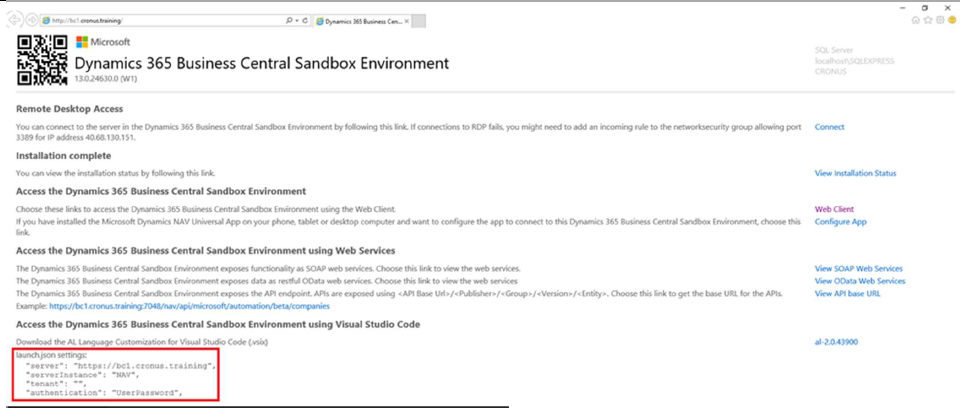
**Data exchange with AL language:**

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 5.**

Create your own API

**Sources:**

GitHub Repository with source code: github.com/abaludin/OwnAPI

**Step 1**

First of all we need data tables and pages for our API. It will be box table and box lines table. Also we need List page for boxes and card with line subform.

Create Box Header table with next fields:

field("Box No."; Code[20])

field("Box Ready"; Boolean)

field(Delivered; Boolean)

field("Shipment Date"; Date)

field("Receive Date"; Date)

field("Sales Order No."; Code[20])

{TableRelation = "Sales Header"."No." where("Document Type" = filter(Order));}

field("External partner sent"; Boolean)

field("Sent date"; Date)

Create Box Header Lines table with next fields:

field("Box No."; Code[20])

{TableRelation = "AWR\_Box Header"."Box No}

field("Box Line No."; Integer)

field(Quantity; Decimal)

field("Item No."; Code[20])

{TableRelation = Item."No."}

field(Description; Text[100])

{CalcFormula = lookup (Item.Description where("No." = field("Item No.")))}

key("Box No.", "Box Line No.")

Full tables code:

<https://github.com/ABaludin/OwnAPI/blob/master/src/table/Tab50140.AWR_BoxHeader.al>

<https://github.com/ABaludin/OwnAPI/blob/master/src/table/Tab50141.AWR_Boxline.al>

Box List page:

<https://github.com/ABaludin/OwnAPI/blob/master/src/page/Pag50142.AWR_BoxList.al>

Box Card Page:

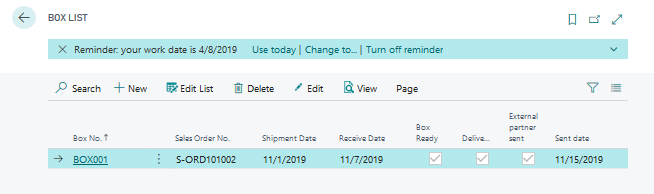
<https://github.com/ABaludin/OwnAPI/blob/master/src/page/Pag50141.AWR_BoxCard.al>

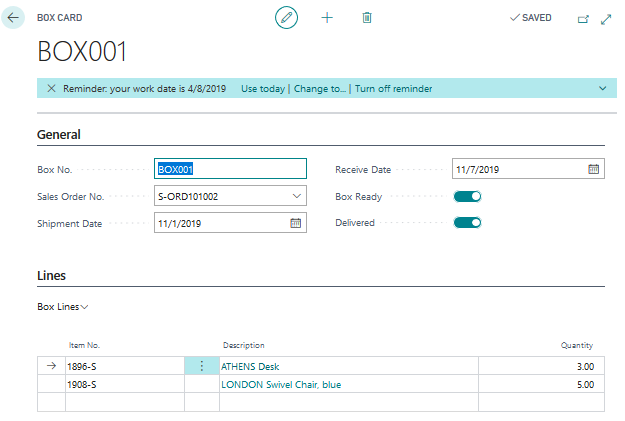
Box Lines subform:

<https://github.com/ABaludin/OwnAPI/blob/master/src/page/Pag50140.AWR_BoxLines.al>

**Step 2**

* Now publish this code, open box list and create some data in it:





**Step 3**

Create new page of API type (use tpage snippet)

API pages has a set of exclusive mandatory options. Add next:

* APIGroup = 'ndt19';
* APIPublisher = 'your name';
* APIVersion = 'v1.0';
* EntityName = 'box';
* EntitySetName = 'boxes';
* DelayedInsert = true;

Use “Box Header” table as source table

Add next fields to page:

field(boxNo; "Box No.")

field(boxReady; "Box Ready")

field(delivered; Delivered)

field(shipmentDate; "Shipment Date")

field(receiveDate; "Receive Date")

Now we need to add box lines as page part. See – we adding not card part page to API page and it will works as part of API! Page part also needs EntityName and EntitySetName properties:

## part(boxLines; "AWR\_Box Lines")

## {

## ApplicationArea = All;

## Caption = 'boxLines';

## EntityName = 'boxLine';

## EntitySetName = 'boxLines';

## SubPageLink = "Box No." = FIELD("Box No.");

## }

Publish your extension

**Step 4**

Call your API with next link:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY ID)/boxes](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY%20ID)/boxes)

