NEO EDITABLE GRID
USER MANUAL
VERSION 1.0.3.6 - NOVEMBER 19, 2018

Contents

I. What is Neo Editable Grid?

Neo Editable Grid (NeoGrid) is a tool that generates editable grids in Dynamics 365. NeoGrid dramatically improves user experience by making record editing a lot easier for users with following features:

- Edit multiple records in Excel like grid and save them all with one click.
- Dynamics CRM/365 security model is inherited. Read/Update/Delete records only if you can do in CRM forms.
- In addition to adding Data Columns, *Custom* columns can be added to grid to provide user-friendliness.
- Use arrow keys to move in editable cells.
- Resize grid and columns widths, recall user preference.
- Allow paging. User can change page size and it is recalled.
- Allow filtering on client side with different style for each column, like text search, select multiple options, date range and number range.
- Add new records to grid with default values including lookup and optionset attribute types.
- Provides easy to configure event handling with integrated Javascript API.
- Nicely handle editing lookup type fields. Search and populate based on partial input.
- Optionset field options are dynamically populated. No need to re-publish the grid, if optionset attribute options are changed.
- Allow editing related entity fields up to three levels.
- Totally integrated solution with Dynamics CRM/365: generates Html and JavaScript web resources. The generated JavaScript uses Dynamics CRM/365 JavaScript API and Web services.
- It does not require any external web application.
- Easily insert generated html web resource into CRM forms.

II. How to Obtain the CRM Solution

Neo Editable Grid is delivered as a managed CRM solution file when you register for FREE product from our website, https://www.neoxrm.com. At the time of registration, you need to provide the CRM organization unique name that you want to install the solution. CRM organization unique name can be found under Settings -> Customization -> Developer Resources page. The delivered solution file will only work for this specific CRM instance. You can always contact us for support either via email at support@neoxrm.com or via phone 1-844-NEOXRM1.

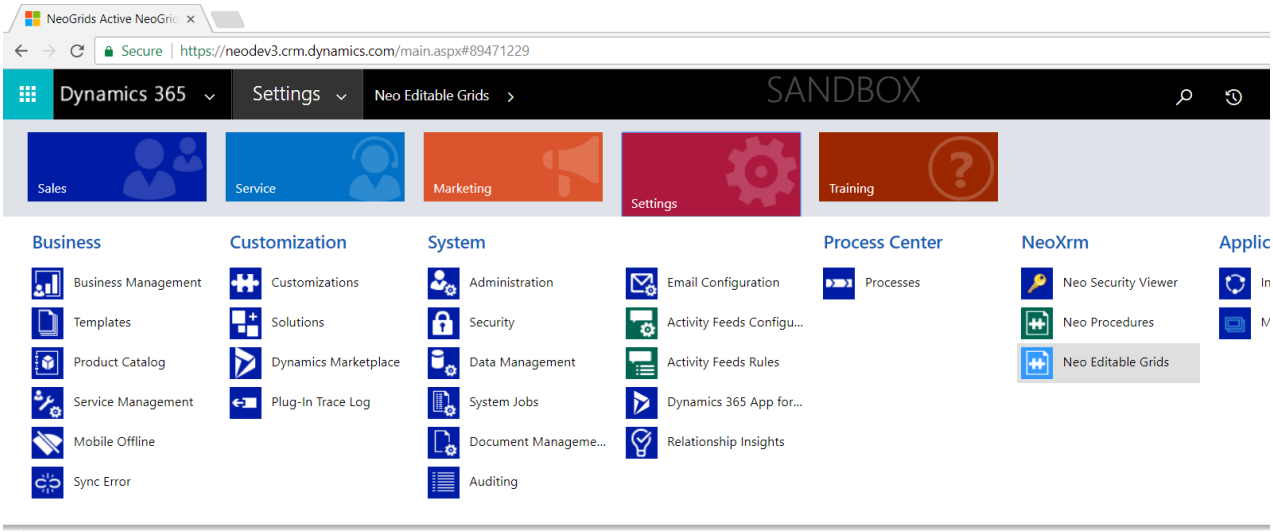
III. How to Install the CRM Solution

The next step is to import the downloaded CRM solution into your CRM environment. You need to have system admin or system customizer role to be able to import CRM solution in general. Once the solution is imported successfully, your NeoGrid is ready to use.

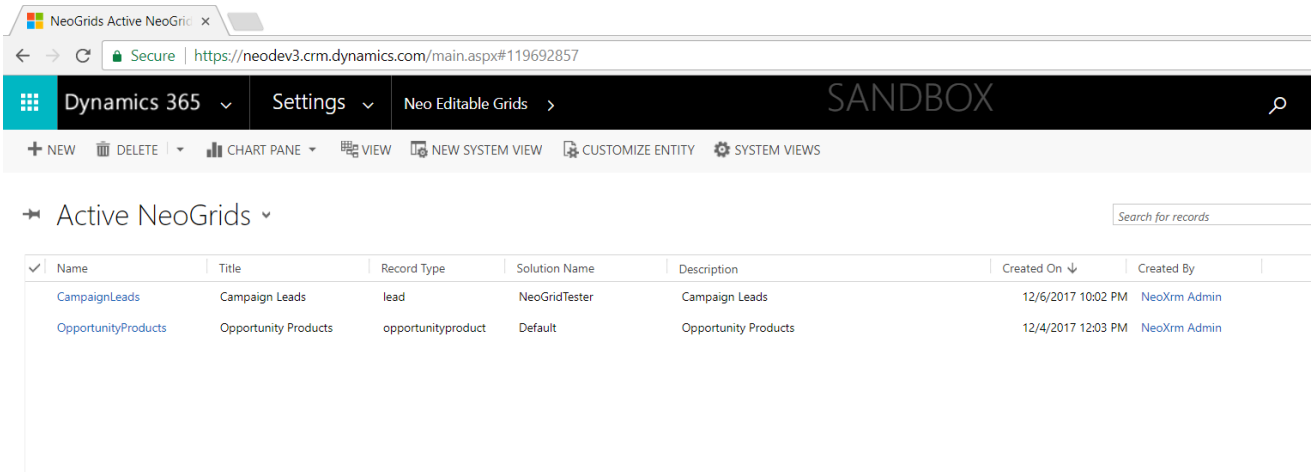
IV. How to Create a New Editable Grid

The NeoGrid solution includes a security role, which allows users to create new grid and modify existing grids. Users with this security role (and Administrator users, of course) can create new grids. You do not need any extra security role to be able to use generated grids; Dynamics CRM/365 security model applies to create/read/modify records in the grid.

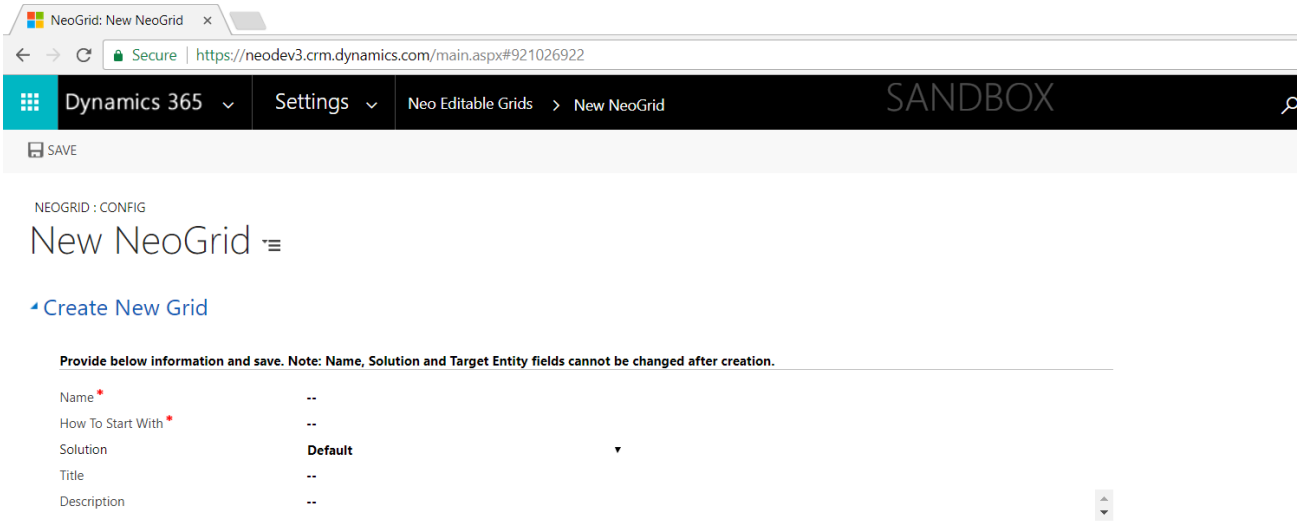
Once the NeoGrid solution is imported into Dynamics CRM/365, Neo Editable Grid entity is listed under the Setting / NeoXrm section:



Click on the **NeoGrids** button, the opened view will list all grids.

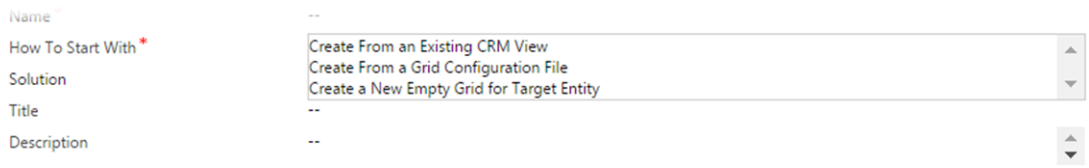


Click on the **New** button to create a new grid. It will open a new NeoGrid form.



