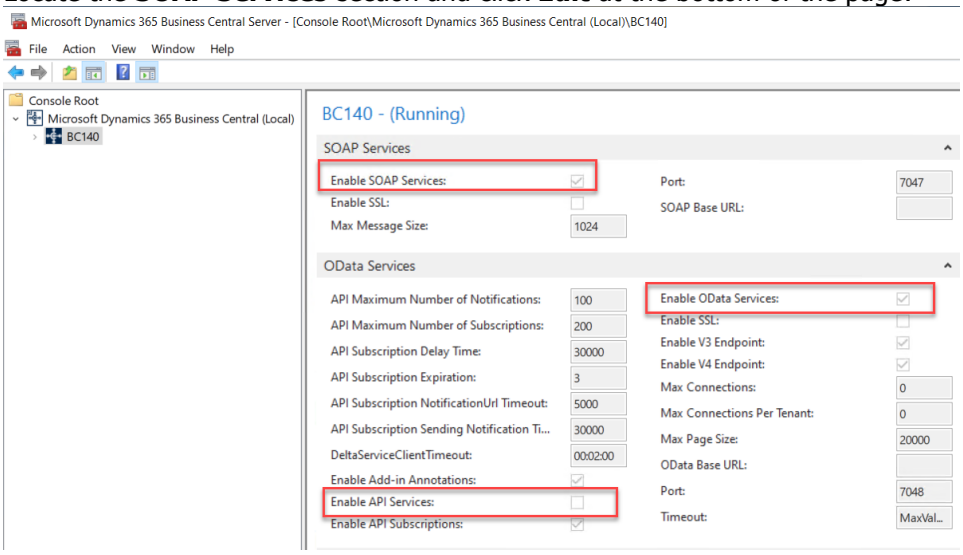
High-Level steps:

- Open the Dynamics 365 Business Central Administration tool
- Enable SOAP Services
- Enable OData Services
- Enable API Services
- Restart the Service tier

Detailed steps:

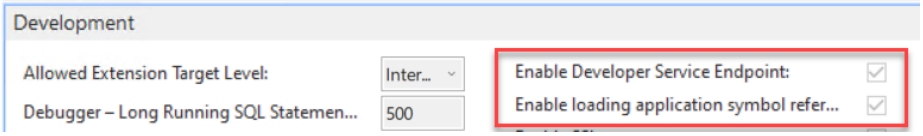
1) Open the Dynamics 365 Business Central Administration tool

2) Open the **Microsoft Dynamics 365 Business Central (Local)** entity and select the **BC140** Service tier, this might be named differently depending on your installation.
3) Locate the **SOAP Services** section and click **Edit** at the bottom of the page:



Make sure that the **Enable SOAP services** here and that **Enable OData services** and **Enable API services** fields in the OData section are checked

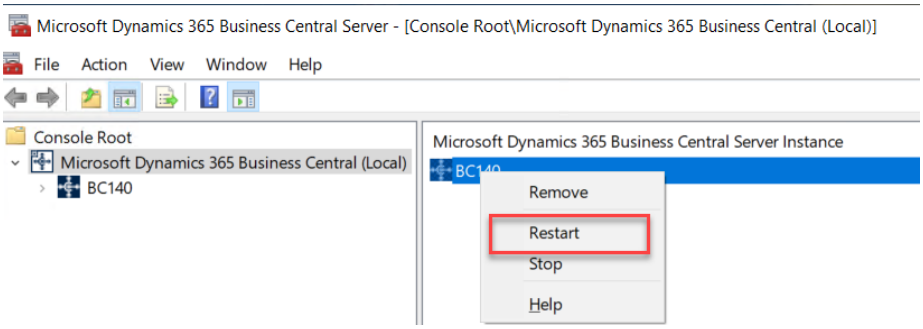
4) Scroll to the Development section:



5) Ensure that the **Enable Developer Service Endpoint** and **Enable loading application symbol references at server startup** fields are checked.

6) Click Save

7) Select the **Microsoft Dynamics 365 Business Central (Local)**, right-click on the **BC140** Service tier, and select **Restart**:



8) Close the Dynamics 365 Business Central Administration tool

Lab 2: Publish a test extension to Dynamics 365 BC

Prerequisite to this lab:

In order to complete this lab, there are a number of prerequisites:

- Previous labs must be completed
- Visual Studio Code must be installed locally
- The AL Language must be installed in VS Code

High-Level steps:

- Create a new project with the name: "API Test"
- Change the launch.json file to connect to the Dynamics 365 BC database
- Change the app.json file meet the minimum requirements for a Dynamics 365 BC extension. Add "D.E. Veloper" as the Publisher and "API Test" as the Extension Name
- Publish the extension to the database
- Verify that the web client opens and that the "Hello World" message appears
- Verify that the "API Test" extension is available in the Extension Management window.
- Remove the HelloWorld.al file from the project
- Remove the **"StartupObjectId"** line, the **"StartupObjectType"** line and the preceding comma.

Detailed steps:

1) Open VS Code
2) Click **Ctrl+Shift+P** or click the menu **View/Command Palette**
3) Write or select the **AL:Go!** Command
4) Replace **Project_** with **API Test** and press **Enter**
5) Expand the **.vscode** folder
6) Open the **launch.json** file
7) In the **name** setting, replace "Your own server" with the server name of Dynamics 365 BC server without http:// and any trailing information

```
"name": "dyn365test",
"server": "https://dyn365test.westeurope.cloudapp.azure.com",
```

Or

```
"name": "localhost",
"server": "http://localhost",
```

8) In this case, use **http://localhost**
9) Enter the URL to the server as shown above
10) Change the *ServerInstance* to **BC140**
11) Change the *Authentication* to **Windows** for on-premise installations or **UserPassword** for cloud
12) Then save the file by clicking **Ctrl+S**

13) Download the symbols by clicking **Ctrl+Shift+P** or clicking the menu **View/Command Palette**
14) Write or select the **AL:Download Symbols** Command
15) If Dynamics 365 BC requires user id and password, provide the user id and press **Enter** then give the password and press **Enter** again.
16) Verify that the message "All reference symbols have been downloaded" is shown In the Output Window
17) Open the **app.json** file
18) Change the **publisher** setting to be "D.E. Veloper"
19) Change the **name** setting to be "**API Test**"
20) Then save the file by clicking **Ctrl+S** or activate **Auto Save** in the *File* menu
21) Open the **HelloWorld.al** file
22) Change the **"App published: Hello World"** text to something else
23) Then save the file by clicking **Ctrl+S**
24) Publish the extension by clicking **Ctrl+F5** (Start without Debugging)
25) Verify that the message **Success: The package has been published to the server** is shown in the Output Window, and that the Web client opens with a message box.
26) Click the search  icon in the Web client and search for the **"Extension Management"** page
27) Verify that the **"API Test"** extension is available in the page
28) Close the Web client
29) Go back to VS Code
30) Stop the debugger on the red button in the debug panel:

31) Delete the "HelloWorld.al" file.
32) Now open the launch.json file and delete the **"startupObjectId": 22** line and delete the **"startupObjectType"** including the comma above.
33) Add an extra line as the last line with the property: **"schemaUpdateMode": "Recreate"**

