Disclaimer: This document and instruction should be used while there is no officially supported Salesforce/Meta integration for sending Conversion Lead signals to Meta. This is intended as a temporary solution for advertisers who cannot implement conversion leads integration using a manual or third party solution. If and when Salesforce and Meta create an official solution, our recommendation would be to disable this solution (by turning off the Lead object trigger), and install the official Salesforce/Meta solution.

Purpose:

Setting up Salesforce to automatically send Conversions API Updates for Conversion Leads

Background:

To enable the Conversion Leads Integration and Optimization, advertisers need to send Lead funnel updates back to Meta when a lead is created/entered within their CRM, and when the lead changes stages/statuses. For Salesforce, you can either write a manual/direct integration, or you can use a third party partner like Zapier or Leadsbridge to accomplish this. An alternative method would be to write code directly within Salesforce, using Apex Triggers, to send the API push updates to Meta as soon as a lead is created and updated.

Requirements:

1. Salesforce must be automatically importing Leads, either through a webhook integration, the Salesforce Lead Capture plugin, or through a third party.
2. Lead object in Salesforce must have a field to store the Facebook Lead ID.
    a. The Salesforce Lead Capture app is commonly used to automatically ingest leads from Facebook to Salesforce, and it creates a field called: **External_Lead_ID**

       This field however, saves the lead ID in a format that needs to be modified for Facebook to understand it. Currently, it is saved like this: "FB-906476827164880", however the "FB-" part of the lead ID must be removed when making updates to Facebooks API. Examples will be shown below.
3. Valid Facebook Pixel ID
4. Valid Facebook Conversions API Token

Steps to implement:

1. Create Util class for sending Conversions API lead update event to Facebook

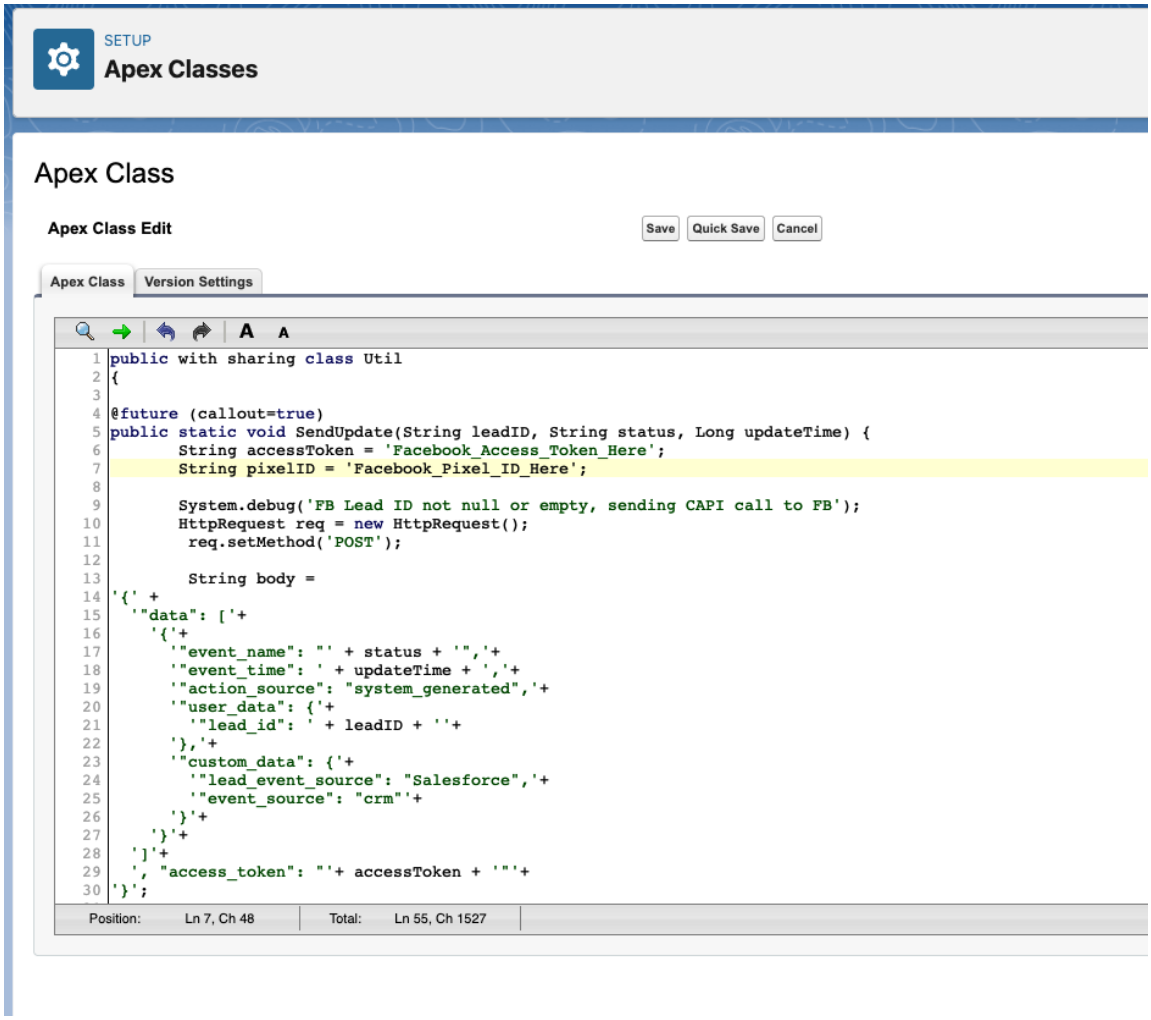
    a. Go to Salesforce -> Setup -> Custom Code -> Apex Classes
    b. Click on New to create a new apex Class

c. Paste the following code
d. Update the values in accessToken and pixelID to match your configuration

Apex Class Code:

```
public with sharing class Util
   {

   @future (callout=true)
   public static void SendUpdate(String leadID, String status, Long
   updateTime) {
          String accessToken = 'Facebook_Access_Token_Here';
       String pixelID = '1234';

          System.debug('FB Lead ID not null or empty, sending CAPI call
   to FB');
          HttpRequest req = new HttpRequest();
           req.setMethod('POST');

           String body =
   '{' +
     '"data": ['+
      '{'+
       '"event_name": "' + status + '",'+
       '"event_time": ' + updateTime + ','+
       '"action_source": "system_generated",'+
       '"user_data": {'+
         '"lead_id": ' + leadID + ''+
       '},'+
       '"custom_data": {'+
         '"lead_event_source": "Salesforce-Apex",'+
         '"event_source": "crm"'+
       '}'+
      '}'+
     ']'+
     ', "access_token": "'+ accessToken + '"'+
   '}';

   System.debug(body);



          req.setHeader('Content-Type','application/json');
```

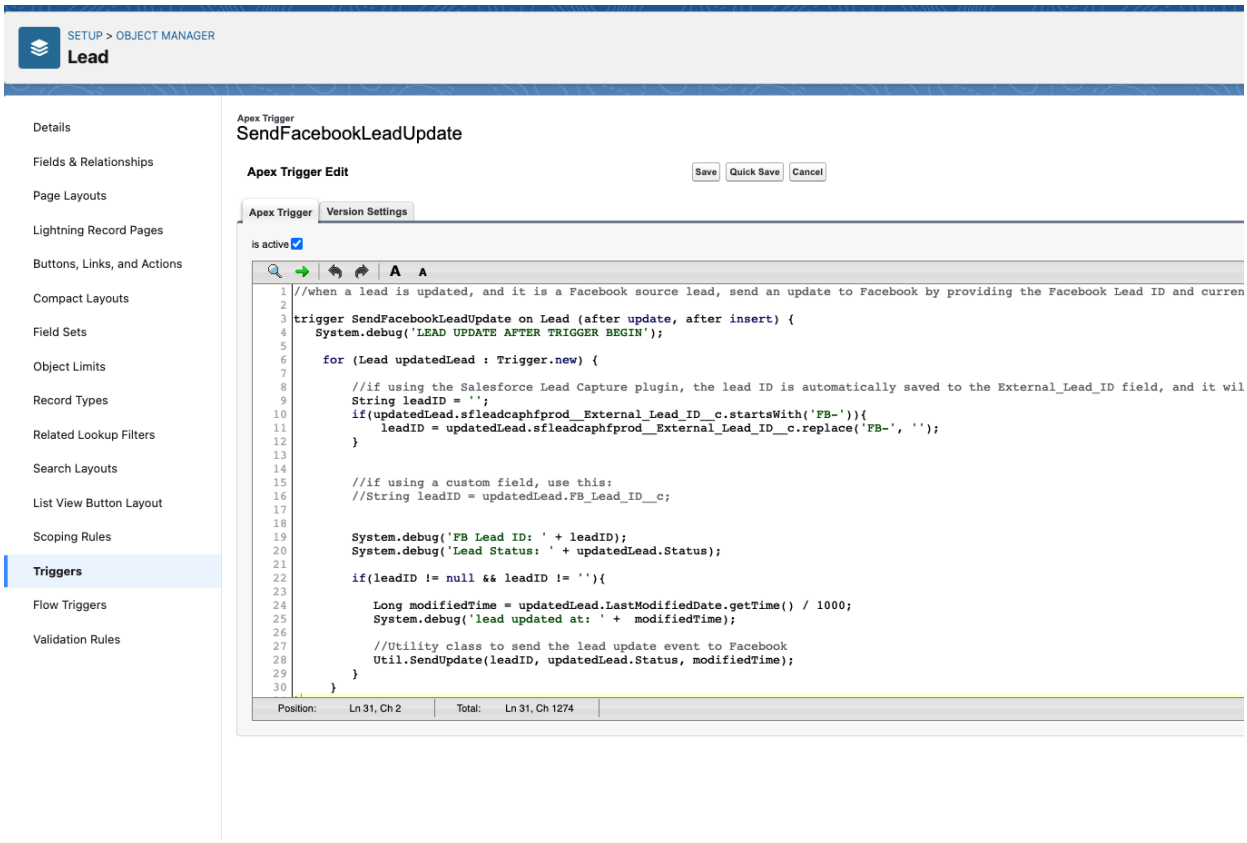
```
        req.setEndpoint('https://graph.facebook.com/v15.0/'+ pixelID
    + '/events');
        req.setBody(body);

        Http http = new Http();
        try{
            System.debug('sending request');