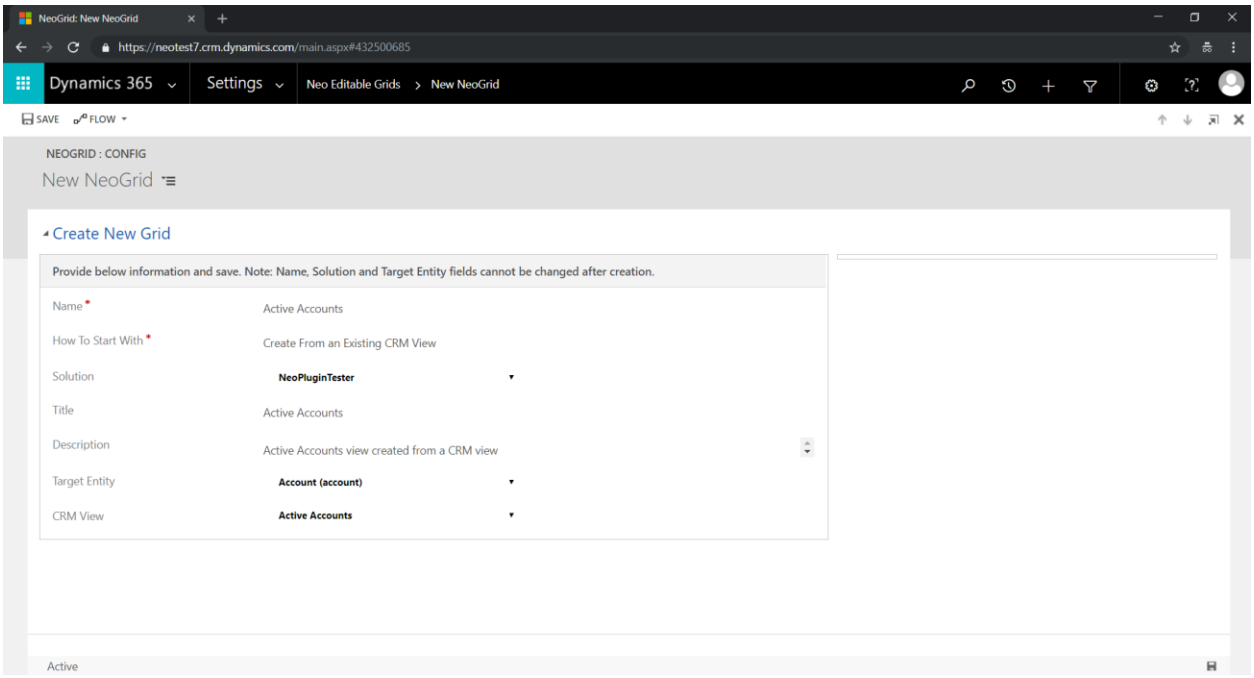
Most fields in this simple form cannot be changed after the initial creation, except for the Title and Description fields.

There are three options to start creating a new grid.



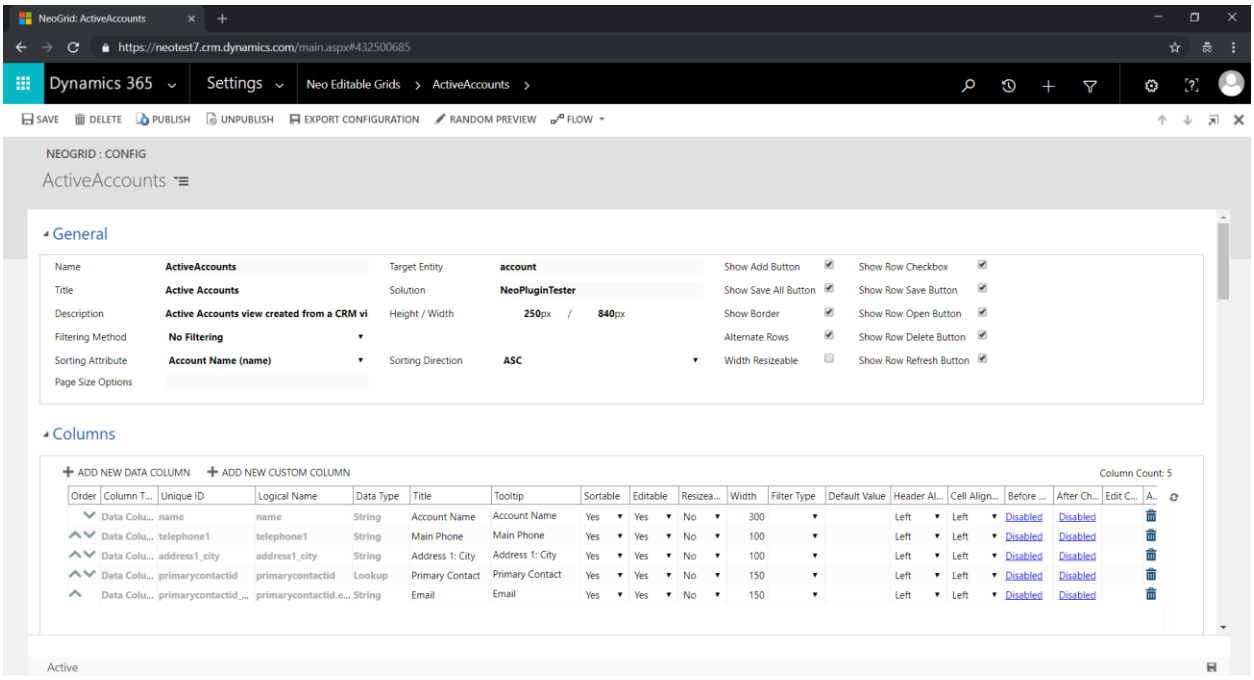
- Quickly create from an Existing CRM View

This is the quickest way to create a new editable grid. All you need to do is the select the entity and the view you want to make editable. If you chose this option, two new fields will show up to select the target entity and a view under that entity.

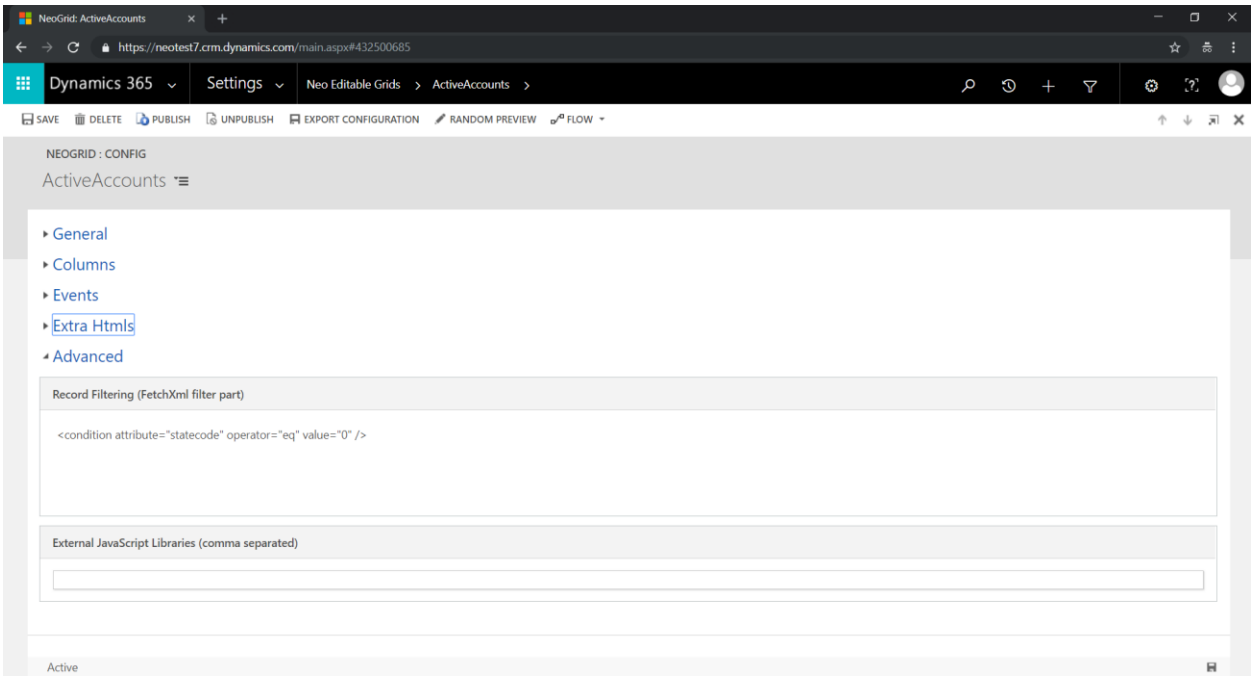


When the target entity is selected, CRM view optionset is populated to list available CRM views under that entity. CRM view is selected and record is saved.

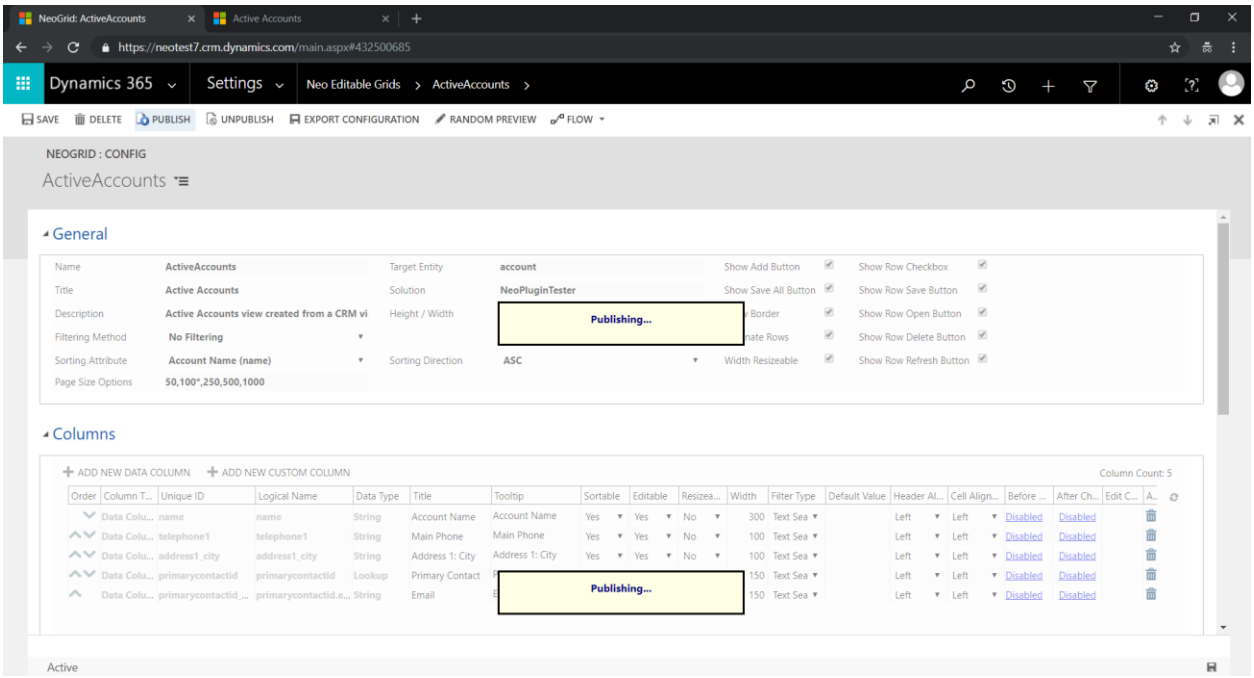
Once the record is saved, more fields become visible to further edit the grid. At this moment, the NeoGrid is ready to publish. List of columns and filtering criteria is brought from the selected view and used in the NeoGrid. Further customization can be done to the grid after this initial creation.