Lab 3: Create the API pages

Prerequisite to this lab:

- Previous labs must be completed

High-Level steps:

- Open VS Code
- Create folders for **Reports**, **APIs** and **Codeunits**
- Create a new page with following information:
    - Object No **50100**
    - Name: **DIR WS Customer SOAP**
    - Type **Card**
    - Fields:
        - **No.**
        - **Name**
        - **Date Filter**
        - **Sales (LCY)**
- Create a new page with following information:
    - Object No **50101**
    - Name: **DIR WS Customer OData**
    - Type **Card**
    - Fields:
        - **No.**
        - **Name**
        - **Date Filter**
        - **Sales (LCY)**
- Create a new page with following information:
    - Object No **50102**
    - Name: **DIR WS Customer API**
    - Type **API**
    - Fields:
        - **No.**
        - **Name**
        - **Date Filter**
        - **Sales (LCY)**

Detailed steps:

1) Open VS Code
2) Right-click the Explorer background and select **New Folder**
3) Create a folder for **Reports**, **APIs** and **Codeunits**
4) Right-Click the **APIs** folder and select New File:
5) Name the file **pag50100_WSCustomerSOAP.al** and press **[Enter]**
6) Enter **tpa** in the first line and select the **tpage, Page of type card** snippet
7) Enter **50100** as the **ID** and "**DIR WS Customer SOAP"** as the **name**

and press enter
8) Set the Source table to be **"Customer"**
9) Add a **Caption** property **"WS Customer SOAP"**
10) Add the **InsertAllowed** property to be **false**
11) Add the **DeleteAllowed** property to be **false**
12) Add the **UsageCategory** property to be **Administration**
13) Add the following fields to the page in the Numbering group:
    a. No.
    b. Name
    c. Date Filter
    d. Sales (LCY)
14) Save the file using **Ctrl+S**
15) Right-Click the **APIs** folder and select New File:
16) Name the file **pag50101_WSCustomerOData.al** and press **[Enter]**
17) Enter **tpa** in the first line and select the **tpage, Page of type card** snippet
18) Enter **50101** as the **ID** and "**DIR WS Customer OData"** as the **name** and press enter
19) Set the Source table to be **"Customer"**
20) Add a **Caption** property **"WS Customer OData"**
21) Add the **InsertAllowed** property to be **false**
22) Add the **DeleteAllowed** property to be **false**
23) Add the **UsageCategory** property to be **Administration**
24) Add the following fields to the page in the Numbering group:
    a. No.
    b. Name
    c. Date Filter
    d. Sales (LCY)
25) Save the file using **Ctrl+S**
26) Right-Click the **APIs** folder and select New File:
27) Name the file **pag50102_WSCustomerAPI.al** and press **[Enter]**
28) Enter **tpa** in the first line and select the **tpage, Page of type API** snippet
29) Enter **50102** as the **ID** and "**DIR WS Customer API"** as the **name**
30) Set the Source table to be **"Customer"**
31) Add a **Caption** property **"WS Customer API"**
32) Set the **APIPublisher** property to **'DirectionsEMEA'**
33) Set the **APIGroup** property to **'APIs'**
34) Set the **APIVersion** property to **'v1.0'**
35) Set the **EntityName** property to **'WSCustomer'**
36) Set the **EntitySetName** property to **'WSCustomers'**
37) Add the **DeleteAllowed** property to be **false**
38) Add the **UsageCategory** property to be **Administration**
39) Add the **ODataKeyFields** property to be **Id**;
40) Add the following fields to the page in the Numbering group:
    a. No.
    b. Name
    c. Date Filter
    d. Sales (LCY)
41) Save the file using **Ctrl+S**

Lab 4: Create the Install Codeunit

Prerequisite to this lab:

- Previous labs must be completed

High-Level steps:

- Open VS Code
- Create a new codeunit with following information:
    - Object No 50100,
    - Name: DIR Install Customer API
    - SubType Install
    - Add code to publish the three web services
- Add code so the three pages are published as web-services

Detailed steps:

1) Open VS Code
2) Right-Click the **Codeunits** folder and select New File:
3) Name the file **cod50100_InstallCustomerAPI.al** and press **[Enter]**
4) Enter **tcu** in the first line
5) Enter **50100** as the **ID** and "**DIR Install Customer API"** as the **name** and press enter
6) Add a **SupType** property to be **"Install"**
7) Remove the **onRun** trigger
8) Remove the **var** section
9) Type **ttr** and select the **ttrigger** snippet
10) Replace the **OnWhat** with **OnInstallAppPerCompany**
11) Create a local variable **WebServiceManagement** referenced to the codeunit **"Web Service Management"**
12) Create a local **Option** variable: **ObjectType** with the following elements:
    TableData,Table,,Report,,Codeunit,XMLport,MenuSuite,Page,Query,System,FieldNumber
13) Add the following command in the first line of the trigger:
    WebServiceManagement.CreateWebService
14) Add the following parameters:
    a. ObjectType::Page
    b. Page::"DIR WS Customer SOAP"
    c. 'WSCustomerSOAP'
    d. true
15) Add another command after the previous:
    WebServiceManagement.CreateWebService
16) Add the following parameters:
    a. ObjectType::Page
    b. Page::"DIR WS Customer OData"
    c. 'WSCustomerOData'
    d. true
17) Add another command after the previous:
    WebServiceManagement.CreateWebService
18) Add the following parameters:
    a. ObjectType::Page
    b. Page::"DIR WS Customer API"
    c. 'WSCustomerAPI'
    d. True
19) Save the file using **Ctrl+S**

Lab 5: Publish the extension to the service tier

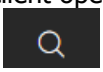
Prerequisite to this lab:

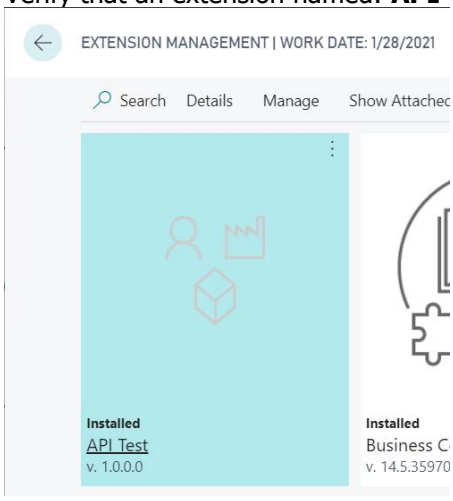
- Previous labs must be completed

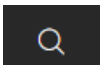
High-Level steps:

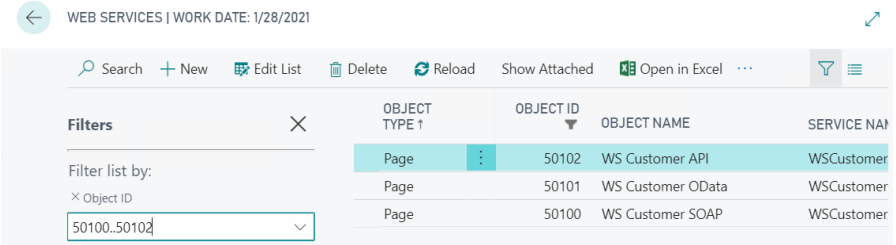
- Open VS Code
- Publish the extension to the service tier
- Open extension management and verify that the extension exists
- Open the web services