Pay attention that you see only box headers, without lines. To get boxes with lines call:

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY ID)/boxes?$expand=boxLines](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY%20ID)/boxes?$expand=boxLines)



**Step 4**

Now let’s change API behavior:

Add OnAfterGetRecord trigger to API page:

## trigger OnAfterGetRecord()

## begin

## "External Partner Sent" := true;

## "Sent date" := Today();

## Modify(false);

## end;

And add OnOpenPage trigger:

## trigger OnOpenPage()

## begin

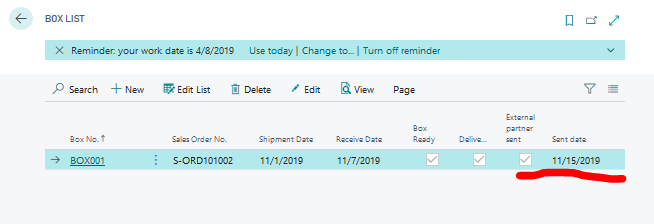
## SetRange("External partner sent", false);

## end;

Publish App again and call

[https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY ID)/boxes?$expand=boxLines](https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANY%20ID)/boxes?$expand=boxLines)

After you’ll get the data – check box list:

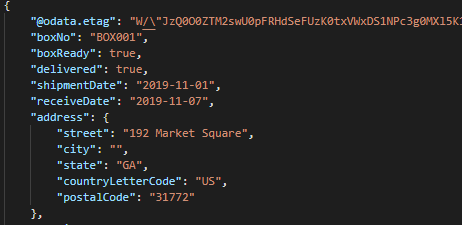


You see that Records now has “External partner sent” Boolean filled and “sent date” field updated. Call API again – you’ll get nothing in response, because now API sends only records which were not sent earlier.

**Step 5:**

And now the last step - Complex types.

Have a look on this response:



We could see address json object inside our box object. Let’s try to create this.

Create PostalAddressJSONText: Text global variable on API page.

Place it as a field in page repeater:

## field(address; PostalAddressJSONText)

## {

## ApplicationArea = All;

## }

Create new GetAddressJSON function on API page. Here I used existing standard functionality for that. You can go to definitions and investigate how it works. In fact – it’s just JSON object creating with values from Sales Header record (remember – we have “Sales Order No.” field in “Box Header” table?):

## local procedure GetAddressJSON(SalesHeader: Record "Sales Header")

## var

## GraphMgtComplexTypes: Codeunit "Graph Mgt - Complex Types";

## begin

## with SalesHeader do

## GraphMgtComplexTypes.GetPostalAddressJSON("Sell-to Address", "Sell-to Address 2",

## "Sell-to City", "Sell-to County", "Sell-to Country/Region Code", "Sell-to Post Code", PostalAddressJSONText);

## end;

And finally add some code to OnAfterGetRecord trigger. Now it will be like this:

## trigger OnAfterGetRecord()

## var

## SalesHeader: Record "Sales Header";

## begin

## SalesHeader.Get(SalesHeader."Document Type"::Order, "Sales Order No.");

## GetAddressJSON(SalesHeader);

## "External Partner Sent" := true;

## "Sent date" := Today();

## Modify(false);

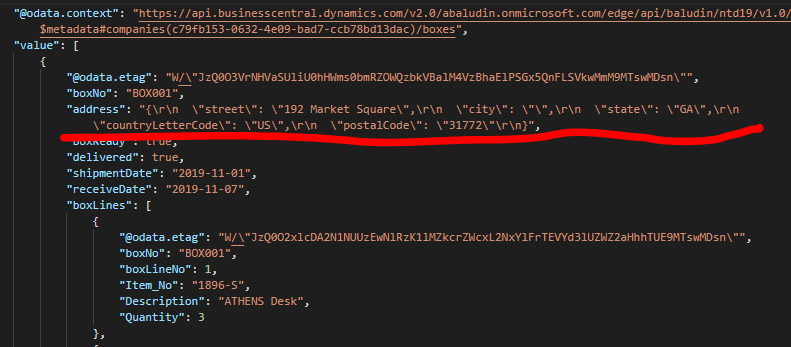
## end;

Publish your app. Do not forget to reset “External Partner Sent” mark – to get data.

Call your API again

<https://VMNAME.ZONE.cloudapp.azure.com:7048/bc/api/PUBLISHER/ntd19/v1.0/companies(COMPANYID)/boxes?$expand=boxLines>

And surprise –



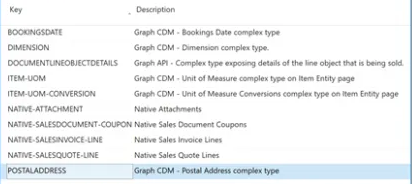
You JSON text not JSON. Just text.

Now you know what is Complex types. By default Business Central can’t show you Json objects as a value.

You need to create separate schema definition for each such object. And unfortunately, it’s possible only with on-premise environment.

All definitions stored in system table 2000000179 OData Edm Type. And it’s not allowed for cloud extensions (now).

But table contains a set of pre-defined Edm types for standard APIs. And our Postal address is one of them:



To use it – you need to set special property to your PostalAddressJSONText field:

## field(address; PostalAddressJSONText)

## {

## ApplicationArea = All;

## ODataEDMType = 'POSTALADDRESS';

## }

Publish your App again, call your API and finally:

