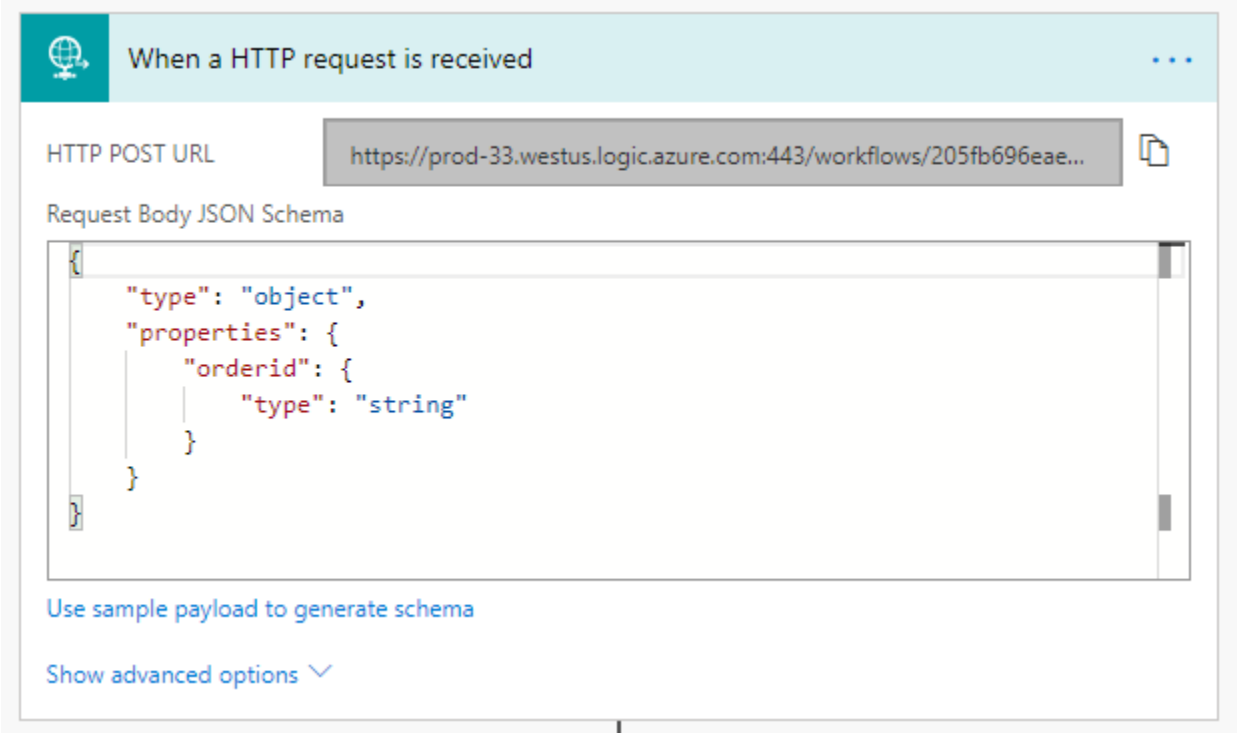
How to Trigger a Flow with a Button in Dynamics 365 Customer Engagement Using an HTTP Request

We recently encountered a client request to migrate orders from Dynamics 365 Customer Engagement to Finance and Operations on demand when the end user clicked a ribbon button. We had been leveraging the Prospect to Cash solution's "Activate" button to simple set the Order to "Active" at which point the PowerApps data integrator would do the heavy lifting. This, however, took too long for certain use cases.

After looking at various options, we decided that leveraging Flow would give us the desired outcome and still leave the process within the D365 ecosystem and allow the customer easy access to modify the solution when needed.

At first, we tried using the Dynamics 365 "When Record is Updated" trigger, but we found the execution to be inconsistent because of the non-specific nature of the update step. So, we started looking at other methods to fire our Flow, and landed on the HTTP Request trigger.

As defined by Microsoft, the HTTP Request trigger is an incoming API call that can trigger the flow. Simply put, this trigger generates a URL the user can then put into whatever function they have to call this Flow on demand.



For our Flow, all we need to get is the GUID value of the order record. So, all we have to do is click “Use sample payload to generate schema,” and add a quick JSON snippet to generate our request schema:

Now that we have our POST URL and schema ready to go, all we have to do is prep it to be added to a JavaScript call within D365 CE.

We used Postman as out test bench to test the call and mock up our JavaScript:

JavaScript Jquery AJAX Copy to Clipboard

```
var settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://prod-33.westus.logic.azure.com:443/workflows
      /205fb696eaed443986e1c662772d9bb5/triggers/manual/paths/invoke?api-version=2016-06-01&sp
      =%2Ftriggers%2Fmanual%2Frun&sv=1.0&sig=kUsLobKsav0hKheWyObOajzidlF0RSpU7Ely46QhraA",
  "method": "POST",
  "headers": {
    "Content-Type": "application/json",
    "cache-control": "no-cache",
    "Postman-Token": "8961268e-99b9-4928-9542-6d04548d294e"
  },
  "processData": false,
  "data": "{\n    \"orderid\": \"21684f93-6cf9-e811-a983-000d3a1bee29\"\n}"
}

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

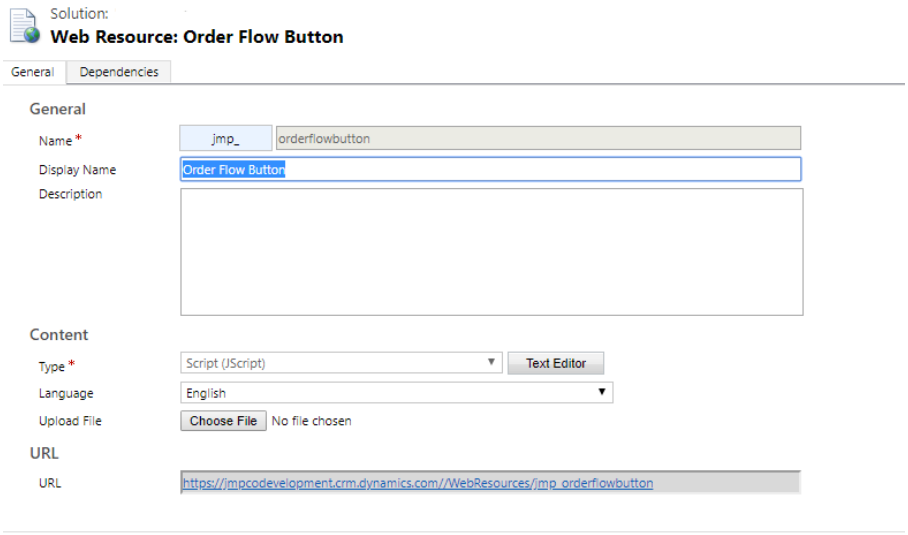
Then, all we had to do was add some logic to dynamically pull the Order ID into the function. First, we give the function a name. In this case we named it “callFlow”. Then, we set a up a variable for the order ID using the Xrm.Page.data.entity.getId() method. The trouble here is that this returns the GUID with braces, which will break the JSON schema, so we have to add a quick snippet to strip the braces in the variable. Lastly, we add the variable to the data string to be consumed by the call:

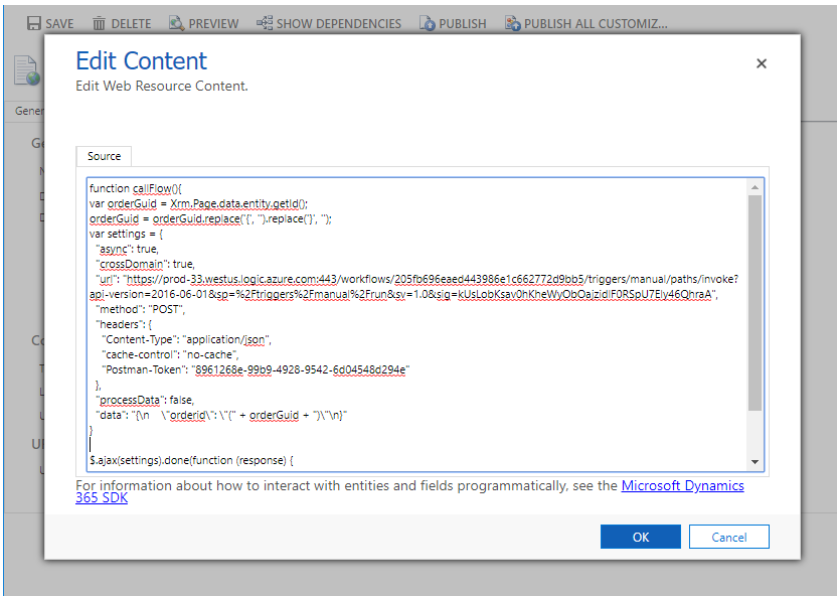
```
function callFlow(){
var orderGuid = Xrm.Page.data.entity.getId();
orderGuid = orderGuid.replace('{', '').replace('}', '');
var settings = {
  "async": true,
  "crossDomain": true,
  "url": "https://prod-33.westus.logic.azure.com:443/workflows/
  "method": "POST",
  "headers": {
    "Content-Type": "application/json",
    "cache-control": "no-cache",
    "Postman-Token": "8961268e-99b9-4928-9542-6d04548d294e"
  },
  "processData": false,
  "data": "{\n    \"orderid\": \"(" + orderGuid + ")\"\n}"
}

$.ajax(settings).done(function (response) {
  console.log(response);

});
}
```

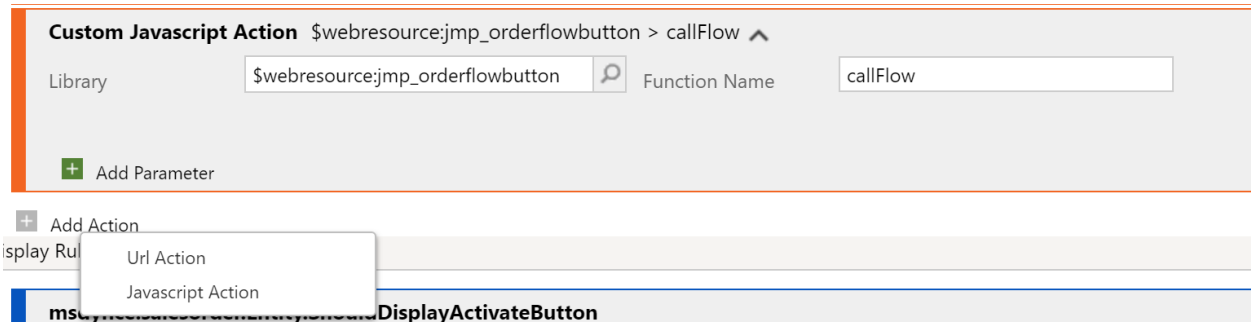
With the JavaScript all cleaned, it needed to be added to a web resource in our solution:



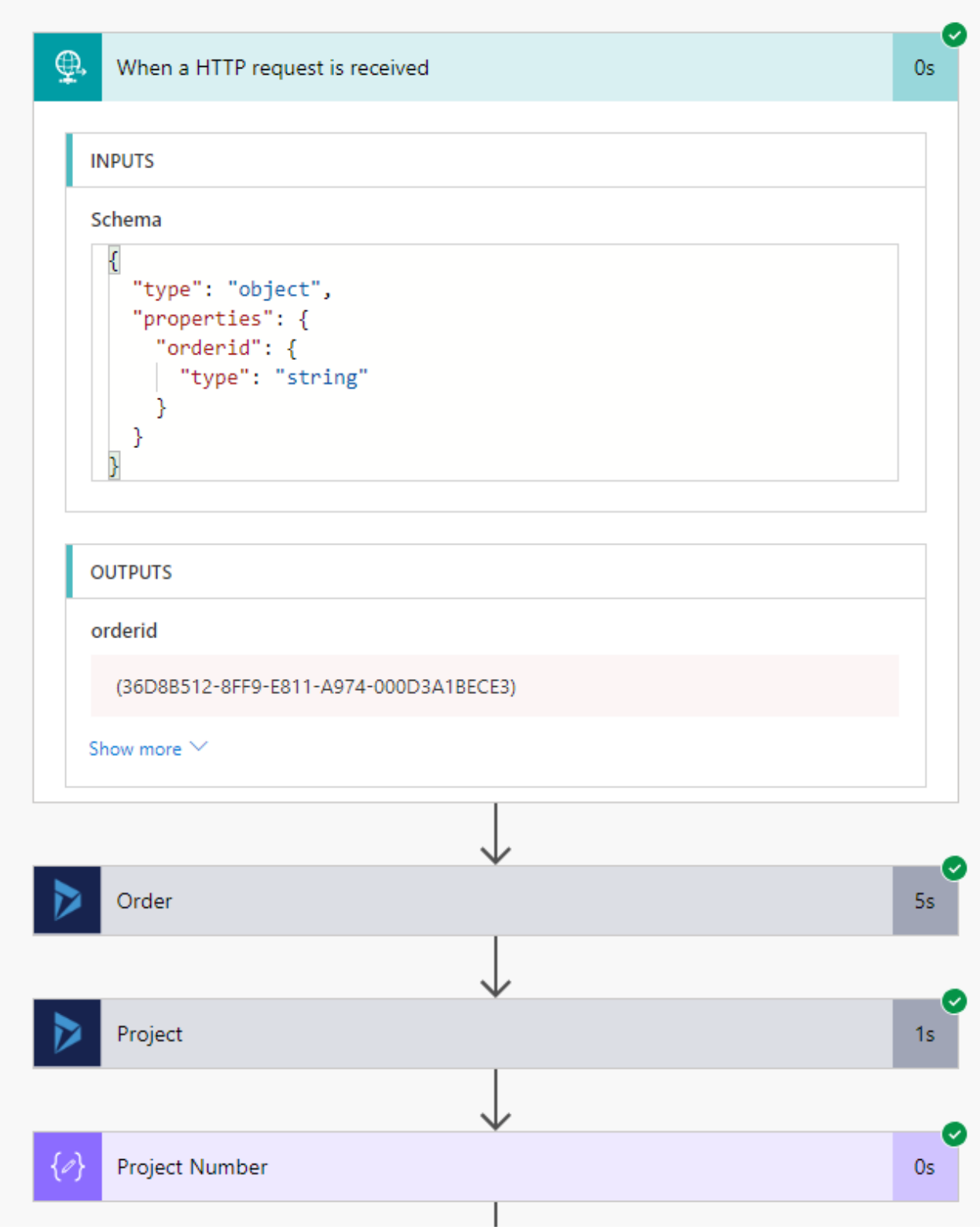


The next thing to do was add the JavaScript Web Resource to the button we already have using the Ribbon Workbench. If you need to add a button, refer to Develop1's quick start guides for help.

After launching the ribbon workbench, select the button you want to edit, or create a new one. In the button's command, click the "Add Action" button, and select "JavaScript Action". Select the JavaScript library containing your API call, and enter the name of the function:



When this is done, publish the updated ribbon, and you'll be able to successfully call your Flow on demand.



This is about as simple as it gets, but you can always expand your schema and JavaScript to pull in as much data as you want from the record and into the Flow. This is a very versatile method of integrating data quickly with the click of a button.