            HTTPResponse res = http.send(req);
            //Executing web service call
            System.debug('STATUS:' + res.getStatus());
            System.debug('STATUS_CODE:' + res.getStatusCode());
            System.debug('Response:' + res.getBody());
        }
        catch(System.CalloutException e){
            //Exception handling goes here..
            system.debug(e);
        }
    }

    }
```



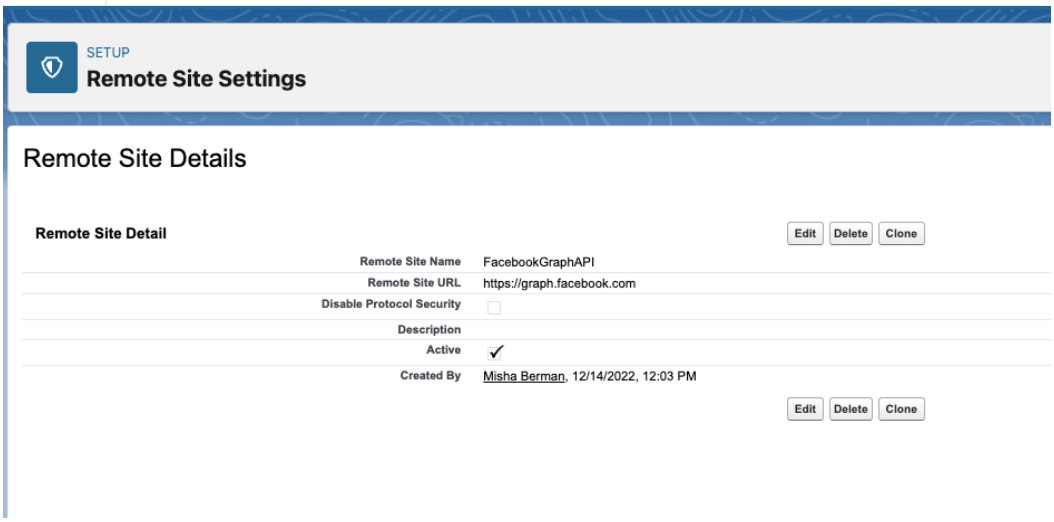
2. Create Apex Trigger that sends Facebook a lead update event, whenever a Facebook ingested lead has changed stages
    a. Go to Salesforce -> Setup -> Objects and Fields -> Object Manager -> Lead
    b. Click on Triggers
    c. Click New to create a new Trigger
        i. In the code below, there are 2 ways you can get the Lead ID, either using the field which Salesforce Lead Capture plugin automatically maps to the Salesforce Lead ID (sfleadcaphfprod__External_Lead_ID__c), or using another field that you defined manually. If using the sfleadcaphfprod__External_Lead_ID__c field, to check if it is a lead that came from Facebook (as opposed to Google), you need to see if it starts with “FB-”, and if it does, make sure to send that lead ID back to Facebook without the “FB-”. The code below shows how to check and strip the value so that it is formatted correctly.

