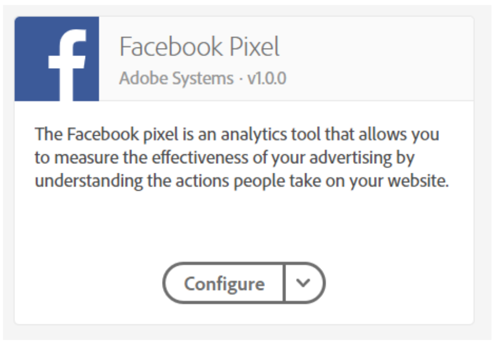
Adobe Launch – Mapping Table

This document provides you with some example use cases as well as some advanced techniques of how to use this extension.

Contents

Simple Facebook Pixel based on URLs

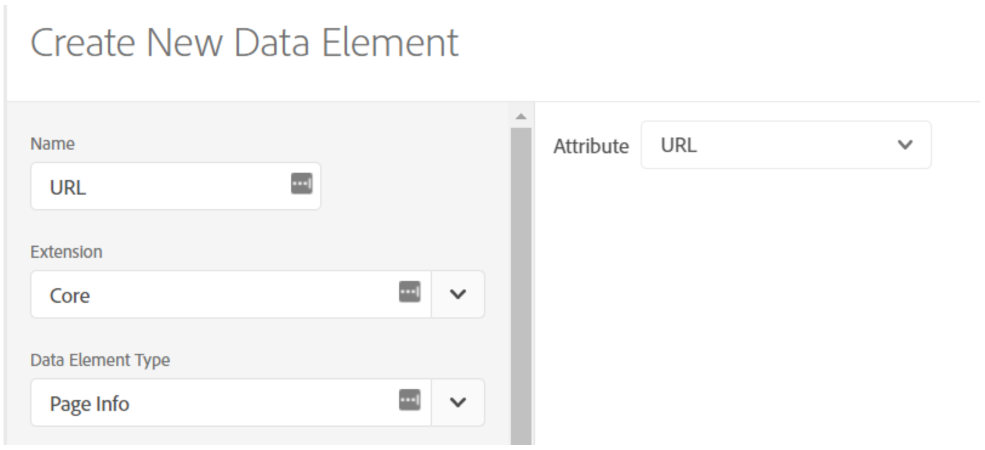
Install the FB Pixel extension:



Facebook Pixel
Adobe Systems · v1.0.0

The Facebook pixel is an analytics tool that allows you to measure the effectiveness of your advertising by understanding the actions people take on your website.

Create a URL Data Element:



Create New Data Element

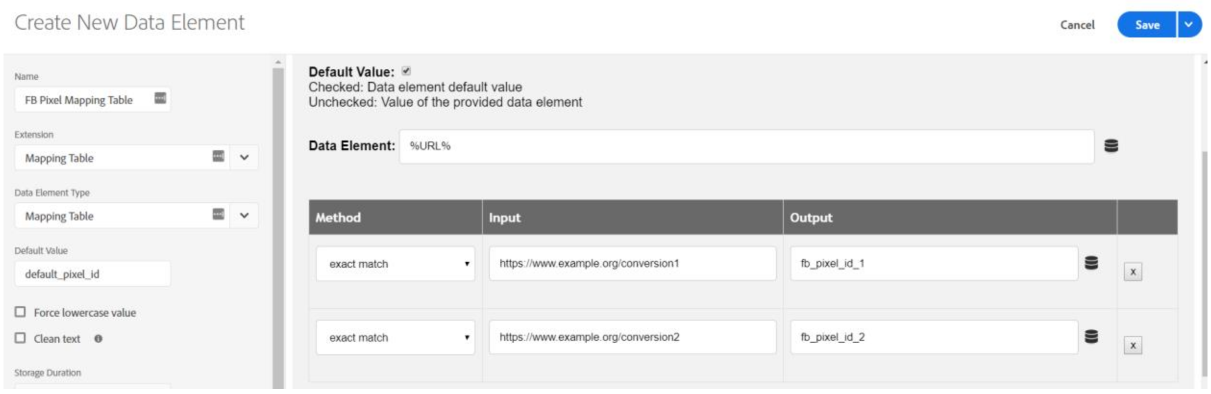
Name: URL

Extension: Core

Data Element Type: Page Info

Attribute: URL

Configure the Mapping Table:



Create New Data Element

Name: FB Pixel Mapping Table

Extension: Mapping Table

Data Element Type: Mapping Table

Default Value: default_pixel_id

☐ Force lowercase value

☐ Clean text

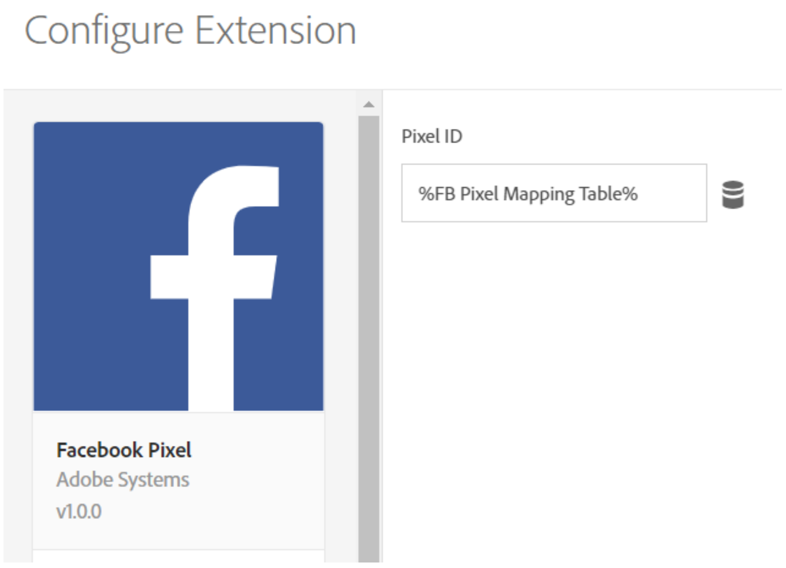
Storage Duration

Default Value: ✓
Checked: Data element default value
Unchecked: Value of the provided data element

Data Element: %URL%

Method	Input	Output
exact match	https://www.example.org/conversion1	fb_pixel_id_1
exact match	https://www.example.org/conversion2	fb_pixel_id_2

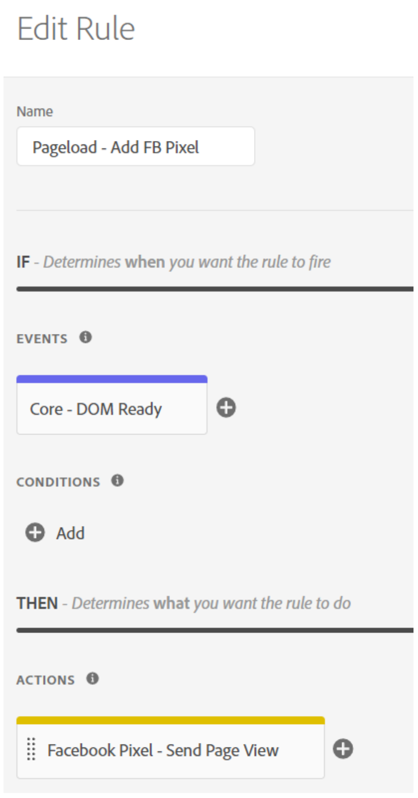
Configure the FB Pixel extension and select the mapping table data element:


Configure Extension

Facebook Pixel
Adobe Systems
v1.0.0

Pixel ID: %FB Pixel Mapping Table%

Create a rule that fires the pixel on page load:


Edit Rule

Name: Pageload - Add FB Pixel

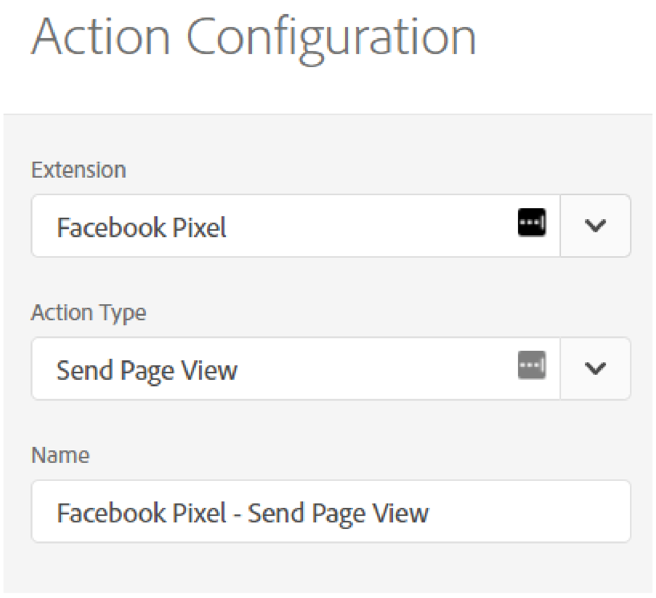
IF - Determines when you want the rule to fire

EVENTS: Core - DOM Ready

CONDITIONS: Add

THEN - Determines what you want the rule to do

ACTIONS: Facebook Pixel - Send Page View


Action Configuration

Extension: Facebook Pixel

Action Type: Send Page View

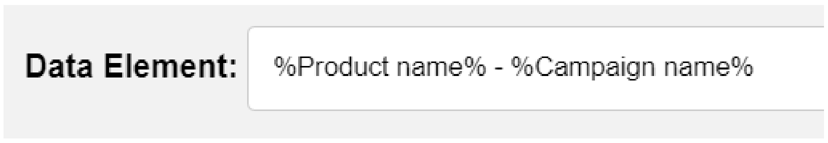
Name: Facebook Pixel - Send Page View

Done! The Facebook Pixel will now fire on the respective pages, based on your mapping table configuration.

Mapping Table based on multiple input data elements

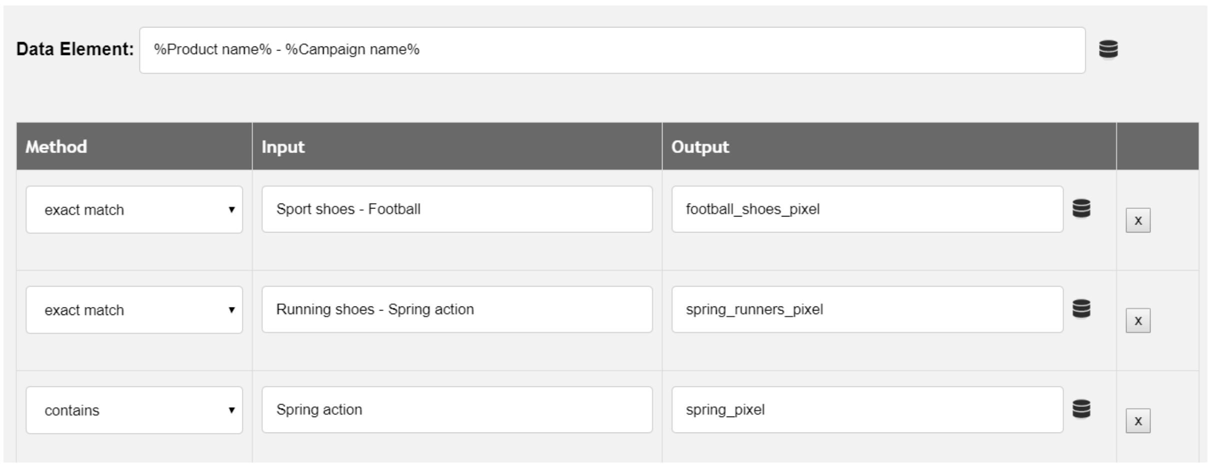
Sometimes it's not enough to just rely on the URL. Often customers want to create mappings based on the product name **and** the current campaign.

Let's see how this could be done using the Adobe Launch Mapping Table extension.



Data Element: %Product name% - %Campaign name%

By concatenating two data elements, one can use the combined value to be matched against. This can be done with any number of data elements.



Data Element: %Product name% - %Campaign name%

Method	Input	Output	
exact match	Sport shoes - Football	football_shoes_pixel	x
exact match	Running shoes - Spring action	spring_runners_pixel	x
contains	Spring action	spring_pixel	x

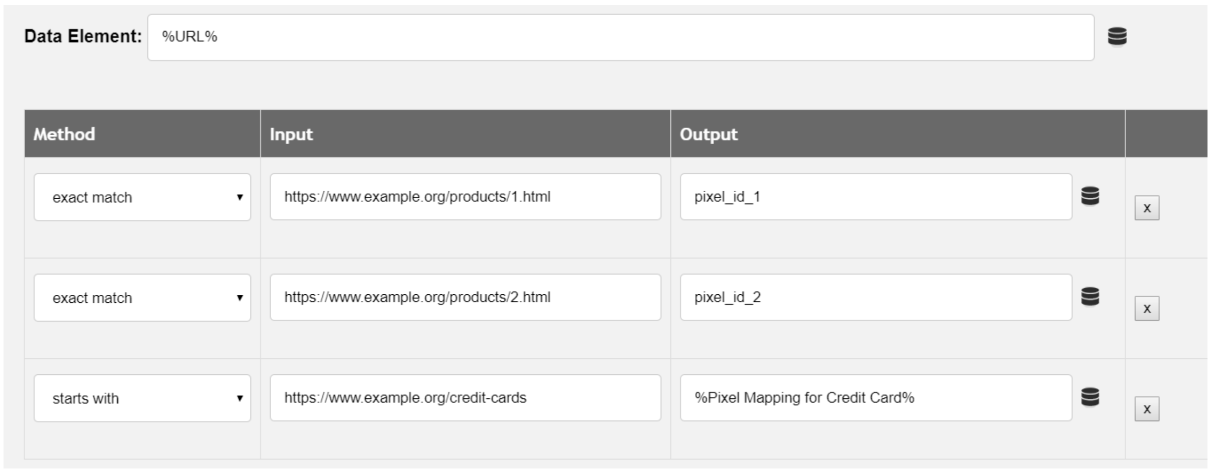
In the table above, we define two exact matches for a combination of a product and a campaign, as well as a fallback for all "Spring action" campaigns.

Advanced mapping – Nested tables

Sometimes very complex structures shall be covered by a mapping table, where it is not enough to concatenate the input values like in the previous example, but to have a completely new mapping.

This is possible with the Adobe Launch Mapping Table extension.

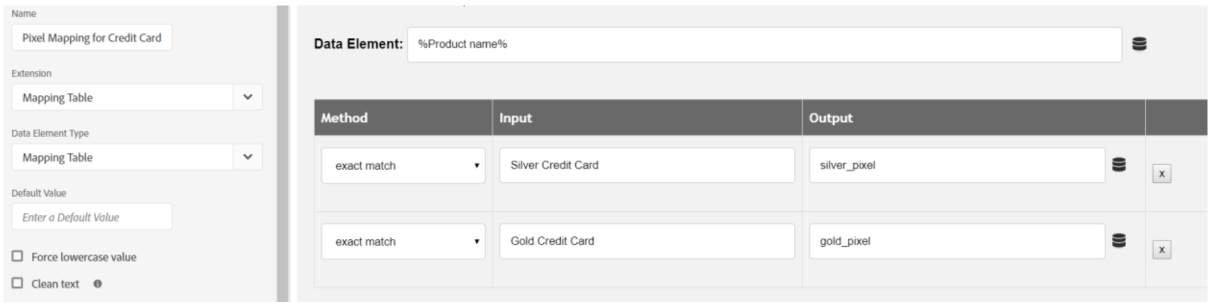
Let's assume that we want to have a mapping for some sites based on a URL, but for all the sites containing a credit card offer, we want to additionally consider the product name.



Data Element: %URL%

Method	Input	Output
exact match	https://www.example.org/products/1.html	pixel_id_1
exact match	https://www.example.org/products/2.html	pixel_id_2
starts with	https://www.example.org/credit-cards	%Pixel Mapping for Credit Card%

By defining the output of the credit card match to be a data element, we can simply specify another mapping table, which will calculate which value should be served.



Name	Pixel Mapping for Credit Card
Extension	Mapping Table
Data Element Type	Mapping Table
Default Value	Enter a Default Value

☐ Force lowercase value

☐ Clean text

Data Element: %Product name%

Method	Input	Output
exact match	Silver Credit Card	silver_pixel
exact match	Gold Credit Card	gold_pixel

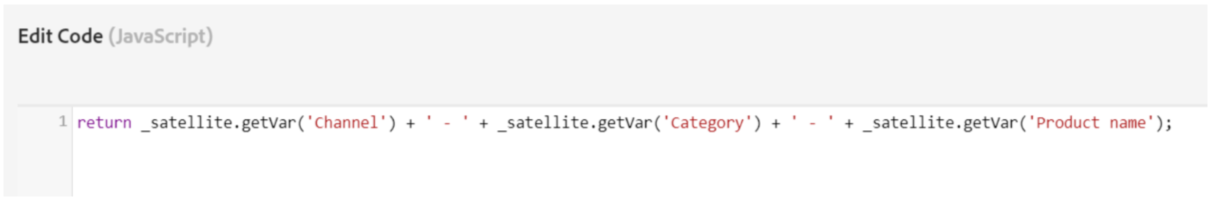
In the second table, we simply define the desired output values, based on the product name.

There is no limitation of how many levels the mapping table can be nested. Keep in mind that this increases the complexity and makes it harder to understand which value will be returned in the end.

Creating a data element which concatenates values of other data elements

Sometimes it is handy to have a data element which consists of a concatenation of other data elements. For example, a page name could consist of “<Channel> - <Category> - <Product name>”.

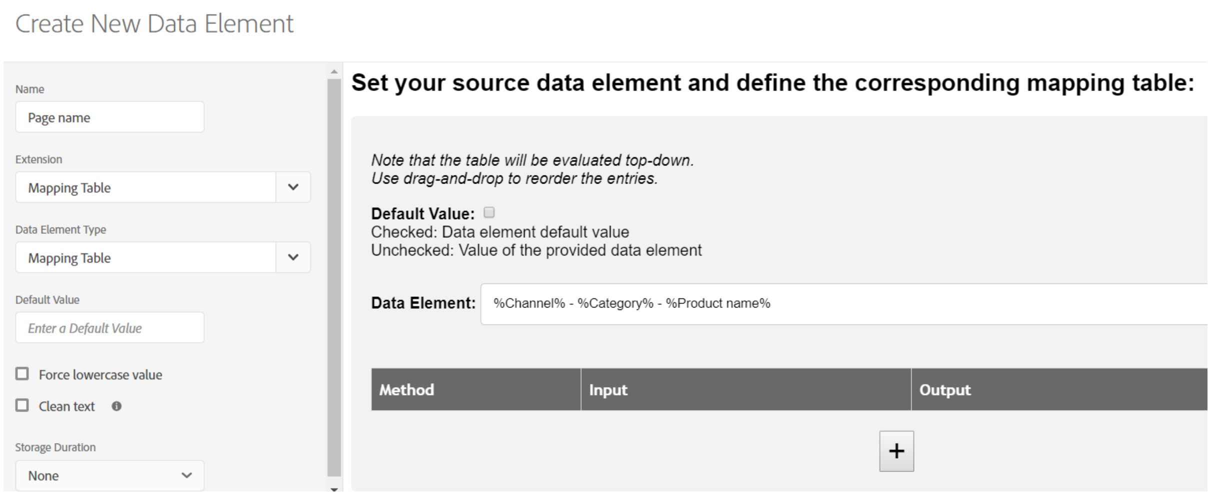
Now if we would like to have a data element for “Page name” we would have to use custom code. For example like this:



Edit Code (JavaScript)

```javascript
return _satellite.getVar('Channel') + ' - ' + _satellite.getVar('Category') + ' - ' + _satellite.getVar('Product name');
```

By using the mapping table in an unusual way, one can overcome the custom code:



Create New Data Element

Name	Page name
Extension	Mapping Table
Data Element Type	Mapping Table
Default Value	Enter a Default Value
☐ Force lowercase value	
☐ Clean text	
Storage Duration	None

Set your source data element and define the corresponding mapping table:

Note that the table will be evaluated top-down.
Use drag-and-drop to reorder the entries.

Default Value: ☐
Checked: Data element default value
Unchecked: Value of the provided data element

Data Element: %Channel% - %Category% - %Product name%

Method	Input	Output

By unchecking the “default value” checkbox, the mapping table will return the value of the input data element in case there is no match. As we didn’t provide any matching rules, the mapping table will always return the concatenated values as we provided it in the input field.

Use Cases beyond Marketing pixels

Even though the most obvious use case for this extension is the implementation of marketing pixels, there are a few other use cases it can be used for:

- Complex page names
- Adobe Target at_property tokens
- Definition of which Adobe Analytics event should be fired based on URL, event, target link, …
- Etc.

There are virtually no limitations of how the mapping table can be used. Be creative :)