i

Data Activation 3rd Party Integration – Facebook

Instructions

Version 1.4.2 8th October 2020

Contents

1 Introduction 4

1.1 Purpose of this Document 4

1.2 Target Audience 4

1.3 Considerations 4

2 Common Use Cases 5

2.1 To enhance re-targeting capabilities for known customer 5

2.2 To build a Unified customer data view 5

3 Design Concepts 6

3.1 What is retargeting? 6

3.2 Customer Data Platform (CDP) 6

3.2.1 Identity Mapping 6

3.2.2 What is Pixel tracking 7

3.2.3 What is Server-to-Server tracking 7

4 Identity Mapping 9

4.1 SAS Support Code for FB 9

4.2 Custom Event for identity mapping 9

5 Pixel Integration (Client Side) 13

5.1 Requirements 14

5.1.1 Add the Facebook pixel to the web (known user) 14

5.1.2 SAS Support Code for FB 14

5.1.3 CI360 Tag Deployed 14

5.2 CI360 Controlled or Tag Manager Intermediator Pixel Calls 15

5.2.1 Create Spot in CI360 15

5.2.2 Create Custom Event 16

5.2.3 Create creative with FB pixel (CI360 Controlled, Option #1) 19

5.2.4 Create creative with Tag Manager rule code (Tag Manager Intermediate) 19

5.2.5 Define Web Task 20

5.2.6 Trigger FB Pixel via SAS Creative 24

5.3 Tag Manager/Framework Controlled Pixel 24

5.3.1 Create Custom Event 24

5.4 Verification 26

5.5 Example 28

5.5.1 EXAMPLE: USE FB EVENT IN OTHER PLATFORMS 28

6 API (Server-to-Server) 29

6.1 Requirements 29

6.1.1 Add the Facebook pixel to the web (known user) 29

6.1.2 SAS Support Code for FB 29

6.1.3 CI360 Tag Deployed 29

6.2 Connector setup 30

6.3 Event setup 32

6.4 Task Setup 34

6.5 Verification 35

7 Appendix 36

7.1 External References 36
7.2 Source Code 36
7.2.1 Facebook Pixel Example: 36
7.2.2 Facebook API Endpoint 37
7.2.3 SAS CI360 Tag Example Code 37
7.2.4 SAS Generic Pixel Placeholder 37
7.2.5 SAS Facebook Placeholder 37
7.3 Connector JSON Payloads 38

1 Introduction

1.1 Purpose of this Document

The purpose of this document is to

guide in the integration of CI360 with Facebook products. This includes sending user identifiers after segmentation by CI360 platforms.

1.2 Target Audience

Those who will be implementing Facebook integrations for Customer Intelligence 360

1.3 Considerations

Platforms change and due to the fast changing nature of APIs and technology be sure to read the Facebook documentation to understand what is needed as some things might have changed.

2 Common Use Cases

There are two main use cases for data activation on 3rd Party platforms:

2.1 To enhance re-targeting capabilities for known customer

Digital Advertising can reach very large audiences in both new and known visitors. It is great way to tell brand stories at scale and in context through ads on various devices or channels. However, digital campaigns may waste or miss the target audience without a re-targeting solution to manage the customer experience. The combination of third-party platform and first-party data will enrich the re-targeting capabilities for digital campaigns which run off site and increase marketing ROI through improved relevance and accuracy.

2.2 To build a Unified customer data view

A unified customer data view combines all the data points of visitors from data collected off site (third-party) and on site (first-party). These data are able to give a better picture of known visitors and to provide the right offer in the right moment in time regardless they were looking advertisement from the internet or navigating within the website. The unified customer data view is capable of enriching re-targeting capabilities for online advertising and marketing campaign as well as in site next best action offering.

3 Design Concepts

3.1 What is retargeting?

Retargeting is a method for putting the organization's brand in front of consumers after they have left the website or app and persuading them to reconsider an offer or value proposition from the brand. It often refers to reaching out to lost visitors by Display Ads and its typically performed by using the third-party platform's event (tracking pixel and cookie) to achieve.

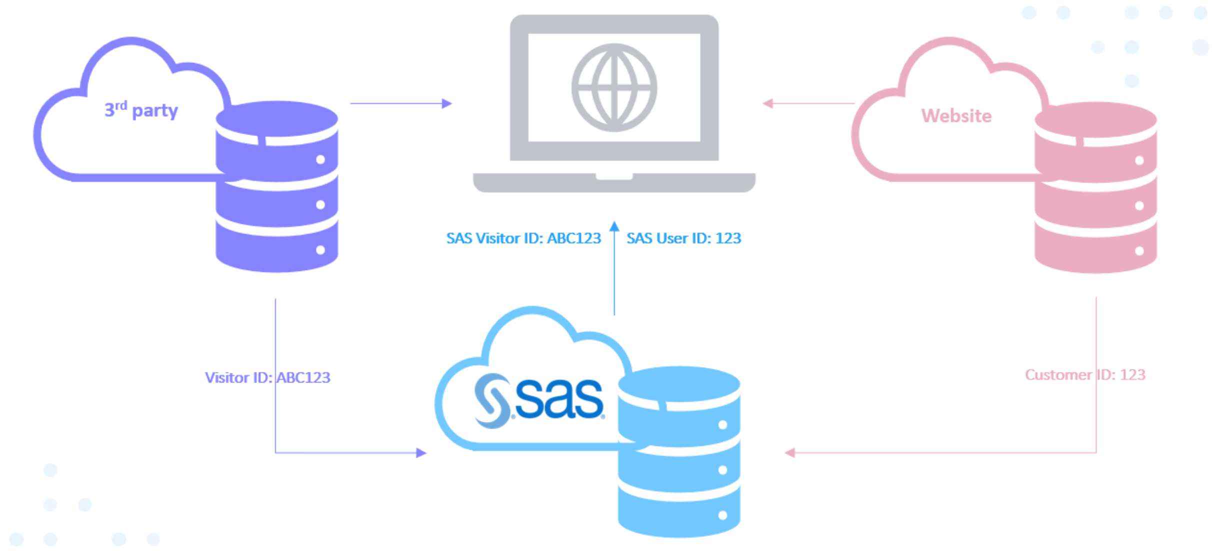
3.2 Customer Data Platform (CDP)

In this article, we are going to use CI360 to combine 1st party data (online and offline), marketing data, and third-party data. Hence, CI360 can be act as CDP to provide unified customer data view for enrich the re-targeting for online campaign which run off-site platform (such as Google Ads), and there are two steps and three methods to achieve this:

1. Identity Mapping (between each platform solutions)
2. Sending Data to third-party platform:
    - Pixel Tracking
    - Server-to-Server Tracking

3.2.1 Identity Mapping

The following diagram shows how marketing tool act as "middle agent" to map identifier and later to exchange data between organization website / app, marketing automation tool as well as third-party platform:

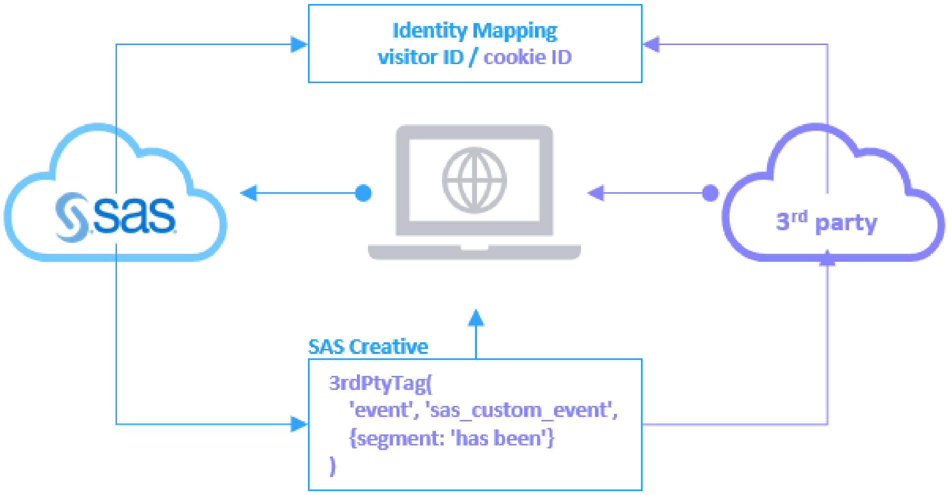


In CI360, the identity mapping can be done by Custom Event with additional attribution, it will be cover in later sections.

3.2.2 What is Pixel tracking

Pixel tracking are snippets of code provided from third-party platform that are loaded on the website, in specific pages, to track visitors in-site or in-app behavior. The code is usually executed on client side (visitor's web browser or app) which sending the information along with unique cookie ID / visitor ID to third-party platform.

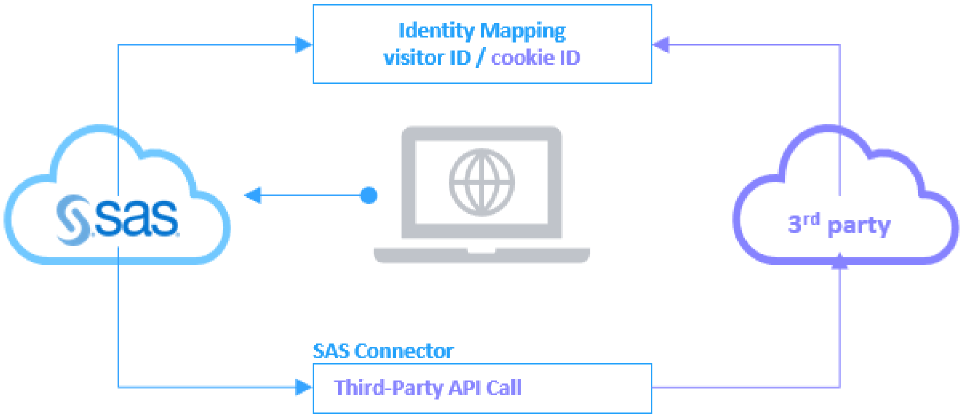
In this offering, we are going to use pixel integration via SAS CI360 Creative to serve third-party tracking code.   This approach is different to traditionally third-party tracking which only embedded in the specified page; CI360 will decide the right moment of the customer journey to fire the event or share the segment.



3.2.3 What is Server-to-Server tracking

Server-to-Server tracking is using APIs call to fire tracking event directly from the solution backend to third party platform which extended the tracking beyond visitor exited the website or app.   It is also enhancing the site security as server to serve tracking doesn't require any third-party coding to be installed on the site and tracking event can be control and triggered by solution owned by the organizations.

In this offering, we are going to use Server-Side API framework (Connector) to execute third-party tracking server call.   This approach allows to share event / segment without embedded third-party code on the website site to maintain security / data privacy.   As well as, for the visitor beyond the site exit where pixel integration unable to reach out.


Identity Mapping
visitor ID / cookie ID
sas
3rd party
SAS Connector
Third-Party API Call

4 Identity Mapping

In order to get Pixel integration and Server-to-Server (API) with third-party platforms, the first step is mapping third-party platform ID with SAS ID.

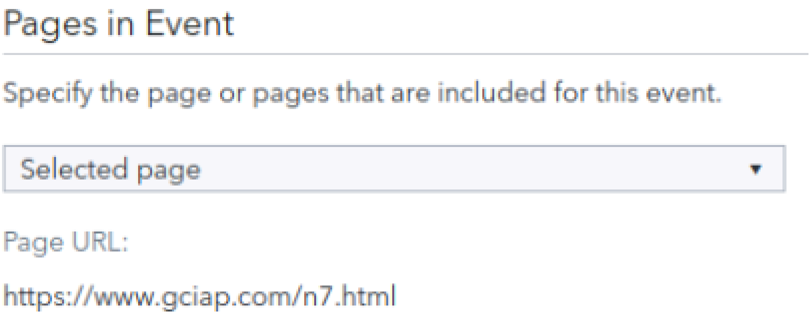
4.1 SAS Support Code for FB

In contrast to other similar integrations we must use some support code to collect Facebook identifier information with SAS CI360. This allows for collections of the pixel identifiers.

4.2 Custom Event for identity mapping

Here we will create a Custom Event, this is required as the code used in our creative to trigger the pixel will use merge tags in order to replace information to be sent to FB. The instructions will provide how to create an event and which data will be sent to FB but you can create your own event with different criteria and naming.

a. From navigation bar, click **Events**
b. Click the button for create a new event
c. Select **Custom Event**
d. Select **Page View** as event type
e. Enter the **Page URL** for re-targeting
f. Click **Create Event**
g. In the **Event Details** Tab
   i. Go to **Event Configuration**
      1. To confirm the site is correct:



Pages in Event

Specify the page or pages that are included for this event.

Selected page ▾

Page URL:

https://www.gciap.com/n7.html

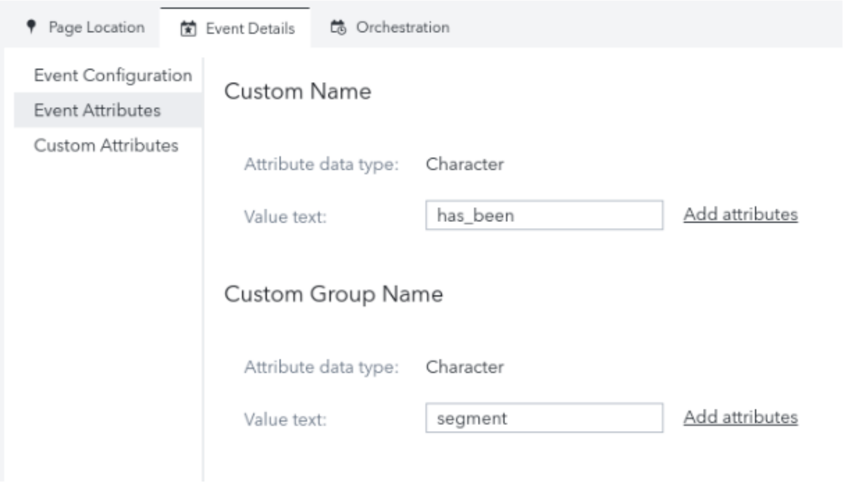
   ii. Go to **Event Attributes**
      1. In Custom Name's value text: type in "**has_been**":

      This is a custom value that will be collected in Facebook and stored as custom parameter field defined as below.