Apex Trigger Code:

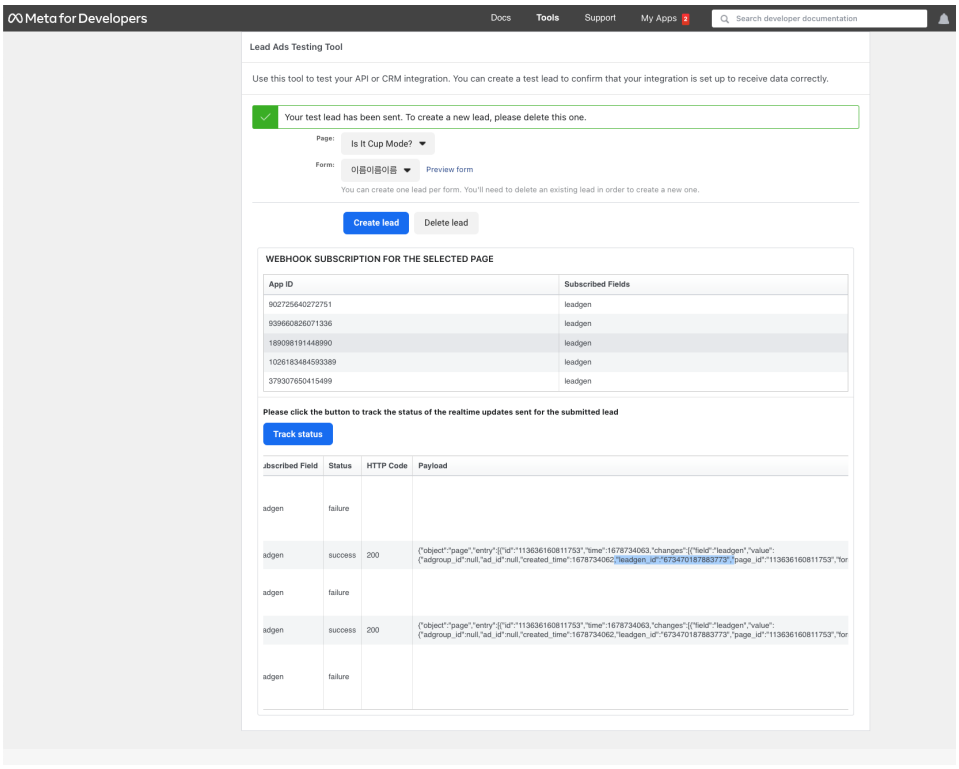
```
//when a lead is updated, and it is a Facebook source lead, send an update
to Facebook by providing the Facebook Lead ID and current lead status/stage

trigger SendFacebookLeadUpdate on Lead (after update, after insert) {
   System.debug('LEAD UPDATE AFTER TRIGGER BEGIN');

    for (Lead updatedLead : Trigger.new) {

        //if using the Salesforce Lead Capture plugin, the lead ID is
automatically saved to the External_Lead_ID field, and it will start with
FB- if it is a Facebook lead
        String leadID = '';

if(updatedLead.sfleadcaphfprod__External_Lead_ID__c.startsWith('FB-')){
            leadID =
updatedLead.sfleadcaphfprod__External_Lead_ID__c.replace('FB-', '');
        }


        //if using a custom field, use this:
        //String leadID = updatedLead.FB_Lead_ID__c;


        System.debug('FB Lead ID: ' + leadID);
        System.debug('Lead Status: ' + updatedLead.Status);

        if(leadID != null && leadID != ''){

           Long modifiedTime = updatedLead.LastModifiedDate.getTime() /
1000;
           System.debug('lead updated at: ' +  modifiedTime);

           //Utility class to send the lead update event to Facebook
           Util.SendUpdate(leadID, updatedLead.Status, modifiedTime);
        }
     }
}
```