Detailed steps:

1) Open VS Code
2) Press Ctrl+F5 to publish the extension
3) Verify that the output says:
   Success: The package 'D. E. Veloper_API Test_1.0.0.0.app' has been published to the server.
4) Verify that the web-client opened automatically
5) Click the search icon  in earlier versions it might be a lightbulb
6) Search for **ext man** and select **Extension Management**
7) Verify that an extension named: **API Test** exists



8) Click the search icon  and search for **web ser** and select Web Services
9) Filter for Object ID 50100..50102
10) Verify that the three web services exist:



11) Leave the web Client open

Lab 6: Test the SOAP end-point

Prerequisite to this lab:

- Previous labs must be completed

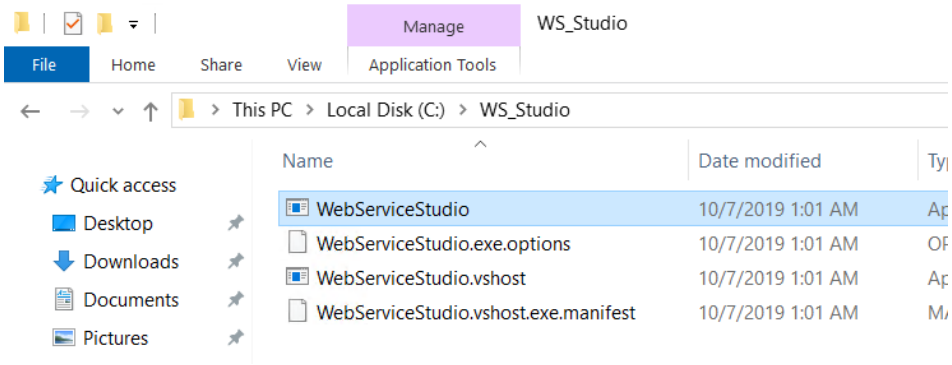
High-Level steps:

- Download the WSStudio or
- Download Postman for Windows or
- Add wizdler for Google Chrome or Firefox
- Test the endpoint

Detailed steps:

You can either:

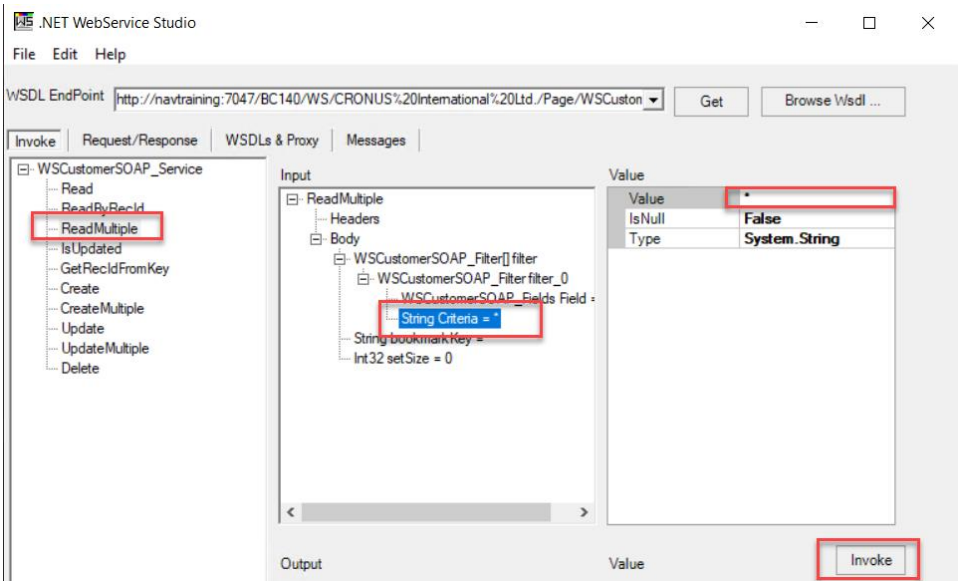
1) Open a web browser and download the WSStudio from: http://ba-consult.dk/downloads/WS_Studio.zip
2) Unpack the zip file to the C:\ drive
3) Start WS Studio



4) Switch to the web client and copy the SOAP URL end point for **WSCustomerSOAP**

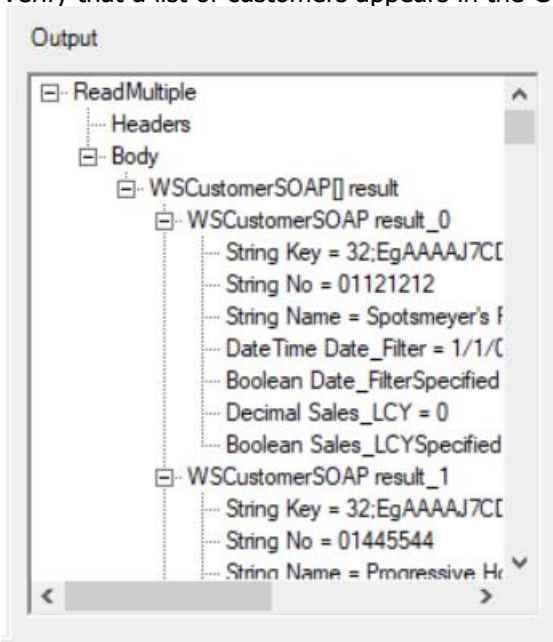
5) Paste the end-point to WS Studio:

6) Click the **Get** button
7) Choose the ReadMultiple method
8) Add an * to the **No.** String Criteria
9) Add a **0** to the **setSize**



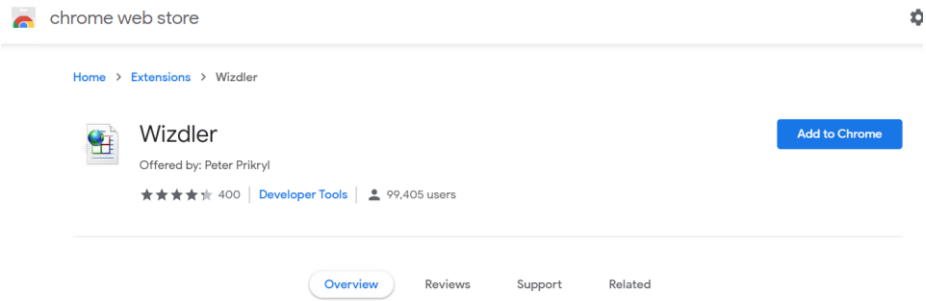
10) Click **Invoke**

11) Verify that a list of customers appears in the **Output** window:



Alternately:

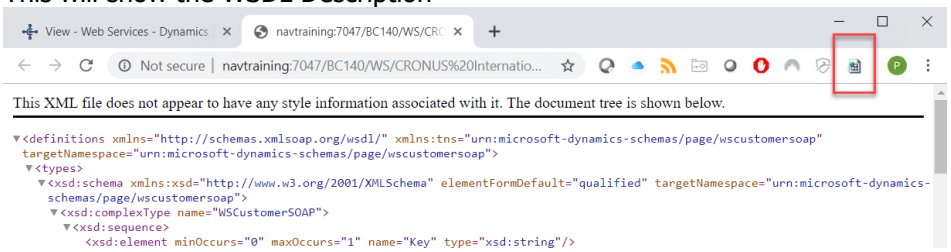
12) Open a Google Chrome or a Firefox browser
13) Search for the **Wizdler** extension
14) Add the extension to the browser



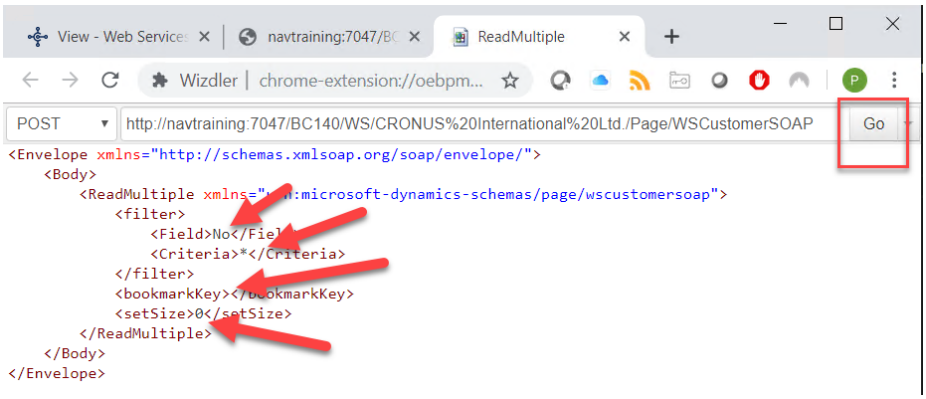
15) Switch to the web client and copy the SOAP URL end point for **WSCustomerSOAP**

16) Click on the end-point in the Chrome address bar

17) This will show the WSDL Description

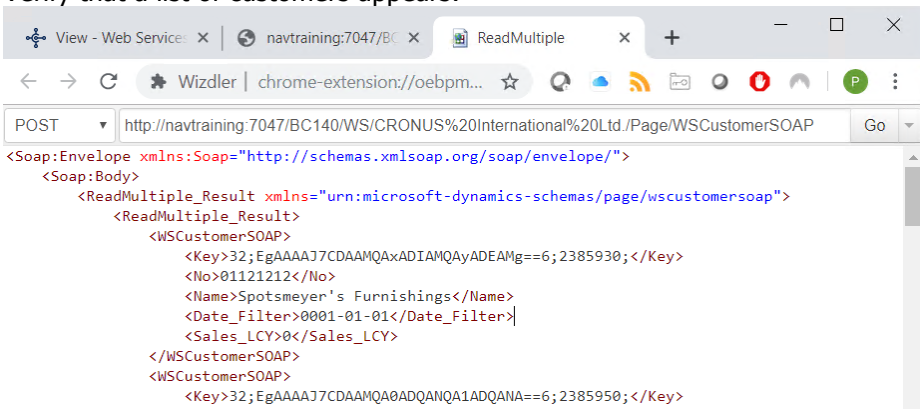


18) Click the **Wizdler** button
19) Select the **ReadMultiple** method
20) Replace the [string] in the **Field** tag with **No**
21) Replace the [string] in the **Criteria** tag with **No**
22) Remove the [string?] in the **bookmarkKey** tag
23) Replace the [int] in the **setSize** tag with **0**



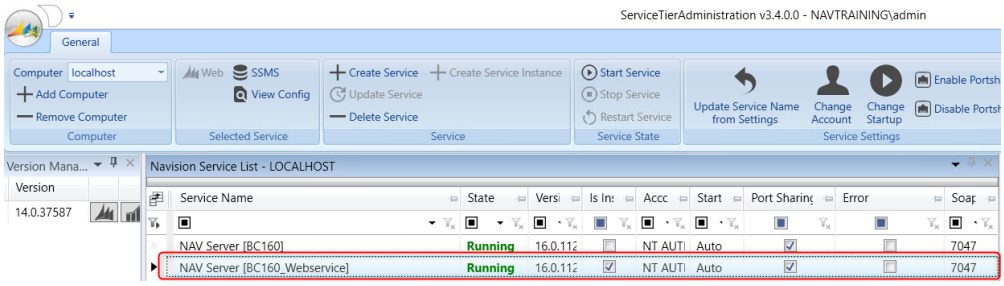
24) Click **Go**

25) Verify that a list of customers appears:

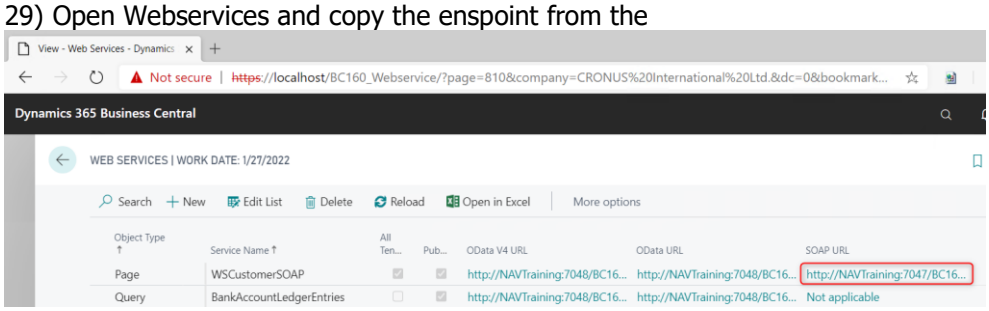


Alternately

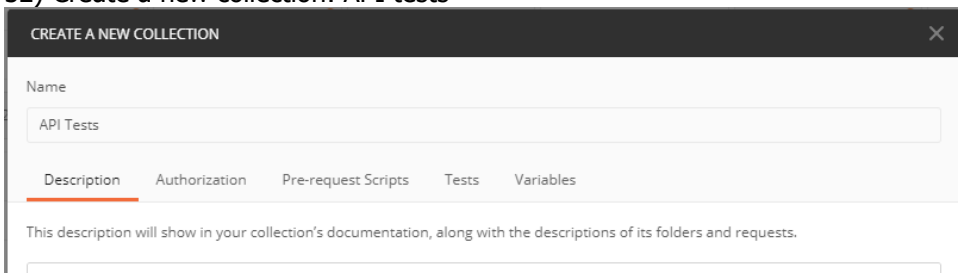
26) Open the Service Tier Administration tool and start the NAV Server BC160_Webservice:



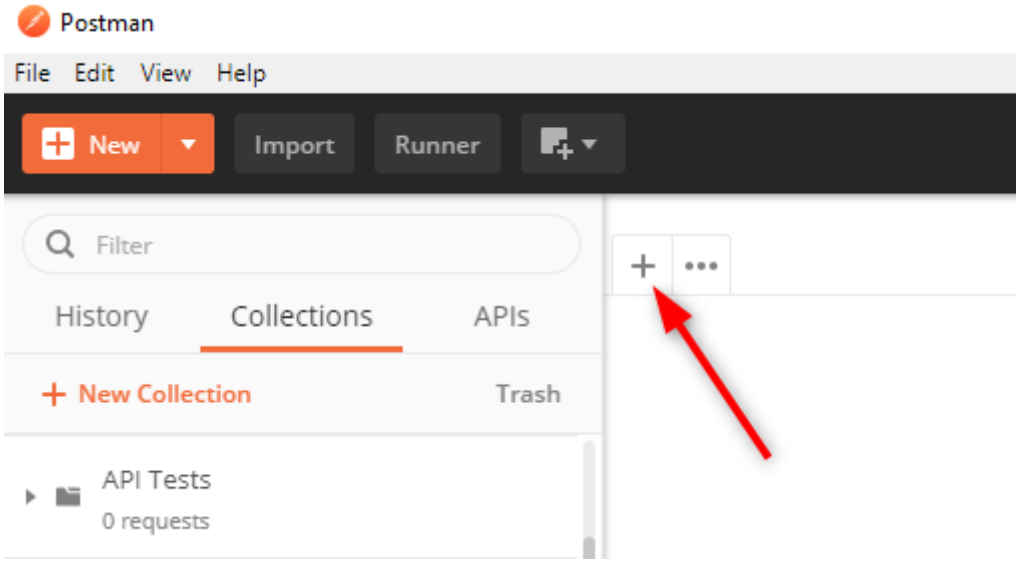
27) Close the Service Tier Administration tool
28) Then open the BC client and point to the new service tier:
    https://navtraining/BC160_NavUser
29) Open Webservices and copy the enspoint from the



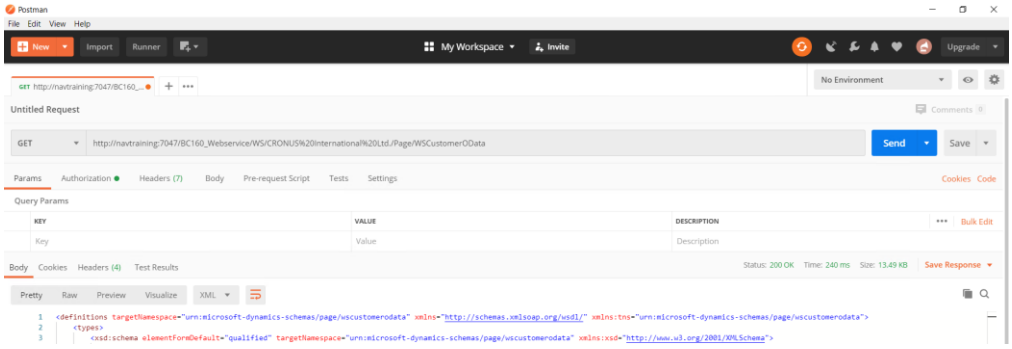
30) Download Postman for Google from web page
    https://www.postman.com/
31) Open Postman
32) Create a new collection: API tests



33) Click the API collection and add a new Request by clicking the +



34) In the new request, paste the endpoint into the request URL and click **Send**:


Postman
File Edit View Help
New
Import
Runner
My Workspace
Invite
Upgrade
GET http://navtraining:7047/BC160_...
No Environment
Untitled Request
Comments 0
GET
http://navtraining:7047/BC160_Webservice/WS/CRONUS%20International%20Ltd./Page/WSCustomerOData
Send
Save
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 240 ms
Size: 13.49 KB
Save Response
Pretty
Raw
Preview
Visualize
XML
1 <definitions targetNamespace="urn:microsoft-dynamics-schemas/page/wscustomerodata" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:microsoft-dynamics-schemas/page/wscustomerodata">
2 <types>
3 <xsd:schema elementFormDefault="qualified" targetNamespace="urn:microsoft-dynamics-schemas/page/wscustomerodata" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
35) Verify that a result shows up in the **Body** section

Lab 7: Test the OData end-point in a browser

Prerequisite to this lab:

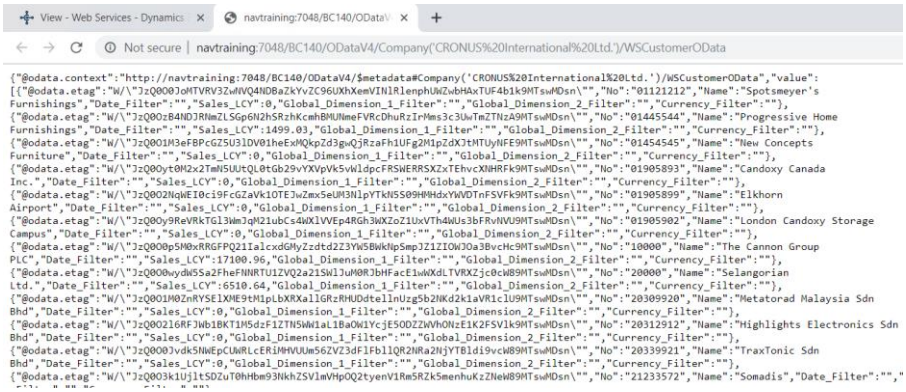
- Previous labs must be completed

High-Level steps:

- Copy the OData v. 4 endpoint from the OData web service
- Test it in a web browser

Detailed steps:

1) Open the web-client
2) Click on the OData v. 4 endpoint from the OData web service
3) Verify that a json list of customers appears:



{"@odata.context":"http://navtraining:7048/BC140/ODataV4/$metadata#Company('CRONUS%20International%20Ltd.')/WSCustomerOData","value":
[{"@odata.etag":"W/\"JzQ0O0JoMTVRV3ZwNVQ4NDBaZkYvZC96UXhXemVINlRlenphUWZwbHAxTUF4b1k9MTswMDsn\"","No":"01121212","Name":"Spotsmeyer's
Furnishings","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0OzB4NDJRNmZLSGp6N2hSRzhKcmhBMUNmeFVRcDhuRzIrMms3c3UwTmZTNzA9MTswMDsn\"","No":"01445544","Name":"Progressive Home
Furnishings","Date_Filter":"","Sales_LCY":1499.03,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O1M3eFBPcGZ5U3lDV01heExMQkpZd3gxQjRzaFh1UFg2M1pZdXJtNTUyNFE9MTswMDsn\"","No":"01454545","Name":"New Concepts
Furniture","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0Oyt0M2x2TmN5UUtQL0tGb29vYXVpVk5vWldpcFRSWERRSXZxTEhvcXNHRFk9MTswMDsn\"","No":"01905893","Name":"Candoxy Canada
Inc.","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O2NqWEI0ci9FcGZaVk1OTEJwZmx5eUM3NlpYTkN5S09HMHdxYWVDTnFSVFk9MTswMDsn\"","No":"01905899","Name":"Elkhorn
Airport","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0Oy9ReVRkTGl3WmJqM21ubCs4WXlVVEp4RGh3WXZoZ1UxVTh4WUs3bFRvNVU9MTswMDsn\"","No":"01905902","Name":"London Candoxy Storage
Campus","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O0pS M0xRRGFPQ21IalcxdGMyZzdtd2Z3YW5BWkNpSmpJZ1ZIOWJOa3BvcHc9MTswMDsn\"","No":"10000","Name":"The Cannon Group
PLC","Date_Filter":"","Sales_LCY":17100.96,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O0wydW5Sa2FheFNNRTU1ZVQ2a21Sw1JuM0RJbHFacE1wWXdLTVRXZjc0cW89MTswMDsn\"","No":"20000","Name":"Selangorian
Ltd.","Date_Filter":"","Sales_LCY":6510.64,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O1M0ZnRYSElXME9tM1pLbXRXallGRzRHUDdtellnUzg5b2NKd2k1aVR1clU9MTswMDsn\"","No":"20309920","Name":"Metatorad Malaysia Sdn
Bhd","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O2l6RFJWb1BKT1M5dzF1ZTN5WW1aL1BaOW1YcjE5ODZZWVhONzE1K2FSV1k9MTswMDsn\"","No":"20312912","Name":"Highlights Electronics Sdn
Bhd","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O0JvdkNWEpCUwRLcERiMHVUUmS6ZV23dF1Fb1lQR2NRa2NjYTBldi9vcW89MTswMDsn\"","No":"20339921","Name":"TraxTonic Sdn
Bhd","Date_Filter":"","Sales_LCY":0,"Global_Dimension_1_Filter":"","Global_Dimension_2_Filter":"","Currency_Filter":""},
{"@odata.etag":"W/\"JzQ0O3k1UjltSDZuT0hHbm93NkhZSVlmVHpOQ2tyenV1Rm5RZk5menhuKzZNeW89MTswMDsn\"","No":"21233572","Name":"Somadis","Date_Filter":""

