Build Your Custom API

Contents

[Create a new workspace 2](#_Toc2141472)

[Create objects 2](#_Toc2141473)

[Create a table 2](#_Toc2141474)

[Solution: table 50101 Vehicle 3](#_Toc2141475)

[Create the API page 5](#_Toc2141476)

[Solution: page 50101 "Vehicle Entity" 6](#_Toc2141477)

[Create the UI page 8](#_Toc2141478)

[Solution: Page 50102 Vehicles.al 8](#_Toc2141479)

[Test the custom API 10](#_Toc2141480)

[Publish the custom API 10](#_Toc2141481)

[Obtain Business Central web services basic authentication credentials 10](#_Toc2141482)

[Obtain Business Central tenant ID and your sandbox URL 10](#_Toc2141483)

[Test the extension in Business Central 11](#_Toc2141484)

# Create a new workspace

In this exercise you create a new workspace. Name it “Custom API”. After you create your workspace, remove the HelloWorld.al file and configure the launch.json file to connect to your Business Central cloud sandbox instance.

If you are not familiar with this procedure, there are step-by-step instructions in the **General – Hands-on – Beginner instructions** document.

# Create objects

In this exercise, you’ll create a table that represents your database schema. Then, you’ll create an API page to allow reading and writing the data from your table. Finally, you’ll create a simple UI page that you can use to validate data changes.

## Create a table

1. In Visual Studio Code, create a new file and name it **Table 50101 Vehicle.al**.
2. In the file, declare table 50101 Vehicle.
3. Declare the **fields** section, and in it declare the following fields:

|  |  |  |
| --- | --- | --- |
| ID | Name | Type |
| 1 | Code | Code[10] |
| 2 | Make | Text[50] |
| 3 | Model | Text[50] |
| 4 | Year | Integer |
| 5 | Registration Plate | Code[20] |
| 8000 | Id | Guid |
| 8001 | Last Modified Date Time | DateTime |

1. Fields **Id** and **Last Modified Date Time** must not be editable.
2. For the **Registration Plate** field, define the OnValidate trigger. Inside, make sure that the year is not lower than 2000, and not higher than current year.

trigger OnValidate();

var

ErrorTooOld: Label 'The vehicle is too old.';

ErrorNotManufactured: Label 'The vehicle has not been manufactured yet. Please come back in %1 and configure it.';

begin

if Year < 2000 then

Error(ErrorTooOld);

if Year > Date2DMY(Today(), 3) then

Error(ErrorNotManufactured, Year);

end;

1. Declare the **keys** section, and in it declare a primary key over the **Code** field. Make the primary key clustered.
2. Define the OnInsert trigger for the table. Inside, it Id is not yet assigned, assign a new GUID to the **Id** field, and then assign the current date and time to the **Last Modified Date Time** field.

trigger OnInsert();

begin

if (IsNullGuid(Id)) then

Id := CreateGuid();

"Last Modified Date Time" := CurrentDateTime();

end;

1. Define the OnModify trigger for the table. Inside, assign the current date and time to the **Last Modified Date Time** field.

trigger OnModify();

begin

"Last Modified Date Time" := CurrentDateTime();

end;

### Solution: table 50101 Vehicle

table 50101 Vehicle

{

Caption = 'Vehicle';

fields

{

field(1; Code; Code[10])

{

Caption = 'Code';

}

field(2; Make; Text[50])

{

Caption = 'Make';

}

field(3; Model; Text[50])

{

Caption = 'Model';

}

field(4; Year; Integer)

{

Caption = 'Year';

trigger OnValidate();

var

ErrorTooOld: Label 'The vehicle is too old.';

ErrorNotManufactured: Label 'The vehicle has not been manufactured yet. Please come back in %1 and configure it.';

begin

if Year < 2000 then

Error(ErrorTooOld);

if Year > Date2DMY(Today(), 3) then

Error(ErrorNotManufactured, Year);

end;

}

field(5; "Registration Plate"; Code[20])

{

Caption = 'Registration Plate';

}

field(8000; Id; Guid)

{

Caption = 'Id';

Editable = false;

}

field(8001; "Last Modified Date Time"; DateTime)

{

Caption = 'Last Modified Date Time';

Editable = false;

}

}

keys

{

key(Primary; Code) { Clustered = true; }

}

trigger OnInsert();

begin

if (IsNullGuid(Id)) then

Id := CreateGuid();