      2. In Custom Group Name's value text: type in "**segment**":

This is a custom value will be used as custom parameter field name in FB event's custom data

iii. The configuration should look similar to the below:



Custom Name

Attribute data type: Character

Value text: has_been

Custom Group Name

Attribute data type: Character

Value text: segment

iv. Create a new Custom Attribute by clicking ⊕

v. Choose **Page Details > JavaScript Variable**

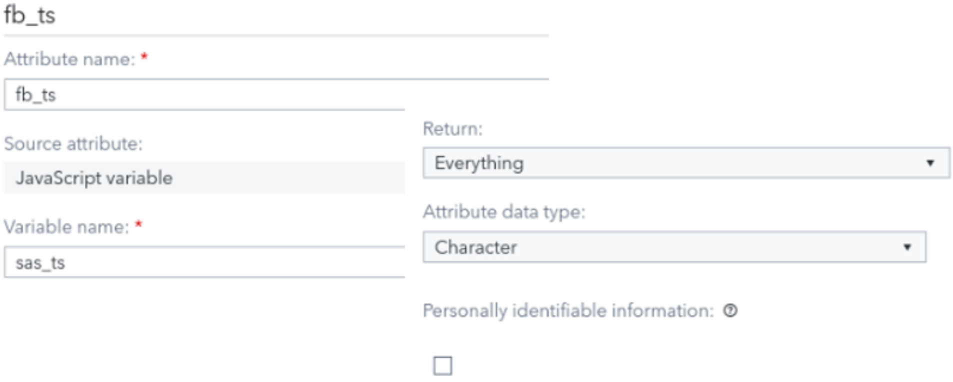
vi. Set the attribute name to **fb_ts**

vii. Variable name to **sas_ts**

viii. Return: **Everything**

ix. Attribute data type: **Character**

x. Personally identifiable information: **Unchecked** (note that we keep this unchecked so we can connect other information from other systems and connect it with our own identifiers if needed later in our data)



fb_ts

Attribute name: * fb_ts

Source attribute: JavaScript variable

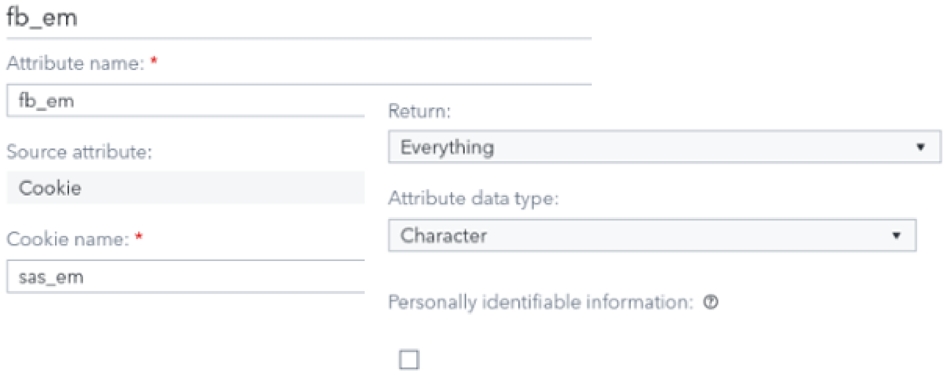
Variable name: * sas_ts

Return: Everything

Attribute data type: Character

Personally identifiable information: ⓘ

□



fb_em

Attribute name: *
fb_em

Source attribute:
Cookie

Cookie name: *
sas_em

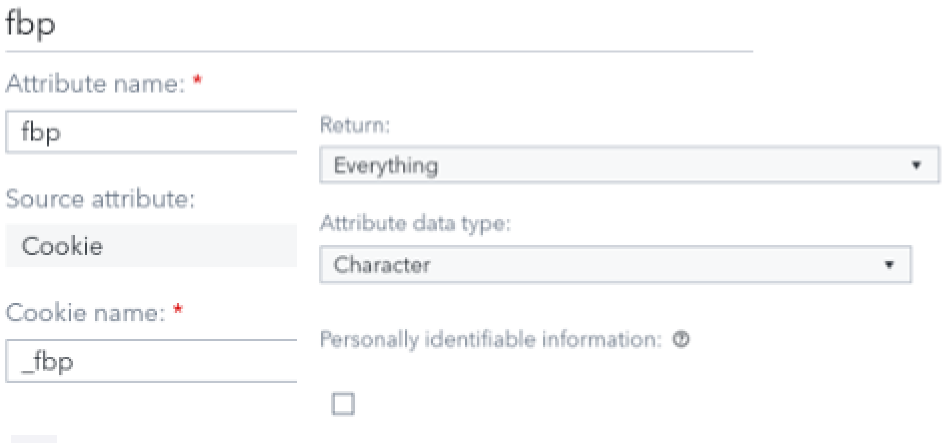
Return:
Everything ▾

Attribute data type:
Character ▾

Personally identifiable information: ⓘ
□

fb_ts are coming from the "SAS Support Code" and they designed for data mapping and API integration which will be covered in later sections.

xi. Create a new Custom Attribute by clicking ⊕
xii. Choose **URL Details > Cookie**
xiii. Set the attribute name to **fbp**
xiv. Variable name to **_fbp**
xv. Return: **Everything**



fbp

Attribute name: *
fbp

Source attribute:
Cookie

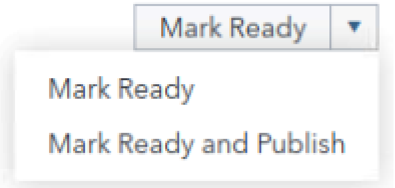
Cookie name: *
_fbp

Return:
Everything ▾

Attribute data type:
Character ▾

Personally identifiable information: ⓘ
□

h. click the 💾 button to save the Custom Event
i. type in "**Event: FB Pixel**" (this is just a placeholder name, you can use any event you want), then click **Save**
j. Go to the **Orchestration** Tab
    i. Click on **Mark Ready and Publish** from drop-down menu:



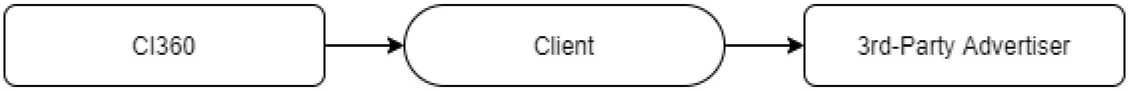
ii. Click **Yes** when prompted by **Publishing the item will change the status to Scheduled or Active**

5 Pixel Integration (Client Side)

Facebook (FB) is using the Facebook Pixel Code to collect web and similar method for native app (ios and android) with Facebook SDK. This information can be formed into group of users (FB called ***Custom Audience****) as ad targeting criteria in Facebook Ads. This document attempts to standardize the process of SAS Ci360 to control FB event pixel. This also assumes that there is one CI360 tenant and one FB instance, some code might need to be modified for testing environments or otherwise.*

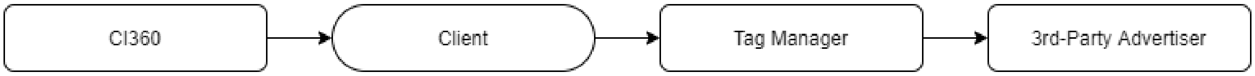
Planning is essential with Pixel Integration as there are many ways to accomplish this. With the changes to allow for JavaScript spots we can now deploy snippets of JavaScript code directly into spots but users may opt to centralize all 3rd party JavaScript into one platform like Google Tag Manager or Adobe Launch, etc.

1. *CI360 Controlled*



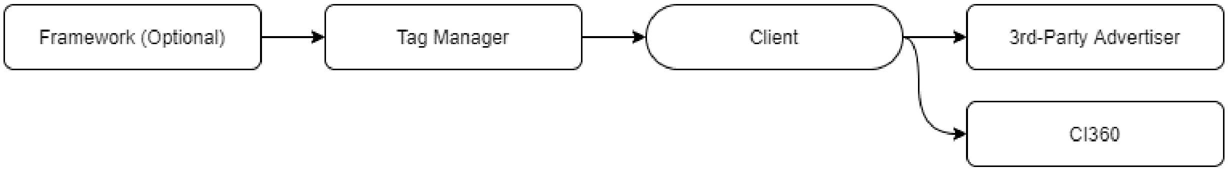
JavaScript integrations are made and hosted by CI360 and implemented directly into the spots. Implementation this way only requires that we have CI360 installed on the website and it is collecting data data.

2. *Tag Manager Intermediator*



CI360 is used to host code that then calls on a Tag Manager rule to fire. This can be beneficial to allow the tag manager, such as Google Tag Manager or Adobe Launch/DTM to manage all 3rd party scripts executed in a single platform which is very commonly used these days. Adding on another script can be simplified and governance policies are easier to manage with already existing processes.

3. *Tag Manager/Framework Controlled (Using CI360 JS API)*

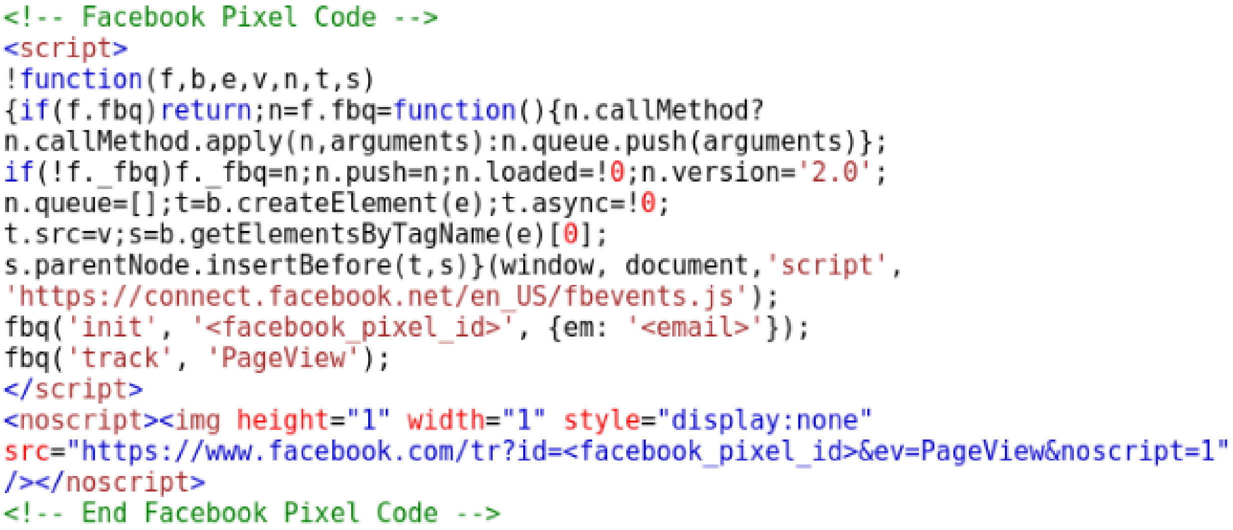


In this instance, you build rules either using an 1st party system or in the Tag manager which then delivers to 3rd Party Advertiser at the same time. The Framework rules or and/or the Tag Manager rules execute and deliver the 3rd Party Pixels to the client which then executes the JavaScript/Pixels for both CI360 and 3rd Party Advertisers at the same time. This version does not require any special implementation beyond using JS API 2.0 call to CI360.

5.1 Requirements

5.1.1 Add the Facebook pixel to the web (known user)

Below is the basic Facebook Pixel base code to embed to the website for tracking their audience activities outside Facebook Apps or Pages and using Email as **identifier**:

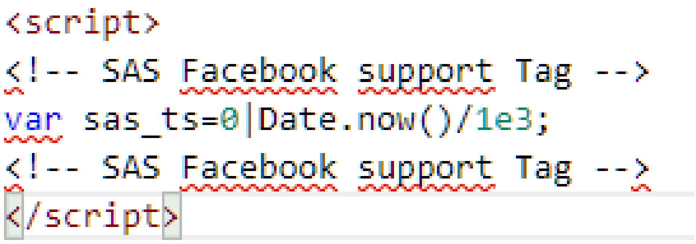


```
<!-- Facebook Pixel Code -->
<script>
!function(f,b,e,v,n,t,s)
{if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};
if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;n.version='2.0';
n.queue=[];t=b.createElement(e);t.async=!0;
t.src=v;s=b.getElementsByTagName(e)[0];
s.parentNode.insertBefore(t,s)}(window, document,'script',
'https://connect.facebook.net/en_US/fbevents.js');
fbq('init', '<facebook_pixel_id>', {em: '<email>'});
fbq('track', 'PageView');
</script>
<noscript><img height="1" width="1" style="display:none"
src="https://www.facebook.com/tr?id=<facebook_pixel_id>&ev=PageView&noscript=1"
/></noscript>
<!-- End Facebook Pixel Code -->
```

Note. The {em: '<email>'} properties can be removed for anonymous user.

