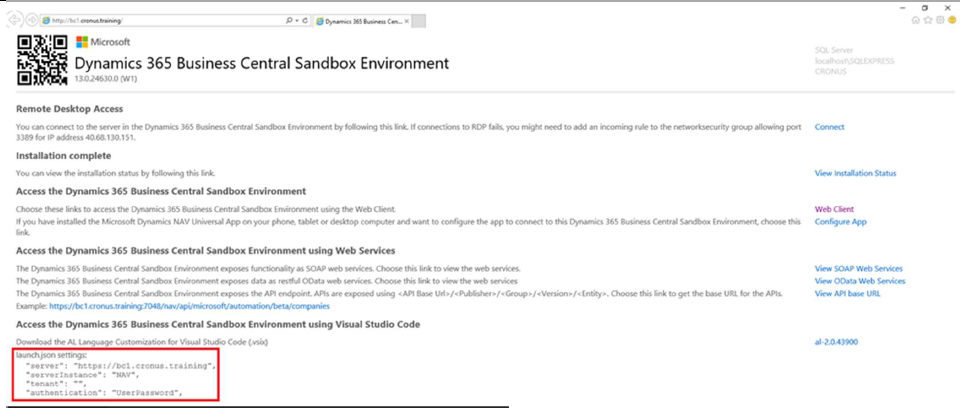
**Data exchange with AL language:**

1. **Creating new App:**

* Start VS Code
* Press “F1” (“Command Palette”)
* Enter “AL: Go!”
* Enter path (e.g. “…\Documents\Task4”)
* Select the latest target platform (4.0)
* In the dropdown list, pick “Your own server”
* When it asks you for credentials, don’t log in, just press ESC
* Set up the “launch.json”
* Make sure the settings are the same as the ones from your own landing page
* You can find those settings when you click “landing page” on the desktop of the VM.





Set up the “app.json”:

* Change “name” to “Task <Number of Task>”
* Change “publisher” to “<<your Name and Surname>>”
* Add feature “TranslationFile”



Download the Symbols:

* Use the Command Palette (F1) to search for “AL: Download symbols”

If necessary:

* Provide username and password from boarding pass



Delete “Hello world.al” file.

**Task 3.**

Creating records in Business Central through API in Selected Company

**Sources:**

1. Your BC API URL:

https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies

1. Items API:

<https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(CompanyID)/items>

1. Customers API:
2. https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(CompanyID)/customers
3. GitHub Repository with source code: github.com/abaludin/RecordCreator

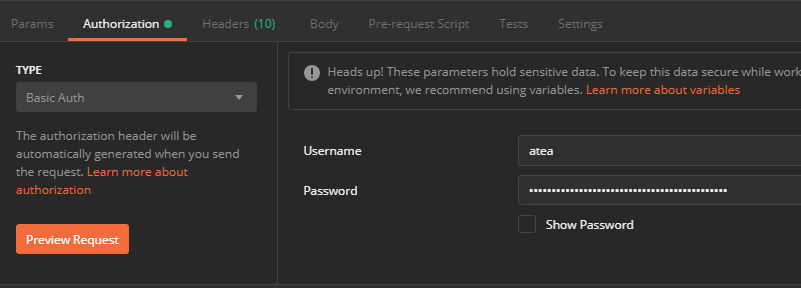
**Step 1**

First of all we need to check Business central standard APIs.

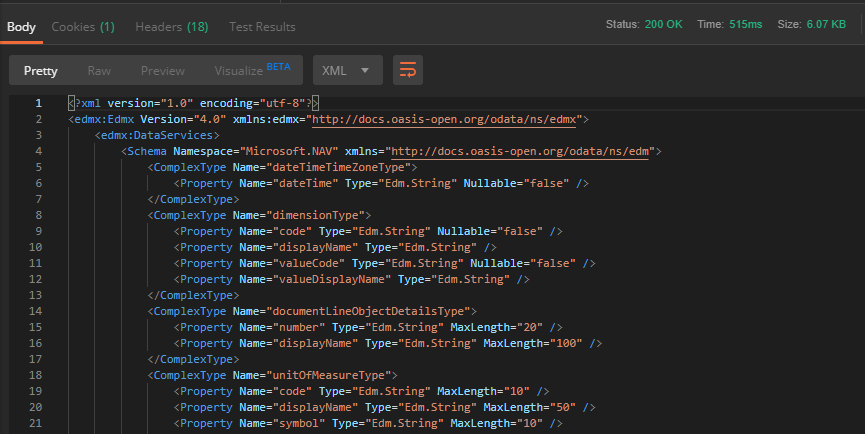
1. Open Postman, select GET method and use <https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/$metadata> as URL.

Open your user card in Business central, Create Web Access Key. Use it as password.