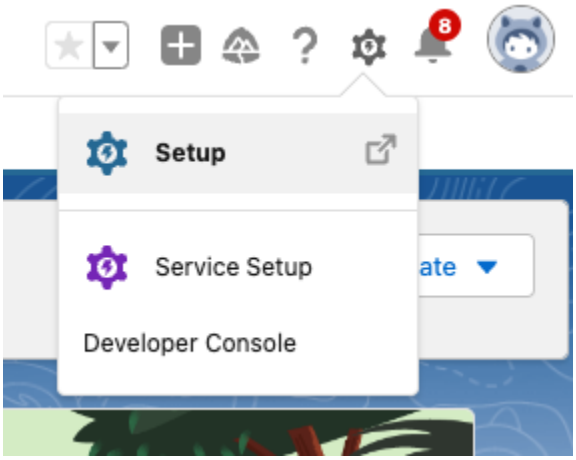
3. **Enable Remote Site Details**, so that outgoing API connections to graph.facebook.com are allowed
    a. Go to Setup -> Security -> Remote Site Settings
    b. Click on **New Remote Site**
    c. For Remote Site Name, enter “FacebookGraphAPI”
    d. For Remote Site URL, enter “https://graph.facebook.com”
    e. Click Save



4. Test the code in the **Developer Console** to make sure updates are being sent whenever a new lead is created, and when that lead is updated
    a. Get a valid, existing, Facebook Lead ID from one of your current imported leads.
        i. You can also use the Lead Ads Testing Tool from Facebook to generate a new lead. After you pick a page/form and click **Create lead**, click on **Track status** a few times until you can see the Payload of a new lead, and copy the leadgen_id value (should be a 15-16 digit number, for example `673470187883773`)



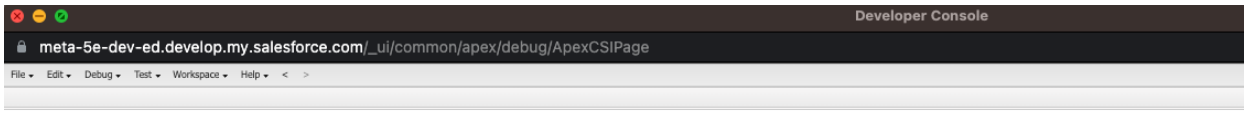
b. Click on the Setup gear wheel at the top right of Salesforce, and go to Dev Console