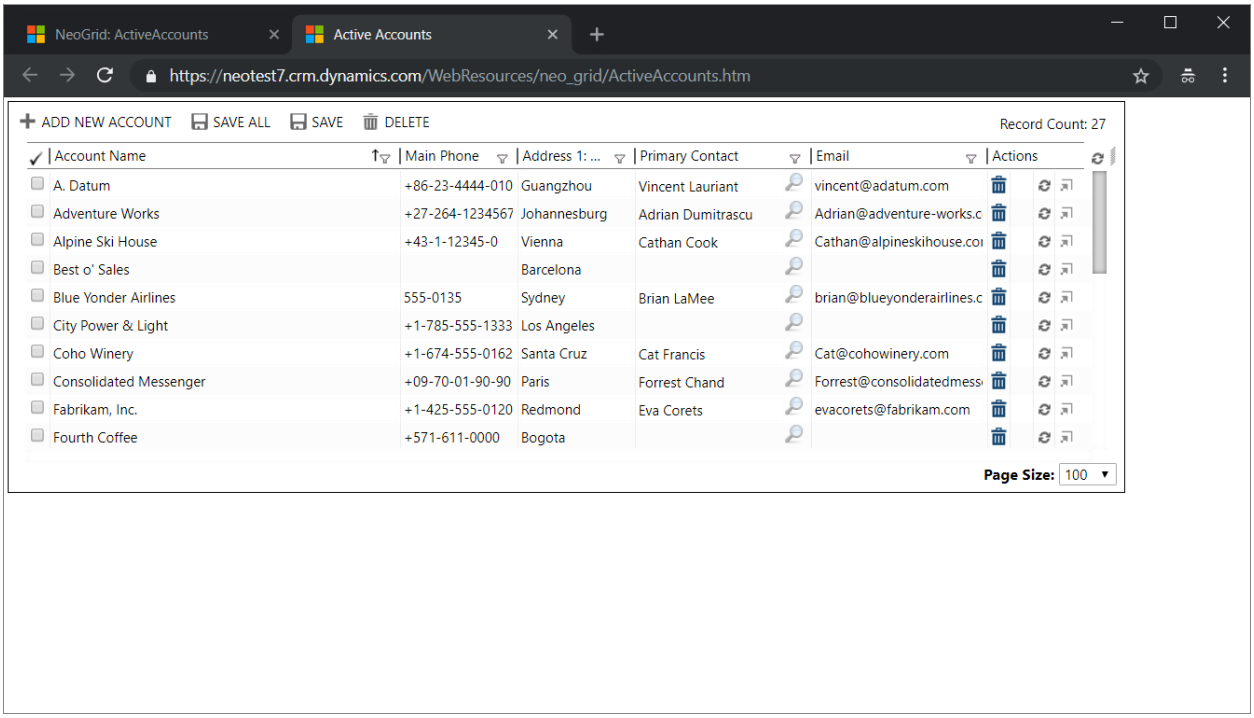
If there is any filtering criteria in the CRM view (as we have in the ***Active Accounts*** View), that criteria is carried over to NeoGrid, as seen under the Record Filtering section.



You make further changes to the grid and it is ready to publish. To publish the grid, click on the Publish button. Behind the scene, this action will create two web resources in CRM: neo_grid/ActiveAccounts.htm and neo_grid/ActiveAccounts.js.



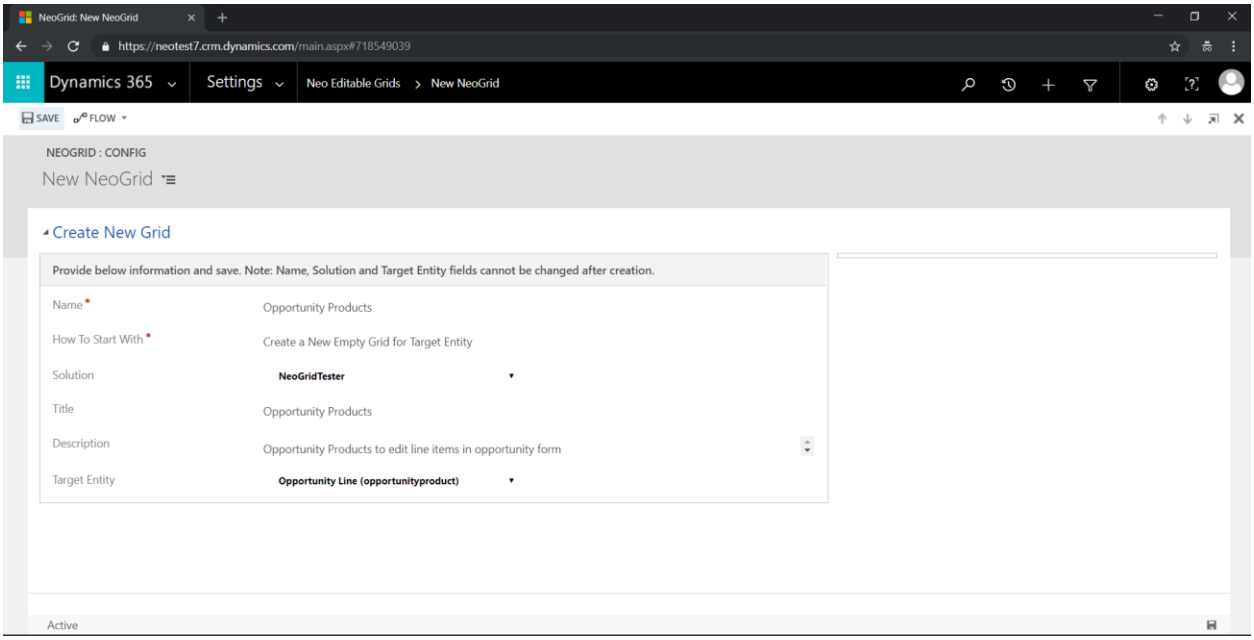
By clicking on the Random Preview button, neo_grid/ActiveAccounts.htm web resource is opened in a new window. Bu sure that Pop-up blocker is disabled for the CRM site.



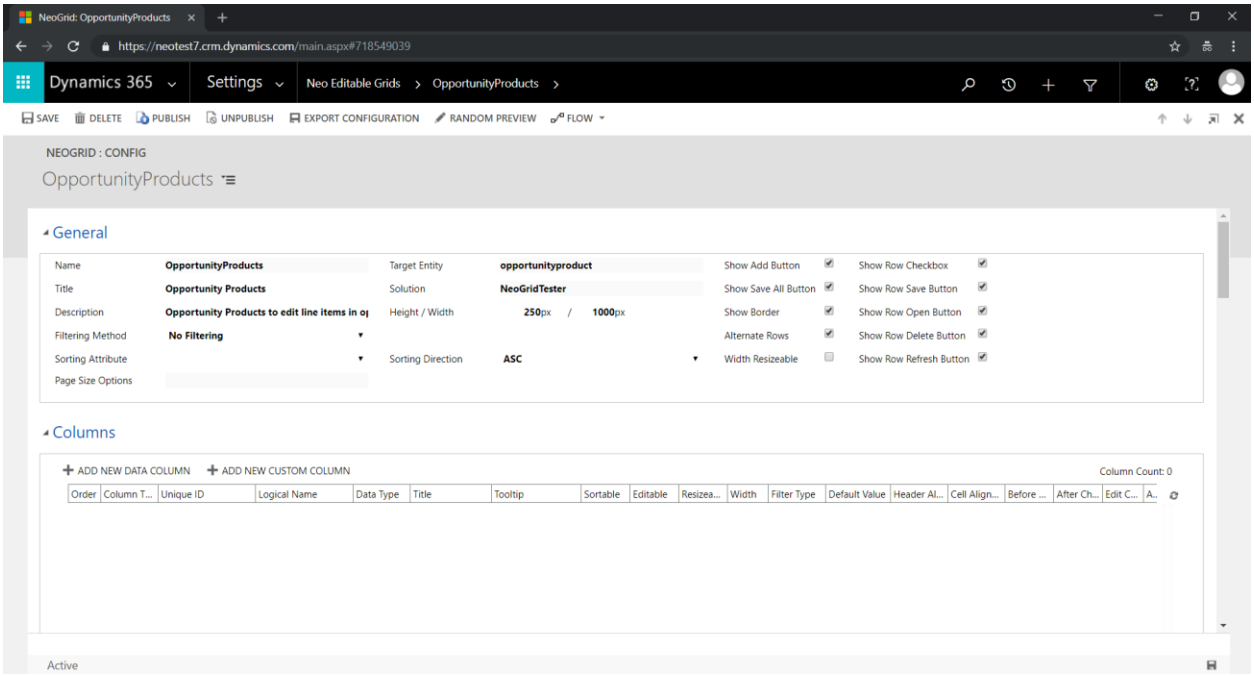
By clicking on the Export Configuration button on top, the grid configuration can be exported to be able to import the grid configuration into another environment or to clone in the same environment. An xml file is saved to local machine like: ActiveAcccounts_config.xml.

- Create a New Grid From Scratch

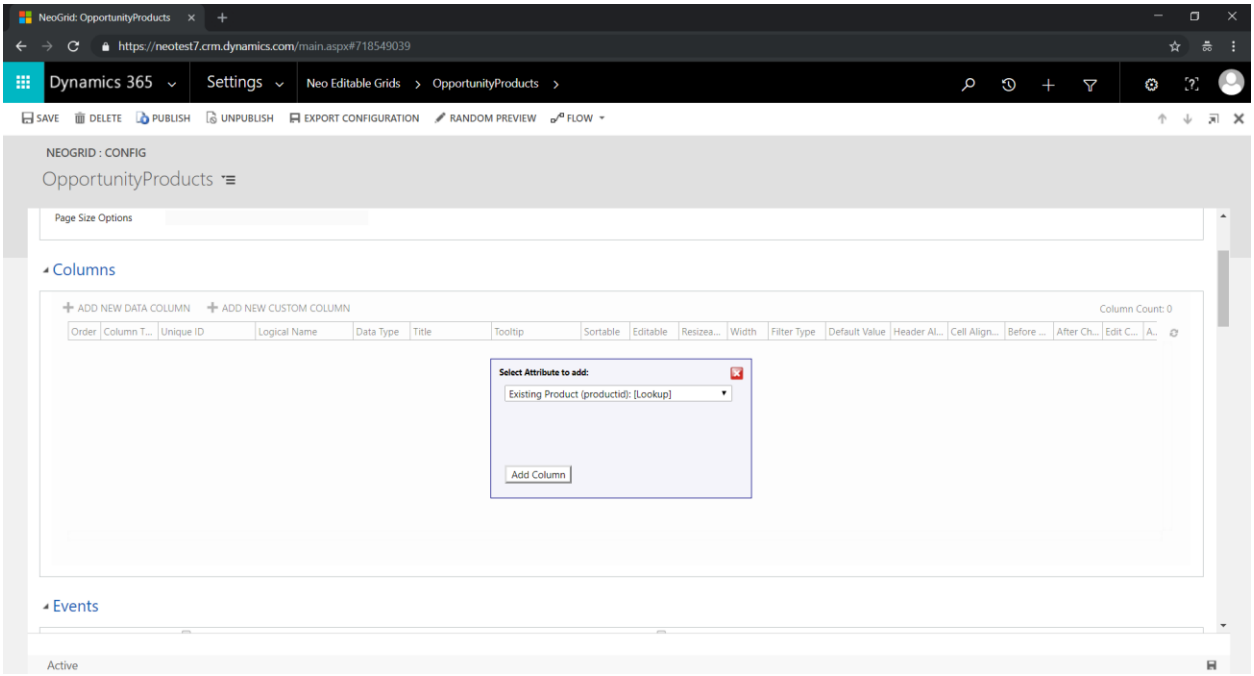
A new grid can be created from scratch by selecting the option “Create an empty grid for Target Entity”, and then selecting the Target Entity from the appearing list.