5.1.2 SAS Support Code for FB

With SAS, we required to embed the below code right after the Facebook Pixel for mapping FB activities and SAS Events in later sections:



```
<script>
<!-- SAS Facebook support Tag -->
var sas_ts=0|Date.now()/1e3;
<!-- SAS Facebook support Tag -->
</script>
```

5.1.3 CI360 Tag Deployed

Add in the CI360 tag into the site (note that if you're making several pixel calls for different services, it would be best practice to deploy the below SAS Pixel Placeholder code with several different div's using different IDs, as setting them up the same way will conflict). Please copy the code from your CI360 from the **General Settings > SAS Tag Instruction** section.

```
SAS Ci360 Tag Example:
<!-- SAS Ci360 Tag -->
<script>
(function(ci){
    var ef=window[ci]=function(){
        return ef.q.push(arguments);
    };
    ef.q=[];ef.a={};
})('ci360');
</script>
<script async data-efname='ci360' id='ob-script-async'
        a='<tenant ID>'
        src='https://<server>/js/ot-all.min.js'></script>
```

5.2 CI360 Controlled or Tag Manager Intermediator Pixel Calls

5.2.1 Create Spot in CI360

This will create a spot for CI360 where we can then inject content, in this case we will inject a tag in order to send data from the website client to the Facebook servers

a. From navigation bar, click **Spots**

b. Click the button to create a new spot

c. Select **JavaScript** as the spot type

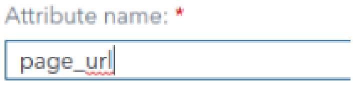
d. Navigate to the **URL** of the web page with **SAS CI360 Tag** and **SAS Pixel placeholder**

e. Then, click **Create Spot**

f. In **Spot Details Tab**,

- i. select **Spot Attribution** from the menu bar

    1. click the ⊕ button to add new Attribution for the spot
    2. select **URL path (entire path)** as Attribute
    3. click **OK**
    4. amend the Attribute name to “**page_url**”:



Attribute name: *

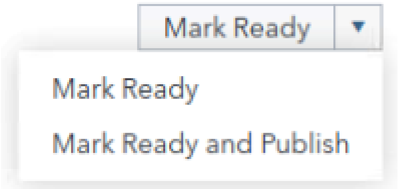
page_url

g. click the button to save the Spot

h. type in a name for the Spot and then click **Save**

i. Go to the **Orchestration Tab**

- i. Click on **Mark Ready and Publish** from drop-down menu:



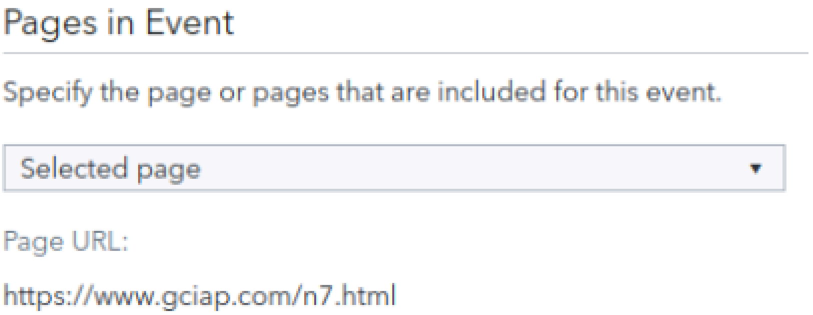
- ii. Click **Yes** when promoted by **Publishing the item will change the status to Scheduled or Active**

Tip: The example below fires a FB event if user had been visited X page Y times during Z period.

5.2.2 Create Custom Event

Here we will create a Custom Event, this is required as the code used in our creative to trigger the pixel will use merge tags in order to replace information to be sent to FB. The instructions will provide how to create an event and which data will be sent to FB but you can create your own event with different criteria and naming.

1. From navigation bar, click **Events**
2. Click the ✱ button for create a new event
3. Select **Custom Event**
4. Select **Page View** as event type
5. Enter the **Page URL** for re-targeting
6. Click **Create Event**
7. In the **Event Details** Tab
   i. Go to **Event Configuration**
      1. To confirm the site is correct:



Pages in Event

Specify the page or pages that are included for this event.

Selected page ▾

Page URL:

https://www.gciap.com/n7.html

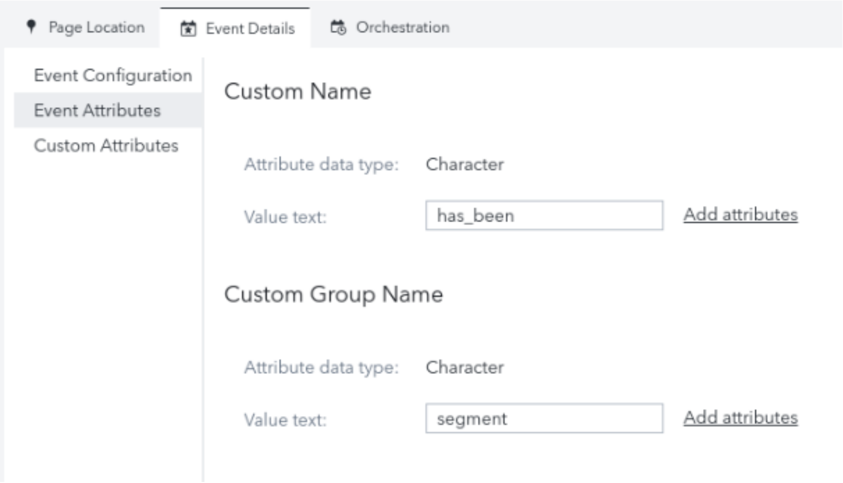
   ii. Go to **Event Attributes**
      1. In Custom Name’s value text: type in “**has_been**”:

         This is a custom value that will be collected in Facebook and stored as custom parameter field defined as below.

      2. In Custom Group Name’s value text: type in “**segment**”:

         This is a custom value will be used as custom parameter field name in FB event’s custom data

iii. The configuration should look similar to the below:



Custom Name

Attribute data type: Character

Value text: has_been

Custom Group Name

Attribute data type: Character

Value text: segment

iv. Create a new Custom Attribute by clicking ⊕

v. Choose **Page Details > JavaScript Variable**

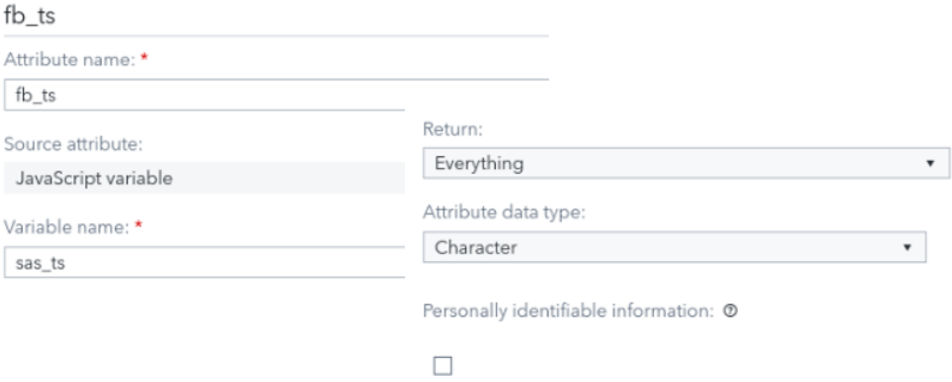
vi. Set the attribute name to **fb_ts**

vii. Variable name to **sas_ts**

viii. Return: **Everything**

ix. Attribute data type: **Character**

x. Personally identifiable information: **Unchecked** (note that we keep this unchecked so we can connect other information from other systems and connect it with our own identifiers if needed later in our data)



fb_ts

Attribute name: * fb_ts

Source attribute: JavaScript variable

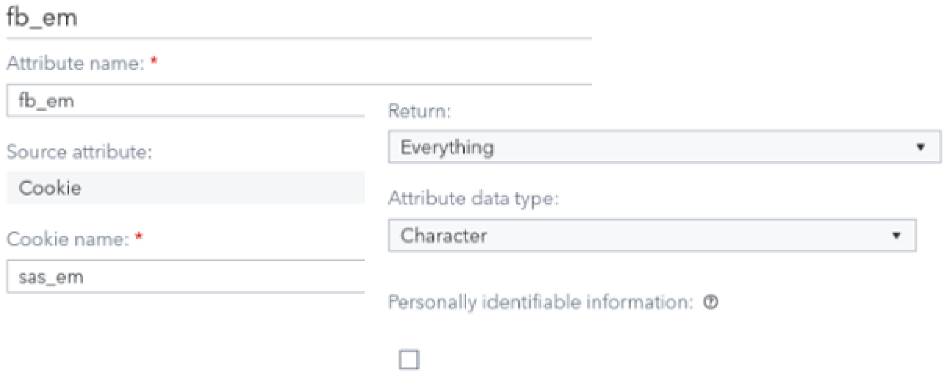
Variable name: * sas_ts

Return: Everything ▾

Attribute data type: Character ▾

Personally identifiable information: ⓘ

□



fb_em

Attribute name: * fb_em

Source attribute: Cookie

Cookie name: * sas_em

Return: Everything

Attribute data type: Character

Personally identifiable information: □

fb_ts comes from the "SAS Support Code" and they designed for data mapping and API integration which will be covered in later sections.

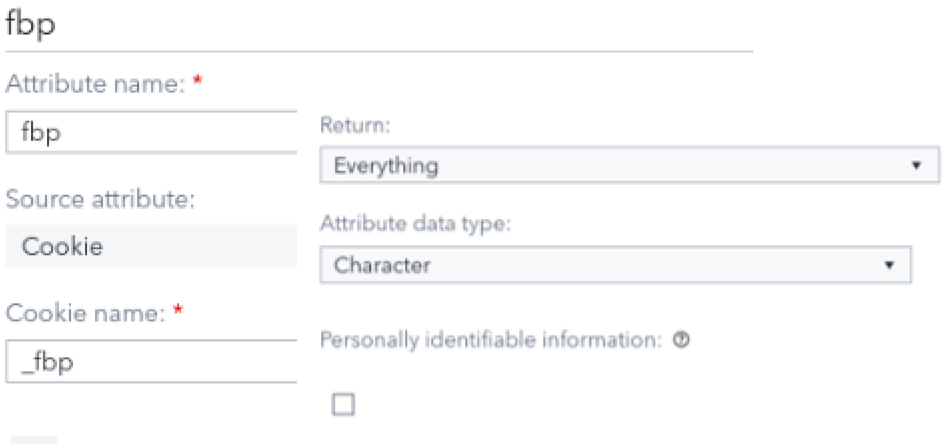
xi. Create a new Custom Attribute by clicking ⊕

xii. Choose **URL Details > Cookie**

xiii. Set the attribute name to **fbp**

xiv. Variable name to **_fbp**

xv. Return: **Everything**



fbp

Attribute name: * fbp

Source attribute: Cookie

Cookie name: * _fbp

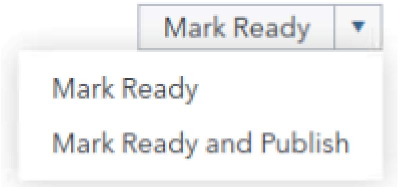
Return: Everything

Attribute data type: Character

Personally identifiable information: □

8. click the 🖫 button to save the Custom Event
9. type in "**Event: FB Pixel**" (this is just a placeholder name, you can use any event you want), then click **Save**
10. Go to the **Orchestration** Tab

xvi. Click on **Mark Ready and Publish** from drop-down menu:



xvii. Click **Yes** when prompted by **Publishing the item will change the status to Scheduled or Active**

5.2.3 Create creative with FB pixel (CI360 Controlled, Option #1)

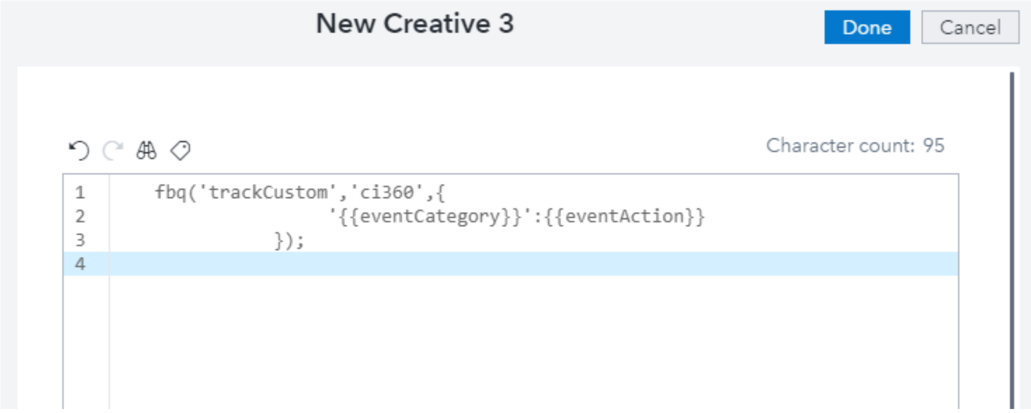
This is where we will define our code to be replaced on the client machine in order to be executed. In this case it will be requesting a pixel from the FB servers with specific link parameters to transfer information

a. From navigation bar, click **Creatives**

b. Click the button for create new Creative

c. Select **Plain Text** as creative type

d. Copy the **Facebook Custom Event Pixel** into the body tag as below



```
    fbq('trackCustom','ci360',{
                    '{{eventCategory}}':{{eventAction}}
              });

```

5.2.4 Create creative with Tag Manager rule code (Tag Manager Intermediate)

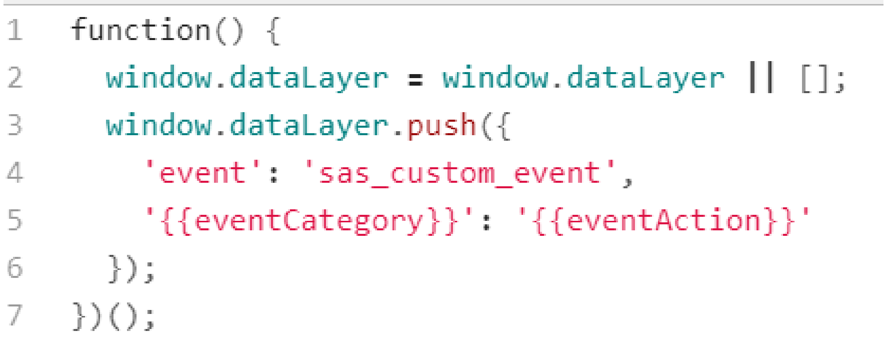
This is where we will define our code to be replaced on the client machine in order to be executed. In this case it will be requesting a pixel from the FB servers with specific link parameters to transfer information

e. From navigation bar, click **Creatives**

f. Click the button for create new Creative

g. Select **Plain Text** as creative type

h. Install the **Tag Manager Script** here (provided below and in the repository linked here for examples)



```
function() {
  window.dataLayer = window.dataLayer || [];
  window.dataLayer.push({
    'event': 'sas_custom_event',
    '{{eventCategory}}': '{{eventAction}}'
  });
})();
```