Open Authorization tab, input your Username and password (web Access Key)



Press Send. You’ll see Status 200 and list of all standard APIs:



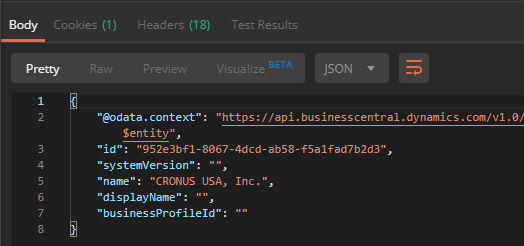
1. Test Companies API the same way:

<https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies>

Copy your Company ID and test next API:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e))

You should get response about single company:



Get Items list:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/items](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/items)

Vendors list:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/vendors](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/vendors)

Customers list:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/customers](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/customers)

now more complex:

Sales Invoices:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/salesInvoices](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/salesInvoices)

And… Sales Invoices with lines:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/salesInvoices?$expand=salesInvoiceLines](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/salesInvoices?$expand=salesInvoiceLines)

2 APIs in one.

By the way, to get single invoice with lines use

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/salesInvoices(<<INVOICE ID>>)?$expand=salesInvoiceLines](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/salesInvoices(%3c%3cINVOICE%20ID%3e%3e)?$expand=salesInvoiceLines)

Filters

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/items?$filter=unitPrice%20gt%20100](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/items?$filter=unitPrice%20gt%20100)

This request returns you all items from database which has Unit Price greater then 100.

%20 used instead of spaces

gt - means greater then

Also you can use:

eq - equal

ne - not equal

lt - less then

Select single token from response:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/name](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/name)

Select only text of token:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/name/$value](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/name/$value)

Now try with other methods. Business Central APIs supports next:

GET - read records

POST - insert record

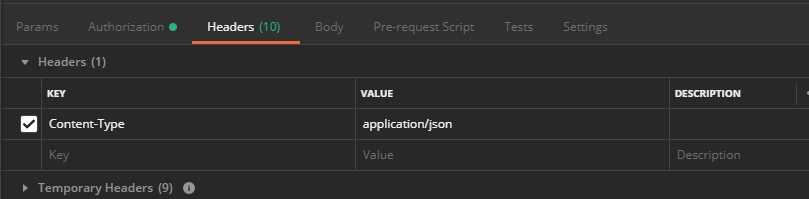
PATCH - modify record

DELETE - delete record

Obtain customers again:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/customers](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/customers)

Change method from GET to POST, click Headers and create Content-Type header with Application/json value:



Click body -> raw and insert

## {

## "displayName": "My customer from API",

## "type": "Company",

## "address": {

## "street": "kineapolis",

## "city": "Antwerp",

## "countryLetterCode": "BE"

## }

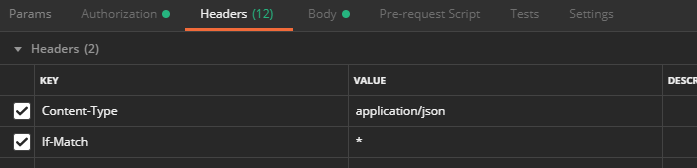
## }

Press Send – you’ll get Status 201 Created

Change method to PATCH, copy your created customer ID to url:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(<<COMPANYID>>)/customers(<<Customer ID>>)](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies(%3c%3cCOMPANYID%3e%3e)/customers(%3c%3cCustomer%20ID%3e%3e))

and add If-Match header with value = \*



Change body as

## {

## "displayName": "My modified customer from API",

## }

Send request and check that customer name is changed.

Remove Body, Change method to DELETE, press Send - > customer will be delete.

Now you know some basics about Business Central APIs and we can use them to create records from itself using AL code and APIs

**Step 2**

Create Setup table and Page (card type) with next fields:

* Entry no. (As you prefer) \*key
* API URL (Text) = https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/v1.0/companies
* Username (text)
* Web Access Key (text)

Table code:

<https://github.com/ABaludin/RecordCreator/blob/master/src/table/Tab50130.AB_RecordCreatorSetup.al>

Page code:

<https://github.com/ABaludin/RecordCreator/blob/master/src/page/Pag50132.AB_RecordCreatorSetup.al>

**Step 3**

Create management codeunit

* Create function to get Setup values:

## local procedure GetSetup()

## begin

## Setup.Get();

## Setup.TestField("API URL");

## Setup.TestField(Username);

## Setup.TestField("Web Access Key");

## end;

* Create function to get API authorization token:

Username + : + Web Access Key – converted to base64 format (using new codeunit Base 64 Convert)

## local procedure AuthorizationToken(): Text

## var

## Base64Convert: Codeunit "Base64 Convert";

## begin

## exit(Base64Convert.ToBase64(Setup.Username + ':' + Setup."Web Access Key"));

## end;

* Create function for testing connection to api (using universal code for HTTP request of course):