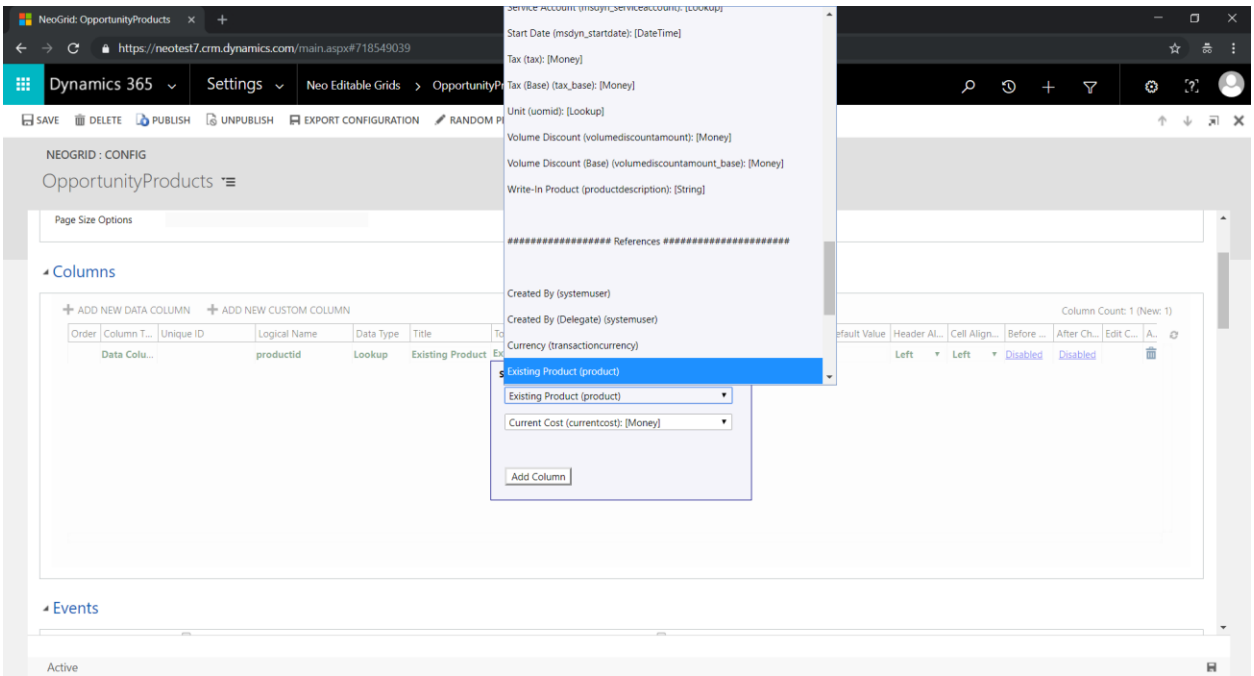
Click on the Save button on top.

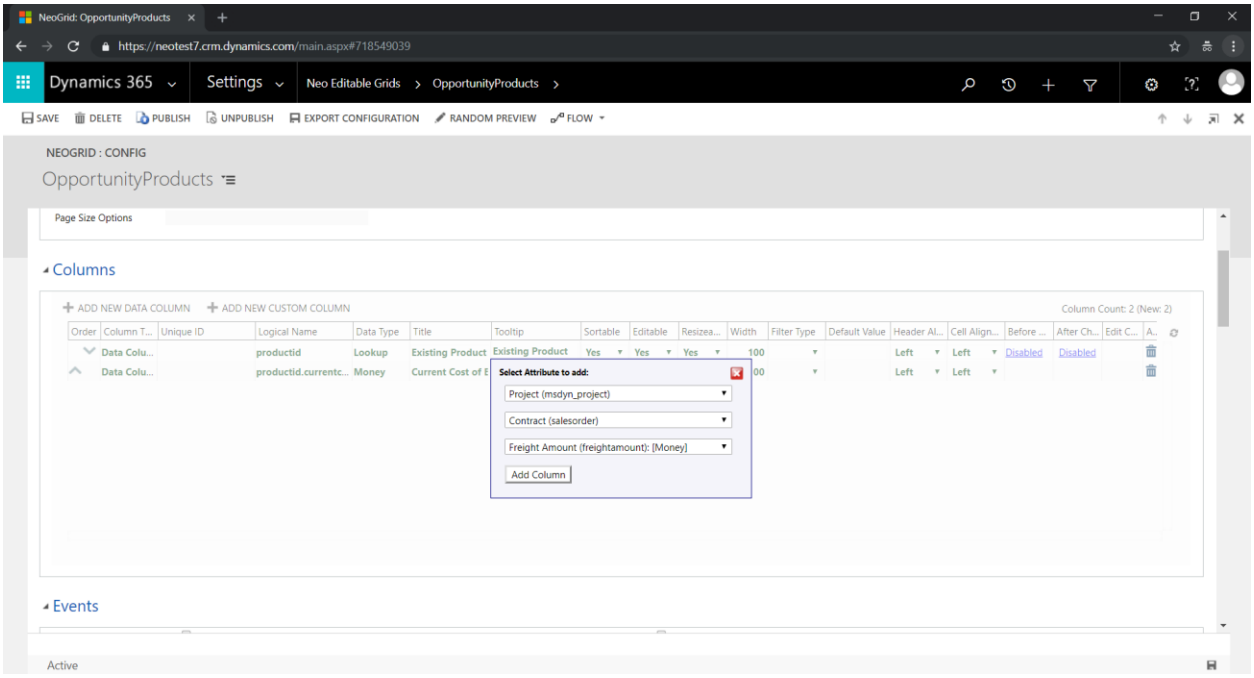


After grid record is created, new columns can be added to the grid. Click on the “Add New Data Column” button and select an attribute.



In addition to attributes of the target entity, attributes of the related entities (lookup) can be added, by selecting the lookup field first and then attribute of the other entity.



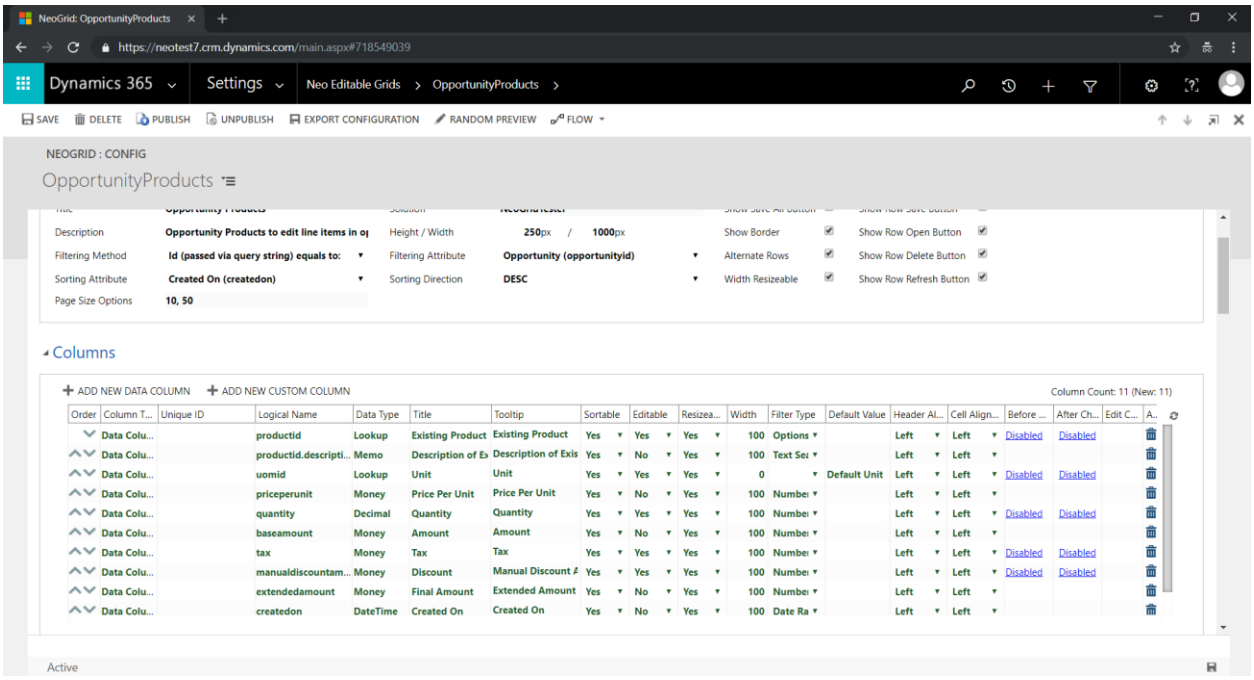


Column orders on the grid can be changed by using the up and down arrows.

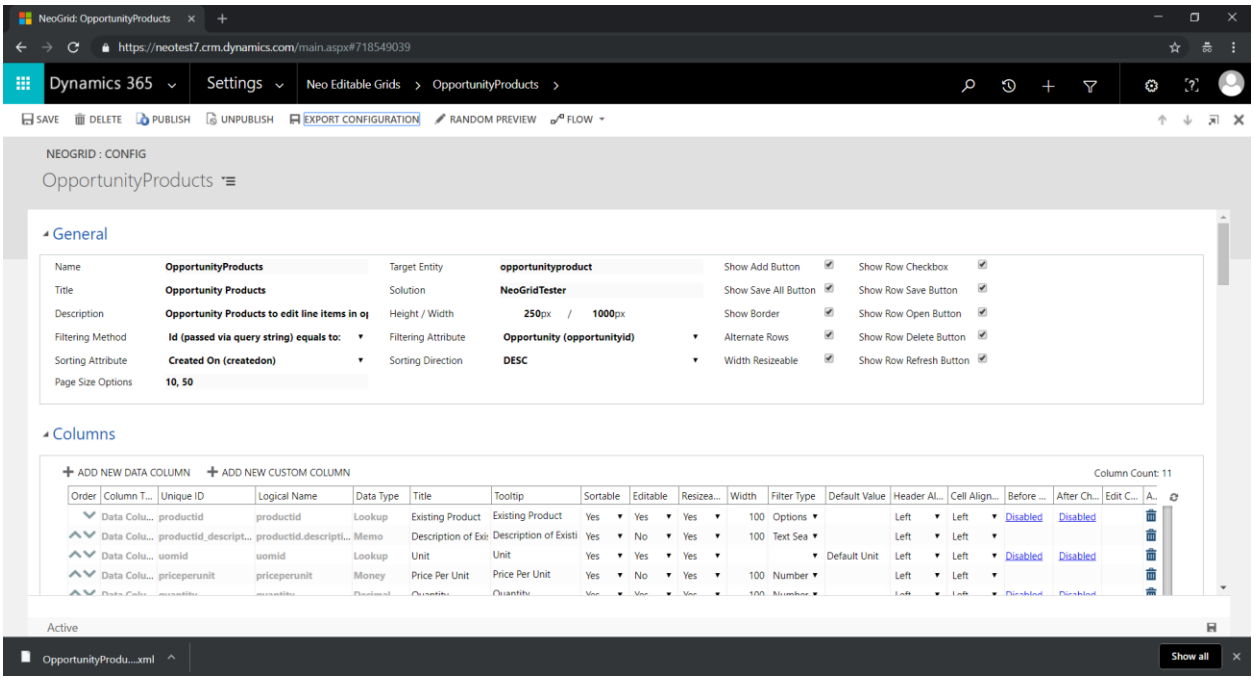
Since we intend to list opportunities for a specific account only, we set the filtering method to "*Id (passed via query string) equals to:*", and set the Filtering Attribute to "*Account (parentaccountid)*".