4) If only part of the dataset is needed e.g. the **No** and **Name** fields, it is possible to add this to the end of the end-point:

   ?$select=No,Name

5) Any filtering can be made by adding ?$filter=No eq '10000'

Lab 8: Test the API end-point in a browser

Prerequisite to this lab:

- Previous labs must be completed

High-Level steps:

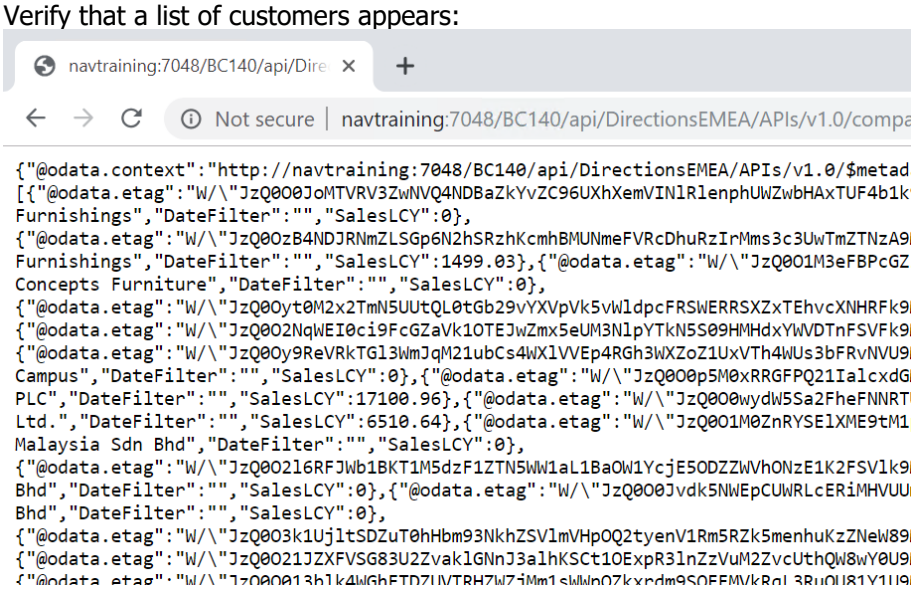
- Copy the OData v. 4 endpoint from the API web service
- Test it in a web browser

Detailed steps:

1) Copy the OData v. 4 endpoint from the API web service in the web client
2) Open a web browser
3) Paste the endpoint into the address bar.
4) Change the endpoint to be
5) http://server:port/instance/api/v1.0/companies

   This will give you a list of all companies in the database
6) Next copy the company id and add to the endpoint so that iw will look like this:
   http://server:port/instance/api/v1.0/Api Publisher/API Group/companies(id)/API Name
   and example could be:
   http://navtraining:7048/BC140/api/DirectionsEMEA/APIs/v1.0/companies(44da4ef3-d237-45cb-824a-527aa5e69cd9)/WSCustomers

7) Verify that a list of customers appears:



8) Paste the endpoint into a text document to save it for later

Lab 9: Consume the SOAP end-point in AL

Prerequisite to this lab:

- Previous labs must be completed

High-Level steps:

- Create a new report named **rep50100_ConsumeSOAPWS.al**
- Utilize the commands:
  - HttpClient
  - HttpRequestMessage
  - HttpResponseMessage
  - HttpContent
  - XmlReadOptions
  - XmlDocument
  - XmlNode

  To consume the SOAP webservice and show an message or an error showing the XML Document

Detailed steps:

1) Open VS Code
2) Right-Click the **Reports** folder and select New File:
3) Name the file **rep50100_ConsumeSOAPWS.al** and press **[Enter]**
4) Enter **trep** in the first line and select **treport**
5) Enter **50100** as the **ID** and "**DIR Consume SOAP WS"** as the **name**
6) Add the **Caption** property and set it to **'Consume SOAP WS'**
7) Add the **ProcessingOnly** property and set it to **true**
8) Add the **UseRequestPage** property and set it to **false**
9) Add the **UsageCategory** property and set it to **ReportsAndAnalysis**
10) Delete the **dataset** section
11) Delete the **requestpage** section
12) Delete the **var** section
13) Add a trigger by typing **ttr** on the last line before the end bracket (})
14) Select **ttrigger** and press enter
15) Delete OnWhat, press Ctrl+[Space] and select OnInitReport()
16) Add the following local variables:
    a. XMLText: Text;
    b. HttpContent: HttpContent;
    c. HttpRequestMessage: HttpRequestMessage;
    d. HttpHeaders: HttpHeaders;
    e. HttpClient: HttpClient;
    f. Url: Text;
    g. HttpResponse: HttpResponseMessage;
    h. XMLoptions: XmlReadOptions;
    i. XMLDoc: XmlDocument;
    j. XmlNodeList: XmlNodeList;
    k. XmlNode: XmlNode;
    l. TempCust: Record Customer temporary;
    m. BalanceLCY: Decimal;

17) In the trigger, create an XML Requestmessage:

18) XMLText := '<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">'+

Add these lines

```
'  <soap:Body>'+
'   <ReadMultiple xmlns="urn:microsoft-dynamics-schemas/page/
wscustomersoap">'+
'     <filter>'+
'       <Field>No</Field>'+
'       <Criteria />'+
'     </filter>'+
'     <bookmarkKey />'+
'     <setSize>0</setSize>'+
'   </ReadMultiple>'+
'  </soap:Body>'+
'</soap:Envelope>';
```