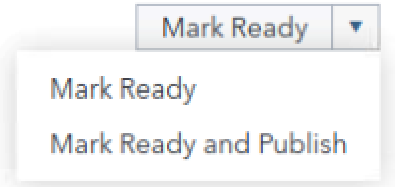
i. Click **Done**

j. click the button to save the creative

k. type in "**Creative: Tag Manager: Custom Event**" then click **Save**

l. Go To **Orchestration Tab**

   a. Click on **Mark Ready and Publish** from drop-down menu:



   b. Click **Yes** when prompted by **Publishing the item will change the status to Scheduled or Active**

m. **Setup the rule in your Tag Manager and make sure to place the Pixel code into that rule to execute**

5.2.5 Define Web Task

Here we will create a task, this connects the Spot and the Creative with Segmentation in order to conditionally fire the FB pixel on the client machine. We will use an event and information defined in the event, combined with merge tags to create the send information to FB servers

a. From navigation bar, click **Task**

b. Click the button for create new Task

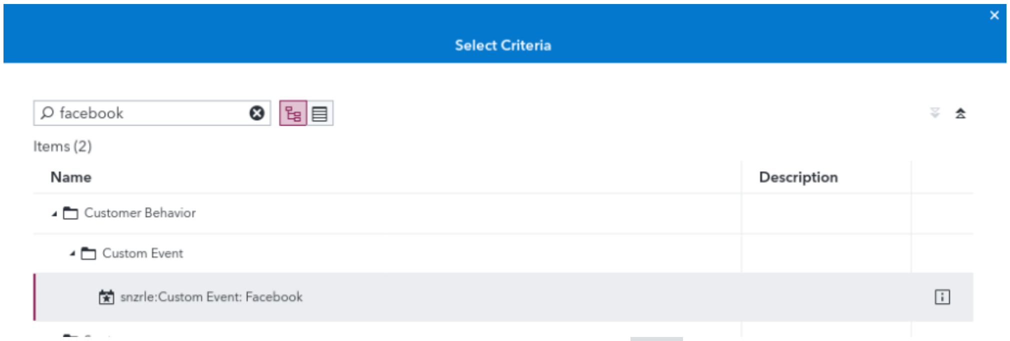
c. Select **Web** as Task type

d. Choose the **Spot** created from "Create Spot in CI360" section

e. Go to **Targeting** Tab

   i. Click the + button to add criteria

   ii. Select the **Custom Event** which defined from Step #2

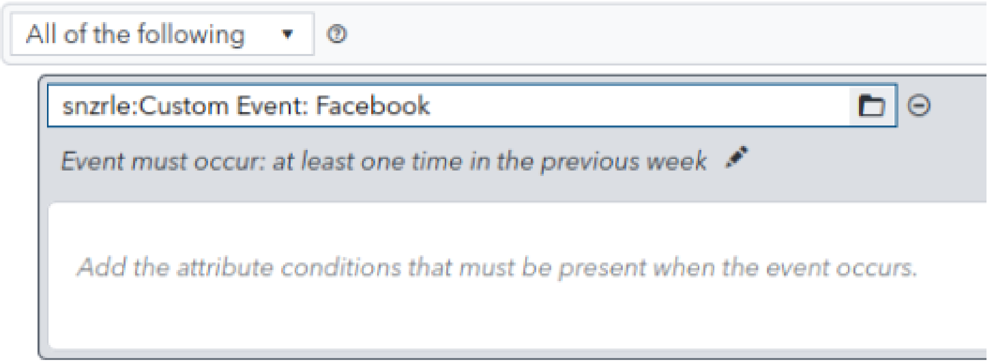


facebook

Items (2)

Name	Description
Customer Behavior	
Custom Event	
snzrle:Custom Event: Facebook	

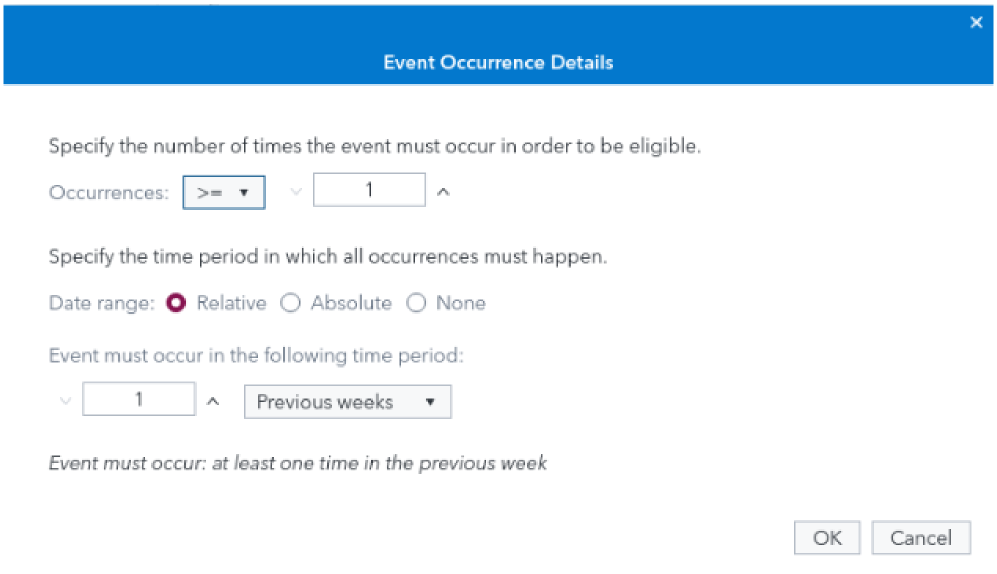
iii. Optional: Define Even Occurrence Detail by click the ✎ button:



snzrle:Custom Event: Facebook

Event must occur: at least one time in the previous week

Add the attribute conditions that must be present when the event occurs.



Specify the number of times the event must occur in order to be eligible.

Occurrences: >= 1

Specify the time period in which all occurrences must happen.

Date range: ● Relative ○ Absolute ○ None

Event must occur in the following time period:

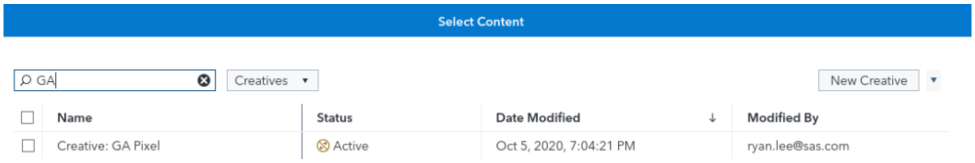
1 Previous weeks

Event must occur: at least one time in the previous week

iv. Or leave it by default as **once per pervious weeks**

f. Go to **Content** Tab
   i. Click **Select Content**
   ii. Choose **Creatives** from the drop-down menu

iii. Select the **Creative** defined from “Create creative with GA pixel installed” section

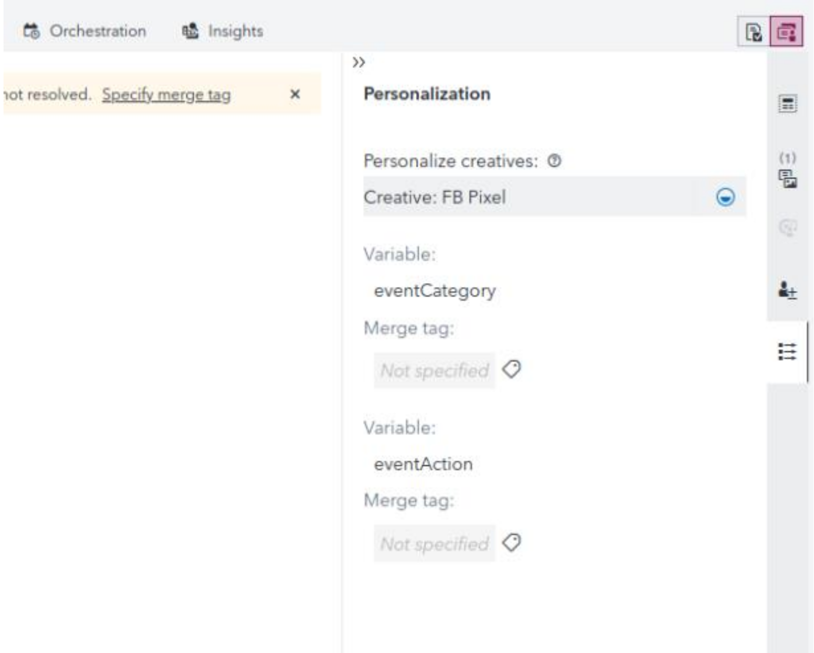


□	Name	Status	Date Modified ↓	Modified By
□	Creative: GA Pixel	Active	Oct 5, 2020, 7:04:21 PM	ryan.lee@sas.com

iv. Click **OK**

v. Click the ☰ button from the right-hand side menu bar

vi. Screen should be displayed as below:



ɔot resolved. Specify merge tag ×

Personalization

Personalize creatives:

Creative: FB Pixel

Variable: eventCategory

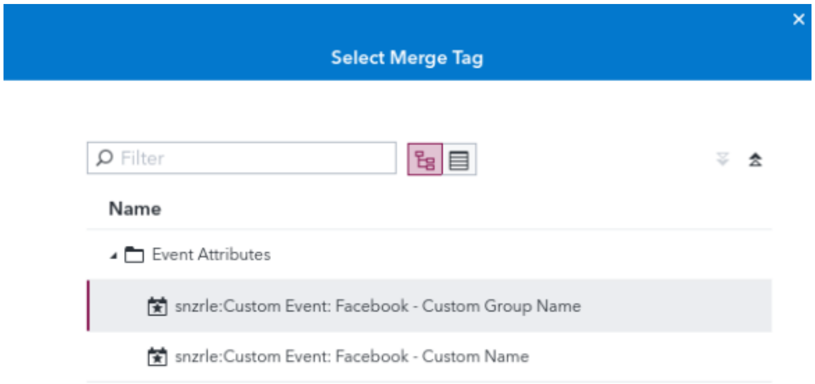
Merge tag: Not specified

Variable: eventAction

Merge tag: Not specified

vii. Click the 🏷 button from the **eventAction** merge tag field

viii. Select the Event Attribute with **Custom Group Name**:



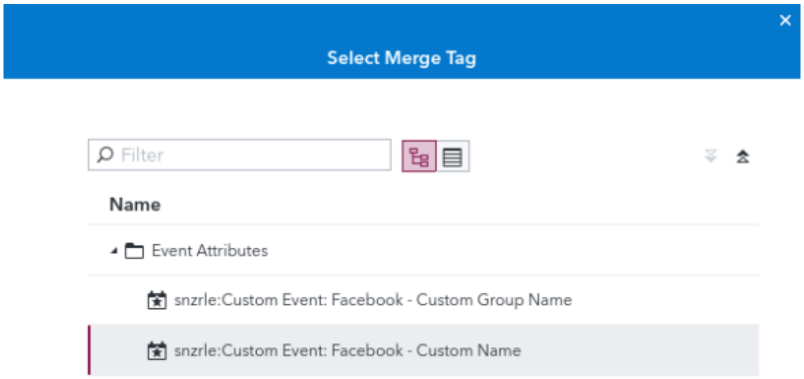
Name

- Event Attributes
  - snzrle:Custom Event: Facebook - Custom Group Name
  - snzrle:Custom Event: Facebook - Custom Name

ix. then, click **OK**

x. Click the 🏷 button from the **eventCategory** merge tag field

xi. Select the Event Attribute with **Custom Name**:



Select Merge Tag

Filter

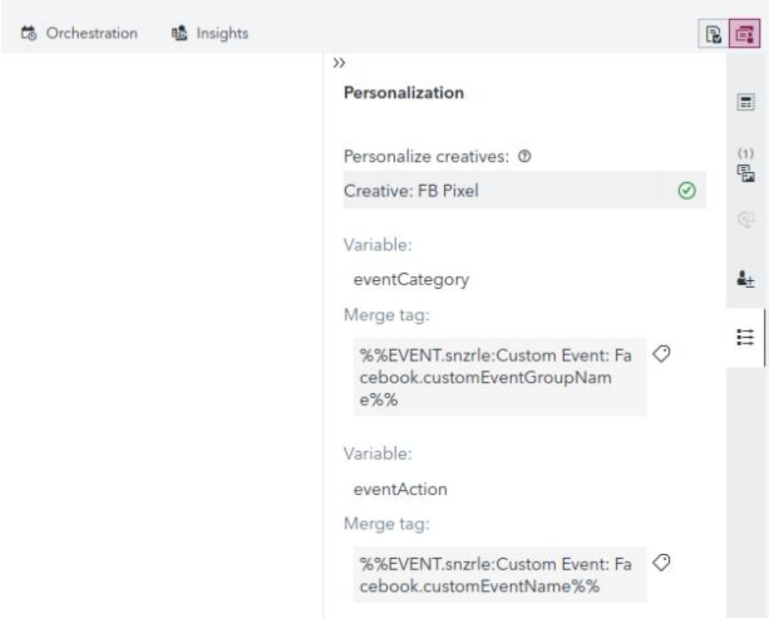
Name

Event Attributes

snzrle:Custom Event: Facebook - Custom Group Name

snzrle:Custom Event: Facebook - Custom Name

xii. then, click **OK**.