After adding any column, it becomes available in the "Sort Attribute" property of the grid. Select the sort attribute and sort direction. Set other properties as well. Click on the Save button on top.

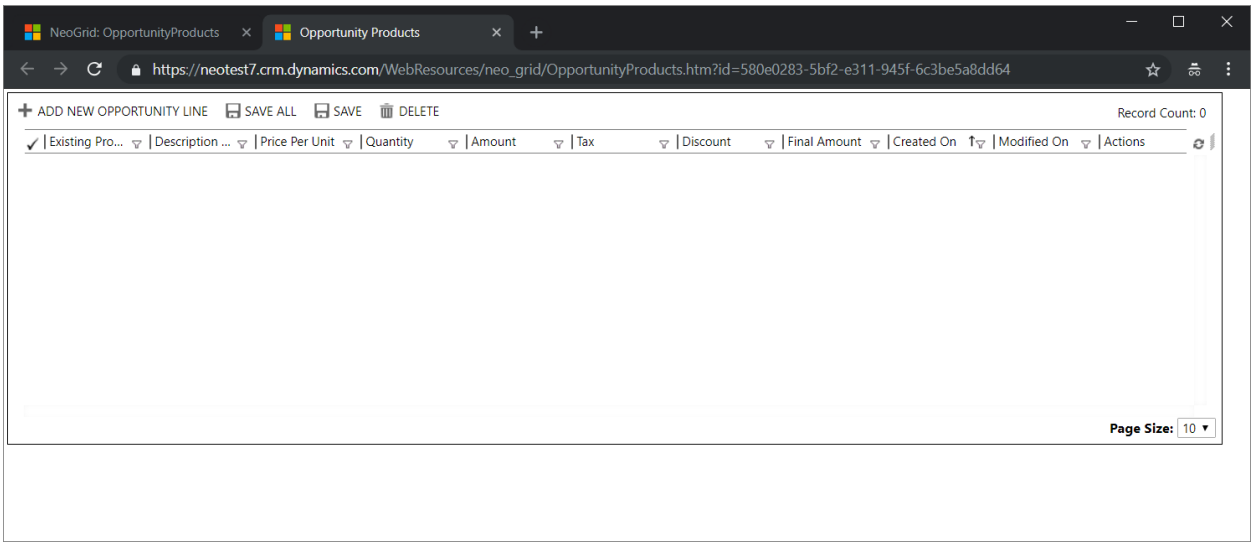
After successful saving, you will notice that each column was assigned a unique id, which can be used during JavaScript coding for event handling.



By clicking on the Export Configuration button, the whole configuration of the NeoGrid can be exported to a xml file and can be used to create another NeoGrid. (OpportunityProducts_config.xml)



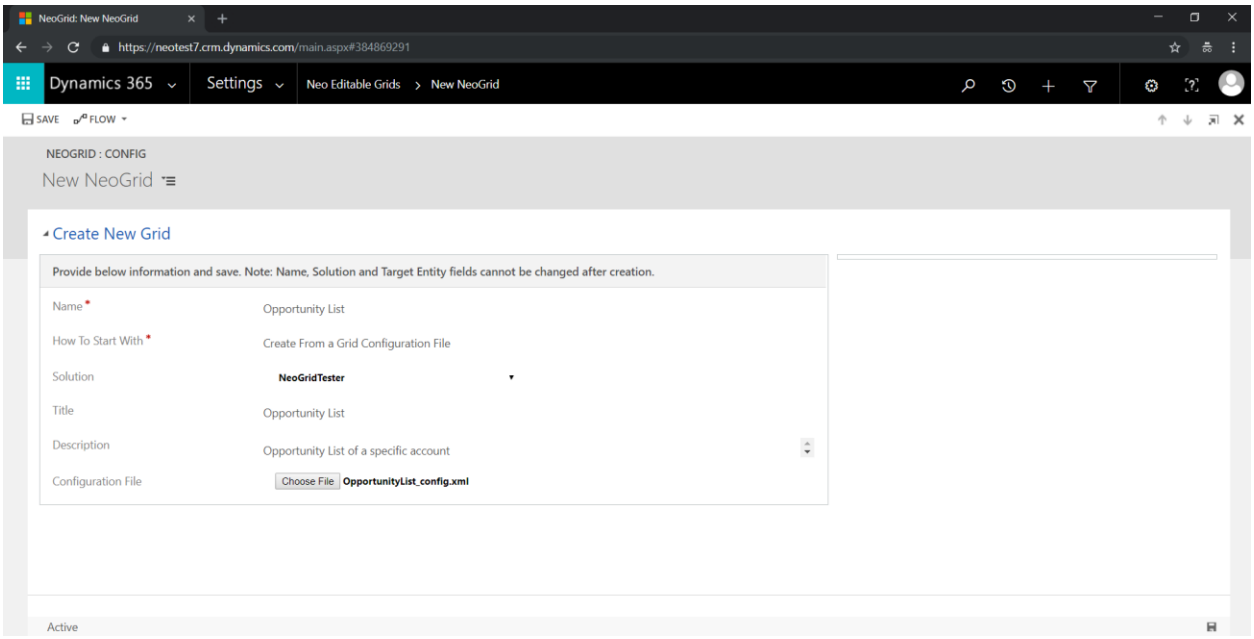
Once all features of the grid is set, it can be published by clicking on the Publish button. Clicking on the “Random Preview” button will select a random opportunity record and open the web resource in a new tab and pass the opportunity id to the new grid.


NeoGrid: OpportunityProducts
Opportunity Products
https://neotest7.crm.dynamics.com/WebResources/neo_grid/OpportunityProducts.htm?id=580e0283-5bf2-e311-945f-6c3be5a8dd64
ADD NEW OPPORTUNITY LINE
SAVE ALL
SAVE
DELETE
Record Count: 0
Existing Pro...
Description ...
Price Per Unit
Quantity
Amount
Tax
Discount
Final Amount
Created On
Modified On
Actions
Page Size: 10

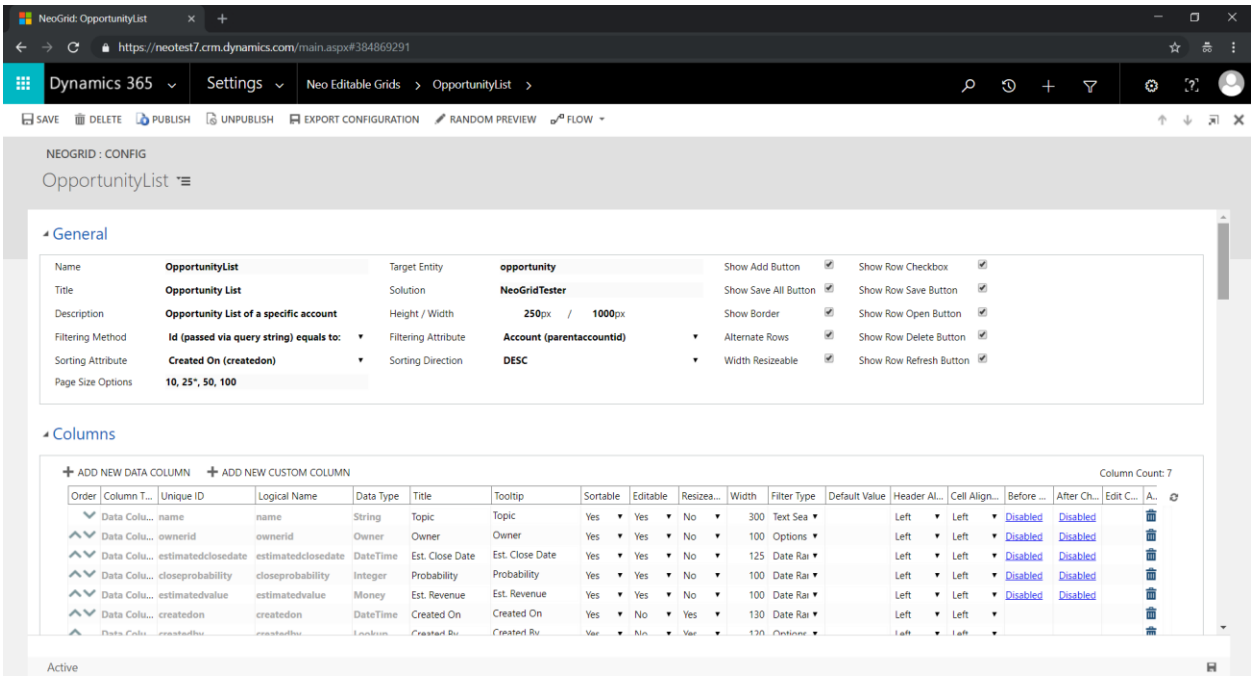
After

- Import a Previously Exported NeoGrid

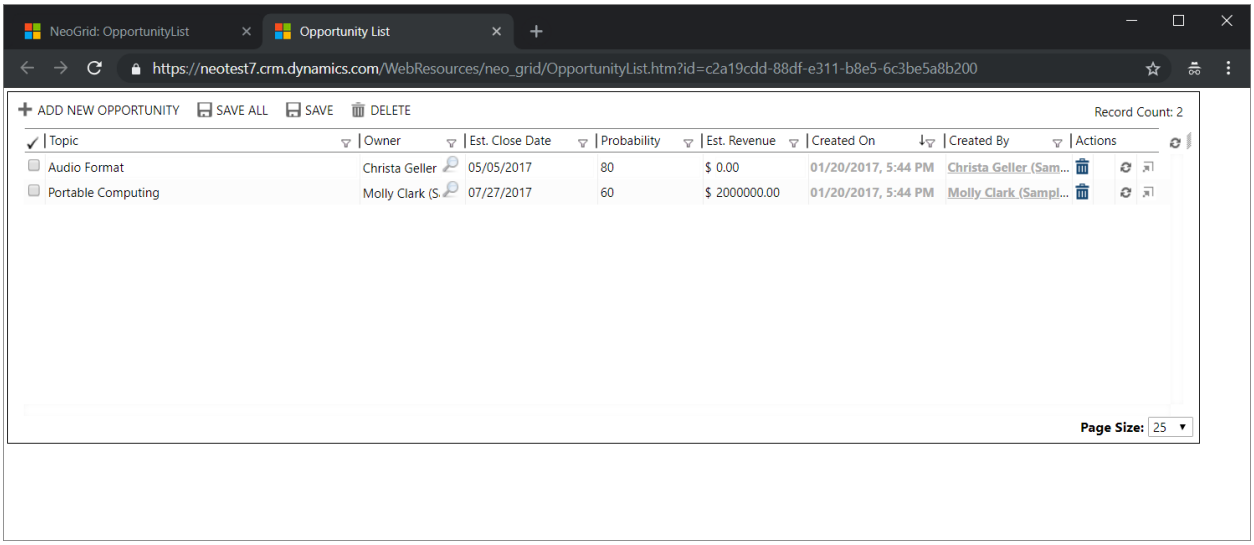
Previously exported configuration file can be used to create a new grid.



Select the option “Create From a Grid Configuration File” and select the Configuration File. Click on the Save button. A new NeoGrid will be created with exactly same features.



After the initial creation, the user can further customize the grid and then publish.



V. How to Publish and Un-publish a Neo Editable Grid

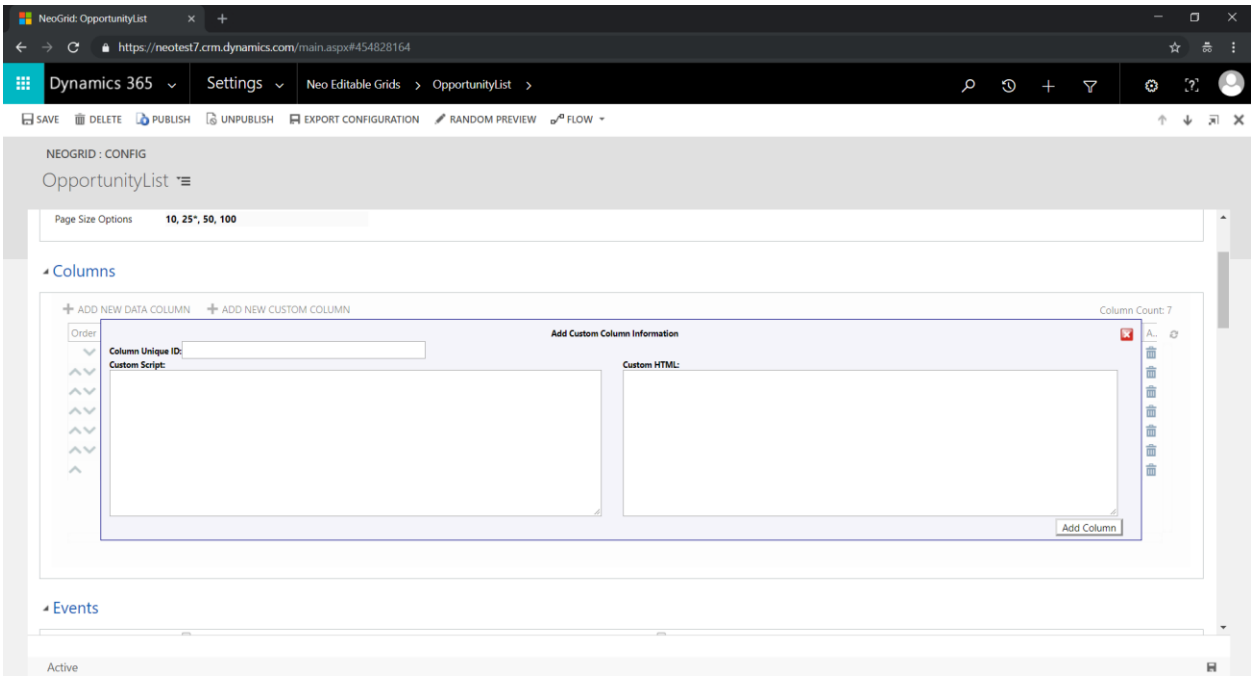
Click on the Publish button to generate the web resources. This process may take a few seconds. Without publishing the grid, changes on the grid configuration will not be in effect.

Click on the Unpublish button to delete the generated web resources. If web resources are used in any CRM form, they cannot be unpublished. You need to remove them from CRM forms, first.

VI. Adding Custom Columns

In addition to data columns, custom columns can be added to grid to provide user more functionality. For example, we want to add a “Follow up” button to opportunity list for records where estimated close date not provided or estimated close date already passed. By clicking to this button users can send an automated follow up email to account executive without opening the opportunity or contact record.

To achieve that you need two things: 1- some data to be collected dynamically and 2- some html to make the data visible. Click on the “Add New Custom Column” button. It will open a small window with three fields.

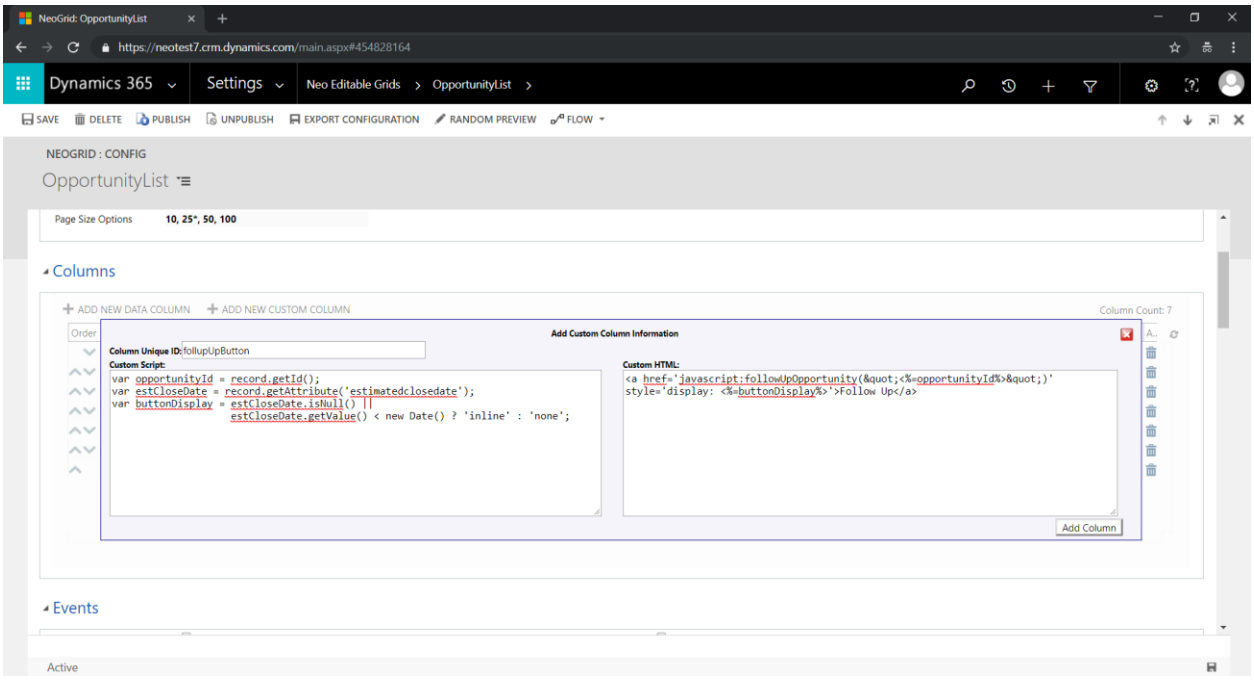


Give a unique name to the new column and set the “Custom Script” to:

```
var opportunityId = record.getId();
var estCloseDate = record.getAttribute('estimatedclosedate');
var buttonDisplay = estCloseDate.isNull() ||
                    estCloseDate.getValue() < new Date() ? 'inline' : 'none';
```

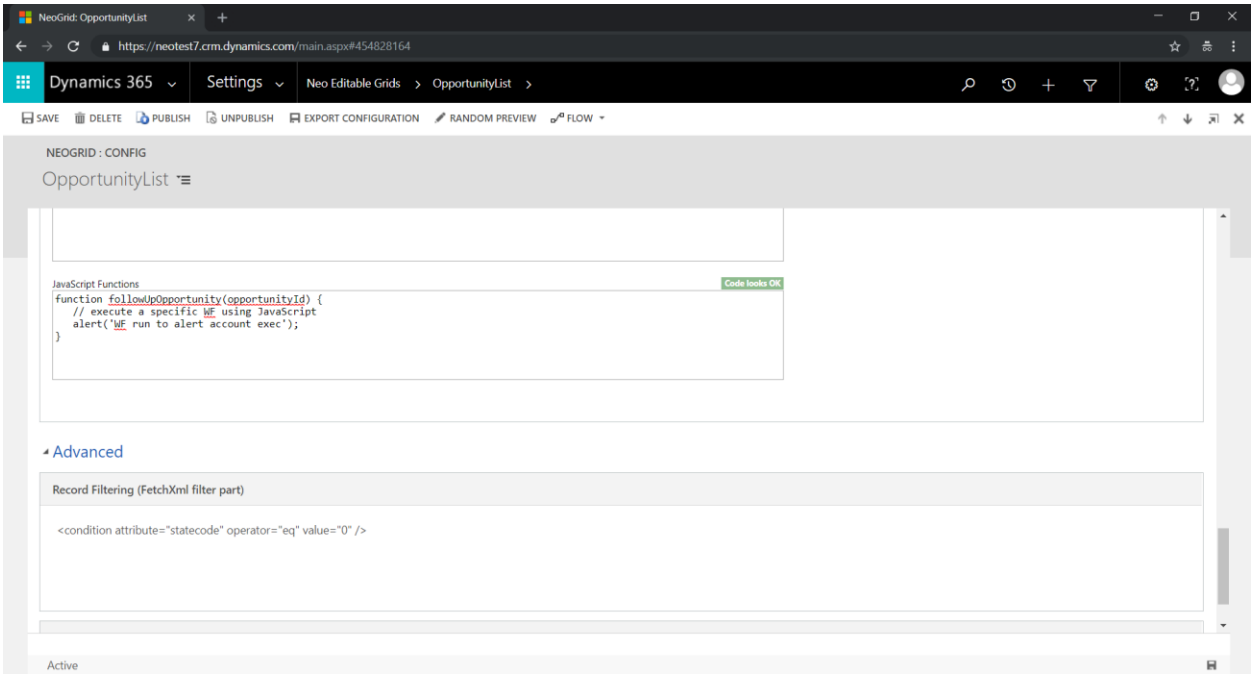
Set the “Custom HTML” to:

```
<a href='javascript:followUpOpportunity(&amp;quot;<%=opportunityId%>&amp;quot;)' style='display:
<%=buttonDisplay%>'>Follow Up</a>
```