19) Verify that the XMLtext looks like this:

```
begin
    XMLText :=  '<soap:Envelope xmlns:soap="http://sch
                '  <soap:Body>'+
                '   <ReadMultiple xmlns="urn:microsoft
                '     <filter>'+
                '         <Field>No</Field>'+
                '         <Criteria />'+
                '     </filter>'+
                '     <bookmarkKey />'+
                '     <setSize>0</setSize>'+
                '   </ReadMultiple>'+
                '  </soap:Body>'+
                '</soap:Envelope>';

end;
```

20) Add the URL:
Url :=
'http://navtraining:7047/BC140/WS/CRONUS%20International%20Ltd./
Page/WSCustomerSOAP';

21) Prepare the HttpRequestMessage and get the HttpResponseMessage:

```
HttpRequestMessage.SetRequestUri(URL);
HttpRequestMessage.Method('POST');
HttpContent.WriteFrom(XMLtext);
HttpContent.GetHeaders(HttpHeaders);
HttpHeaders.Remove('Content-Type');
HttpHeaders.Add('Content-Type', 'application/xml;charset=utf-8');
HttpRequestMessage.Content := HttpContent;
HttpRequestMessage.GetHeaders(HttpHeaders);
HttpHeaders.Add('SOAPAction', 'urn:microsoft-dynamics-schemas/page/WSCustomerSOAP');
HttpClient.UseWindowsAuthentication('Admin', '1<3VScode', 'NavTraining');
HttpClient.send(HttpRequestMessage, HttpResponse);
```

22) If the web service requires basic authentication, another four variables are required:

```
AuthTxt : Text;
UserName : Text;
Password : Text;
TempBlob : Record TempBlob;
```

23) And instead of the UseWindowsAuthentication, this code is required:

```
// Basic authentication
if UserName <> '' then begin
    AuthTxt := strsubstno('%1:%2', UserName, Password);
    TempBlob.WriteAsText(AuthTxt, TextEncoding::Windows);
    HttpHeaders.Add('Authorization', StrSubstNo('Basic %1', TempBlob.ToBase64String()));
end;
```

24) Handle the result and show an error with the XML Document:

```
if not HttpResponse.IsSuccessStatusCode() then
    error(format(HttpResponse.HttpStatusCode()) + ' , ' + HttpResponse.ReasonPhrase())
else begin
    clear(XMLtext);
    HttpResponse.Content().ReadAs(XMLtext);
    XMLoptions.PreserveWhitespace := true;
    XmlDocument.ReadFrom(xmlText, XMLoptions, XMLDoc);
    error('%1', XMLDoc);
```

25) Verify that a list of customers appears:


```
<?xml version="1.0" encoding="utf-16"?>
<Soap:Envelope
xmlns:Soap="http://schemas.xmlsoap.org/soap/envelope/">
<Soap:Body>
<ReadMultiple_Result xmlns="urn:microsoft-dynamics-
schemas/page/wscustomersoap">
<ReadMultiple_Result>
<WSCustomerSOAP>
<Key>32;EgAAAAJ7CDAAMQAxADIAMQAyADEAMg==6;238593
0;</Key>
<No>01121212</No>
<Name>Spotsmeyer's Furnishings</Name>
<Date_Filter>0001-01-01</Date_Filter>
<Sales_LCY>0</Sales_LCY>
</WSCustomerSOAP>
<WSCustomerSOAP>
<Key>32;EgAAAAJ7CDAAMQA0ADQANQA1ADQANA==6;23859
50;</Key>
<No>01445544</No>
<Name>Progressive Home Furnishings</Name>
<Date_Filter>0001-01-01</Date_Filter>
<Sales_LCY>1499.03</Sales_LCY>
</WSCustomerSOAP>
<WSCustomerSOAP>
<Key>32;EgAAAAJ7CDAAMQA0ADUANAA1ADQANQ==6;23859
```

Lab 10: Consume the OData end-point in AL

Prerequisite to this lab:

- Previous labs must be completed

High-Level steps:

- Create a new report named **rep50101_ConsumeODataWS.al**
- Utilize the commands:
  - HttpClient
  - HttpResponseMessage
  - jsonToken
  - JsonArray

  To consume the OData webservice and show a message or an error showing the Json response

Detailed steps:

1) Open VS Code
2) Right-Click the **Reports** folder and select New File:
3) Name the file **rep50101_ConsumeODataWS.al** and press **[Enter]**
4) Enter **trep** in the first line and select **treport**
5) Enter **50101** as the **ID** and "**DIR Consume OData WS"** as the **name**
6) Add the **Caption** property and set it to **'Consume OData WS'**
7) Add the **ProcessingOnly** property and set it to **true**
8) Add the **UseRequestPage** property and set it to **false**
9) Add the **UsageCategory** property and set it to **ReportsAndAnalysis**
10) Delete the **dataset** section
11) Delete the **requestpage** section
12) Delete the **var** section
13) Add a trigger by typing **ttr** on the last line before the end bracket (}}
14) Select **ttrigger** and press enter
15) Delete **OnWhat**, press Ctrl+[Space] and select **OnInitReport()**
16) Add the following local variables:
    a. HttpClient: HttpClient;
    b. ResponseMessage: HttpResponseMessage;
    c. JsonToken: JsonToken;
    d. JsonObject: JsonObject;
    e. JsonArray: JsonArray;
    f. JsonText: text;
    g. Customer: Record Customer temporary;
    h. Url: Text;
17) In the trigger, add the Url in the first line:
    Url:='http://NAVTraining:7048/BC140/ODataV4/Company("CRONUS%20International%20Ltd.")/WSCustomerOData'

18) Set the authentication for windows authentication
    HttpClient.UseWindowsAuthentication('Admin', '1<3VScode', 'NavTraining')

19) Get the Httpclient and show an error if it doesn't work:

    if not HttpClient.Get(Url, ResponseMessage) then

Error('The call to the web service failed.');

if not ResponseMessage.IsSuccessStatusCode then
error('The web service returned an error message:\\' + 'Status code: %1\' + 'Description: %2', ResponseMessage.HttpStatusCode, ResponseMessage.ReasonPhrase);

20) Transfer the Responsemessage to the JsonText
ResponseMessage.Content.ReadAs(JsonText);

21) Show the Response as a message or an error.
error(JsonText);

```
report 50101 "DIR Consume OData WS"
{
    Caption = 'Consume OData WS';
    UsageCategory = ReportsAndAnalysis;
    ApplicationArea = All;
    ProcessingOnly = true;
    UseRequestPage = false;

    trigger OnInitReport()
    var
        HttpClient: HttpClient;
        HttpResponseMessage: HttpResponseMessage;
        JsonToken: JsonToken;
        JsonObject: JsonObject;
        JsonArray: JsonArray;
        JsonText: text;
        Customer: Record Customer temporary;
        Url: Text;
    begin
        Url := 'http://NAVTraining:7048/BC140/ODataV4/Company(''CRONUS%20International%20Ltd.'')/WSCustomerOData';
        HttpClient.UseWindowsAuthentication('Admin', '1<3VScode', 'NavTraining');
        if not HttpClient.Get(Url, HttpResponseMessage) then
            Error('The call to the web service failed.');
        if not HttpResponseMessage.IsSuccessStatusCode then
            error('The web service returned an error message:\\' + 'Status code: %1\' + 'Description: %2', HttpRespon
        HttpResponseMessage.Content.ReadAs(JsonText);
        error(JsonText);
        JsonText := '[' + JsonText + ']';
        if not JsonArray.ReadFrom(JsonText) then
            Error('Invalid response, expected an JSON array as root object');
        foreach jsonToken in JsonArray do begin
            JsonObject := JsonToken.AsObject;
        end;
    end;
}
```