"Last Modified Date Time" := CurrentDateTime();

end;

trigger OnModify();

begin

"Last Modified Date Time" := CurrentDateTime();

end;

}

## Create the API page

1. Create a new file and name it Page 50101 Vehicle Entity.al.
2. In this file, declare page 50101 "Vehicle Entity", and set the following properties:

|  |  |
| --- | --- |
| Property | Value |
| Caption | vehicles |
| PageType | API |
| SourceTable | Vehicle |
| DelayedInsert | true |
| APIPublisher | redCarpet |
| APIGroup | demos |
| APIVersion | beta |
| EntityName | vehicle |
| EntitySetName | vehicles |
| ODataKeyFields | Id |

1. Declare the layout section.
2. Inside the layout section, declare the Content area.
3. Inside the Content area, declare a repeater, and name it **Vehicles**.
4. Inside the **Vehicles** repeater, declare all fields from the **Vehicle** table. Start with the Id field first, then declare all the other fields.
5. For each field, make sure that name of each field uses camelCase. Also, explicitly set caption and make sure it’s in camelCase, too. Also, make sure that the ApplicationArea is set to All for all fields.
6. Define the OnInsertRecord trigger for the page. From the trigger, insert the record explicitly and make sure you invoke the OnInsert trigger on the table. Exit the trigger with **false** as the return value.

trigger OnInsertRecord(BelowxRec: Boolean): Boolean;

begin

Insert(true);

exit(false);

end;

1. Define the OnModifyRecord trigger for the page. From the trigger, make sure that the **Id** field didn’t change. Then locate the **Vehicle** record that has the same Id, and then check if **Code** was changed. If **Code** was changed, transfer the field values from the current record to the record you located, then invoke Rename on that record. Then, transfer all fields back to the current record.

trigger OnModifyRecord(): Boolean;

var

Vehicle: Record Vehicle;

ErrorIdImmutable: Label 'Value of Id is immutable.';

begin

if xRec.Id <> Id then

Error(ErrorIdImmutable);

Vehicle.SetRange(Id, Id);

Vehicle.FindFirst();

if Vehicle.Code <> Code then begin

Vehicle.TransferFields(Rec);

Vehicle.Rename(Code);

TransferFields(Vehicle);

end;

end;

### Solution: page 50101 "Vehicle Entity"

page 50101 "Vehicle Entity"

{

Caption = 'vehicles';

PageType = API;

SourceTable = Vehicle;

DelayedInsert = true;

APIPublisher = 'redCarpet';

APIGroup = 'demos';

APIVersion = 'beta';

EntityName = 'vehicle';

EntitySetName = 'vehicles';

ODataKeyFields = Id;

layout

{

area(Content)

{

repeater(Vehicles)

{

field(id; Id)

{

Caption = 'id';

ApplicationArea = All;

}

field(code; Code)

{

Caption = 'code';

ApplicationArea = All;

}

field(make; Make)

{

Caption = 'make';

ApplicationArea = All;

}

field(model; Model)

{

Caption = 'model';

ApplicationArea = All;

}

field(year; Year)

{

Caption = 'year';

ApplicationArea = All;

}

field(registrationPlate; "Registration Plate")

{

Caption = 'registrationPlate';

ApplicationArea = All;

}

field(lastModifiedDateTime; "Last Modified Date Time")

{

Caption = 'lastModifiedDateTime';

ApplicationArea = All;

}

}

}

}

trigger OnInsertRecord(BelowxRec: Boolean): Boolean;

begin

Insert(true);

exit(false);

end;

trigger OnModifyRecord(): Boolean;

var

Vehicle: Record Vehicle;

ErrorIdImmutable: Label 'Value of Id is immutable.';

begin

if xRec.Id <> Id then

Error(ErrorIdImmutable);

Vehicle.SetRange(Id, Id);

Vehicle.FindFirst();

if Vehicle.Code <> Code then begin

Vehicle.TransferFields(Rec);

Vehicle.Rename(Code);

TransferFields(Vehicle);

end;

end;

}

## Create the UI page

1. Create a new file and name it **Page 50102 Vehicles.al**.
2. In the file, declare page 50102 Vehicles and set the following properties:

|  |  |
| --- | --- |
| Property | Value |
| Caption | Vehicles |
| PageType | List |
| SourceTable | Vehicle |
| UsageCategory | List |
| ApplicationArea | All |

1. Declare the layout section.
2. Inside the layout section, declare the Content area.
3. Inside the Content area, declare a repeater, and name it **Vehicles**.
4. Inside the **Vehicles** repeater, declare all fields from the **Vehicle** table, except for the **Id** and **Last Modified Date Time** fields. Make sure to declare the fields in the same order in which the table declares them.
5. For each field, make sure that the ApplicationArea is set to All.

### Solution: Page 50102 Vehicles.al

page 50102 Vehicles

{

Caption = 'Vehicles';

PageType = List;

SourceTable = Vehicle;

UsageCategory = Lists;

ApplicationArea = All;

layout

{

area(Content)

{

repeater(Vehicles)

{

field(Code; Code) { ApplicationArea = All; }

field(Make; Make) { ApplicationArea = All; }

field(Model; Model) { ApplicationArea = All; }

field(Year; Year) { ApplicationArea = All; }

field("Registration Plate"; "Registration Plate")

{

ApplicationArea = All;

}

}

}

}

}

# Test the custom API

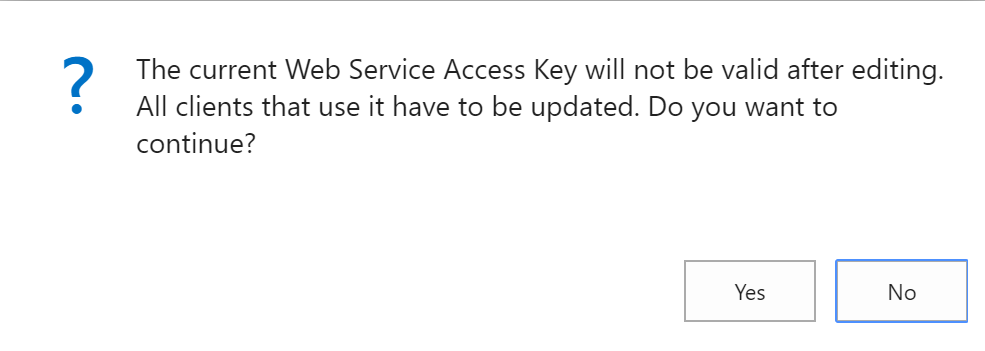
In this exercise, you publish your custom API to your Business Central cloud sandbox instance and then test the API from Postman. You’ll use the Business Central web client to check the data.

## Publish the custom API

1. In Visual Studio Code, press Ctrl+F5 to build, deploy, and run the extension.

## Obtain Business Central web services basic authentication credentials

1. Start your Business Central web client.
2. Use the **Tell me** feature to search for and run the **Users** list page.
3. In the list of users, locate your user account and click its name. This opens the **User Card** page.
4. Expand the **Web Service Access** FastTab.
5. Click **Process > Change Web Service Key**. Business Central asks you to confirm this dialog:



1. Click **Yes**.
2. In the **Edit – Set Web Service Access Key** dialog, select **Key Never Expires**, and click **OK**.
3. Copy the value from the **Web Service Access Key** field and paste it somewhere. You will need it later.
4. Close the **User Card** page.

## Obtain Business Central tenant ID and your sandbox URL

1. Use the **Tell me** feature to search for and run the **Web Services** page.
2. Copy the OData4 field value from any of the rows. It may look something like this:

https://api.businesscentral.dynamics.com/v1.0/**7878d308-4c84-4c38-931a-928fa9b5cb1b**/sandbox/ODataV4/Company('CRONUS%20USA%2C%20Inc.')/Power\_BI\_Cust\_Item\_Ledg\_Ent?$select=No,Item\_No,Quantity

1. The GUID (for your convenience it’s marked with **red bold underline**) is your Tenant ID, in this example it is **7878d308-4c84-4c38-931a-928fa9b5cb1b**. Copy that value and paste it somewhere.
2. Your sandbox URL has this form: [https://api.businesscentral.dynamics.com/v1.0/<tenant\_id>/sandbox/api/beta/](https://api.businesscentral.dynamics.com/v1.0/%3ctenant_id%3e/sandbox/api/beta/). Replace the <tenant\_id> part with your Tenant ID, for example: <https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/beta>

## Test the extension in Business Central

1. Start Postman.
2. Close the **Create New** dialog.
3. Click (+) (or press Ctrl+T) to create a new tab.
4. In the new tab, make sure that the method is GET, and put your sandbox root URL into the URL box.
5. Click **Authorization** (under the GET method URL).
6. In the **Type** combo box, select **Basic Auth**.
7. In the **Username** and **Password** fields, put your user credentials. Your username is exactly as listed in your User Card, and your password is the Web service access key you copied from the User Card.
8. Click **Send**. The response section of the Postman screen will show you the JSON response containing all of standard APIs.
9. Modify the GET URL by replacing /beta with /redCarpet/demos/beta. Your full URL might now look like this:

<https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta>

1. Click **Send** again.
2. Your response JSON now lists your custom APIs:

{

"@odata.context": "https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/$metadata",

"value": [

{

"name": "vehicles",

"kind": "EntitySet",

"url": "vehicles"

},

{

"name": "companies",

"kind": "EntitySet",

"url": "companies"

},

{

"name": "subscriptions",

"kind": "EntitySet",

"url": "subscriptions"

}

]

}

1. Append /companies to the URL and click **Send** again. You get the list of companies:

{

"@odata.context": "https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/$metadata#companies",

"value": [

{

"id": "500fb248-b517-4d7c-99b8-0daf25b479f8",

"systemVersion": "28186",

"name": "CRONUS USA, Inc.",

"displayName": "CRONUS USA, Inc.",

"businessProfileId": ""

},

{

"id": "fbbad9d1-b822-4efb-977c-24bb5206ce6c",

"systemVersion": "28186",

"name": "My Company",

"displayName": "redcarpetdemos.info",

"businessProfileId": ""

}

]

}

1. Copy the id of the company you are currently signed into in your Business Central web client and append the (<company\_id>)/vehicles to the URL, where <company\_id> is the id you copied. For example, the entire URL may now look like this:

https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/companies(500fb248-b517-4d7c-99b8-0daf25b479f8)/vehicles

1. You will receive the following JSON:

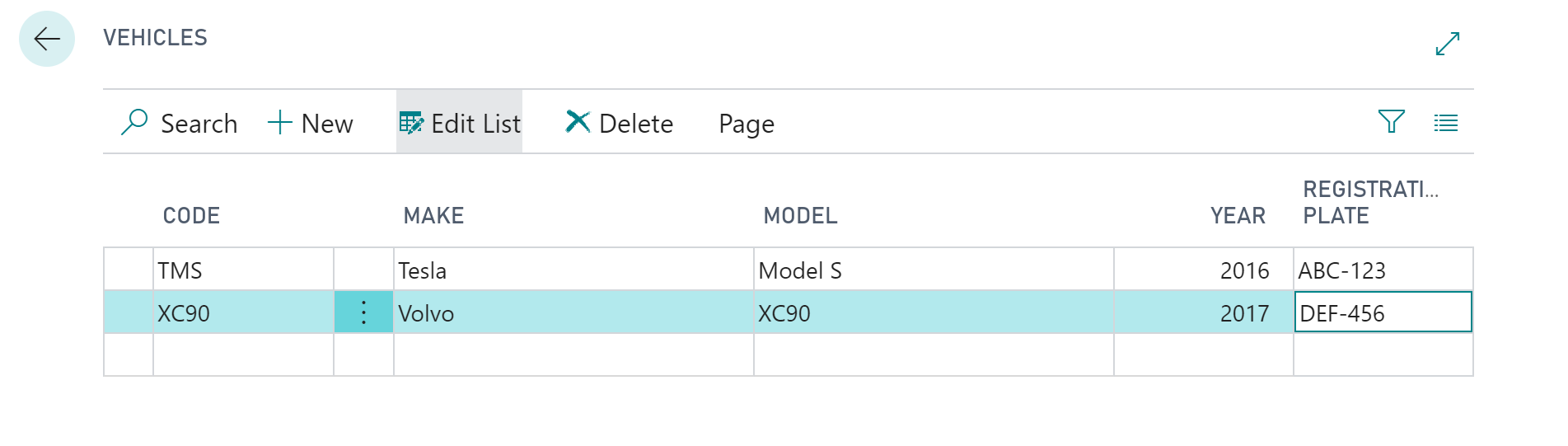
{

"@odata.context": "https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/$metadata#companies(500fb248-b517-4d7c-99b8-0daf25b479f8)/vehicles",

"value": []

}

1. Back in your Business Central client, use the **Tell me** feature to search for and run the **Vehicles** page.
2. In the **Vehicles** page, click **Edit List**.
3. Enter a couple of vehicle records with arbitrary information, for example:



1. Back in Postman, click **Send** without changing the URL. You will receive JSON containing your new vehicle information:

{

"@odata.context": "https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/$metadata#companies(500fb248-b517-4d7c-99b8-0daf25b479f8)/vehicles",

"value": [

{

"@odata.etag": "W/\"JzQ0O3dZUWc1aGNuejRqS0I0ZS9wc2tJS1kxbWxtSzIzMHd2WUNwaTFJZjc2U2s9MTswMDsn\"",

"id": "e2e1d24f-f6e6-4a35-8750-56b7d6b39071",

"code": "XC90",

"make": "Volvo",

"model": "XC90",

"year": 2017,

"registrationPlate": "DEF-456",

"lastModifiedDateTime": "2019-02-27T13:54:53.187Z"

},

{

"@odata.etag": "W/\"JzQ0OzVVdmFZLzA2eDhPU2VERFVDSVM4ZWFsbUlaMVZWNGt3M3AydVdDU3RyQjA9MTswMDsn\"",

"id": "4f0c39b1-037b-405b-9403-cd3f89859773",

"code": "TMS",

"make": "Tesla",

"model": "Model S",

"year": 2016,

"registrationPlate": "ABC-123",

"lastModifiedDateTime": "2019-02-27T13:49:09.327Z"

}

]

}

1. Copy the id of one of the listed vehicles, append (<copied\_id>) to the URL, and then click **Send**. This will obtain information about a single vehicle:

{

"@odata.context": "https://api.businesscentral.dynamics.com/v1.0/7878d308-4c84-4c38-931a-928fa9b5cb1b/sandbox/api/redCarpet/demos/beta/$metadata#companies(500fb248-b517-4d7c-99b8-0daf25b479f8)/vehicles/$entity",

"@odata.etag": "W/\"JzQ0O3dZUWc1aGNuejRqS0I0ZS9wc2tJS1kxbWxtSzIzMHd2WUNwaTFJZjc2U2s9MTswMDsn\"",

"id": "e2e1d24f-f6e6-4a35-8750-56b7d6b39071",

"code": "XC90",

"make": "Volvo",

"model": "XC90",

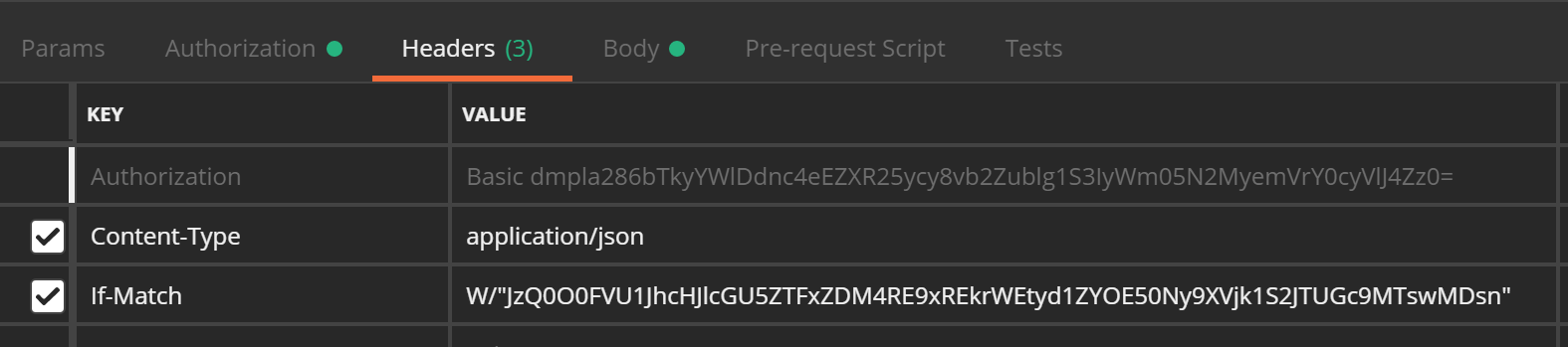
"year": 2017,

"registrationPlate": "DEF-456",

"lastModifiedDateTime": "2019-02-27T13:54:53.187Z"

}

1. Copy the value from the @odata.etag value, declare a request header named If-Match, and put the copied value in there. Make sure to replace \" with " (the \" is simply a JSON escaped representation of "). For example:



1. Click Body, and write the following JSON in the body:

{

"year": 2019,

"registrationPlate": "GHI-789"

}

1. Change the request type from GET to PUT.
2. Click Send. You will receive a JSON document indicating the new state of the record.
3. Back in the NAV web client, press F5 to refresh the Vehicles page. Verify if the **Year** and **Registration Plate** have been updated for the vehicle you chose to update.