Personalization

Personalize creatives:

Creative: FB Pixel

Variable:

eventCategory

Merge tag:

%%EVENT.snzrle:Custom Event: Facebook.customEventGroupName%%

Variable:

eventAction

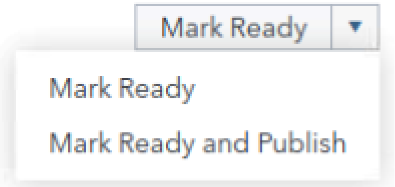
Merge tag:

%%EVENT.snzrle:Custom Event: Facebook.customEventName%%

xiii. click the button to save the task

g. Go To **Orchestration** Tab

i. Click on **Mark Ready and Publish** from drop-down menu:



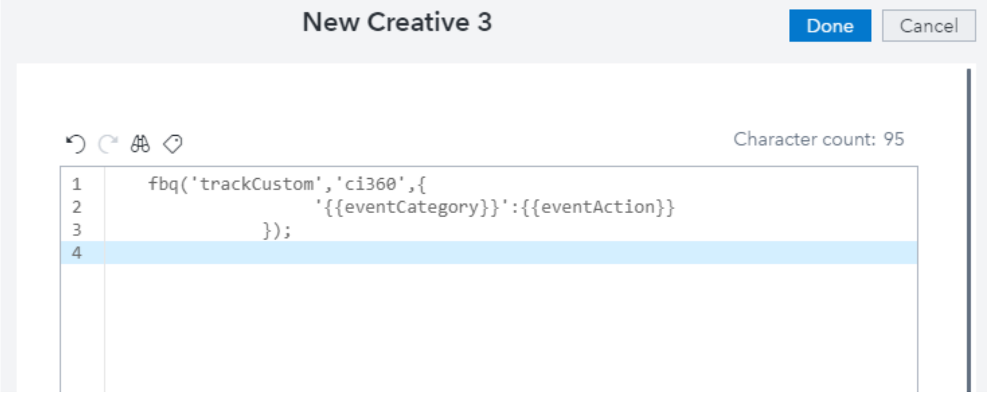
ii. Click **Yes** when promoted by **Publishing the item will change the status to Scheduled or Active**

5.2.6 Trigger FB Pixel via SAS Creative

a. Go to the page where your **Custom Event** is fired

b. Then refresh the page or visit other page contain the **CI360 Pixel Placeholder**

c. The “**Event: FB Pixel**” Custom Event should be now triggered the Facebook Pixel via SAS Creative, as below



```
fbq('trackCustom','ci360',{
                '{{eventCategory}}':{{eventAction}}
          });
```

5.3 Tag Manager/Framework Controlled Pixel

Note: This method requires to develop and maintain a naming system outside of CI360. Be sure to note this in either the event name or somewhere to keep everything organized!

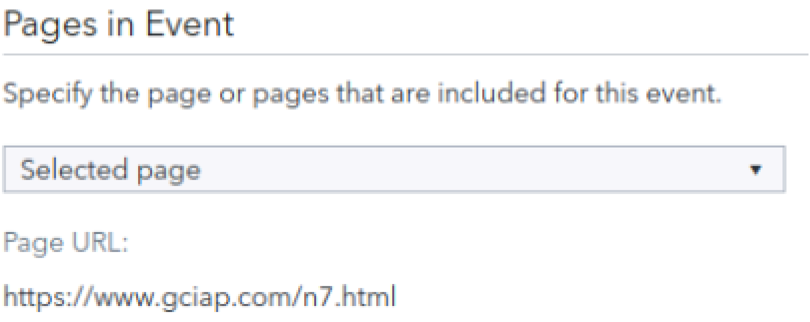
5.3.1 Create Custom Event

Here we will create a Custom Event, this will be a JavaScript 2.0 API call event setup to fire from the Tag Manager

1. From navigation bar, click **Events**
2. Click the button for create a new event
3. Select **Custom Event**
4. Select **JavaScript API** as event type
5. Click **Create Event**
6. Enter the **API Event Key**
7. In the **Event Details** Tab

   a. Go to **Event Configuration**

      1. To confirm the site is correct:



Pages in Event

Specify the page or pages that are included for this event.

Selected page ▾

Page URL:

https://www.gciap.com/n7.html

   b. Go to **Event Attributes**

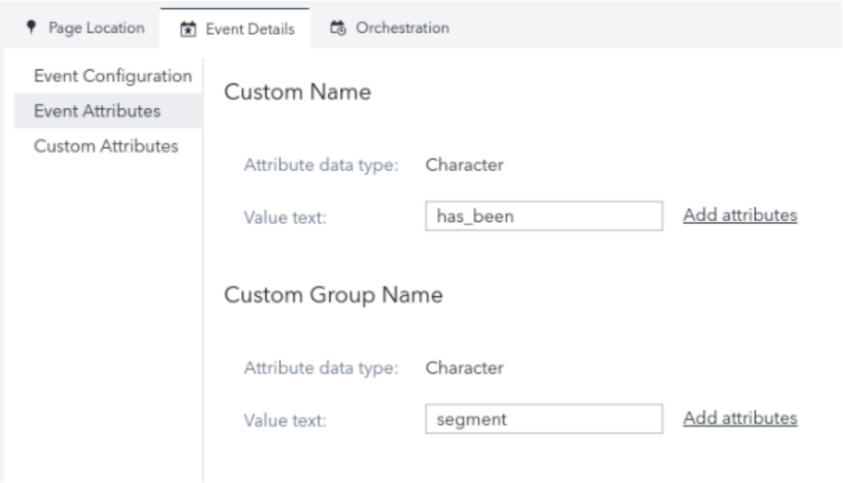
      2. In Custom Name’s value text: type in “**has_been**”:

This is a custom value that will be collected in Facebook and stored as custom parameter field defined as below.

3. In Custom Group Name’s value text: type in “**segment**”:

   This is a custom value will be used as custom parameter field name in FB event’s custom data

c. The configuration should look similar to the below:



Custom Name

Attribute data type: Character

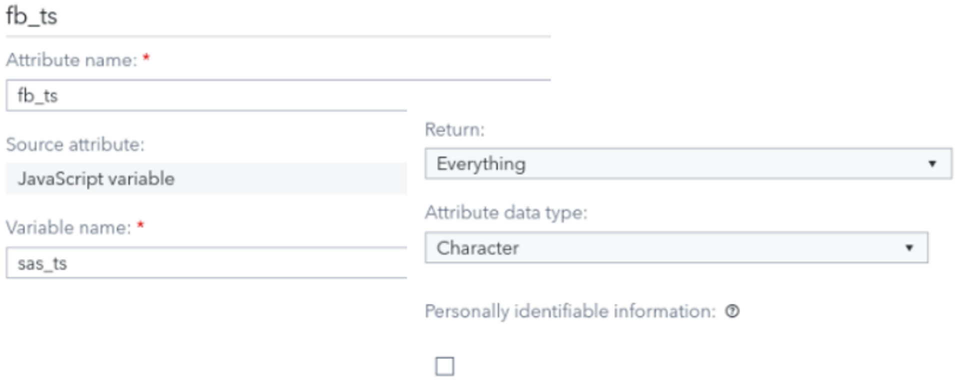
Value text: has_been

Custom Group Name

Attribute data type: Character

Value text: segment

d. Create a new Custom Attribute by clicking ⊕
e. Choose **Page Details > JavaScript Variable**
f. Set the attribute name to **fb_ts**
g. Variable name to **sas_ts**
h. Return: **Everything**
i. Attribute data type: **Character**
j. Personally identifiable information: **Unchecked** (note that we keep this unchecked so we can connect other information from other systems and connect it with our own identifiers if needed later in our data)



fb_ts

Attribute name: * fb_ts

Source attribute: JavaScript variable

Variable name: * sas_ts

Return: Everything

Attribute data type: Character

Personally identifiable information: ☐

k. Create another new Custom Attribute by clicking ⊕
l. Choose **Page Details > JavaScript Variable**

fb_ts comes from the “SAS Support Code” and they designed for data mapping and API integration which will be covered in later sections.

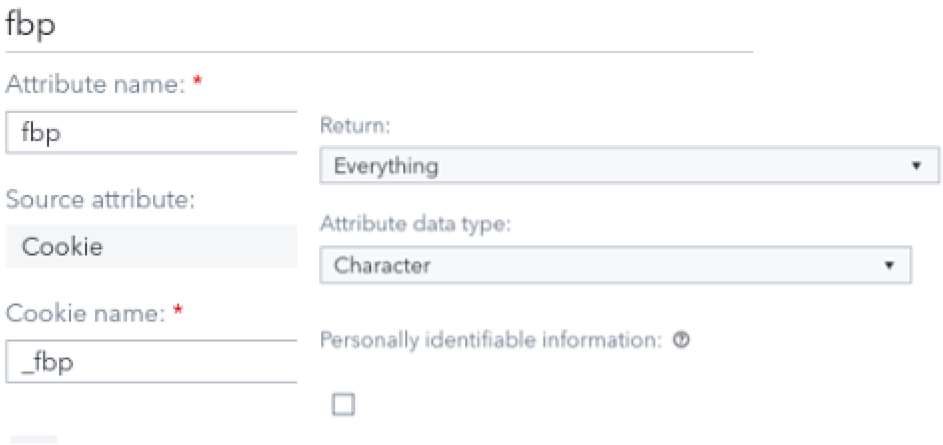
m. Create a new Custom Attribute by clicking ⊕

n. Choose **URL Details > Cookie**

o. Set the attribute name to **fbp**

p. Variable name to **_fbp**

q. Return: **Everything**



fbp

Attribute name: * fbp

Source attribute: Cookie

Cookie name: * _fbp

Return: Everything ▾

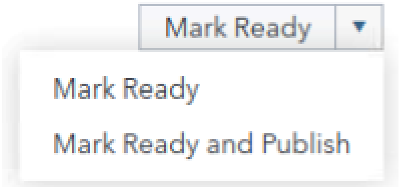
Attribute data type: Character ▾

Personally identifiable information: ⓘ

□

8. click the 🖫 button to save the Custom Event
9. type in “**Event: FB Pixel**” (this is just a placeholder name, you can use any event you want), then click **Save**
10. Go to the **Orchestration** Tab

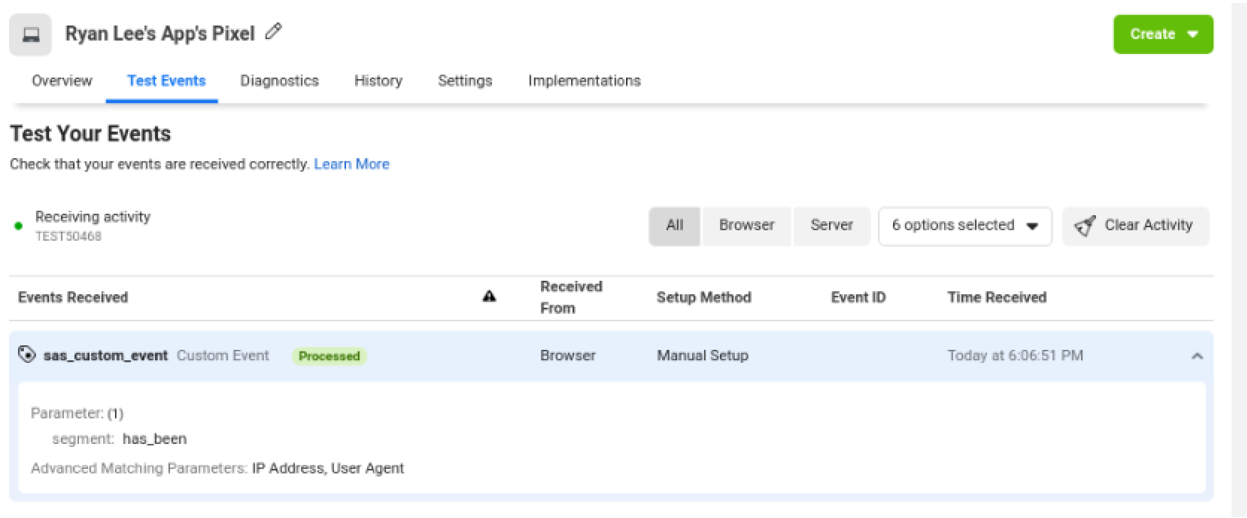
r. Click on **Mark Ready and Publish** from drop-down menu:



s. Click **Yes** when prompted by **Publishing the item will change the status to Scheduled or Active**

5.4 Verification

Here we will verify if the Pixel is properly sending data to FB servers. Using the values we send to Facebook we will see if those values show up in the **Overview** section of the FB interface, our for testing purpose, it can be display Test Your Events Tab:



Test Your Events

Check that your events are received correctly. Learn More

● Receiving activity
TEST50468

Events Received	⚠	Received From	Setup Method	Event ID	Time Received
sas_custom_event Custom Event Processed		Browser	Manual Setup		Today at 6:06:51 PM

Parameter: (1)
segment: has_been
Advanced Matching Parameters: IP Address, User Agent