For the Dynamics 365 Business Central sandbox:

```
report 50103 "DIR Consume OData SB WS"
{
    Caption = 'Consume OData SB WS';
    UsageCategory = ReportsAndAnalysis;
    ApplicationArea = All;
    ProcessingOnly = true;
    UseRequestPage = false;

    trigger OnInitReport()
    var
        HttpClient: HttpClient;
        HttpResponseMessage: HttpResponseMessage;
        HttpRequestMessage: HttpRequestMessage;
        HttpContent: HttpContent;
        HttpHeaders: HttpHeaders;
        JsonToken: JsonToken;
        JsonObject: JsonObject;
        JsonArray: JsonArray;
        JsonText: text;
        Customer: Record Customer temporary;
        Url: Text;
        UserName: Text;
        Password: Text;
        AuthTxt: Text;
        TempBlob: Record TempBlob;

    begin
        Url := 'https://api.businesscentral.dynamics.com/v1.0/Tenant/Sandbox/ODataV4/Company(''CRONUS%26

        UserName := 'user';
        Password := 'Web service key';
        HttpRequestMessage.SetRequestUri(URL);
        HttpRequestMessage.Method('GET');
        HttpContent.GetHeaders(HttpHeaders);
        HttpHeaders.Remove('Content-Type');
        HttpHeaders.Add('Content-Type', 'application/xml;charset=utf-8');
        HttpRequestMessage.GetHeaders(HttpHeaders);
        HttpHeaders.Add('SOAPAction', 'urn:microsoft-dynamics-schemas/page/WSCustomerSOAP');
        // Basic authentication
        if UserName <> '' then begin
            AuthTxt := strsubstno('%1:%2', UserName, Password);
            TempBlob.WriteAsText(AuthTxt, TextEncoding::Windows);
            HttpHeaders.Add('Authorization', StrSubstNo('Basic %1', TempBlob.ToBase64String()));
        end;
        HttpClient.send(HttpRequestMessage, HttpResponseMessage);

        HttpResponseMessage.Content.ReadAs(JsonText);
        error(JsonText);
        JsonText := '[' + JsonText + ']';
        if not JsonArray.ReadFrom(JsonText) then
            Error('Invalid response, expected an JSON array as root object');
        foreach jsonToken in JsonArray do begin
            JsonObject := JsonToken.AsObject;
        end;


    end;

}
```

Lab 11: Consume the API end-point in AL

Prerequisite to this lab:

- Previous labs must be completed

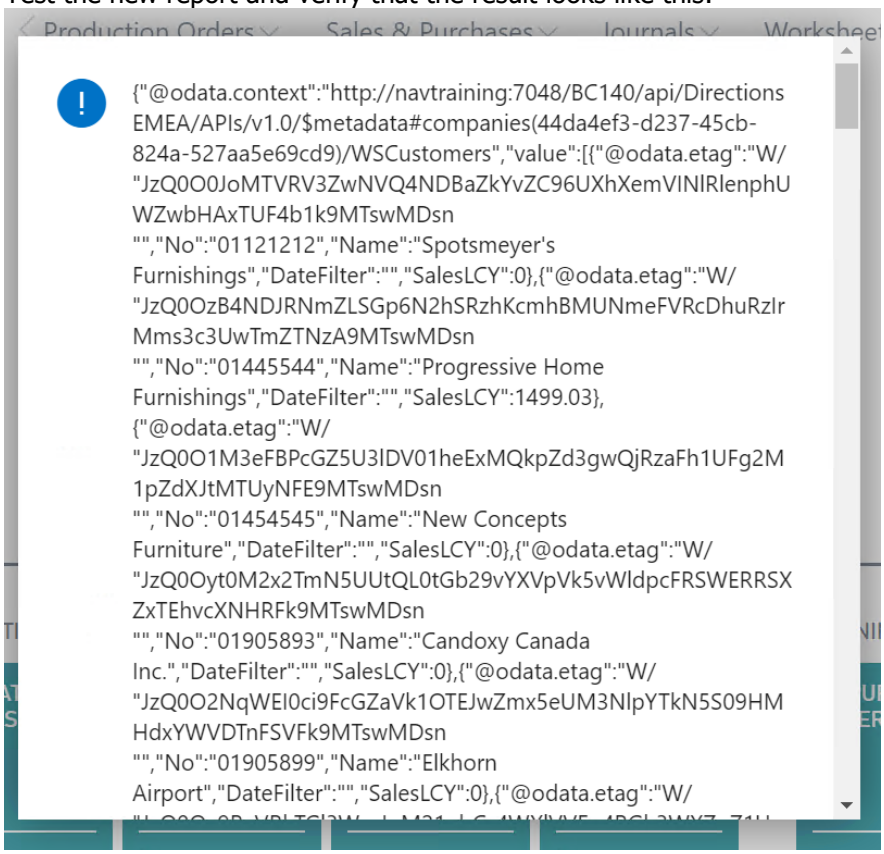
High-Level steps:

- Create a new report named **rep50103_ConsumeAPIWS.al** as a copy of 50102
- Change the endpoint to the saved end-point from lab 8

  Consume the API webservice and show a message or an error showing the Json response

Detailed steps:

1) Open VS Code
2) Right-Click the **rep50102** file in the **Reports** folder and select **copy**
3) Right-Click the **Reports** folder and select **paste**
4) Name the file **rep50103_ConsumeAPIWS.al**
5) Enter **50103** as the **ID** and "**DIR Consume API WS"**
6) Change the caption to "**DIR Consume API WS"** as the **name**
7) Copy the end-point from lab 8 and paste it into the URL
8) Press Ctrl+F5 to publish
9) Test the new report and verify that the result looks like this:



Resources

Links that can help:

http://www.tharangac.com/2018/11/getting-started-with-dynamics-365.html

https://www.kauffmann.nl/2017/06/24/al-support-for-rest-web-services/

https://community.dynamics.com/nav/b/andreysnavblog/posts/how-to-obtain-data-from-nav-standard-apis

https://www.crt-insights.com/2019/05/16/publish-and-test-soap-odata-web-services-in-nav/

https://saurav-nav.blogspot.com/2018/01/microsoft-dynamics-nav-2018-api-part-1.html

https://saurav-nav.blogspot.com/2018/01/microsoft-dynamics-nav-2018-api-part-2_22.html

https://saurav-nav.blogspot.com/2018/01/microsoft-dynamics-nav-2018-api-part-3.html
https://docs.microsoft.com/da-dk/dynamics365/business-central/dev-itpro/developer/devenv-restapi-overview