## procedure TestConnection()

## var

## Client: HttpClient;

## ResponseMessage: HttpResponseMessage;

## ResponseText: Text;

## begin

## GetSetup();

## Client.DefaultRequestHeaders().Add('Authorization', 'Basic ' + AuthorizationToken());

## Client.DefaultRequestHeaders().Add('Accept', 'application/json');

## if not Client.Get(Setup."API URL", ResponseMessage) then

## Error(Text001\_Err);

## ResponseMessage.Content().ReadAs(ResponseText);

## if not ResponseMessage.IsSuccessStatusCode() then

## error(Text002\_Err, ResponseMessage.HttpStatusCode(), ResponseText);

## Message(Text003\_Msg);

## end;

## var

## Setup: Record "AB\_RecordCreator Setup";

## Text001\_Err: Label 'Service inaccessible';

## Text002\_Err: Label 'The web service returned an error message:\ Status code: %1\ Description: %2';

## Text003\_Msg: Label 'Connection OK!';

And place an action on Setup page to run Test connection:

## action(TestConnection)

## {

## ApplicationArea = All;

## Image = Interaction;

## Caption = 'Test Connection';

## trigger OnAction()

## var

## RecordCreatorMgt: Codeunit "AB\_Record Creation Management";

## begin

## RecordCreatorMgt.TestConnection();

## end;

## }

Here you can publish your intermediate solution, fill values on Setup table and Test connection to APIs.

**Step 4**

Now we prepare a page for record creation. We will fill some values on our page, generate Request body from them and POST it to standard API. Also we need a company selection to choose company where we want to create record. Intercompany posting.

We need page without Source table, just variables:

* Company Name (TableRelation = Company.Name)
* Record Type – option or enum with Item and customer values
* Description

Page Source code:

<https://github.com/ABaludin/RecordCreator/blob/master/src/page/Pag50130.AB_DataCreator.al>

Enum for Record type:

<https://github.com/ABaludin/RecordCreator/blob/master/src/enum/Enum50130.AB_RecordTypes.al>

**Step 5**

* Make function CreateItemObject which creates JsonObject like this:

{

"displayName": "Description",

"baseUnitOfMeasure": {

"code": "PCS",

"displayName": "Piece"

}

}

## local procedure CreateItemObject(Description: Text) ItemRequest: Text

## var

## ItemObject: JsonObject;

## UoMObject: JsonObject;

## begin

## ItemRequest := '';

## ItemObject.Add('displayName', Description);

## UoMObject.Add('code', 'PCS');

## UoMObject.Add('displayName', 'Piece');

## ItemObject.Add('baseUnitOfMeasure', UoMObject);

## ItemObject.WriteTo(ItemRequest);

## end;

* Create “Create Item” function in codeunit, accepting Company ID and Description (from page from Step 4), Send item object to Items API (in selected company) with POST method. Do not forget to add Content-type header.

## local procedure CreateItem(CompanyID: Text; Description: Text)

## var

## Client: HttpClient;

## Content: HttpContent;

## Headers: HttpHeaders;

## ResponseMessage: HttpResponseMessage;

## ResponseText: Text;

## URL: Text;

## ResponseObject: JsonObject;

## ResponseToken: JsonToken;

## begin

## Client.DefaultRequestHeaders().Add('Authorization', 'Basic ' + AuthorizationToken());

## Client.DefaultRequestHeaders().Add('Accept', 'application/json');

## CompanyID := DelChr(CompanyID, '=', '{}');

## URL := Setup."API URL" + '(' + CompanyID + ')/items';

## Content.WriteFrom(CreateItemObject(Description));

## Content.GetHeaders(Headers);

## Headers.Remove('Content-Type');

## Headers.Add('Content-Type', 'application/json');

## if not Client.Post(URL, Content, ResponseMessage) then

## Error(Text001\_Err);

## ResponseMessage.Content().ReadAs(ResponseText);

## if not ResponseMessage.IsSuccessStatusCode() then

## error(Text002\_Err, ResponseMessage.HttpStatusCode(), ResponseText);

## ResponseObject.ReadFrom(ResponseText);

## ResponseObject.Get('number', ResponseToken);

## Message(Text004\_Msg, ResponseToken.AsValue().AsText(), Description);

## end;

Put the code calling CreateItem function on Data creator page.

* Check in BC that Item created

**Step 6**

Create the same functions for customers entity.

Object will be:

{

"displayName": "Description",

"type": "Company",

"address": {

"street": "Kineapolis",

"city": "Antwerp",

"countryLetterCode": "BE"

}

}

Full management codeunit code:

<https://github.com/ABaludin/RecordCreator/blob/master/src/codeunit/Cod50130.AB_RecordCreationManagement.al>

**Final Result:**