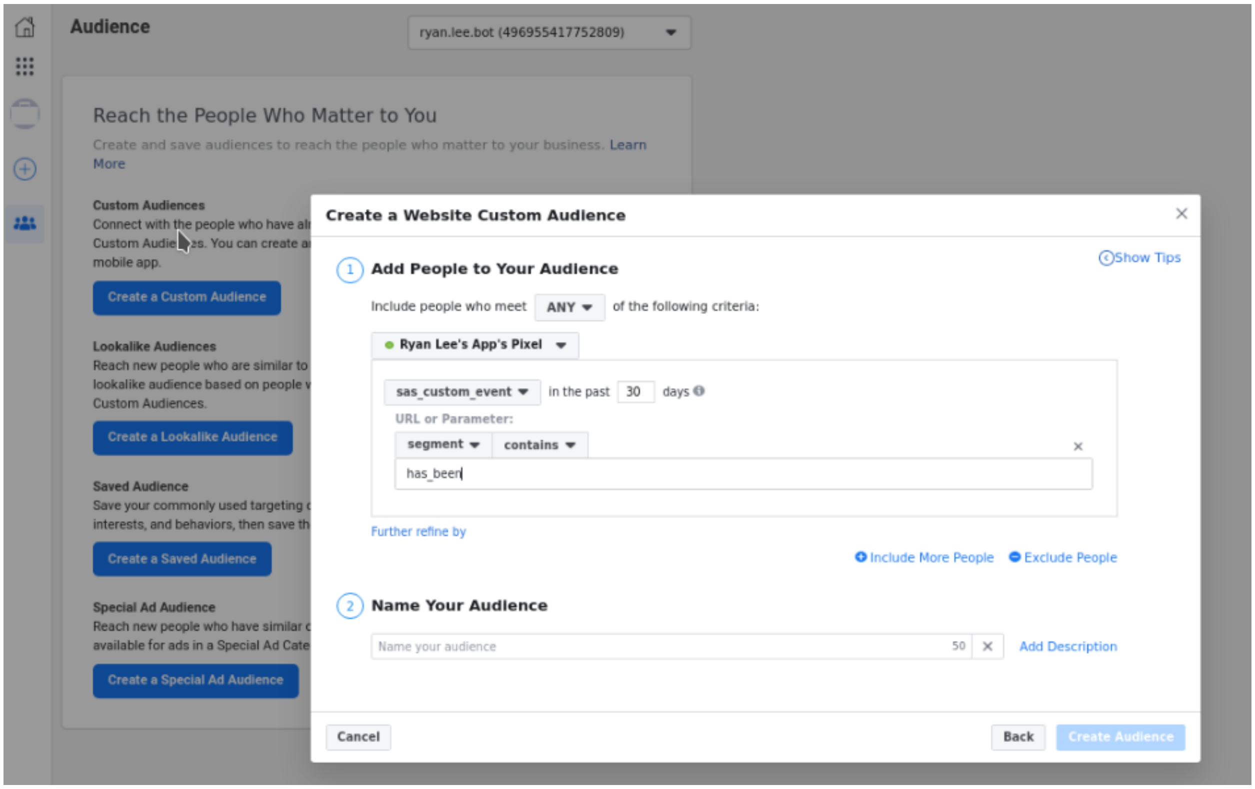
Once verified we can then use these in segments for activation on other platforms connected to FB

5.5 Example

5.5.1 EXAMPLE: USE FB EVENT IN OTHER PLATFORMS

Once FB pixel is active (has been tracked), the data will be available in FB Audience for define as Custom Audience Group:

FB Audience > Click on “Create a Custom Audience” > select “sas_custom_event” from the drop down menu > click on “refine by” and “select URL/Parameter” > choose “segment” then type in “has_been” value.



Audience

ryan.lee.bot (496955417752809)

Reach the People Who Matter to You

Create and save audiences to reach the people who matter to your business. Learn More

Custom Audiences
Connect with the people who have al
Custom Audie es. You can create a
mobile app.

Create a Custom Audience

Lookalike Audiences
Reach new people who are similar to
lookalike audience based on people v
Custom Audiences.

Create a Lookalike Audience

Saved Audience
Save your commonly used targeting c
interests, and behaviors, then save th

Create a Saved Audience

Special Ad Audience
Reach new people who have similar c
available for ads in a Special Ad Cate

Create a Special Ad Audience

Create a Website Custom Audience

1 Add People to Your Audience

Show Tips

Include people who meet ANY of the following criteria:

Ryan Lee's App's Pixel

sas_custom_event in the past 30 days

URL or Parameter:

segment contains

has_been

Further refine by

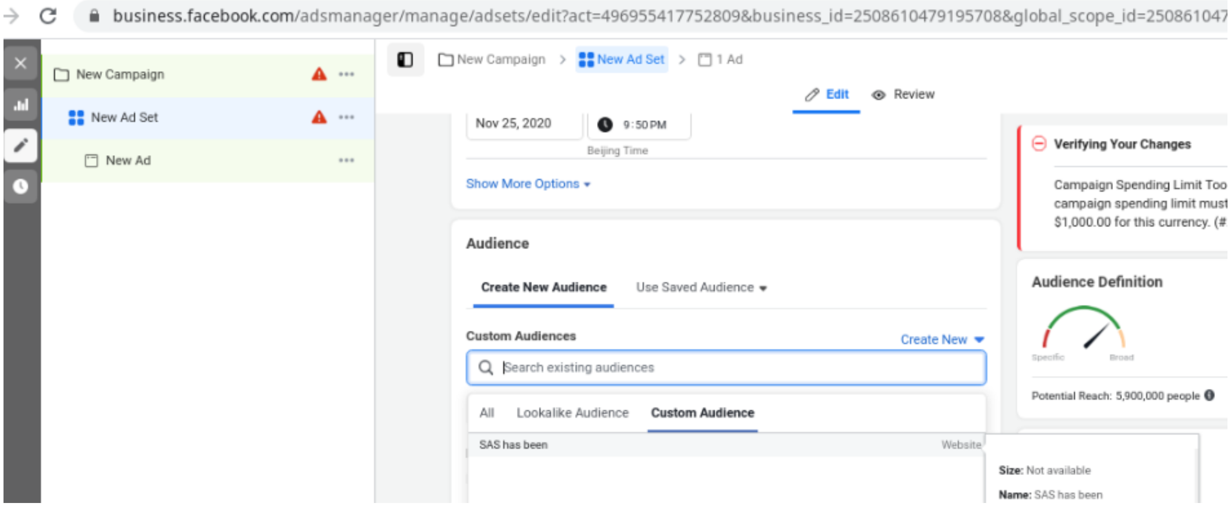
Include More People Exclude People

2 Name Your Audience

Name your audience 50 Add Description

Cancel Back Create Audience

Then, it will become Custom Audience Target which can be choose under Facebook Ad:



business.facebook.com/adsmanager/manage/adsets/edit?act=496955417752809&business_id=2508610479195708&global_scope_id=250861047

New Campaign
New Ad Set
New Ad

New Campaign > New Ad Set > 1 Ad

Edit Review

Nov 25, 2020 9:50 PM
Beijing Time

Show More Options

Verifying Your Changes
Campaign Spending Limit Too
campaign spending limit must
$1,000.00 for this currency. (#

Audience

Create New Audience Use Saved Audience

Custom Audiences Create New

Search existing audiences

All Lookalike Audience Custom Audience

SAS has been Website

Audience Definition

Potential Reach: 5,900,000 people

Size: Not available
Name: SAS has been

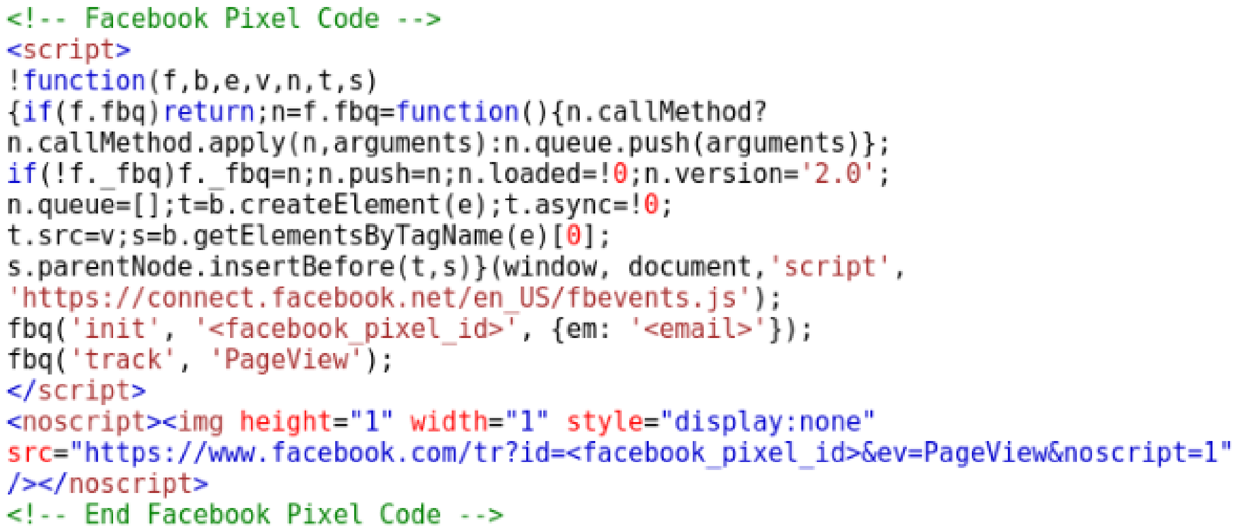
6 API (Server-to-Server)

This section will show how to make a server-to-server connection using a CI360 Connector. This doesn't require any bridge code as it is a one-way communicator from the CI360 tenant to the Facebook servers. This can also be proxied through on-premise agent if needed.

6.1 Requirements

6.1.1 Add the Facebook pixel to the web (known user)

Below is the basic Facebook Pixel base code to embed to the website for tracking audience activities outside Facebook Apps or Pages and using Email as **identifier**:

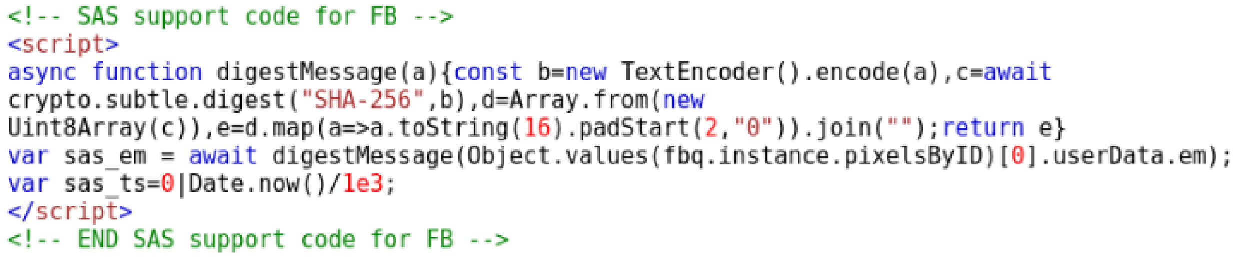


```
<!-- Facebook Pixel Code -->
<script>
!function(f,b,e,v,n,t,s)
{if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};
if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;n.version='2.0';
n.queue=[];t=b.createElement(e);t.async=!0;
t.src=v;s=b.getElementsByTagName(e)[0];
s.parentNode.insertBefore(t,s)}(window, document,'script',
'https://connect.facebook.net/en_US/fbevents.js');
fbq('init', '<facebook_pixel_id>', {em: '<email>'});
fbq('track', 'PageView');
</script>
<noscript><img height="1" width="1" style="display:none"
src="https://www.facebook.com/tr?id=<facebook_pixel_id>&ev=PageView&noscript=1"
/></noscript>
<!-- End Facebook Pixel Code -->
```

Note. The {em: '<email>'} properties can be removed for anonymous user.

6.1.2 SAS Support Code for FB

With SAS, we required to embed the below code right after the Facebook Pixel for mapping FB activities and SAS Events in later sections:



```
<!-- SAS support code for FB -->
<script>
async function digestMessage(a){const b=new TextEncoder().encode(a),c=await
crypto.subtle.digest("SHA-256",b),d=Array.from(new
Uint8Array(c)),e=d.map(a=>a.toString(16).padStart(2,"0")).join("");return e}
var sas_em = await digestMessage(Object.values(fbq.instance.pixelsByID)[0].userData.em);
var sas_ts=0|Date.now()/1e3;
</script>
<!-- END SAS support code for FB -->
```

6.1.3 CI360 Tag Deployed

Add in the CI360 tag into the site (note that if you're making several pixel calls for different services, it would be best practice to deploy the below SAS Pixel Placeholder code with several different div's using different IDs, as setting them up the same way will conflict). Please copy the code from your CI360 from the **General Settings > SAS Tag Instruction** section.

```
SAS Ci360 Tag Example:
<!-- SAS Ci360 Tag -->
<script>
(function(ci){
    var ef=window[ci]=function(){
        return ef.q.push(arguments);
    };
    ef.q=[];ef.a={};
})('ci360');
</script>
<script async data-efname='ci360' id='ob-script-async'
        a='<tenant ID>'
        src='https://<server>/js/ot-all.min.js'></script>
```

6.2 Connector setup

a. Navigate to **General Settings > Connectors**

b. Click on the ✳ button

c. Enter the details

   i. Name: **Facebook Event Manager**

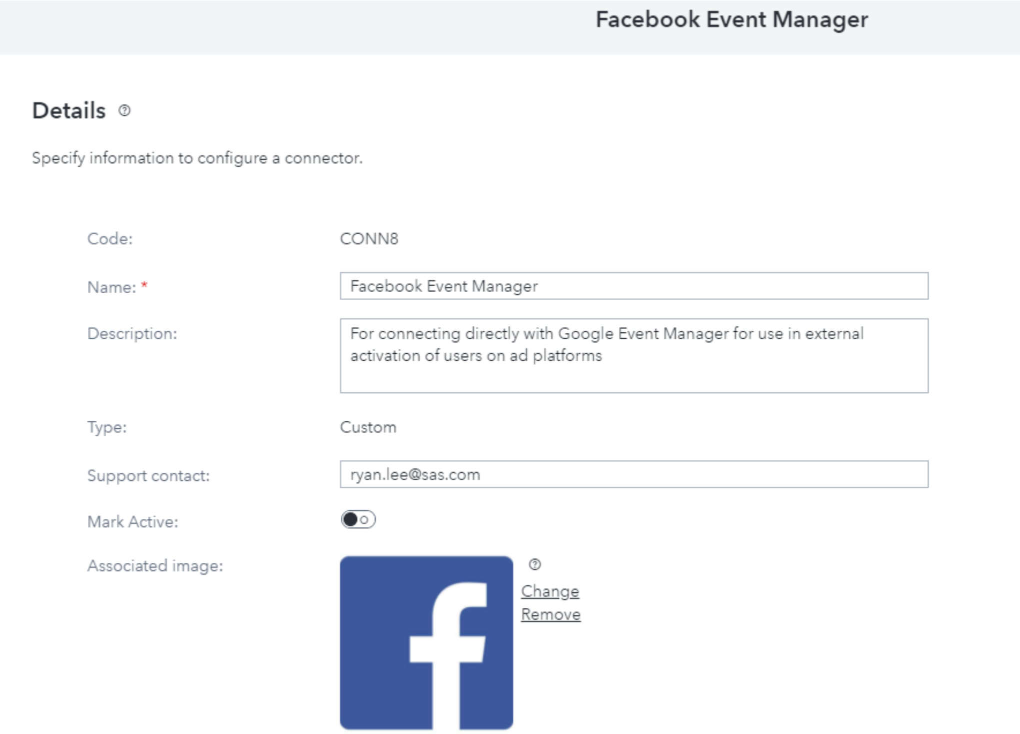
   ii. Description: ***Anything you want can go here to describe this***

   iii. Support Contact: Set this up as anyone that would receive emails in case of errors

   iv. Set as **Active** by changing the slider from ●○ to ●

   v. Attach an image if you want (can find an image through Google for “Facebook” or otherwise)

   vi. Click **Apply**



Facebook Event Manager

Details

Specify information to configure a connector.