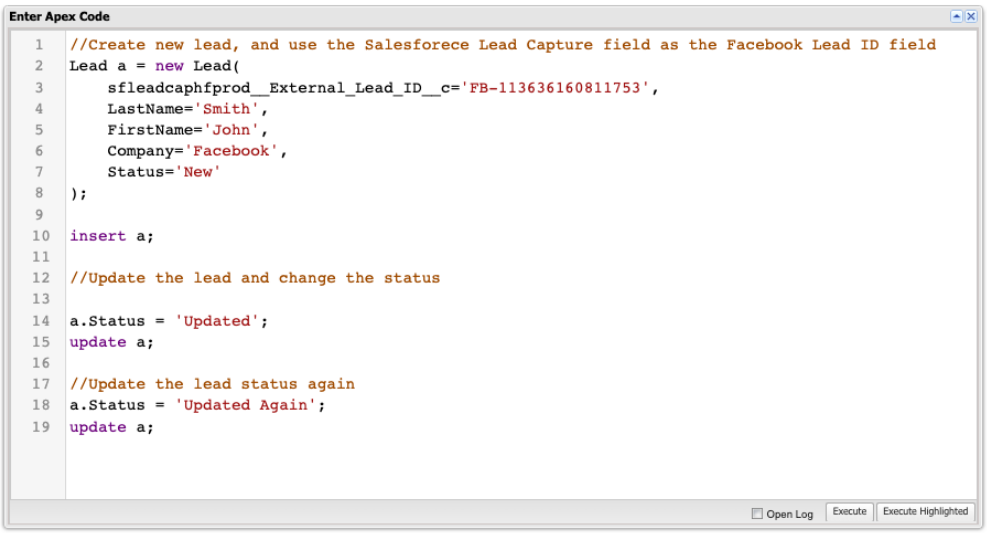


c. In the Debug menu, click on **Open Execute Anonymous Window**
d. Type the following Code which will simulate creating a new lead, and then updating it
    i. Note, make sure to match whatever lead ID field you setup in the Lead Apex Trigger in this test code (either use the sfleadcaphfprod__External_Lead_ID__c field or your custom predefined

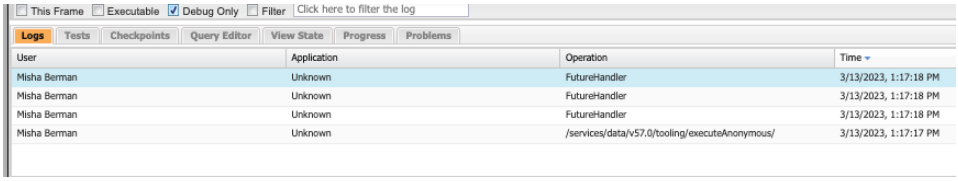
Facebook lead ID field)

```
//Create new lead, and use the Salesforece Lead Capture field as the
Facebook Lead ID field
Lead a = new Lead(
      sfleadcaphfprod__External_Lead_ID__c='FB-673470187883773',
    LastName='Smith',
    FirstName='John',
    Company='Facebook',
    Status='New'
);

insert a;

//Update the lead and change the status

a.Status = 'Updated';
update a;

//Update the lead status again
a.Status = 'Updated Again';
update a;
```

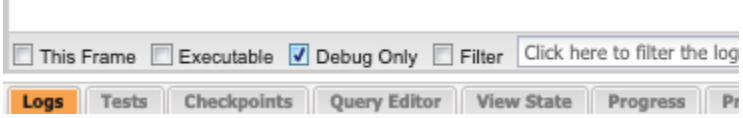




e. Click on **Execute** to run the code
f. View the logs to make sure the API calls were sent. You should see 3 successful API calls, 1 for the newly created lead, and 2 for the updates to the statuses
    i. Double click on the first line in the Logs window, it should have the Operation Column as /services/data/v57.0/tooling/**executeAnonymous**

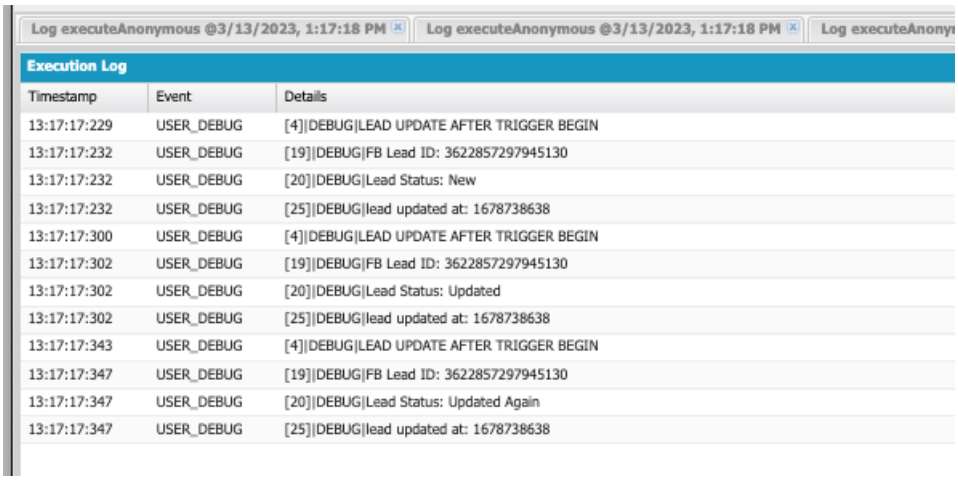


    ii. Click on **Debug Only** in the filters so that you can see the debug messages from the trigger and class