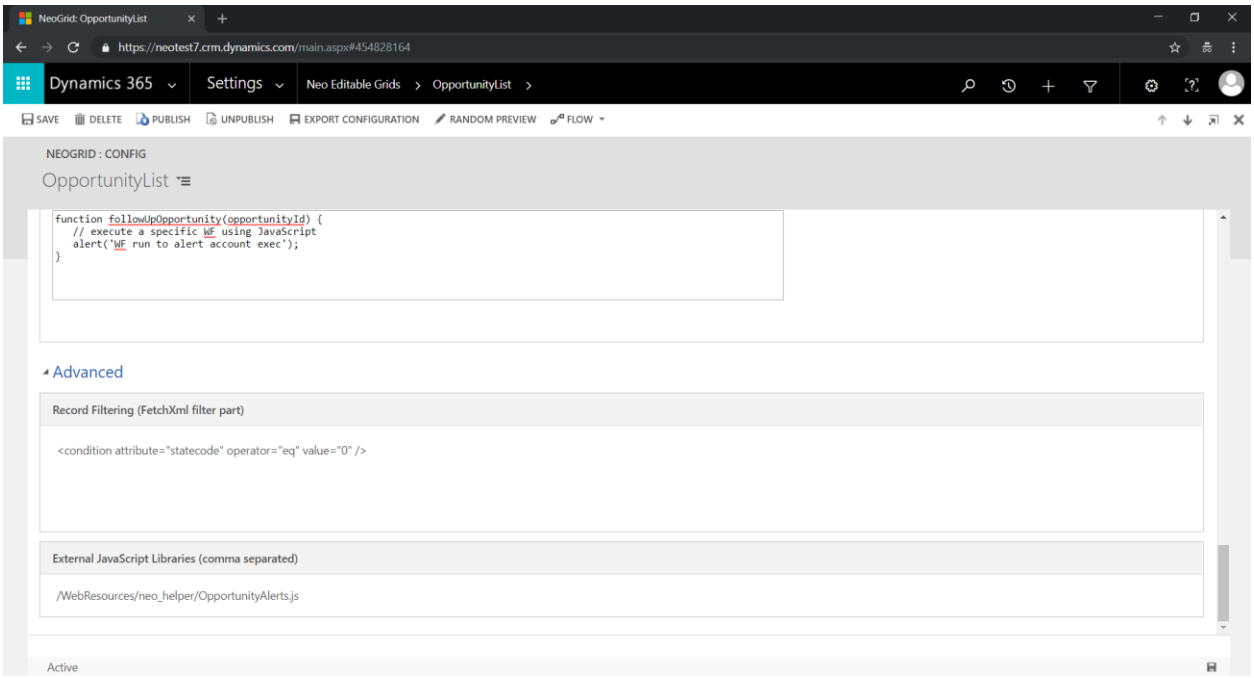


The texts `<%=opportunityId%>` and `<%=buttonDisplay%>` will be replaced with the values of `opportunityId` and `buttonDisplay` variables calculated with given script. The button click is calling a JavaScript function called `followUpOpportunity` which accepts an opportunity id and sends a “Follow Up” email to account executive of the opportunity by firing a CRM workflow.

This function can be provide in the Javascript field in the Extra Htmls section.

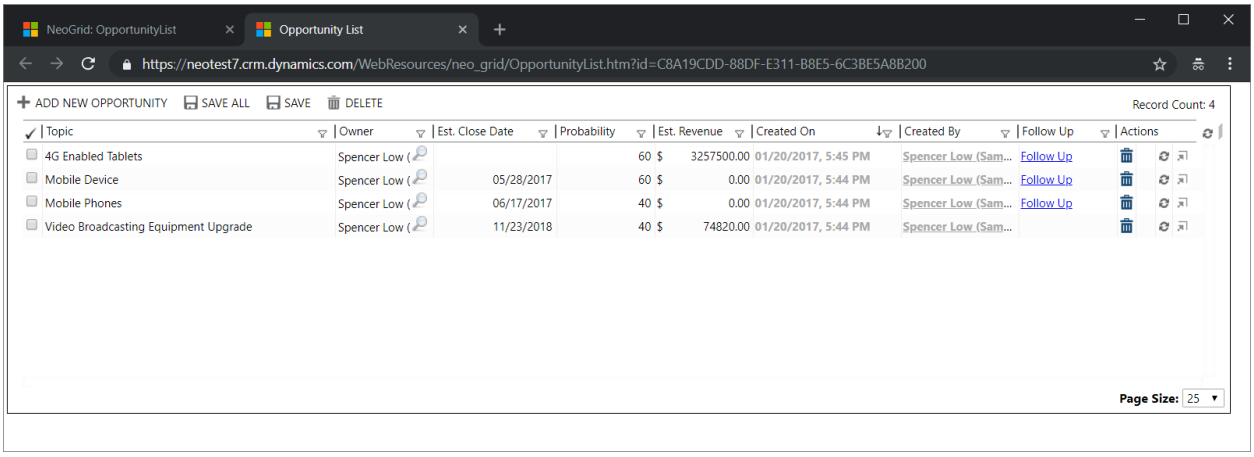


Or alternatively (and more preferably in complex cases), this function can be implemented in another JavaScript web resource, for example /WebResources/neo_helper/OpportunityAlerts.js, which can be referenced by adding to External Javascript Libraries under the Advanced tab.

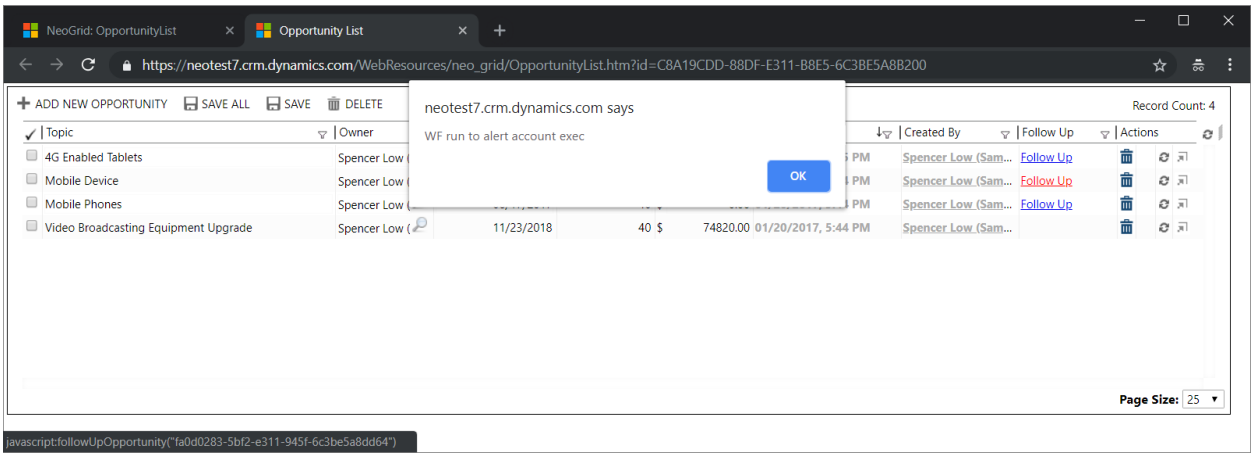


Please refer to section VII for Javascript API. A ***record*** object refers to current row in the grid.

After publishing the grid that is what we see.



When clicked on the “Follow Up” link:


NeoGrid: OpportunityList
Opportunity List
https://neotest7.crm.dynamics.com/WebResources/neo_grid/OpportunityList.htm?id=C8A19CDD-88DF-E311-B8E5-6C3BE5A8B200
ADD NEW OPPORTUNITY
SAVE ALL
SAVE
DELETE
neotest7.crm.dynamics.com says
WF run to alert account exec
OK
Record Count: 4
Topic
Owner
Created By
Follow Up
Actions
4G Enabled Tablets
Mobile Device
Mobile Phones
Video Broadcasting Equipment Upgrade
11/23/2018
40 $
74820.00
01/20/2017, 5:44 PM
Page Size: 25
javascript:followUpOpportunity("fa0d0283-5bf2-e311-945f-6c3be5a8dd64")

VII. JavaScript Event Handling

NeoGrid provides a JavaScript API for client-side event handling.

Object **RecordSet**
Represents a list of Records

Functions

int getCount()
Returns the number of records in the recordSet

Record getRecordByIndex(index)
Returns the Record object within the given index (0-based)

Record getRecordById(id)
Returns the Record object with given id

Object **Record**
Represents one row in the grid

Functions

Attribute getAttribute()
Returns one attribute (cell) of the record

Object **Attribute**
Represents one data cell in the row.

Functions

object getValue()
Returns the value of the attribute

void setValue(object data)
Sets the value of the attribute

void setLooupValue(name, id, type, typeCode)
Sets the value of the lookup type attribute

void setNull()
Sets the attribute null

boolean isDirty()
Returns true, if the attribute is dirty (edited)

boolean isNull()
Returns true, if the attribute value is null

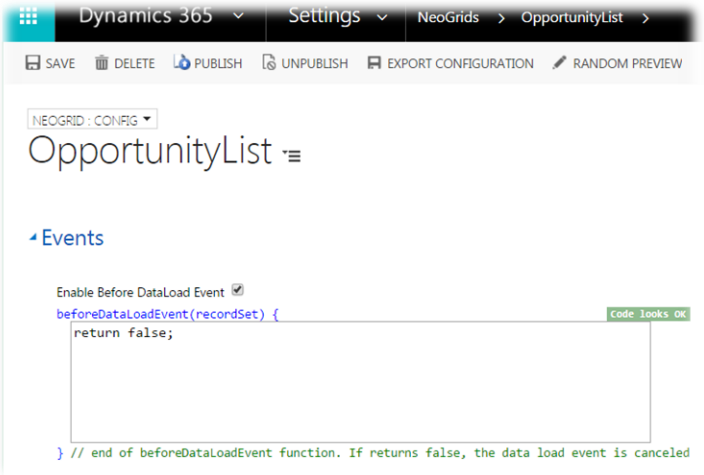
- Grid-Level Events:

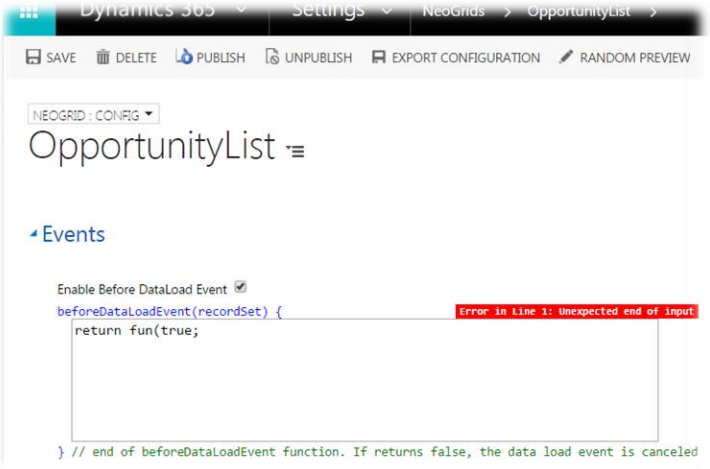
- BeforeDataLoadEvent

This event occurs just before grid data is being loaded from CRM. Input of the function is the current recordset. At this moment, it is possible to stop data being loading by returning false.

```
beforeDataLoadEvent(recordSet) {
   // fill in the function body
}
```

The syntax of the function body is checked, and it gives a message about the error, if there is any.





- AfterDataLoadEvent

This event occurs just after grid data was loaded from CRM. Input of the function is the new recordSet just retrieved from CRM. Returning false from this function, does NOT have any effect on the event flow.

```
afterDataLoadEvent(recordSet) {
   // fill in the function body
}
```

You can use these two events to measure data load time, by starting the timer on before data load event and stopping the timer on after data load event.

- BeforeRenderEvent

  This event occurs just before rendering the grid html. Input of the function is the new recordset. At this moment, it is possible to stop the flow by returning false. You can also manipulate the data just before being displayed.

```
beforeRenderEvent(recordSet) {
   // fill in the function body
}
```

- AfterRenderEvent

  This event occurs just after rendering the grid html is completed.

- BeforeAddLineEvent

  This event occurs when user click on the “Add New Record” button. It is possible the stop adding new line, by returning true. Inputs of the function are the current recordSet and the new record going to be added.

```
beforeAddLineEvent(recordSet, record) {
   // fill in the function body
}
```

- AfterAddLineEvent

  This event occurs after the new record was added to the recordSet. Inputs of the function are the new recordSet and the new record that was just added.

```
afterAddLineEvent(recordSet, record) {
   // fill in the function body
}
```

- BeforeSaveEvent

  This event occurs just before the saving. It is possible to stop saving by retuning false. The data can be manipulated just before saving.

- AfterSaveEvent

  This event occurs after ASYNCHRONOUS save request was made to the CRM server. At this moment, we don’t know yet, if the save was successful or failed.

- SaveCompleted

  This event occurs when all records are saved successfully.

- SaveFailedEvent

  This event occurs when any records is failed to save. Save operation can be to create a new record, or to update an existing record, or to delete an existing record. The inputs of the function are the specific records failed to save, the action (“Create”, “Update” or “Delete”), and the error message returned from the CRM server.

```
saveFailedEvent(record, action, error){
   // fill in the function body
}
```

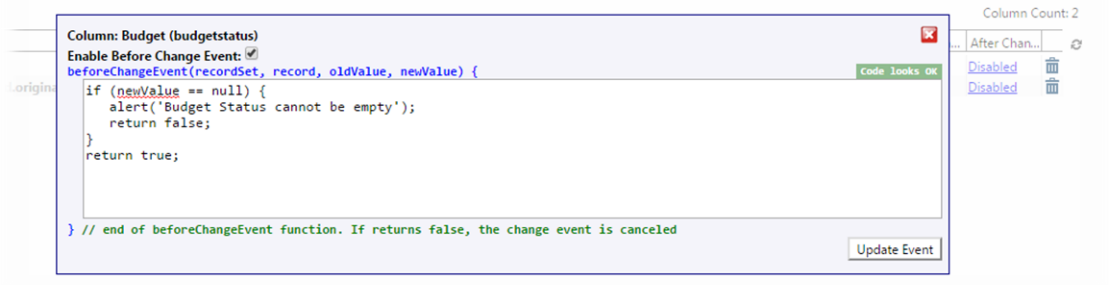
- Cell-Level Events:

- BeforeChangeEvent

This event occurs when a specific attribute of a record changed by the user. At this moment, the change does not apply to recordSet yet, and the change can be canceled by returning false with this function. Data manipulation is also possible. The inputs of the function are the current recordSet and record before update, the old and new values of the field.

```
beforeChangeEvent(recordSet, record, oldValue, newValue) {
    // fill in the function body
}
```

To active the beforeChange event of a column, click on the Disabled button on the Before Change column, and check the box, “Enable Before Change Event”.

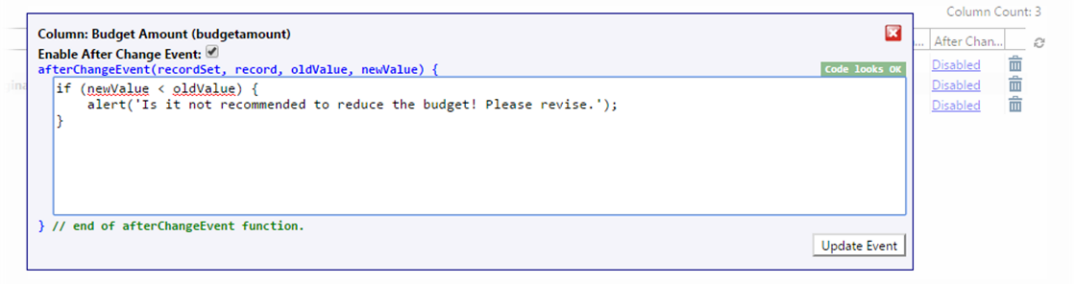


- AfterChangeEvent

This event occurs after a specific attribute of a record changed by the user. The recordSet reflects the change.

```
afterChangeEvent(recordSet, record, oldValue, newValue) {
    // fill in the function body
}
```

To active the afterChange event of a column, click on the Disabled button on the Before Change column, and check the box, “Enable Before Change Event”.

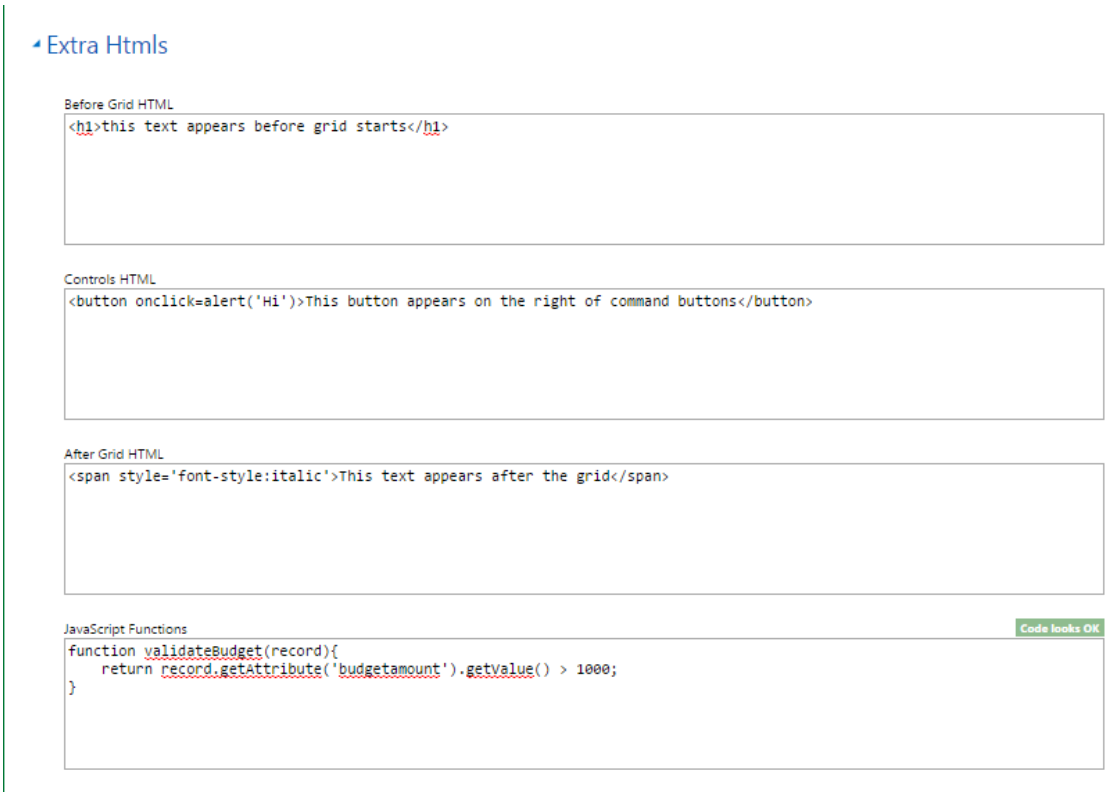


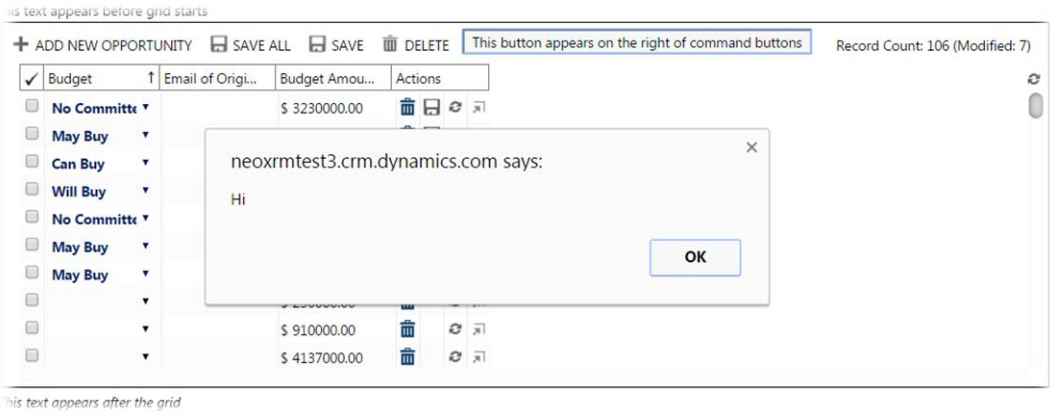
VIII. Custom Filtering

A custom filtering can be applied to fetch XML by providing the filtering criteria into Custom Filtering textbox. Be sure that it is a valid criteria for the entity, otherwise you will receive an error from the server.

IX. Custom JavaScript and Html

It is possible to add custom html code segments and JavaScript code into designated text areas.





X. How to Get Support for NeoGrid

You can always get help for NeoGrid Viewer by contacting to our support team via email at support@neoxrm.com or via phone 1-844-NEOXRM1.