Code: CONN8

Name: * Facebook Event Manager

Description: For connecting directly with Google Event Manager for use in external activation of users on ad platforms

Type: Custom

Support contact: ryan.lee@sas.com

Mark Active: ●○

Associated image:

d. Re-Open the Facebook Connector

e. Click **New Endpoint**

  i. Name: **Facebook Server Event**

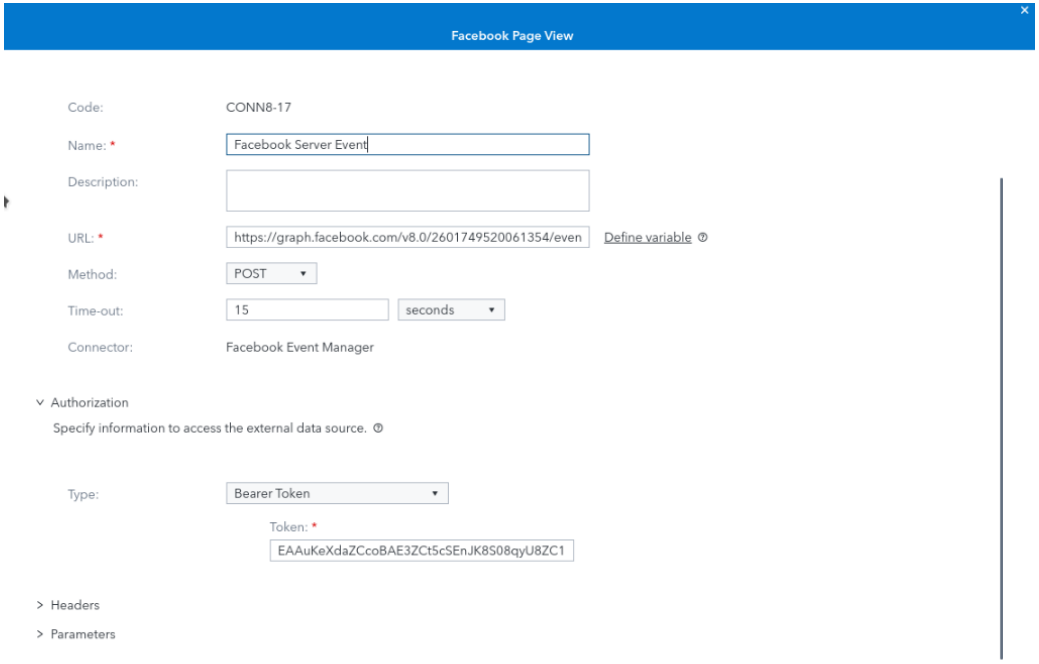
  ii. Description: ***Anything you want can go here to describe this***

  iii. URL: https://graph.facebook.com/v8.0/<pixel_id>/events?data=[{event_name:"{{$eventname}}",event_time:{{$timestamp}},user_data:{fbq:"{{$fbq}},em:{{$fbem}}"},custom_data:{{{$eventcategory}}:"{{$event}}"}}]&test_event_code=TEST50468
  (note the **test_event_code** highlighted in the URL, it is the flag that can be found under Facebook Pixel's Test Events Tab to allow instant result from the Server Events)

  iv. Method: **POST**

  v. Time-out: **15 seconds**

  vi. Authorization: **Bearer Token** (the Access Token is proved from Facebook Event Manager > Pixel > Generate an Access Token section)



Facebook Page View

Code: CONN8-17

Name: * Facebook Server Event

Description:

URL: * https://graph.facebook.com/v8.0/2601749520061354/even

Method: POST

Time-out: 15 seconds

Connector: Facebook Event Manager

Authorization

Specify information to access the external data source.

Type: Bearer Token

Token: * EAAuKeXdaZCcoBAE3ZCt5cSEnJK8S08qyU8ZC1

  vii.

f. Click on **define variables** (this will replace the {{variables}} in the URL with values from the JSON payload that will be sent each time the connector is invoked and these values will be populated from values we create in the custom event and task we create). Unfortunately at this time, there are no easy way to view the JSON payloads without setting up your own server to capture these values first but once you know their locations you can use JSON path selectors to define them (Check appendix for website to test your JSON payload and selectors) Check to see if merge tags available from other.

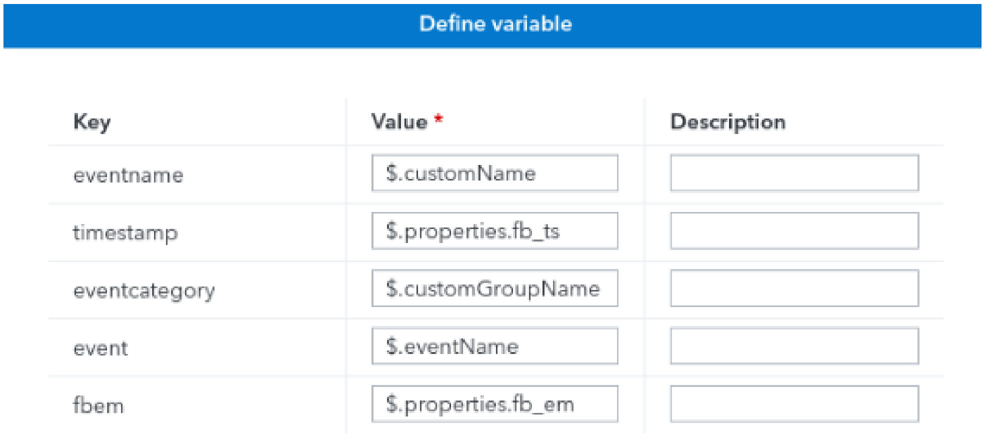
  i. Set eventname to "**$.customName**"

  ii. Set timestamp to "**$.properties.fb_ts**"

  iii. Set fbq to "**$.properties.fbq**"

  iv. Set eventcategory to "**$.customGroupName**"

v. Set event to **“$.eventName”**



Define variable

Key	Value *	Description
eventname	$.customName	
timestamp	$.properties.fb_ts	
eventcategory	$.customGroupName	
event	$.eventName	
fbem	$.properties.fb_em	

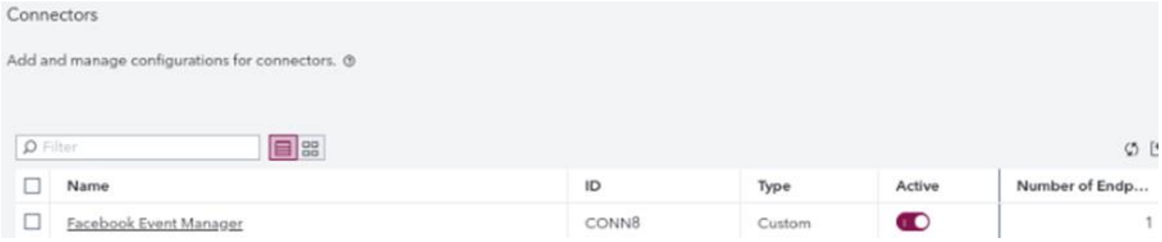
vi. Click **OK**

vii. Click **Save** to save the endpoint

viii. In the **Endpoints** section, click ● to enable the endpoint

ix. Click **Apply**

x. You will now have a new connector as show below with 1 endpoint setup for Facebook Event Manager



Connectors

Add and manage configurations for connectors.

	Name	ID	Type	Active	Number of Endp...
□	Facebook Event Manager	CONN8	Custom	●	1

Optional: Setup the connector to proxy through an on-premise agent

6.3 Event setup

This event will be setup to create to collect the Facebook Browser ID from the client’s computer. This ID we want is stored in the _fbp cookie. As well as the event time stamp which we added into the facebook pixel code on the page. Note that this event won’t necessarily fire our external event and Server-to-Server communication through the API, we will do that with an **External System Task** using this event as a trigger which we can further customize to add additional conditions and segmentation before sending data to FB.

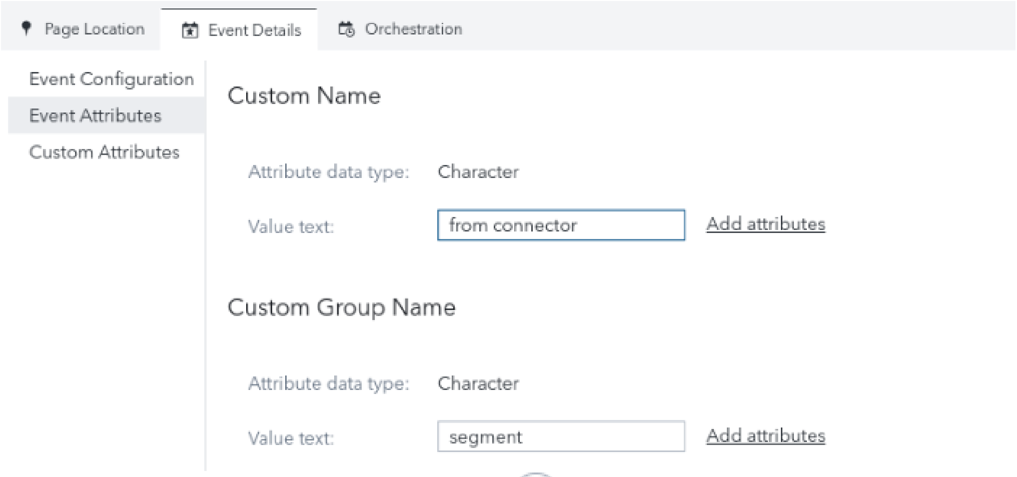
a. Click **Events** in the left navigation menu

b. Create a new event by clicking on ⊞

c. You can create any type of event, but for this we will create a **Custom Event** based on a **Page View**

d. Choose the page you want this to fire on (this could simply be every page if you restrict using segments in a later step, or it could be specific pages)

e. Set the **Custom Name** and the **Custom Group Name**, note that these are the values that will be sent to Facebook. (Write these down)



Custom Name

Attribute data type: Character

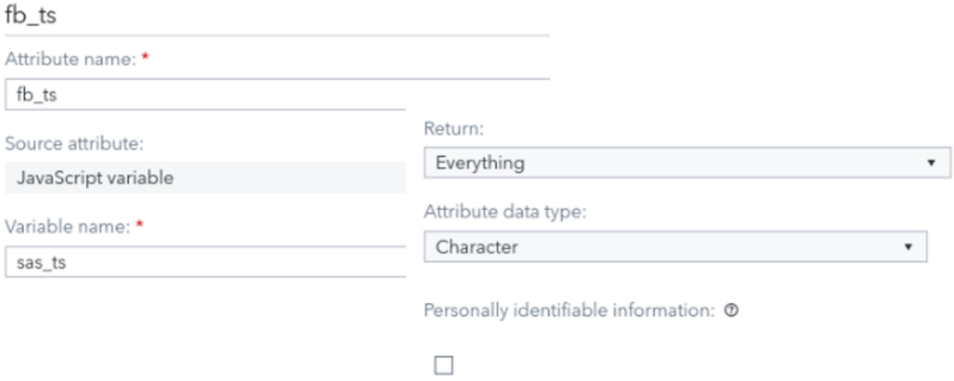
Value text: from connector Add attributes

Custom Group Name

Attribute data type: Character

Value text: segment Add attributes

f. Create a new Custom Attribute by clicking ⊕
g. Choose **Page Details > JavaScript Variable**
h. Set the attribute name to **fb_ts**
i. Variable name to **sas_ts**
j. Return: **Everything**
k. Attribute data type: **Character**
l. Personally identifiable information: **Unchecked** (note that we keep this unchecked so we can connect other information from other systems and connect it with our own identifiers if needed later in our data)



fb_ts

Attribute name: * fb_ts

Source attribute: JavaScript variable

Variable name: * sas_ts

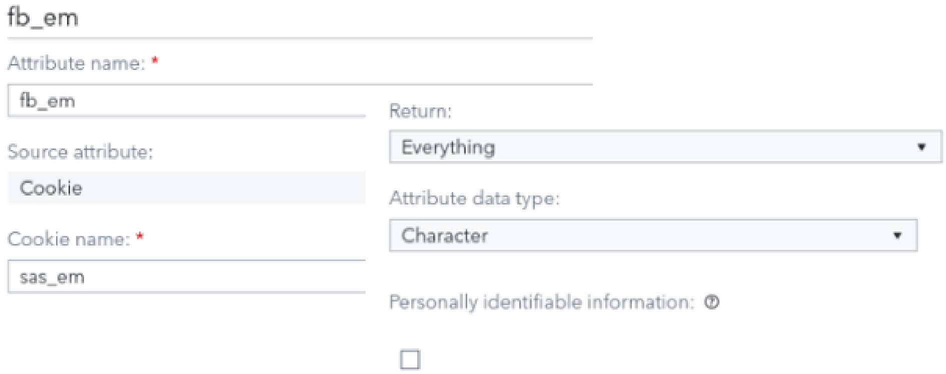
Return: Everything ▾

Attribute data type: Character ▾

Personally identifiable information: ⓘ

□

m. Create another new Custom Attribute by clicking ⊕
n. Choose **Page Details > JavaScript Variable**
o. Set the attribute name to **sas_em**
p. Personally identifiable information: **Unchecked** (note that we keep this unchecked so we can connect other information from other systems and connect it with our own identifiers if needed later in our data)



fb_em

Attribute name: *
fb_em

Source attribute:
Cookie

Cookie name: *
sas_em

Return:
Everything

Attribute data type:
Character

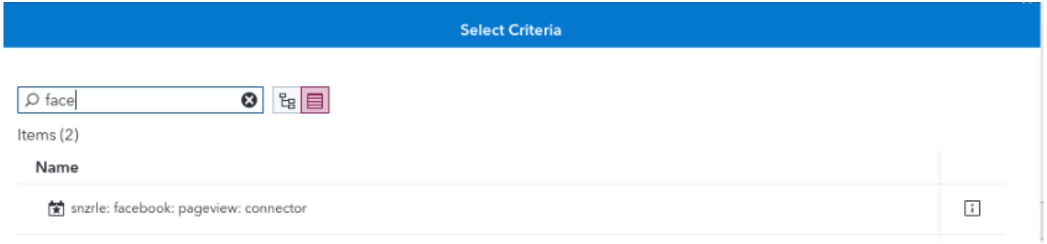
Personally identifiable information:
□

q. Click 💾 to save the event naming it whatever you want (remember the event name won't be used to send to FB, the Event Attributes will be used instead)

6.4 Task Setup

Here we will create an external system task to invoke the connector using the event we created last step as a trigger. In the case of FB, all the information is sent through link parameters so we don't need any bridge code, but if needed we can use the content creation combined with merge tags to send additional information to the external system.

a. Click on **Tasks** in the side panel
b. Create a new task by clicking on ✱
c. Choose **External System**
d. Here you have the choice to select an Agent (even if you are using an on-premise agent to send your connector through, selecting an Agent is not required here) so simply click on ③ Select Connector Endpoints to skip this section
e. Check the **Endpoint Name/Connector Name** we set before (Facebook Event Manager/ Facebook Server Events)
f. Click **Create Task**
g. Navigate to the **Orchestration Tab > Trigger** section and specify the Event you created in last step



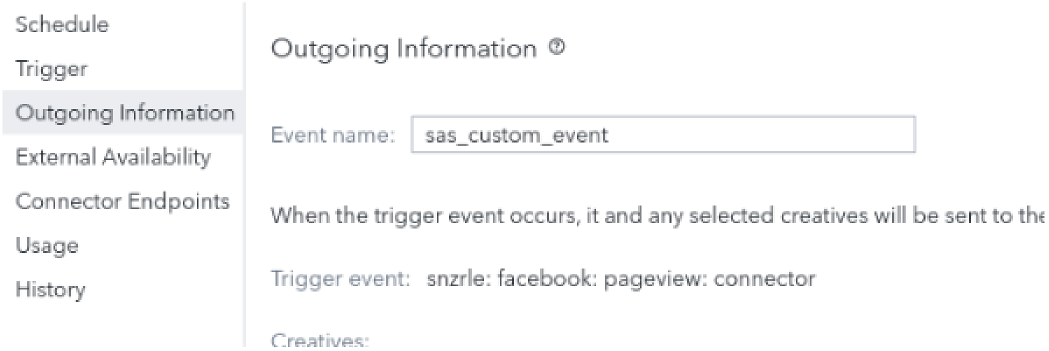
Select Criteria

face

Items (2)

Name
snzrle: facebook: pageview: connector

a. Click **Ok**
b. Go to **Outgoing Information**
c. Type in **Event name to** "sas_custom_event" this value is going to display in FB.



Outgoing Information

Event name: sas_custom_event

When the trigger event occurs, it and any selected creatives will be sent to the

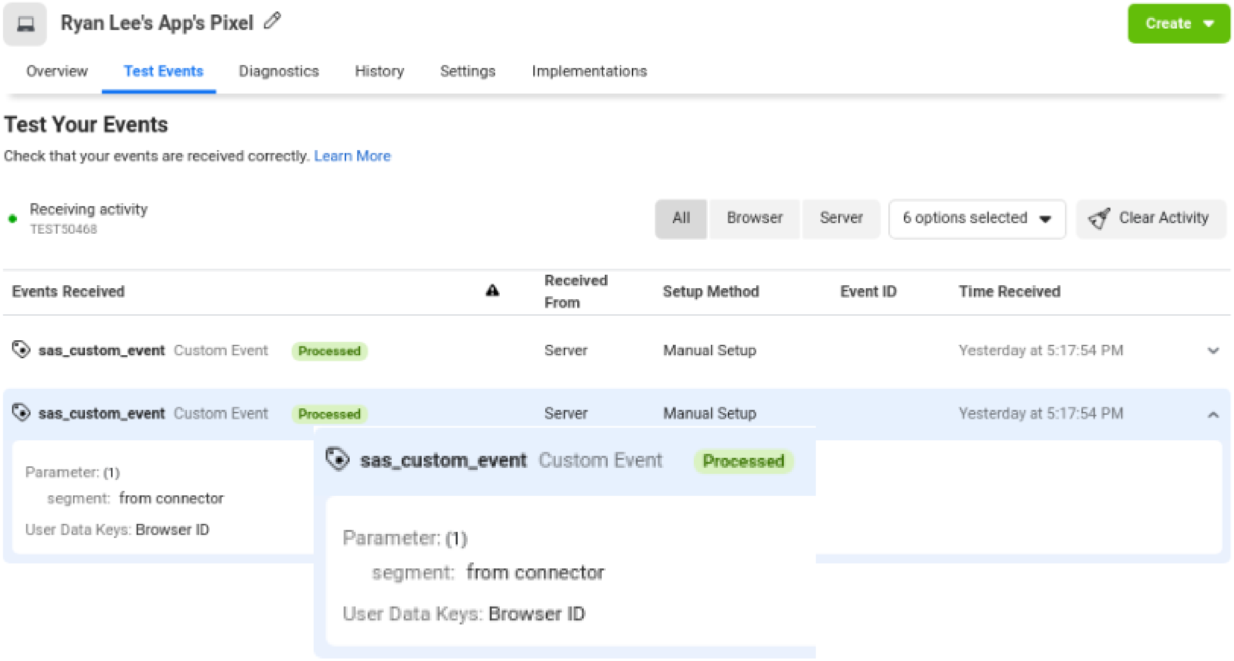
Trigger event: snzrle: facebook: pageview: connector

Creatives:

d. Optional
    i. Add any other targeting criteria you want based on your requirements by navigating to the **Targeting** tab
    ii. Add a schedule making sure this task only fires during specific times

6.5 Verification

Here we will verify if the Pixel is properly sending data to FB servers. Using the values we send to Facebook we will see if those values show up in the **Custom Report** section of the GA interface. The Custom Report setting as per screen shot:



Test Your Events

Check that your events are received correctly. Learn More

Receiving activity
TEST50468

Events Received		Received From	Setup Method	Event ID	Time Received
sas_custom_event Custom Event Processed		Server	Manual Setup		Yesterday at 5:17:54 PM
sas_custom_event Custom Event Processed		Server	Manual Setup		Yesterday at 5:17:54 PM

Parameter: (1)
segment: from connector
User Data Keys: Browser ID

sas_custom_event Custom Event Processed

Parameter: (1)
segment: from connector
User Data Keys: Browser ID

7 Appendix

7.1 External References

Item	Link
Sending Data to (Pixel/API)	https://developers.facebook.com/docs/marketing-api/conversions-api/using-the-api
Upload Custom Audience	https://developers.facebook.com/docs/marketing-api/audiences/guides/custom-audiences
JSON Selector (API/Connectors)	http://jsonpath.com/

7.2 Source Code

7.2.1 Facebook Pixel Example:

Note: You do need to replace **"<pixel_id>"** part of the code!

```
<html>
<head>
  <meta content="text/html;charset=UTF-8" http-equiv="Content-Type" /
  <title></title>
</head>
<body>
<!-- Facebook Pixel Code -->

<script>
!function(f,b,e,v,n,t,s)
{if(f.fbq)return;n=f.fbq=function(){n.callMethod?
n.callMethod.apply(n,arguments):n.queue.push(arguments)};
if(!f._fbq)f._fbq=n;n.push=n;n.loaded=!0;n.version='2.0';
n.queue=[];t=b.createElement(e);t.async=!0;
t.src=v;s=b.getElementsByTagName(e)[0];
s.parentNode.insertBefore(t,s)}(window, document,'script',
'https://connect.facebook.net/en_US/fbevents.js');
fbq('init', '<pixel_id>');
fbq('track', 'PageView');
</script>
<noscript><img height="1" width="1" style="display:none"
src="https://www.facebook.com/tr?id=<pixel_id>&ev=PageView&noscript=1"
/></noscript>
<!-- End Facebook Pixel Code -->

</body>
</html>
```

7.2.2 Facebook API Endpoint

https://graph.facebook.com/v8.0/<fb_pixel_id>/events?data=[{event_name:"{{$eventname}}",event_time:{{$timestamp}},user_data:{fbq: "{{$fbp}}"},custom_data:{{{$eventcategory}}:"{{$event}}"}}]&test_event_code=TEST50468

i. Set eventname to “**$.customName**”
ii. Set timestamp to “**$.properties.fb_ts**”
iii. Set fbq to “**$.properties.fbq**”
iv. Set eventcategory to “**$.customGroupName**”
v.

7.2.3 SAS CI360 Tag Example Code

Note: This is for examples use only, please copy the code from your CI360 from the General Settings > SAS Tag Instruction section

SAS Ci360 Tag Example:

```
<!-- SAS Ci360 Tag -->
<script>
(function(ci){
  var ef=window[ci]=function(){
    return ef.q.push(arguments);
  };
  ef.q=[];ef.a={};
})('ci360');
</script>
<script async data-efname='ci360' id='ob-script-async'
    a='<tenant ID>'
    src='https://<server>/js/ot-all.min.js'></script>
```

7.2.4 SAS Generic Pixel Placeholder

For generic spot creation

```
<!-- SAS Pixel Placeholder -->
<div id="sas_pixel"></div>
```

7.2.5 SAS Facebook Placeholder

For Facebook spot creation

```
<!-- SAS Pixel Placeholder -->
<div id="sas_facebook_pixel"></div>
```

7.3 Connector JSON Payloads

Below is a cut down version of the json payload for reference, you can use http://www.jsonpath.com/ to find the value you want using JSON selector notation

7.3.1.1 Custom Event

Example JSON Selectors

Selector	**Value (using the JSON example below)**
$.guid	d4a1b51b-cc9b-4b25-8d76-1b5021625a5a
$.eventName	Outgoing_1
$.customName	has_been
$.customGroupName	segment
$.sessionID	e105513fe064112b551dccc3
$.channelType	external
$.date.generatedTimestamp	1597647116198
$.externalTenantId	Abcdef0123456789
$.internalTenantId	xxxx
$.identityId	2d25b302-6ee9-35fc-ba41-8c3033f90266
$.visitId	930981fa7e3a4c58623bc80d
$.properties.ga_cid	1832800649.1603260950
$.properties.externalCode	TSK_104
$.identity.identityId	2d25b302-6ee9-35fc-ba41-8c3033f90266
$.identity.sessionId	e105513fe064112b551dccc3

Ex. Using the date value below in a connector variable you would use $.date.generatedTimestamp and the value returned would be 1597647116198

7.3.1.2 Example JSON Payload

It is highly recommended that you setup a server to receive the JSON payloads in order to see any differences and confirm the code is correct. This should be used for reference only!

```
{
  "guid": "d4a1b51b-cc9b-4b25-8d76-1b5021625a5a",
  "apiEventKey": null,
  "eventDesignedId": "d780a02f-2e96-4e46-a80a-9885e23bb4ec",
  "eventDesignedName": "event", //Defined in the external task under Orchestration > Outgoing Information >
Event Name
  "eventName": "Outgoing_1", //Defined in the external task under Orchestration > Outgoing Information > Event
Name
  "customName": "has_been", //Custom Name defined in the trigger event (custom event) under Event Details >
Event Attributes > Custom Name > Value text
```

```
  "eventType": "outboundSystem", //Defines which kind of task, external tasks are outboundSystem tasks, events would be customEvent etc.
  "sessionId": "e105513fe064112b551dccc3", //Defines the sessionID of the current user
  "channelId": "6ffeb2f866cb3044264262ab",
  "channelType": "external", //Defines the channelType
  "ipAddress": null,
  "date": {
    "generatedTimestamp": 1597647116198, //Timestamp in Unix Epoch format
  },
  "externalTenantId": "abcdef0123456789", //The external Tenant ID (used in the tag on the site)
  "internalTenantId": xxxx, //The internal Tenant ID
  "identityId": "2d25b302-6ee9-35fc-ba41-8c3033f90266", //The datahub ID used to define a user
  "page": {
    "loadId": "9df574667a3a4c587dca6ec7",
    "viewSequenceNum": 0,
  },
    "visitId": "930981fa7e3a4c58623bc80d", //The visitID for the user associated with this event
    "visitorGroup": null,
    "visitorState": "returning",
    "visitSequenceNum": null
  },
  "properties": { //This section contains most of the custom attributes and custom properties
    "ga_cid ": "1832800649.1603260950",
    "parent_eventname": "load",
    "externalCode": "TSK_104", //The external code defined in the external task under Properties
    "event_datetime_utc": "1597647116198", //Event time in Unix Epoch
    "parent_event": "customEvent",
    "contributing_guid_1": "a217d4c4-34f9-460a-afb7-0b7ade117c7e"
  },
  "identity": {
    "identityId": "2d25b302-6ee9-35fc-ba41-8c3033f90266",
    "identityType": null,
    "identitySource": null,
    "identityEventName": null,
    "identityAttribute": null,
    "identityAssociation": null,
    "userId": null,
    "visitId": null,
    "ipAddress": null,
    "sessionId": "e105513fe064112b551dccc3",
    "visitorId": null,
    "loginEventType": null,
    "loginType": null,
    "loginValue": null
  },
  "customGroupName": "adobeaam",
  "extendedCustomEventWithRevenueFlag": false,
  "parentEventUid": "9155c61d-dd69-4fbb-9b49-c6f73490b900"
}
```