You should see something like this, which shows the lead being created

and then being update twice



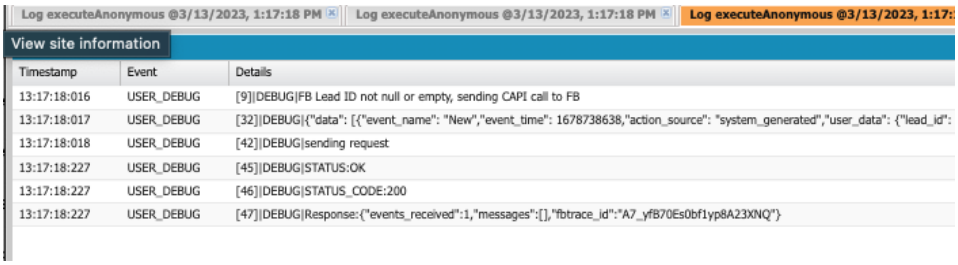
Timestamp	Event	Details
13:17:17:229	USER_DEBUG	[4]|DEBUG|LEAD UPDATE AFTER TRIGGER BEGIN
13:17:17:232	USER_DEBUG	[19]|DEBUG|FB Lead ID: 3622857297945130
13:17:17:232	USER_DEBUG	[20]|DEBUG|Lead Status: New
13:17:17:232	USER_DEBUG	[25]|DEBUG|lead updated at: 1678738638
13:17:17:300	USER_DEBUG	[4]|DEBUG|LEAD UPDATE AFTER TRIGGER BEGIN
13:17:17:302	USER_DEBUG	[19]|DEBUG|FB Lead ID: 3622857297945130
13:17:17:302	USER_DEBUG	[20]|DEBUG|Lead Status: Updated
13:17:17:302	USER_DEBUG	[25]|DEBUG|lead updated at: 1678738638
13:17:17:343	USER_DEBUG	[4]|DEBUG|LEAD UPDATE AFTER TRIGGER BEGIN
13:17:17:347	USER_DEBUG	[19]|DEBUG|FB Lead ID: 3622857297945130
13:17:17:347	USER_DEBUG	[20]|DEBUG|Lead Status: Updated Again
13:17:17:347	USER_DEBUG	[25]|DEBUG|lead updated at: 1678738638

iii. Next, click on each of the **FutureHandler** entries in the Logs (make sure you still have debug filtering on)

You should see 3 events showing that there was an event name sent to Facebook called ‘New’, ‘Updated’, and ‘Updated Again’.

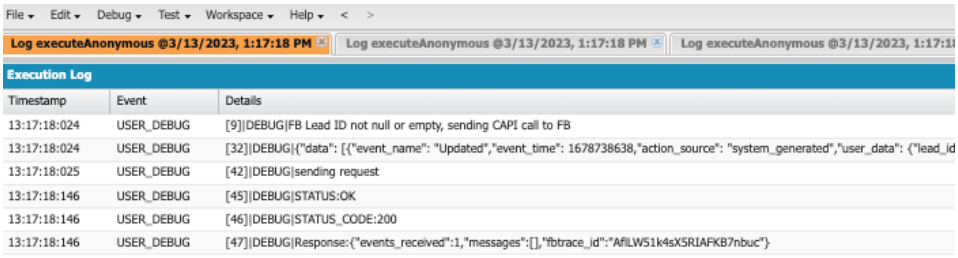
Each should also show a status code of OK/200, meaning it was successful

New:



Timestamp	Event	Details
13:17:18:016	USER_DEBUG	[9]|DEBUG|FB Lead ID not null or empty, sending CAPI call to FB
13:17:18:017	USER_DEBUG	[32]|DEBUG|{"data": [{"event_name": "New","event_time": 1678738638,"action_source": "system_generated","user_data": {"lead_id":
13:17:18:018	USER_DEBUG	[42]|DEBUG|sending request
13:17:18:227	USER_DEBUG	[45]|DEBUG|STATUS:OK
13:17:18:227	USER_DEBUG	[46]|DEBUG|STATUS_CODE:200
13:17:18:227	USER_DEBUG	[47]|DEBUG|Response:{"events_received":1,"messages":[],"fbtrace_id":"A7_yfB70Es0bf1yp8A23XNQ"}

Updated:



Updated Again: