WEB

Section 1 - Getting Started

1.1 Prerequisites

The Accengage Web SDK provides a JavaScript module that will enable your website to use the Accengage Push Notification software very easily.

The current Web SDK version is **3.5.x** and currently works with **Chrome 42+** , **Firefox 44** (Windows/Apple desktop and Android devices) , and **Safari 11.1 (and higher) for Mac**.

In order to integrate the Web SDK, it is recommended to have some knowledge in **HTML** , **CSS and JavaScript** .

For the basic integration and customization, it is possible to proceed **without a deep knowledge**, especially if you are used to Tag Management Systems. For the advanced customization and tagging, it is however recommended to have a good knowledge of these.

1.2 Compatibility Matrix

The current version of the Accengage Web SDK supports the following browsers:

- Chrome 42+ on PC, MAC and Android phones
- Firefox 44+ on PC and MAC and Android phones
- Safari 11.1+ for Mac

Accengage complies with the **Push API W3C standard**, which means that any browser which complies with this standard, will be compatible with our Web Push SDK. This includes:

- Chrome
- Firefox

Safari on Mac also supports Web Push Notifications, but uses another protocol, very similar to how iOS supports Push Notifications.

Note that Safari on iOS does not support Web Push Notifications, and neither do any other browser.

Browser	version	Compatible
Chrome	44+	✓
Firefox	46+	✓
Safari	11.1+	✓
Opera	/	✗

Below is the rendering of Web Push Notifications following the format of the message:

Message	Images
Simple	
Button & Images (can be seperated)	

The Accengage SDK is compatible with **Progressive Web Apps 'PWA'** (on Android Chrome Browser), and is not compatible with Accelerated Mobile Pages 'AMP' (on mobile devices).

1.3 Setup

1.3.1 Google Chrome Setup

In order to activate Web Push Notifications for your website, you will need to create and obtain a **Google API Key**, and an **Android Sender Id**.

These credentials will enable Accengage to be the authorized sender for your Web Push Notifications.

They are associated to your GCM (Google Cloud Messaging) or FCM (Firebase Cloud Messaging) project. If you do not have a FCM or GCM project already, you will first need to create a project on Firebase by following this link : FCM console.

Once your project is created, go to **"Settings > Cloud Messaging"**, and retrieve your **Server Key** which will be used as your Google API Key, and your **Sender ID** which will be used as your Android Sender ID.

To send Web Push Notifications on Firefox, there is no need for this type of credentials. Configuring the Accengage Dashboard will give you access to Web Push Notifications on Firefox.

1.3.2 Safari setup

Safari compatibility is available from SDK 3.4.x . In order to target Safari users, please update your SDK version in the apps settings of the Accengage's dashboard.

From Safari 12.1, native permission can't be displayed automatically to the user.

Indeed, since this version it is required to get an action from the user to display the permission.

For the "none" scenario you must need to use the method "requestBrowserPermission" behind the action button that the user will click on. More information on this page.

In order to send push notification to Safari users, you will need to generate and obtain a **Push Certificate**, and a **Website Push ID** from Apple developer website.

In the Apple Developer website, go to the " ***Certificates, IDs & Profiles"*** secton.

Then go to ***"Identifiers > Website Push IDs".***

Then click on the ***"+"*** icon to create a new ***"website push id".***

You shall be redirected to this section :

> Please note the identifier you have just set (example : web.com.mycompany.myapp), since it will be required to inform it later on the Accengage configuration interface (see 1.4).

Click on ***"Continue"*** and save your new website push id.

Next, click on ***"Certificates*** **> Production"** (or Certificates > Development - for a testing environment)

Select ***"website Push ID Certificate"*** .

Click on "**Continue**".

In the next page, select the website push id previously created.

Then click on ***"Continue"*** . Follow the next steps provided by Apple to generate your certificate.

At the end, you should be provided a ***".p12"*** file.

Save this file as you will need to upload it in Accengage configuration interface (See 1.4)

1.3.3 Firefox setup

To send Web Push Notifications on Firefox, there is no credentials needed

1.4 Configure the Accengage dashboard

Once you have subscribed to Accengage services, our support team will provide you with a **Partner Id** and a **Private Key** for each website and each environment

- Development
- Production

As well a '**MASTER DOMAIN'**. These are only available for registered customers on our platform. If you do not have a partner id, private key and MASTER DOMAIN yet, please contact our support at http://ticket.accengage.com

Within the Accengage Dashboard, you will see the created applications in **Settings > Manage** your applications.

Our support team will create your Web Application. To do so, we will ask you to provide some **basic informations**:

Field	Description
Registered Domains	Domain names with which you want to use our Web SDK. For example if you have different top-level-domains *".com"* and *".fr"*, you must register them as two distinct domains. The same logic applies if you have an optional *"www."* subdirectory. ***For Safari****, domain names have to be written entirely and without wildcards (*)*
Google API Key	Refer to the part "Activate Web Push Notifications" Create a FCM project to obtain these credentials.
Android Sender ID	

Default URL on Notification Click	A default URL (generally your website's home page) used to open a new window when the user clicks on your notifications. Obviously, you will be able to use a more specific URL while creating a message.
Default Notification Image	A default image displayed on your notifications. Same as above, you can use a specific image when editing messages. Please note that **for Safari** *Default Notification Image* is mandatory to generate a valid push package.
Certificate	When creating a certificate on the Apple Developer website, a ".p12" file has been generated. We must need this file to configure your application in the Accengage Dashboard.
Website push id	When creating a certificate on the Apple Developer website, it has been asked to you to register a Website Push ID. We must need it to configure properly your application in the Accengage Dashboard

That's it! You've now finished the initial configuration for your web application. Let's now look at how to integrate the Accengage Web SDK into your website

When you play the video, click on HD to enhance video quality

Section 2 - Web SDK Integration

2.1 Overview

The Accengage Web SDK consists of a Javascript module, which simplifies and manages the interaction with the Accengage Push Notification software.

By adding the Accengage's snippet inside your html code, you will be able to handle :

- The Web Push Notification Opt-in process, in order to collect user permission (this is detailed in the next section)
- The Web Push Notification reception, display and tracking
- User profile and behaviour tagging in order to collect the data you wish and store it on the Accengage servers for targeting purposes (detailed in the section "User Profile and Behaviour tracking").

The successive steps in order to integrate and use the Accengage Web Push software are:

1. Copy and paste Accengage Snippet (see below) into your website, that will call and integrate the Accengage SDK
2. Customize the opt-in process, to maximize opt-in user collection and adapt it to your design and user experience
3. Tag relevant information on users, using Javascript methods, which will interact with the Accengage SDK.

When you play the video, click on HD to enhance video quality

2.2 Mandatory web SDK integration steps

2.2.1 Paste a code snippet into your website pages

Accengage has developed its Web Push Notification SDK so as to simplify at maximum the integration process for you. As a result, you just need to copy and paste a Snippet into your website pages, that will call and integrate the Accengage SDK, and most of the work will be done! No need to host files, maintain them etc.

The integration of our library in your website is simple.
Once your Web app is registered on the Accengage Interface, you'll receive your partnerID and Master Domain name (see Section 1)
.
You just need to paste a small piece of JavaScript into your pages, replacing the MASTER DOMAIN referenced below with the one which was provided to you:

Code snippet to paste

```
(function(l,o,a,d,i,n,g,w,e,b){
  g='AccengageWebSDKObject';w='script';l[g]=l[g]||{};l[g][n]=d;
  l[d]=l[d]||[];l[d].p={'date':1*new
Date(),'window':l,'document':o,'params':a};
  e=o.createElement(w);b=o.getElementsByTagName(w)[0];e.async=1;
  e.src='https://'+n+i+'/init.js';b.parentNode.insertBefore(e,b);
})(window,document,{},'ACC','/pushweb/assets','MASTER DOMAIN');
// Replace the reference `MASTER DOMAIN` with the domain provided by Accengage
```

Use another reference name

In the above snippet code, If you would like to use another reference name instead of 'ACC', in the snippet code, you will notice in the last line a few defined variables.

You can edit the "ACC" <string> parameter for another one. For instance , if you set the value to "foo" , you would use our library with the foo.push( ) method.

From there, the Web Push Notifications are functional, and you can test sending Web Push Notifications!

We do recommend however to go further and customize the opt-in process (read the next section), and to set some tags for relevant targeting.

In order to customize the experience further or to activate some functions for tracking purposes, you will need to create some commands within the Javascript. To see how to proceed, please read the beginning of Section 4, or the beginning of Section 7

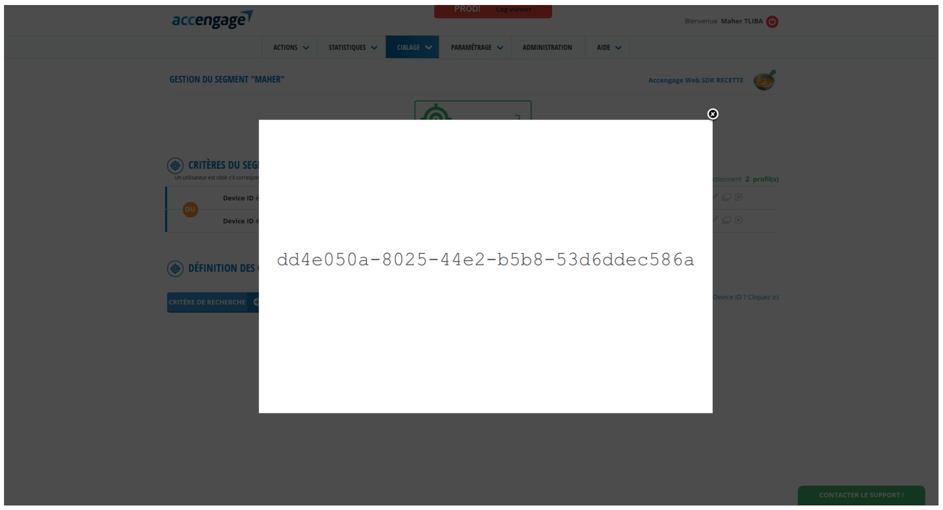
2.2.2 Test sending Web Push Notifications

In order to test a Web Push on your own browser and device, you can send a simple Web Push Notification targeted on yourself.

For that purpose, you need to find your own Device ID and create a test segment within the Accengage dashboard.
Go to the Accengage Dashboard, and open it with the browser you wish to target.

Go to Targeting > Search or to Targeting > Segments/Lists > Add New Segment > Manage Criterion.

Click on the link "Want to find your Device Id? Click here". Your Device ID will appear:



dd4e050a-8025-44e2-b5b8-53d6ddec586a

Now that you have retrieved your Device ID, you can create a test segment to target your device, and send a simple Web Push Notification on this test segment.

If you would like more help on how to create a test segment and how to send a simple Web Push Notification, you can follow these User Guide links:

Segments and static lists

Editing sending and scheduling a web push

That’s it, you should be able to send your first Web Push Notification!

Of course, we recommend you to go further and to:

- Customize the opt-in process see the next section
- Collect user data to enrich the user profiles for more targeted and relevant messages see the section “User Profile and Behaviour tracking”.

2.3 Useful commands

This first step is to edit the "MASTER DOMAIN" <string> with the master domain created with your Accengage Application. Check this section of our guide to have more information about process of creating an Accengage Web Application.

```
// If the snippet is still onload, it will create an array
// Otherwise, it will use the ACC object previously declare in the code
snippet
var ACC = ACC || [];

/*************************************/

// In the most common scenario, the library will be accessible through your
global `window` scope
// If you develop with IIFE, don't forget to export it
(function(global) {
    global.ACC = global.ACC || [];
})(window);
```

In order to use the more advanced functions that will be described in the following sections (for opt-in process customization, or tagging purposes), you will need to create some commands within the Javascript.

In general, when you want to communicate with the SDK, you will need to create a command by calling the main JavaScript Object ‘ACC’ (read this section if you want another reference name). As the code snippet could still be on a loading state (because you are using a tag manager for example), we suggest you declare the ‘ACC’ object as an <array> as soon as possible.

For instance:

```
var ACC = ACC || [];
```

Then you just need to use the ‘push’ method of the ‘ACC’ object to create a new command.

For instance:

```
ACC.push([ "plugin_name:method_name" , options, callbacks ]);
```

The three parameters mostly accepted are:

Argument	Type	Description
1 name	<string>	always required, containing the plugin name and method name to call, separating by a ":"
2 options	<*>	containing the option of the command
3 callbacks	<object>	containing callbacks <function> type, applied when the command has been consumed

To summarize, the **ACC** JavaScript Object is set thanks to the snippet code you have previously added in your page. If the library is not fully loaded when the command is created, it will be put inside a queue, waiting for the library to load. When the library is loaded, the stored commands, or next created commands, will be properly consumed.

2.4 Updating the SDK version (migration guide)

From 3.x.x To 3.3.x

If you want to migrate to version 3.3.x of our SDK Web and benefit of the new and easier optinization process for your users engagement, please follow these next steps.

Important
This new opt-in process requires to work a domain under **HTTPs** protocol.

Note that if you migrate to 3.3.x and have a domain working under **HTTP** protocol, the current opt-in process will continue to work like others versions below 3.3.x of the SDK Web

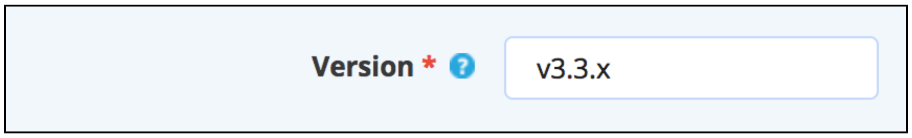
We advise to **follow these steps after each others** for the proper conduct of the migration and let the SDK Web operate effectively.

Mandatory steps :

(1/3) Edit the version number of your application :

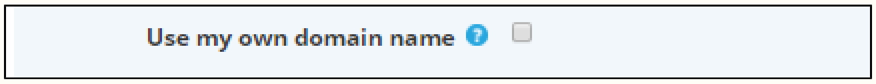
To get access to the new opt-in process of the SDK Web you should set your application version to 3.3.x.

To do so, please go into your Accengage Dashboard : Settings > Manage application > Edit > Field Version : change it to our last version available "**v3.3.x**"



Version * v3.3.x

Important
At this stage, please keep the checkBox 'Use my own domain name' **unchecked**



Use my own domain name ☐

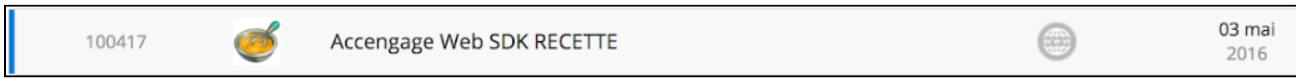
Click on "Save".

Changes will take approximately half an hour before getting live.

(2/3) Download and host to your web server the mandatory files :

Once you've upgraded the version number of your application, to operate the new opt-in process needs a zip file to be downloaded from our interface.

To download the zip file, please go to our interface > Settings > Manage Application > Click on the download icon at the right :



100417 Accengage Web SDK RECETTE 03 mai 2016

Once you've downloaded the ZIP file, you should extract files from this ZIP. You should obtain 3 files :

- acc_sw.js
- acc_ww.js
- manifest.json

Once extracted, **before going to the next step**, please upload those files to the root directory of your website.

The files "***acc_sw.js"*** and "***acc_ww.js"*** will be called by the SDK and the file "***manifest.json"*** just has to be referenced in the pages where the snippet will be present.

The optinization on your own domain needs a service worker running on your domain to operate (acc_sw.js)

If you already host a service worker on your website, you'll have to merge the Accengage one with yours.

To do so, add the lines below at the end of your service worker script :

```
var swURI = 'https://' + this.location.hostname + '/acc_sw.js';
importScripts(swURI);
```

Important
Those 3 files are only available for download to applications configured in the Accengage Dashboard with SDK's version number above 3.2.x (step 1/3)

Those files should be uploaded to the root directory of your website, unless you already host a "manifest.json" file, you should merge the lines generated by the "manifest.json" downloaded from our interface to the "manifest.json" hosted on your server.

Important
You need to reference "manifest.json" in all your pages by including this tag in the <head> :

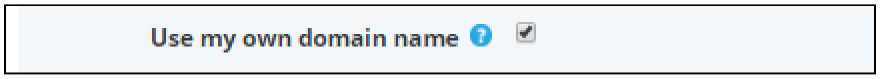
<link rel="manifest" href="/manifest.json">

If the "manifest.json" doesn't reside in your root directory please reference it like below :

<link rel="manifest" href="/path/to/your/manifest.json">

(3/3) Enable optin with your own domain anme :

Finally to set up the opt-in process you should enable it through the Accengage dashboard and the **"modify web application"** m odal screen, to do so please go to our interface > Settings > Manage Application > Edit your Web App > check **"Use my own domain name"**


Use my own domain name ✓

Note
Please keep the field **"master domain name"** __unchanged.__

It should still be the domain provided by Accengage during your first integration.

The master domain is referenced as **.by-accengage.net** or **.notification.group** if you have subscribed to a custom domain.

From 2.x.x To 3.x.x

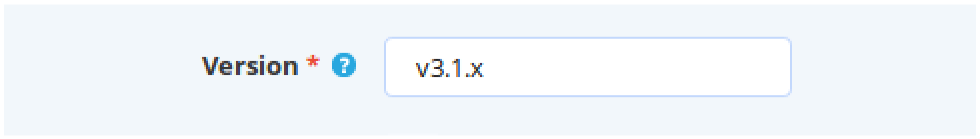
Mandatory steps

(1/2) Change your snippet

First of all, you have to change your old snippet for this new one . This snippet will work with every version. So even if you have a version 2 or 3, this snippet will launch our library in both cases.
Please keep in mind that if you don't upgrade your snippet, the version 3 would not be able to work.

(2/2) Edit the configuration of your application

Only then, you will be able to set your application from a version 2 to 3. To do so, please go to our interface > Edit your application > Version : and change it for our last version available **"v3.1.x".** Changes will take approximately half an hour before getting live.



Version * v3.1.x

Heads up! We have made few changes you need to know

Migrating from Cookie to LocalStorage

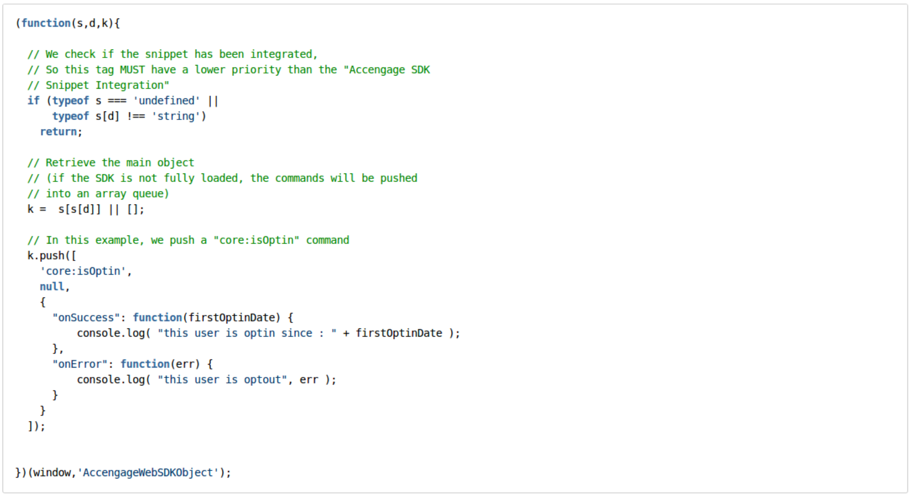
Regarding our built-in HTML Alert, we previously used Cookies to store the state of the component. For this new version, this state is now store within a LocalStorage. You could see some minor issues while migrating from version 2 to 3, e.g. a display of the alert although the user had previously denied it. It won't affect optin users, nor hard-optout users (that have previously denied the permission).

If you have followed one of our previous article about the use of Google Tag Manager

Please note that an edition could be necessary to make some "Tags" still working with the new library version.
It concerned the article published from last year to February 2017, and concerned about one tag made in the "Create basic interaction" and another one with the "Create complex interaction" sections.

What you have to change for the "Create basic interaction"

The documentation advised you to set your tag like this :



```
(function(s,d,k){

  // We check if the snippet has been integrated,
  // So this tag MUST have a lower priority than the "Accengage SDK
  // Snippet Integration"
  if (typeof s === 'undefined' ||
      typeof s[d] !== 'string')
    return;

  // Retrieve the main object
  // (if the SDK is not fully loaded, the commands will be pushed
  // into an array queue)
  k =  s[s[d]] || [];

  // In this example, we push a "core:isOptin" command
  k.push([
    'core:isOptin',
    null,
    {
      "onSuccess": function(firstOptinDate) {
          console.log( "this user is optin since : " + firstOptinDate );
      },
      "onError": function(err) {
          console.log( "this user is optout", err );
      }
    }
  ]);


})(window,'AccengageWebSDKObject');
```

We advise you a simpler approach, since our snippet have changed a little bit for the newly version 3 :

```
(function(global){

  // Retrieve the main object
  // (if the SDK is not fully loaded, the commands will be pushed
  // into an array queue)
  global.ACC = global.ACC || [];

  // In this example, we create a "core:isOptin" command
  // (check our documentation to see other possible commands)
  global.ACC.push([
    'core:isOptin',
    null,
    {
      "onSuccess": function(firstOptinDate) {
          console.log( "this user is optin since : " + firstOptinDate );
      },
      "onError": function(err) {
          console.log( "this user is optout", err );
      }
    }
  ]);
})(window);
```

What you have to change for the "Create complex interaction"

Regarding the Custom HTML Tag "Accengage SDK API", the article guided you to set it like this :

```
(function(w,e,b){

  // We check if the snippet has been integrated,
  // So this tag MUST have a lower priority than the "Accengage SDK
  // Snippet Integration"
  if (typeof w === 'undefined' ||
      typeof w[e] !== 'string')
    return;

  // Retrieve the command, GTM keeps a reference to inner command Object,
  // so we clone the command ...
  var command = JSON.parse(JSON.stringify(b));
  // ... and delete the original
  b[1] = null;
  delete b[1];

  // Retrieve the main object
  // (if the SDK is not fully loaded, the commands will be pushed
  // into an array queue)
  (w[w[e]] || []).push(command);

})(window,'AccengageWebSDKObject',{{Accengage SDK Push Command}});
```

We guide you now with a simpler approach :

```
(function(global,data){

    // Retrieve the main object
    // (if the SDK is not fully loaded, the commands will be pushed
    // into an array queue)<link rel="manifest" href="/manifest.json">
    global.ACC = global.ACC || [];

    // Retrieve the command, GTM keeps a reference to inner command Object,
    // so we clone the command ...
    var command = JSON.parse(JSON.stringify(data));
    // ... and delete the original
    data[1] = null;
    delete data[1];

    // Push the command
    global.ACC.push(command);

})(window,{{Accengage SDK Push Command}});
```

Version 2 documentation

Follow this link to check our previous documentation for the **version 2**

Section 3 - Customization and optimizations of the opt-in process

3.1 Overview

You can collect opt-in users from any website, any domain or subdomain (such as *yourwebsite.com*, *yourwebsite.de*, or *shop.yourwebsite.com*…), whether http or https, and have a unified database for your Web Push Notification messages.

In order to send Web Push Notifications to your website users, you first need to obtain their **permission** (‘opt-in’). We call the whole procedure to obtain user permission, the **‘opt-in process’**.

This process has a strong impact on the volume of opt-in users.

Before explaining how to customize the opt-in process, we want to make you familiar with it and the different steps.

The user permission to send Web Push Notifications is obtained via a browser permission message, which is not customizable. It has a set design and wordings are decided by the browser.

Accengage provides you with 2 opt-in process :

- One is done on the Accengage's domain (or a custom domain):
    - the opt-in process will be done through the Accengage's domain (or a custom domain) and users will be ask to opt-in through a landing page which, when displayed, will call the browser's native permission
    - This optinization process is available for websites in HTTP and HTTPs
    - Check out this documentation to implement it
- The other is done directly on your own domain :
    - the opt-in process will be done through your own domain and the browser's permission will be called directly without the display of a landing page (but you can set up one if needed)
    - to use this method you must integrate our SDK 3.3.x and activate this optinization method in your Accengage dashboard
    - this optinization process is available only for websites in HTTPs
    - Please, check out our FAQ to know how to implement this new process !

Through Accengage domain or a custom one

Also, once the users have answered to this permission request (either with ‘**allow**’ or ‘**block**’), you will not be able to ask this permission again. If users have denied them previously, the only way users can allow them again, is to go through the browser settings... Not ideal !

For this reason, Accengage recommends to first ask the user for its intention, with a **customizable**, full-html and attractive message (which is called ‘**Alert**’), and then, once the user has expressed will allow Web Push Notifications, to trigger the native browser permission.

If the user does not express its intent to allow Web Push Notifications, then the permission will not be triggered, which allows you to ask for permission at a later stage. This is a great way to achieve better opt-in performance than by just triggering the native browser permission.

The Accengage Web SDK provides a Pop-in ‘Alert’ by default, with **numerous options** for it to be **customized simply** (see the ‘basic customization’ paragraph). But Accengage also provides a library of **more advanced and efficient scenarios**, such as interstitial, full-length banners, buttons… (see the ‘advanced scenarios’ paragraph).

We highly recommend you to customize the ‘opt-in’ process, as this can enhance significantly opt-in results.

Also, as Web Push Notification permission can only be triggered on **HTTPS** websites, and users must give their permission for **e ach domain or subdomain** which want to send Web Push Notifications, Accengage has developed a **solution to ease the process**.

This solution consists of the following:

- Once the users have expressed their intention to accept Web Push Notifications through the ‘Alert’
- A new Window is opened and displays an **HTTPS** landing page, with a unique MASTER DOMAIN (provided by Accengage)
- In this landing page, the native browser permission is prompted (see below)
- Once the permission is granted, the landing page registers the user to the Web Push Notifications and communicates all relevant information to Accengage.

As a result, you can collect opt-in users from any website, any domain or subdomain (such as yourwebsite.com, yourwebsite.de, or shop.yourwebsite.com…), whether http or https, and have a unified database for your Web Push Notification messages.

See below an illustrated example used on our website in which we have customized some steps:

Through your own domain

Starting from version 3.3.x of the SDK Push Web, Accengage allows you to ask for the opt-in through your own domain name.

This optout-to-optin process will follow these requirements : a link in your page > display of the browser's native permission

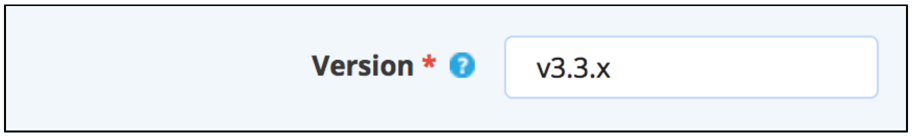
Important
This optout-to-optin process is available only for website working on **HTTPs** protocol.

In order to benefit this process we recommend you to **follow these steps after each other** :

(1/3) Edit the version number of your application :

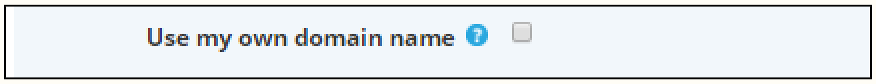
To get access to the new opt-in process of the SDK Web you should set your application version to 3.3.x.

To do so, please go into your Accengage Dashboard : Settings > Manage application > Edit > Field Version : change it to our last version available "**v3.3.x**"



Version * v3.3.x

Important
At this stage, please keep the checkBox 'Use my own domain name' **unchecked**



Use my own domain name ☐

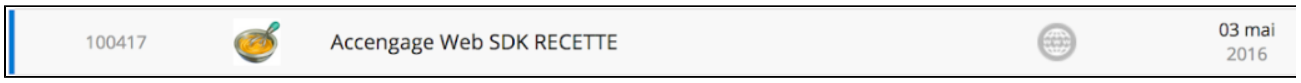
Click on "Save".

Changes will take approximately half an hour before getting live.

(2/3) Download and host to your web server the mandatory files :

Once you've upgraded the version number of your application, to operate the new opt-in process needs a zip file to be downloaded from our interface.

To download the zip file, please go to our interface > Settings > Manage Application > Click on the download icon at the right :



Once you've downloaded the ZIP file, you should extract files from this ZIP. You should obtain 3 files :

- acc_sw.js
- acc_ww.js
- manifest.json

Once extracted, **before going to the next step**, please upload those files to the root directory of your website.

The files "***acc_sw.js"*** and "***acc_ww.js"*** will be called by the SDK and the file "***manifest.json"*** just has to be referenced in the pages where the snippet will be present.

The optinization on your own domain needs a service worker running on your domain to operate (acc_sw.js)

If you already host a service worker on your website, you'll have to merge the Accengage one with yours.

To do so, add the lines below at the end of your service worker script :

```
var swURI = 'https://' + this.location.hostname + '/acc_sw.js';
importScripts(swURI);
```

Important
Those 3 files are only available for download to applications configured in the Accengage Dashboard with SDK's version number above 3.2.x (step 1/3)

Those files should be uploaded to the root directory of your website, unless you already host a "manifest.json" file, you should merge the lines generated by the "manifest.json" downloaded from our interface to the "manifest.json" hosted on your server.

Important
You need to reference "manifest.json" in all your pages by including this tag in the <head> :

<link rel="manifest" href="/manifest.json">

If the "manifest.json" doesn't reside in your root directory please reference it like below :

<link rel="manifest" href="/path/to/your/manifest.json">

(3/3) Enable optin on your own domain name :

Finally to set up the opt-in process you should enable it through the Accengage dashboard and the **"modify web application"** m odal screen, to do so please go to our interface > Settings > Manage Application > Edit your Web App > check **"Use my own domain name"**

Note
Please keep the field **"master domain name"** **unchanged.**

It should still be the domain provided by Accengage during your first integration.

The master domain is referenced as **.by-accengage.net** or **.notification.group** if you have subscribed to a custom domain.

Requesting directly the browser's native permission

Starting from SDK 3.3.x, Accengage gives you different ways to ask a user to opt-in:

- Display the pop-in alert (called showAlert) and then the browser's native permission (2 steps)

or

- Display the browser's permission when a user opens specific pages (1 step)

or

- Trigger the browser's permission when a user clicks on a specific link of your website

What are the benefits of each process ?

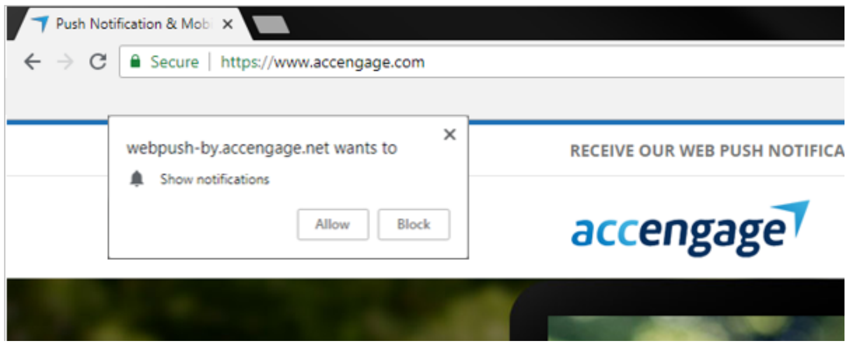
- **Pop-in alert :**

It allows you to explain the benefits of your web push notification to the user.

It can be customized to match with you're website appareance.

Your optin process will contain 2 steps - more details in the section "3.2.1 Edit the JSON plugin configuration"

- **Display the browser's native permission on specific pages :**


Push Notification & Mob
Secure | https://www.accengage.com
webpush-by.accengage.net wants to
Show notifications
Allow
Block
RECEIVE OUR WEB PUSH NOTIFICA
accengage

You can display automatically the browser's native permission when a user opens a specific page of your website with your own domain name.

To use this scenario you have to :

1. Edit your JSON configuration on the Accengage Dashboard and set your scenario as "none" (it will prevent the display of the pop-in alert) - more details in the section "3.2.1 Edit the JSON plugin configuration" .
2. Add the snippet on every page in which you want to display the browser's permission
    a. You want to ask the permission everywhere except page X ? Then add the snippet everywhere except page X.
    b. You want to ask the permission only on page Y; Z, W ? Then add the snippet only on these pages

This method allows you to only have one step in your optinization process.

Information
Please note that if the snippet isn't integrate into a page, you'll not be able to track user's actions on that page.

- **Calling directly the browser's native permission :**

It gives you the possibility to display the browser's permission when a user clicks on a specific link of your website

To do so :

1. Edit your JSON configuration and set your scenario as "none" (it will prevent the display of the pop-in alert) - more details in the section "3.2.1 Edit the JSON plugin configuration" .
2. Trigger the call-to-action method "**requestBrowserPermission**" when a user clicks on a specific link of your website :

Argument	Required	Type	Description
options	yes	<array>	of <string> containing selected anchors to bind. The value must start with "#" or "." (ID or ClassName)
callbacks	optional	<object>	• "onSuccess" : <array> provided elements that have been utterly binded • "onError" <string> label error

For example

```
ACC.push([
 "push:requestBrowserPermission",
 [
  "#myLink",
  ".myOtherLinks"
 ]
]);
```

Quick integration vs. customization

Our solution **can be fully configured**. You can legitimately configure it to create the most adapted process according to the look and feel of your website.

When you play the video, click on HD to enhance video quality

In the following paragraphs, we will look at:

- How to customize very easily the default Popin Alert which Accengage provides, as well as the Landing page and certain options
- How to customize the Popin Alert in a more advanced way
- How to create more advanced and efficient opt-in scenarios, to maximize results (we highly recommend you to create these)
- Some useful commands to help you during these steps.

This page intends to explicit **some common use cases about the optout-to-optin customization**

3.2 Basic Opt-in process customization

3.2.1 Edit the JSON Configuration

As explained in the overview, there are 2 ways to turn a website visitor into an opt-in user :

Through the Accengage domain (or a custom one) or your own domain.

Depending on the optin process you're using, some customization are available.

When asking the optin through the Accengage domain you will :

1. Show an opt-in ‘ **Alert** ’
2. Trigger a ‘ **Landing Page**’ containing the native Browser permission.

Optimizing the ‘opt-in’ process by customizing the opt-in alert and the landing page can **enhance significantly the volumes** of opt-in users, generating better results for your campaigns.

We highly recommend you to customize them and we have created many tools and options to help you to do so.

Starting from SDK 3.3.x, you have different ways to ask a user to opt-in through your domain:

- Display a pop-in alert (called showAlert) and then the browser's native permission (2 steps)
  - Show the pop-in 'Alert'
  - Trigger the browser's permission when a user has clicked on the pop-in "Alert"

or

- Display the browser's permission when a user opens specific pages of your website (1 step)

or

- Trigger the browser's permission when a user has clicked on a specific link of your website

Displaying directly the browser's permission (without displaying the pop-in 'Alert') or triggering it on specific links (option 2 and 3) **can also enhance significantly the volumes** of opt-in users.

If you want to use this process please click here.

Those scenarios have to be managed in the JSON configuration available in the Accengage Dashboard. Please read carefully the indications below.

Accengage has developed a Popin Alert by default, which is customizable and designed to encourage users to accept your Web Push Notifications.

In this first step, you will learn how to very easily and quickly **customize the wording** , the basic design and some display rules for this Popin Alert, directly from the Accengage **dashboard** with **just some HTML knowledge** .

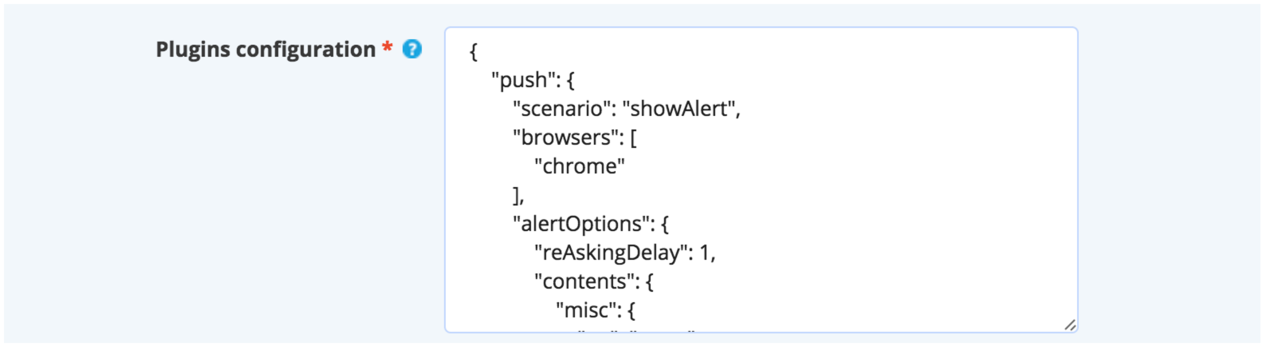
Then we will look at how to **easily customize the Landing Page**, which contains the native Browser permission.

Of course, if you would like to have a more successful and customized experience, you will be able to add your own **CSS** and also go further in terms of **scenarios**, by following the steps in the next paragraphs.

Caution
We highly recommend you to read through the '**advanced scenarios**' part and try and optimize your opt-in process as it is a key driver to the volumes of opt-in users.

This basic customization will imply modifying a **Json file configuration** within the **application settings** in the **Accengage Dashboard**.

To edit these options, you can go directly on our interface. Go to **Settings** > **Manage Applications** > **Edit your application** > **Plugins configuration** .



```
{
   "push": {
      "scenario": "showAlert",
      "browsers": [
         "chrome"
      ],
      "alertOptions": {
         "reAskingDelay": 1,
         "contents": {
            "misc": {
```

Note
When modifying these options, you will have to wait for approximately half an hour to see the changes live in your pages. However, with the Debug mode, you can speed things up (see below the paragraph Useful Commands [LIEN] to learn how you can immediately see the changes).

Within the "push" property in the Plugin configuration, you will be able to set these parameters:

Parameter	Type	Default	Description
scenario	<string>	"none"	This determines which action is automatically launched as soon as the library is loaded . Available values : • "**none**" : use this value if you do not want to show automatically the Popin Alert ◦ typically if you want to customize when to launch it ◦ or if you want a more customized experience ◦ or if you want to display the the permission only on specific pages (starting from SDK 3.3.x) ◦ or if you want to trigger the browser's permission once a user clicks on a specific link of your website (starting from SDK 3.3.x) • "**showAlert**" : use this value to show automatically the HTML alert
alertOptions	<object>	*see below*	Object containing the parameters for the **HTML alert**
landingOptinOptions	<object>	*see below*	Object containing the parameters for the **Landing page**
customError	<object>	*see below*	Object containing the parameters for the **customizing the Chrome Error Notification**. This message is displayed as a Web Push Notification in case the servers cannot be reached.

For Instance:

For example

```
// Plugins configuration
{
 ...

 // Set the "push" plugin options
 "push": {
  "scenario": "showAlert",
  "alertOptions": {
   ...
   "contents": {
    ...
   }
  },
  "landingOptinOptions": {
   ...
   "contents": {
    ...
   }
  }
 }

 ...
}
```

3.2.2 Basic customization of the Html Popin Alert

The **HTML Popin alert** is an interactive block injected by the library inside your page in order to incite users to receive push notifications and to detect their intent when they are not yet opt-in. It is an optional feature and it is the first step of the opt-in process.

Choose when to display the alert

You can choose the period to display the Popin Alert:

- Automatically when a user arrives on the website (in that case, set the **"scenario" option** to **"showAlert"**, click here for more information)
- At a specific moment that you decided thanks to the method **`push:showAlert`** (click here to see how this command can be used in your pages)

> This can be combined with the automatic display of the Popin Alert when the home page is loaded. This can ensure that you get maximum visibility, as well as a relevant exposure based on what the user is doing (e.g.: on the home page automatically AND after finalizing a purchase)

Customize the overall Popin Alert parameters

By editing the following parameters inside the JSON configuration file within the Accengage dashboard, you'll also be able to customize the wording, the basic design, the position as well as some display rules for the Popin Alert:

Parameter	Type	Default	Description
contents	<object>	*see below*	Lists the wordings and logos displayed to the user on the Popin Alert.
theme	<string>	"light"	Choose the theme's desgin to stylize the **HTML Popin Alert**. Available themes are: • "modern" • "light" • "custom" : no CSS will be injected, you have to customize the alert on your own (see Section 3) With the *"custom"* theme, don't forget to create your own CSS rule for the modifier class **. acc--hidden** (with a *"display: none"* for example)
position	<string>	"topRight"	Set the positions of the **HTML Popin Alert**. Available values are : • "topLeft", "topCenter", "topRight" • "middleLeft", "middleCenter", "middleRight" • "bottomLeft", "bottomCenter", "bottomRight"
overlay	<boolean>	false	If true, this will add a semi-transparent layer between your page and the Alert. With this option activated, users won't be able to click on your page's elements (only applied on "modern" theme) **Tip** This is a very interesting option to activate in order to maximize the number of opt-in users.
reAskingDelay	<number>	24	If the user clicks on the "close" button, or hasn't interacted with the alert in a previous page, this parameter defines the number of hours to wait before showing the **HTML Popin Alert** again. When set to 0, no delay will be taken into account

reAskingDelay2	<number>	24	If the user clicks on the "deny" button, this parameter defines the number of days to wait before showing the **HTML alert** again. When set to 0, no delay will be taken into account

If you would like to know all the details about how to set the ‘contents’ object, you can read below the part Useful Commands for more details.

Customize the Popin Alert Contents

Below is the 3 contents fields used in the **HTML Popin Alert** that you can modify by updating the **push.alertOptions.content** object . Note that you can include some **HTML** code within the content.

Content field	Description	Default value
misc	Value can be a: • Valid HTML <string> • Plain string <string> the <string> value hydrating all the ".acc--miscContent" DOM Elements contained in the **HTML alert**	none
deny	Value can be a: • Valid HTML <string> • Plain string <string> Hydrating all the ".acc–denyContent" component	none
accept	Value can be a: • Valid HTML <string> • Plain string <string> Hydrating all the ".acc–acceptContent" component	none

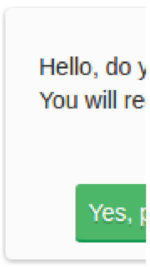
Below are some examples of how you can modify these contents:

Modification examples

Simple text changing example

Expand source

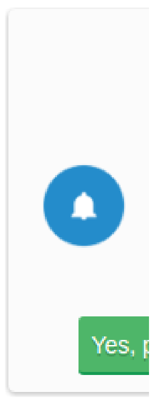
```
"contents": {
 "misc": "Hello, do you want to receive our push notifications
?<br>You will receive nice offers and discounts !",
 "deny": "No ! I prefere to pay full price",
 "accept": "Yes, please !"
}
```



Hello, do y
You will re

Add HTML to add a logo Expand source

```
"contents": {
 "misc": "<div style='text-align:center;padding:0 0 15px'><img
src='https:\/\/websdk.accengage.net\/images\/accengage_logo.png'
width='175'><\/div><div
style='position:relative;width:100%;display:flex'><div
style='width:15%;position:relative;padding:10px 0 0'><img
src='https:\/\/websdk.accengage.net\/images\/alert_content_bell.png'
width='100%' style='max-width:55px;display:block;margin:0
auto'><\/div><div style='width:80%;padding-left:5%'><div
style='color:#0088C3;font-size:16px;padding:0 0
10px;font-weight:bold'>Receive all our promotions !<\/div><div
style='font-size:13px'>Subscribe to our web alerts and don't miss any
offer. You will be noticed as soon as a deal is
online.<\/div><\/div><\/div>",
 "deny": "No ! I prefere to pay full price",
 "accept": "Yes, please Sir !"
}
```



Customize the Popin Alert Theme

Accengage offers preset design templates (or themes) that you can very easily choose from. We also give you the possibility to add your own design and CSS (see Section 3).

To do so, use the content object **"theme"** inside the above JSON Plugin file configuration. Changing the teme will only alter the CSS file that is injected and not the HTML Popin Alert HTML markup.

Below are how the different themes look like:

Theme	Images
modern	
light	
custom (no theme apply)	

Customize the Popin Alert Position

You can easily choose the position of the Popin Alert by modifying the **position property** in the JSON configuration file.

We recommend **“topCenter”** to maximize the volume of opt-in users.

Activate an overlay effect for the Popin Alert

You can decide to add a semi-transparent layer between your page and the Popin Alert. In that case, users won't be able to click on your page elements. Obviously, this will highlight the Popin Alert and will drive maximum attention to it, maximizing the volume of opt-in users!

In order to use this overlay effect, you need to choose the **"modern"** theme.

Customize when to ask for the permission

In order to maximize the number of opt-in users, it is important to ask the user for his permission at the right moment, and in several occasions. In order to customize when to ask the user for his permission, Accengage provides you with several options.

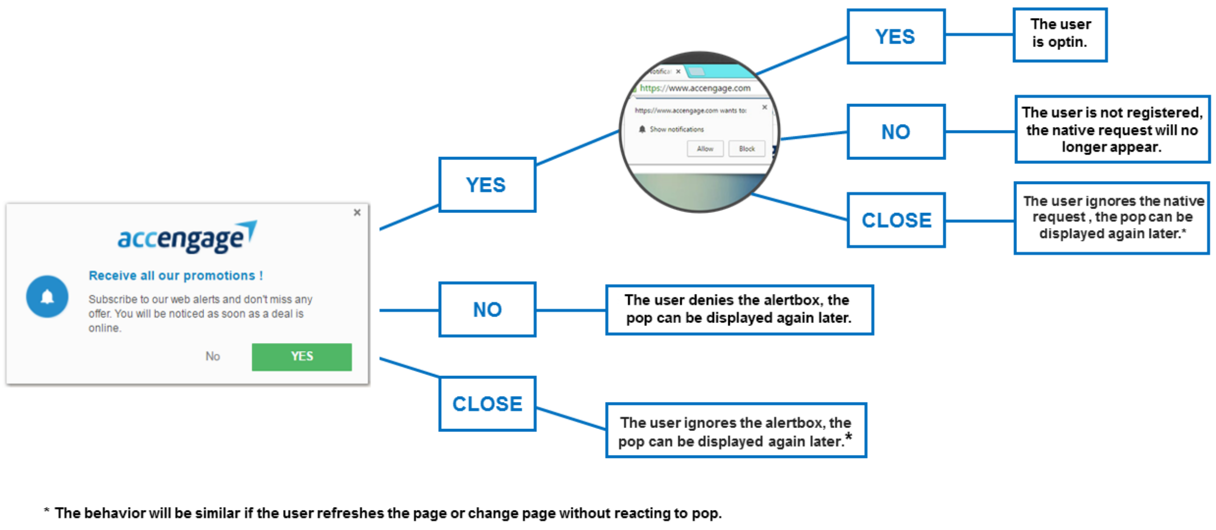
First of all, as seen above, you can determine specifically when to trigger the permission thanks to the method `push:showAlert` (clic k here to see how this command can be used in your pages).

> This can be combined with the automatic display system of the Popin Alert when the homepage is loaded. This help you to ensure maximum visibility, as well as a relevant exposure based on what the user is doing (e.g: on the home page automatically **AND** after finalizing a purchase).

Second of all, Accengage provides you with the **Re-Asking delay options**. Indeed, as long as the user has not answered to the native browser permission, you may **ask them again** for their opt-in, and you can set delays you wish.

Accengage provides 2 Re-Asking delay options in the **JSON Plugin configuration** :

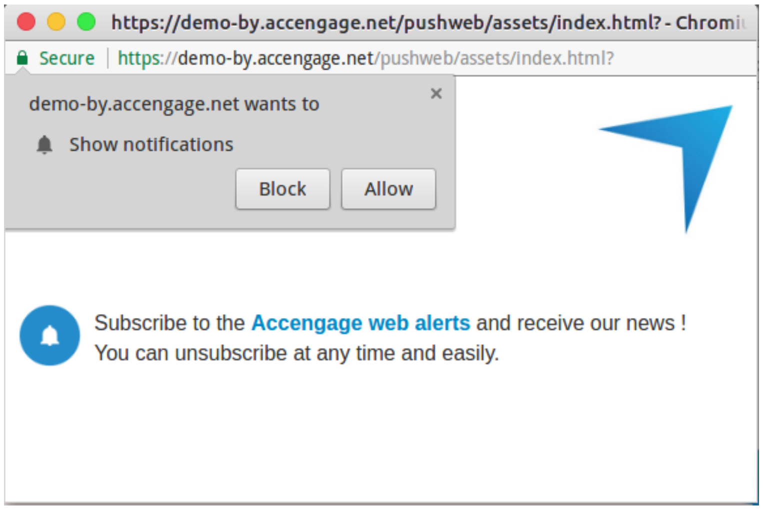
Type	Description
reAskingDelay	Delay, expressed in **hours** (decimals are possible), before showing the HTML alert again, after the user has clicked on the **"close**" button of the Alert, or hasn't interacted with the alert.
reAskingDelay2	Delay, expressed in **days** (decimals are possible), before showing the HTML alert again, after the user has clicked on the **"deny"** button of the Alert.


accengage
Receive all our promotions !
Subscribe to our web alerts and don't miss any offer. You will be noticed as soon as a deal is online.
No
YES
YES
NO
CLOSE
https://www.accengage.com
https://www.accengage.com wants to:
Show notifications
Allow
Block
YES
The user is optin.
NO
The user is not registered, the native request will no longer appear.
CLOSE
The user ignores the native request , the pop can be displayed again later.*
The user denies the alertbox, the pop can be displayed again later.
The user ignores the alertbox, the pop can be displayed again later.*
* The behavior will be similar if the user refreshes the page or change page without reacting to pop.

3.2.3 Basic customization of the Landing Page

As a reminder the Landing Page is a new window which is triggered when the user has expressed its intention to opt-in thanks to the Popin Alert, and in which the native browser permission is displayed.

You can see in the screenshot below that the native notification permission is displayed inside.


https://demo-by.accengage.net/pushweb/assets/index.html? - Chromi
Secure | https://demo-by.accengage.net/pushweb/assets/Index.html?
demo-by.accengage.net wants to
Show notifications
Block
Allow

Subscribe to the **Accengage web alerts** and receive our news !
You can unsubscribe at any time and easily.

It is also interesting to customize its text and design, in order to maximize the volume of opt-in users, as it is the last element which can encourage the users to give their permission.

Typically, it can be interesting to reassure the user on the fact that they can unsubscribe in just 1 click, that they will receive interesting content etc.

Accengage also lets you customize the Landing page easily via the **Plugin configuration**, by modifying the **push.landingOptinOptions** property.

Below you will find the elements that can be edited in the JSON configuration:

Parameter	Type	Default	Description
contents	<object>	*see below*	Lists the contents displayed to the user (see below)
theme	<string>	"light"	Selected theme to customise the **landing page** . Available themes : • "white" • "light" • "dark" • "black"
height	<number>	135	Pixel value defining the height of the popup window
width	<number>	462	Pixel value defining the width of the popup window
withAppIcon	<boolean>	false	a flag defining if you want to display your Application Icon *(The one determined in the section **Getting started > Configure the Accengage Dashboard**)*

If you would like to know all the details about how to set the ‘contents’ object, you can read below the part Useful Commands for more details.

Here are the wordings that can be customized in the landing page, set in the **push.landingOptinOptions.contents** object:

Content field	Description	Default value
ask	Custom <string> value which is displayed immediately on the landing page (for instance to explain the benefits of web push notifications and reassure on the unsubscription)	none

granted	Custom <string> value which is displayed when the user has granted the notification permission	none
denied	Custom <string> value which is displayed when the user has denied the notification permission	none
error	Custom <string> value which is displayed when an error occurs during the opt-in process	• "en" : "An error occured, please try again later" • "fr" : "Une erreur est apparue, veuillez recommencer plus tard"
close	Custom <string> value which is displayed when the page does not close itself	• "en" : "You can now close the page" • "fr" : "Vous pouvez désormais fermer la fenêtre"

3.2.4 Custom Error option

In very rare occasions, when the integration is not properly made or if servers cannot be reached, an error message might be displayed as a Web Push notification.

This message can be customized in the JSON Configuration:

Parameter	Type	Default	Description
contents	<object>	*see below*	a list of selected contents displayed to the user

Content field	Description	Default value
message	notification's message	"An error occurred while displaying a notification"
title	notification's title	"Oups"

If you would like to know all the details about how to set the 'contents' object, you can read below the part Useful Commands for more details.

3.3 Advanced design customization for the pop-in alert

In the previous paragraphs, we've looked at how to customize the default opt-in process in a simple way, with **little HTML knowledge** and for maximum efficiency.
In this section, we'll look at how to go further in customizing the user experience, and adapting the design to your own design.

All the parameters, which you can set inside the JSON Plugin configuration (such as re-asking delays, position of the Alert etc.), can still be set up within the JSON Plugin configuration. The opt-in process will result in the combination of these parameters and the advanced customization done.

You can easily add your own javascript file to override the CSS properties included in the plugin. Please find below two cases where CSS files have been added to modify the esthetics of the Alertbox

Alertbox HTML Structure

Html structure

```
<div class="acc-alert--desktop acc-alert--visible" id="acc-alert">
    <div id="acc-alert-body">
        <a class="acc--closeLink" href="#" id="acc-alert-close"></a>
        <div class="acc--miscContent" id="acc-alert-content">
            MISC OBJECT CONTENT
        </div>
    </div>
    <div id="acc-alert-buttons">
        <a class=
        "acc-alert-button acc-alert-success acc--acceptLink
acc--acceptContent"
        href="#">ACCEPT OBJECT CONTENT</a><a class=
        "acc-alert-button acc-alert-error acc--denyLink acc--denyContent"
        href="#">DENY OBJECT CONTENT</a>
    </div>
</div>
```

Example 1: Modifying pre-existing Light theme.

We saw before that themes are CSS properties added to your page when the plugin is loaded. You can override a few properties with your own CSS files.

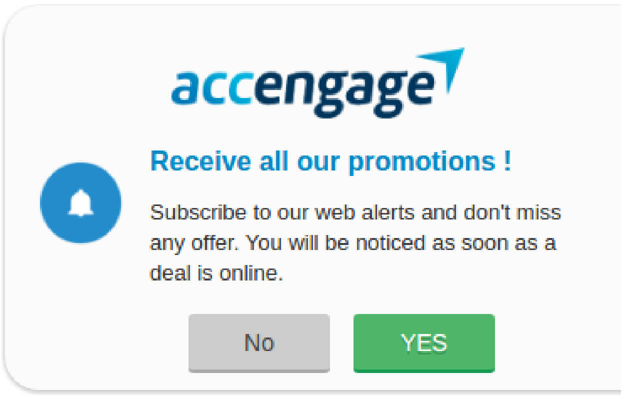
Simple design modification

Css addition

```
/* Switch "deny" and "accept"
buttons' position */
#acc-alert-buttons {
  direction: rtl;
}
#acc-alert-buttons
.acc-alert-button.acc-alert-error {
  margin-right: 10px;
}
#acc-alert-buttons
.acc-alert-button.acc-alert-success
{
  margin-right: 0;
}

/* Delete "close" cross button */
#acc-alert-close {
  display: none;
}

/* Edit component border-radius */
#acc-alert {
  border-radius: 20px;
}
```


Receive all our promotions !

Subscribe to our web alerts and don't miss any offer. You will be noticed as soon as a deal is online.

Example 2: Fresh start from the blank theme

When choosing a theme for the Pop-in within the JSON configuration, you can set it to “custom” (which corresponds to a blank theme).

When you choose the blank theme, the HTML structure of the Alertbox is included to the DOM but NO CSS will be added.

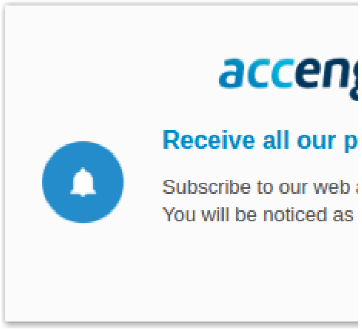
You can then add your own CSS file without having to override the already existing properties.

For example

Css addition › Expand source

```
#acc-alert * {
  box-sizing: border-box;
}

#acc-alert {
  position: fixed;
  width: 100%;
  background: #FBFBFB;
  box-shadow: 1px 1px 5px 1px rgba(0, 0, 0, 0.4);
  overflow: hidden;
```


Receive all our p

Subscribe to our web

You will be noticed as

```
  font-family: Arial, Helvetica, sans-serif;
  z-index: 999;
}

#acc-alert-body {
  position: relative;
  width: 100%;
  z-index: 1;
}

#acc-alert-buttons {
  position: relative;
  width: 100%;
  text-align: right;
}

#acc-alert-close {
  position: absolute;
  top: 6px;
  height: 12px;
  width: 12px;
  opacity: 0.5;
  z-index: 2;
  cursor: pointer;
}

#acc-alert-close:hover {
  opacity: 1;
}

#acc-alert-close:before,
#acc-alert-close:after {
  position: absolute;
  left: 3px;
  content: " ";
  width: 2px;
  background-color: #000;
}

#acc-alert-close:before {
  transform:rotate(45deg);
}

#acc-alert-close:after {
  transform:rotate(-45deg);
}

#acc-alert-content {
  position: relative;
  color: #505050;
}

#acc-alert-buttons {
  direction: rtl;
}

#acc-alert-buttons .acc-alert-button.acc-alert-success
{
  display: inline-block;
  text-align: center;
  text-decoration: none;
  border-radius: 1px;
  background: #50B766;
  color: #fff;
  font-weight: bold;
  margin-left: 35px;
  cursor: pointer;
}

#acc-alert-buttons
.acc-alert-button.acc-alert-success:hover {
  background: #45A85A;
}

#acc-alert-buttons .acc-alert-button.acc-alert-error {
  display: inline-block;
  text-align: center;
  text-decoration: none;
  color: #808080;
  cursor: pointer;
}

#acc-alert-buttons
.acc-alert-button.acc-alert-error:hover {
  color: #505050;
}

#acc-alert.acc-alert--desktop.acc-alert--left {
  right: initial!important;
  left: 10px;
}

#acc-alert.acc-alert--desktop.acc-alert--bottom {
  top: initial!important;
  bottom: 10px;
}

.acc-alert--hidden {
  display: none;
}

#acc-alert.acc-alert--desktop {
  top: 10px;
  right: 10px;
  width: 450px;
  border-radius: 1px;
}

.acc-alert--desktop #acc-alert-body {
  height: auto;
  min-height: 78px;
  padding: 20px 20px 0;
}

.acc-alert--desktop #acc-alert-buttons {
  padding: 10px 20px 15px;
}
```

```
.acc-alert--desktop #acc-alert-close {
  right: 5px;
}

.acc-alert--desktop #acc-alert-close:before,
.acc-alert--desktop #acc-alert-close:after {
  height: 10px;
}

.acc-alert--desktop #acc-alert-buttons
.acc-alert-button.acc-alert-success {
  height: 35px;
  line-height: 35px;
  width: auto;
  padding: 0 8px;
  min-width: 125px;
}

#acc-alert.acc-alert--mobile {
  width: 100%;
  bottom: 0;
  left: 0;
}

.acc-alert--mobile #acc-alert-close {
  right: 8px;
}

.acc-alert--mobile #acc-alert-close:before,
.acc-alert--mobile #acc-alert-close:after {
  height: 20px;
}

.acc-alert--mobile #acc-alert-buttons
.acc-alert-button.acc-alert-success {
  height: 45px;
  line-height: 47px;
  padding: 0;
  width: 155px;
  max-width: 47%;
  min-width: 100px;
}

.acc-alert--mobile #acc-alert-body {
  padding: 25px 5% 0;
}

.acc-alert--mobile #acc-alert-buttons {
  padding: 15px 5% 8px;
}
```

Remarks
You can visualize a nice example of a completely customized Pop-in on our demo page here and you can even download the Stylesheet

Here is an example of a full customization

3.4 Advanced opt-in scenarios

We've seen in the previous sections how to customize the Show Alert which is provided by Accengage by default. However, if you would like to collect more opt-in users, we recommend you to combine and try different scenarios as well as different contexts to ask for the user permission.

For instance, you may ask for the user permission on the home page, but also after a specific action performed (purchase, newsletter registration etc.).

For this reason, Accengage also provides some examples of other types of **'Alert' displays**, such as banners, interstitials, switch buttons etc.

You also have the possibility to create your own template.

Below are some examples of what you can do, feel free to use these examples and adapt them to your website and ideas.

All the parameters, which you can set inside the JSON Plugin configuration (such as reasking delays, position of the Alert etc.), can still be set up within the JSON Plugin configuration. The opt-in process will result in the combination of these parameters and the advanced customization done.

Instertitial

This example is impactful given the attention it drives.

Also, the volumes of opt-in users are maximized as the overlay disables the navigation behind.

You can achieve high numbers using interstitials!

You can visualize it on our demo page and re-use the code if needed.

Illustration of an instertitial

Full width banner

This example is a good compromise between impact and non-intrusiveness.

It looks very similar to the banners which are displayed for cookie acceptancy, so users are accustomed to seeing them on websites.

It has the advantage that you can show it at the top of all the pages of your website during the user navigation. You could also decide to display it from the bottom as the user scrolls down, in which case we recommend you use it in combination with another display seen from the top.

You can visualize it on our demo page and re-use the code if needed.

Opt-in Button

This example can be very useful if you would like to maintain a permanent button on your website to encourage users to give their permission, similar to a “register to our newsletter” button.

The example below shows an animated bell that you can position as an overlay on your website

You can visualize it on our demo page and re-use the code if needed.

Switch Button

This example can be very useful if you would like to maintain a permanent button on your website to encourage users to give their permission. It can also be added during a form registration.

The Accengage website (http://www.accengage.com) shows an example at the top of the screen.

You can visualize an example on our demo page and re-use the code if needed.

Check box in a form

Taking advantage of a form is a good additional way to ask for the user permission.

Note that we recommend not to only rely on this form, as a lot of users do not go through forms. But it’s a good way to catch a high percentage of your more engaged users.

You can visualize an example on our demo page and re-use the code if needed.

The Accengage website (http://www.accengage.com) shows a nice example of that.

You can access the demo page to see an example of the SDK's integration.

3.5 Scenario triggering options

Together with the design, the timing of the Alert display can prove to be a key factor for the opt-in process performance.

Asking the user's permission directly from the Home Page is an important and recommended step, however, asking them in other contexts (e.g.: at the end of a purchase, of a registration, on a deep page etc.) is also important and should be used in combination with a request on the Home Page.

> In order to use the functions described below, you will need to create some commands within the Javascript. To see how to proceed, please read paragraph 3 of Section ‘Web SDK Integration’.

Triggering the Popin Alert when you want

If you want to use the default or customized Show Alert, Accengage gives you the opportunity to choose when to display this it :

- Automatically when a user arrives on the website (set the push plugin "scenario" option to "showAlert", click here for more information)
- At a specific moment that you decide thanks to the method `push:showAlert ` (click here to see how this command can be used in your pages)

Please refer to the API Documentation for complete details

Use the **push:showAlert** command when you want to show the **HTML alert** to the user.

Argument	Required	Type	Description
options	optional	<object>	a new set of parameters that can override the configuration of your application (see the "alert configuration" section to check all the possibilities)
callbacks	optional	<object>	• "onSuccess" : <void> • "onError" : <string> label error

For Instance:

For instance

```
ACC.push(["push:showAlert"]);

/*********************************/

// or, let's say you have a configuration so that every alert are injected
with a specific content
// for this command execution, you can provide other contents (or any other
options !):
var params = {
 theme: "modern",
 contents: {
  misc: "a unique content can be now display!"
 }
};
ACC.push(["push:showAlert", params]);
```

Triggering Landing Page following a custom scenario

In order to launch your own custom scenarios at a desired moment, Accengage has created a method to let you trigger the landing page (where the native web push permission is asked) following your custom Alert scenario.

It's the **push:launchLandingOnClick** command.

Use this command to bind one or more *DOM Anchor Elements* (**</a>** tag) to launch the **optout-to-optin landing page** on click.

Argument	Required	Type	Description
options	yes	<array>	of <string> containing selected anchors to bind. The value must start with "#" or "." (ID or ClassName)

callbacks	optional	<object>	• "onSuccess" : <array> provided elements that have been utterly binded • "onError" <string> label error

For Instance:

For example

```
ACC.push([
 "push:launchLandingOnClick",
 [
  "#myLink",
  ".myOtherLinks"
 ]
]);
```

Before launching your custom opt-in scenario, you may want to know if the user is compliant with Web Push Notification requisites , and/or if the user is already opt-in or not.

To know whether the user is already opt-in to Web Push Notifications, you can use the **core:isCompliantWithPushPlugin** command. It can also tell you why the user is optout,

e.g. if he has denied the native notification permission.

Use this command to know if the user is **compliant with the Push Plugin**

Argument	Required	Type	Description
options	not used		
callbacks	yes	<object>	• "onSuccess" : <void> • "onError" : <void>

For instance

```
ACC.push([
 "core:isCompliantWithPushPlugin",
 null,
 {
  "onSuccess": function() {
   console.log( "this user is compliant with the push plugin !" );
  },
  "onError": function() {
   console.log( "this user is NOT compliant with the push plugin" );
  }
 }
]);
```

You can add some custom listeners to the events which are triggered by the library such as when the landing page is displayed, when the user has granted permission etc.

Please refer to the API Documentation for complete details JS doc.

3.6 Useful commands

3.6.1 How to set up the contents object in the JSON plugin configuration

The contents <object> contains a list of properties, according to the plugin configuration itself.

Hence, here is a possible example of configuration, to manage for instance multi-lingual messages:

```
{
 // In this example, the contents object contains two properties "welcome" and
"goodbye"
 "contents": {

  // You can set the field with a <string>
  "welcome": "Hello ! How are you ?",

  // Or you can set the field with an <object>
  "goodbye": {

   // The key is used for a specific language code
   // So, if the browser language is set to "en", it will display this content
:
   "en": "See you !",

   // Same thing, if the browser language is set to "fr", it will display this
content instead :
   "fr": "Au revoir !"

   // Which content is displayed if the browser is, for example, set to "es" ?
   // -> It will take the first provided language code. In this case, it will
display "See you !"

   // Please, use a lower-cased value for the provided language code.

   }
 }
}
```

Enable Debug Mode to bypass cache settings

Our solution provides a Debug Mode allowing you to keep a stack trace accessible through your browser console.

Also, as explained above, when modifying the JSON plugin configuration options, you need to wait for approximately half an hour to see the changes live in your pages. However, with the Debug mode, you can bypass the application's settings cache and speed things up to see the changes.

It won't destroy the current cache though, other users will keep seeing the cached version.

To use the debug mode, just follow these steps:

- Enable the checkbox on our interface (Settings > Manage Applications > Edit application > Allowing debugging.)
- This will create an "ACCdebugKey". To activate the debug mode, you just have to include it in your page URL as query string parameter
- For example, if your current page is http://myWebsite.com/page.html, just add: http://myWebsite.com/page.html?ACCdebug

Key=XXXX

3.6.2 Enable debug mode to bypass cache settings

Our solution provides a **Debug Mode** allowing you to keep a stack trace accessible through your browser console.

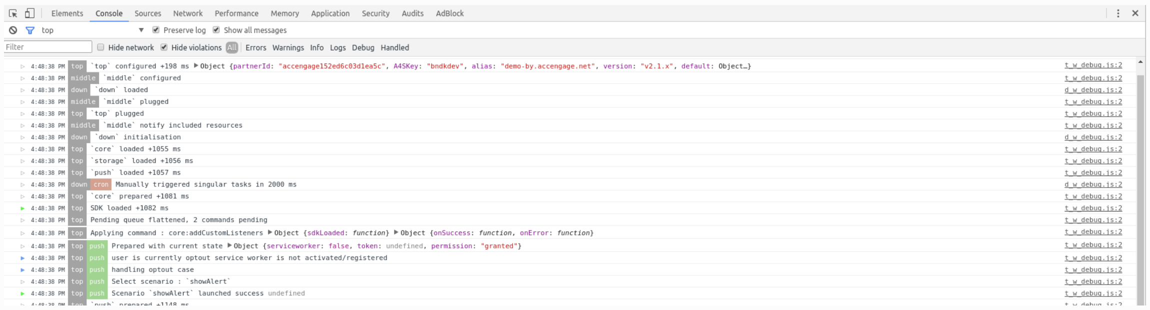
The debug mode can also bypass the application's settings cache. For example, you have made some changes on our interface for your application, and you don't want to wait thirty minutes to see the modifications. Use the debug mode avoid this wait and see the changes (it won't destroy the current cache though, other users will keep seeing the cached version).

To use the debug mode, just follow these steps :

1. Enable the checkbox on our interface > *Settings* > *Manage Applications* > *My application* > ***Allowing debugging***.
2. It will create an "**ACCdebugKey**". To activate the debug mode, you just have to include it in your page URL as query string parameter
3. For example, if your current page is *http://myWebsite.com/page.html*, just add : *http://myWebsite.com/page.html**?ACCdebugKey=XXXX***

On Google Chrome (or Firefox), you can access the console by

- Using the keyboard shortcuts
  - On Windows and Linux: *F12*
  - On Mac: *Cmd + Option + I*
- Then go to the 'Console' panel



```
4:48:38 PM top `top` configured +198 ms ▶Object {partnerId: "accengage152ed6c03d1ea5c", A4SKey: "bndkdev", alias: "demo-by.accengage.net", version: "v2.1.x", default: Object…}   t_w_debug.js:2
4:48:38 PM middle `middle` configured                                              t_w_debug.js:2
4:48:38 PM down `down` loaded                                                      d_w_debug.js:2
4:48:38 PM middle `middle` plugged                                                 t_w_debug.js:2
4:48:38 PM top `top` plugged                                                       t_w_debug.js:2
4:48:38 PM middle `middle` notify included resources                               t_w_debug.js:2
4:48:38 PM down `down` initialisation                                              d_w_debug.js:2
4:48:38 PM top `core` loaded +1055 ms                                              t_w_debug.js:2
4:48:38 PM top `storage` loaded +1056 ms                                           t_w_debug.js:2
4:48:38 PM top `push` loaded +1057 ms                                              t_w_debug.js:2
4:48:38 PM down cron Manually triggered singular tasks in 2000 ms                  d_w_debug.js:2
4:48:38 PM top `core` prepared +1081 ms                                            t_w_debug.js:2
4:48:38 PM top SDK loaded +1082 ms                                                 t_w_debug.js:2
4:48:38 PM top Pending queue flattened, 2 commands pending                         t_w_debug.js:2
4:48:38 PM top Applying command : core:addCustomListeners ▶Object {sdkLoaded: function} ▶Object {onSuccess: function, onError: function}   t_w_debug.js:2
4:48:38 PM top push Prepared with current state ▶Object {serviceworker: false, token: undefined, permission: "granted"}   t_w_debug.js:2
4:48:38 PM top push user is currently optout service worker is not activated/registered   t_w_debug.js:2
4:48:38 PM top push handling optout case                                           t_w_debug.js:2
4:48:38 PM top push Select scenario : `showAlert`                                  t_w_debug.js:2
4:48:38 PM top push Scenario `showAlert` launched success undefined                t_w_debug.js:2
4:48:38 PM top `push` prepared +1148 ms                                            t_w_debug.js:2
```

Section 4 - User profile and behaviour tracking

Tagging relevant information regarding your user profiles and their behaviours is crucial to send personalized and valuable messages.

Although the Accengage SDK collects by default certain useful information (such as first opt-in date, last visit date, number of visits, browser type…), it also offers a number of advanced tagging and tracking features for developers, that will allow you to target notifications, trigger messages at the right time and analyze your campaign performance.

It is important that you check with the product or marketing team what type of actions they would like to perform, so that you can determine which information needs to be tagged. Most of the time they will provide you with a 'tagging plan' which lists all of these. However, a strategy could be to first implement a minimal version of Web Push Notifications in order to speed up the process and learn faster, then implement the tagging plan at a later stage for more relevant messages.

Note that the Accengage Web SDK is compatible with Tag Management Systems, and Section 5 (Tag Management Systems) explains how to proceed.

In the following section, we will look at the various options that you can enjoy to collect and use the right information:
- enrich user profile, thanks to custom device information (or User Attributes), which can be used for targeting and message personalization
- tag user location, for geolocation targeting
- tag user behaviours thanks to events, which offer many benefits (targeting, and measurement).

In order to use the functions described below, you will need to create some commands within the Javascript. To see how to proceed, please read paragraph 3 of Section 'Web SDK Integration'.

4.1 User Attributes

4.1.1 Enrich User profile with custom device information (or User Attributes)

In order to send well-targeted and personalized messages, Accengage lets you enrich your user profiles with additional relevant information, called custom device information (or User Attributes).

These are important for campaign targeting, as well as message personalization.

For instance, if you tag and update the first name of a user, you will be able to personalize a message with “Hello ${firstname}”.

By implementing the library in your site, you will automatically gather certain information about your web visitors: their browser type, browser version, language, country code, time zone, domain where the user became opt-in first, first opt-in date, the number of visits and more.

The Accengage library lets you also collect additional custom device information, which can be attached to the user profile through custom database fields.

Traditional usage of custom device information for instance include first name, account ID (to match with external systems), total number of purchases, last purchased product, last purchase date, favorite product… They can be numeric values (set values, or incremental), strings, dates…

Unlike events (see below), the values of these custom device info (or user attributes) are unique per user. This means the user's associated fields are single-value type fields and each update will erase currently stored values except when it's an incrementation (or decrementation).

Make sure that your custom_field is already created in the database (targeting > database scheme > add a field).
Otherwise, the data will be ignored!
Check our user guide to have an overview of the database associated to your app and how to create a new field.

In order to enrich the user profile with additional user attributes (previously created in the database), you can use the core:updateDeviceInfo command:

core:updateDeviceInfo

Use this command to update the current user's information. Some fields, that we use internally, are filtered.

Argument	Required	Type	Description
options	yes	<object>	a list of fields <string> with value <string|number>
callbacks	optional	<object>	• "onSuccess" : <void> • "onError" : <string> label error

For instance

```
ACC.push([
 "core:updateDeviceInfo",
 {
  "field1": "value1",
  "field2": "value2"
 }
]);
```

Update date-type data:

To update date-type fields, please format your date object in yyyy-MM-dd HH:mm:ss zzz string. The library provides a helper.

Update counter-type data:

To update counter-type data, you can increment (or decrement) your field by prefixing the value with the positive sign + (or negative -).

4.1.2 User geolocation

It is possible to know the user location and to target users based on their last known position.

> This user geolocation targeting is not to be confused with real-time geofencing which is possible with mobile app push notifications, but not yet with web push notifications, even on mobile.

If you wish to target users based on their previous known location, you may use the following solutions below:

- Requesting the native browser permission for user location (this permission needs to be given by the end user, each time you would like to know their location)
- Pass the user location, and present date as the following user attributes (see above how to pass user attributes):

> **Info**
> The value should respect a format which is available below

Params

```
{
  lat: <Number>, // 48.870083
  long:<Number>, // 2.334254
  timestamp: <Number>
}
```

4.2 Behaviour tracking with events

Events are very useful to track user behaviors. One user may have multiple events, and all event occurrences are reported to the Accengage servers.

They can be used for:

- Message targeting (subject to your site being integrated in the new Accengage platform)
- Site or campaign performance measurment.

The library offers specific and useful methods to track standardized events such as ‘add to cart’ events, ‘purchases’ or ‘leads’. You can also track custom events for other user behaviors.

For instance, with events, you could target users who have made 3 purchases over the last 30 days and who have visited more than 10 pages.

You will need to ask Accengage teams to activate events, and then you will be autonomous to create them in the Accengage dashboard (in settings > settings > add an event), so as to select them as targeting criteria.

Events are composed of a type (a numeric value which categorizes the type of event – see below), and parameters, which bring more information and can be targeted on (provided you are on the new platform).

To track events, you will need to interact with the Track plugin of the Web SDK.

Each command related to the Track Plugin uses an option <object> and callbacks <object> (containing <function>).

For instance

```
var options   = {
 "id": 1001
};
var callbacks = {
 "onSuccess": mySuccessFunc(),
 "onError": myErrorFunc( <string> )
};

ACC.push(["track:event", options, callbacks]);
```

The **options** <object> differ from each other. Each command has its own data to provided.

The **callbacks** <object> will tell if your command has been validated and will be treated. In this case the "onSuccess" provided function will be applied. Otherwise the "onError" function will provide you a <string> expliciting the error.

For the sake of clarity, we will only tell how to set your **options** <object> for each command (as the callbacks object is generic and straightforward to understand).

4.2.1 Track leads

track:lead

‘Lead’ events are useful to track behaviours such as sign-up, registrations etc. The Accengage advanced statistic dashboard will also measure these leads.

Options	Required	Type	Description
lead	required	<string>	
value	required	<string>	you can also provide "now()" as the value, it will automatically replace the string as a formatted date

The library provides a simple method to track leads. You might want to track particular special events and actions like an authentification. The type of a lead is 10, but you don’t need to specify it.

This is how to use the track:lead command:

For instance

```
ACC.push(["track:lead", {
 "lead": "foo",
 "value": "bar"
}]);
```

4.2.2 Track 'Add to Cart' events

track:cart

‘Add to cart’ events are used to track and tag the fact that certain items were added to a basket. This is typically useful if you would like to set up “abandoned cart” campaigns, for which you will need to tag ‘add to cart’ events, as well as ‘purchases’.
The library provides a simple method to track items added to a cart. The type of all cart events is 30 but you don’t need to specify it.

This is how to use the track:cart command:

Options	Required	Type	Description
item	required	<object>	The item <object> will respect this format: `{ id: <string> price: <numeric> currency: <string> quantity: <numeric> [default: 1] label: <string> [optional] category: <string> [optional] }`
id	optional	<string>	The id of your **cart**, it could match the id provided in a **purchase** track event

For instance

```
var myItem = {
 "id": "c22d2ea6-9184-11e7-abc4-cec278b6b50a",
 "price": 42,
 "currency": "EUR"
}

ACC.push(["track:cart", {
 "item": myItem
}]);
```

4.2.3 Track purchases

track:purchase

'Purchase' events are used to tag your users purchases within your site or target messages based on them, but also to measure the revenues generated thanks to your notification campaigns.

Indeed, Accengage provides a statistic dashboard which measures this information (Statistics > advanced statistics).

The library provides a simple method to track purchases. The type of all purchase events is 50, but you don't need to specify it. It is recommended to specify all the purchased items.

This is how to use the track:purchase command:

Options	Required	Type	Description
id	required	<string>	
price	required	<numeric>	please provide a valid number (NaN will not be a valid value)
currency	required	<string>	

items	optional	<array>	containing **item** <object> respecting this format:
			`{`
			`id: <string>`
			`price: <numeric>`
			`currency: <string>`
			`quantity: <numeric> [default: 1]`
			`label: <string> [optional]`
			`category: <stringg> [optional]`
			`}`

For instance

```
ACC.push(["track:purchase", {
 "id": "123456789",
 "price": 420,
 "currency": "EUR"
}]);
```

4.2.4 Track custom events

track:event

Apart from the special events seen above, you can tag and track any other event that will be useful for message targeting, thanks to custom events, such as ‘made a search’, ‘invited a friend’ etc.
For all of these custom events, use the track:event command.

> Event types below 1000 are reserved for the library internal usage. You can use custom event types starting from 1001, and you must specify them

Options	Required	Type	Description
id	required	<numeric>	Please provide a valid number (NaN will not be a valid value) Please note that you can create **custom event** in the *Advanced Settings* section of our interface
details	optional	<object>	*key* - *value* set of data, used to details your **event**

For instance

```
ACC.push(["track:event", {"id": 1001}]);
```

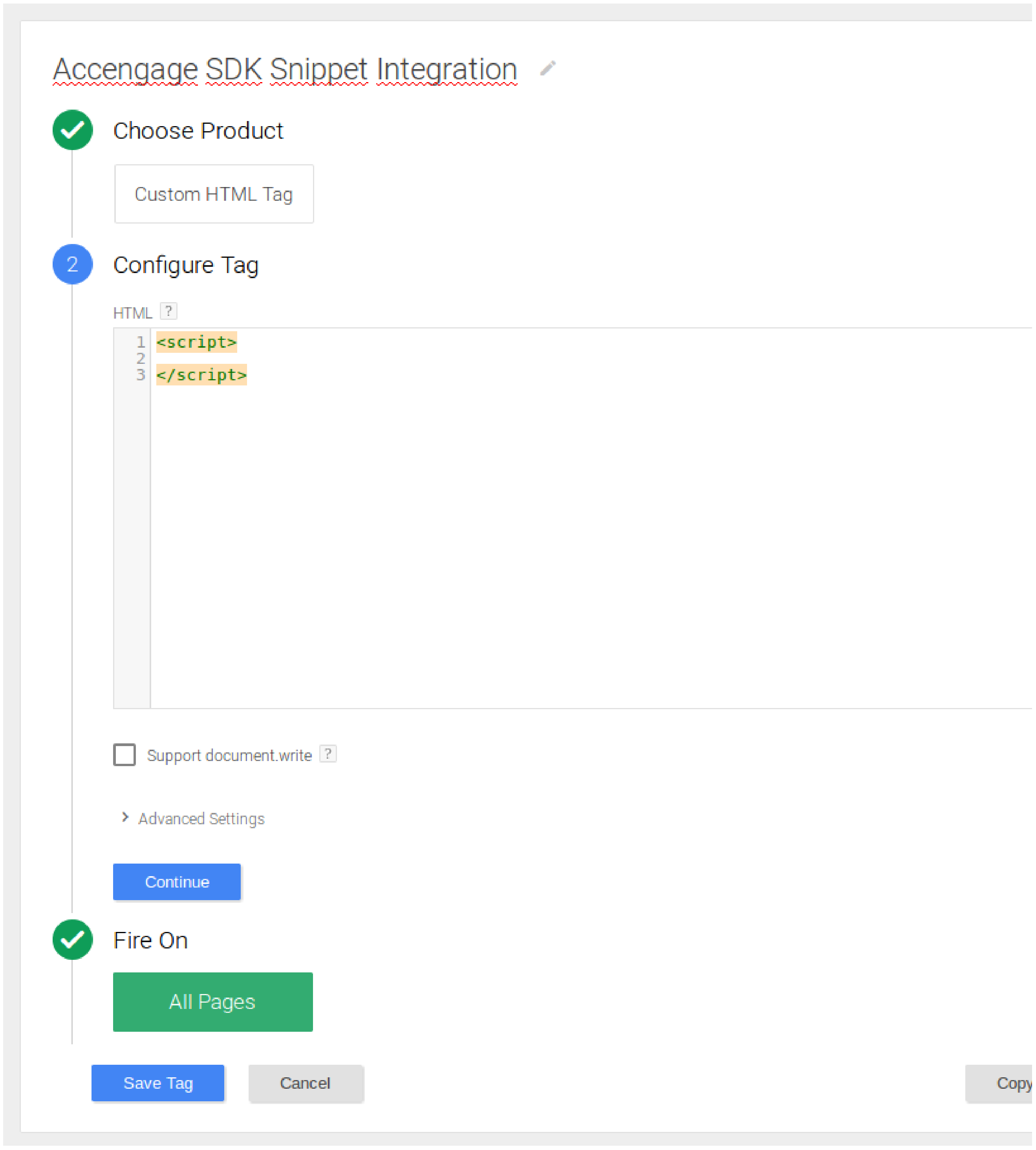
Section 5 - Tag management systems

Integrating the SDK in your site

> **The aim: use the SDK in your pages**

No change in the content of your pages is required. You will only need to create one tag that implements the snippet.

First, you will have to click on the " Tags " menu item, then create a new Tag, and select Custom HTML Tag:


Accengage SDK Snippet Integration

Choose Product

Custom HTML Tag

Configure Tag

HTML

```
<script>

</script>
```

" Accengage SDK Snippet Integration " Custom HTML Tag

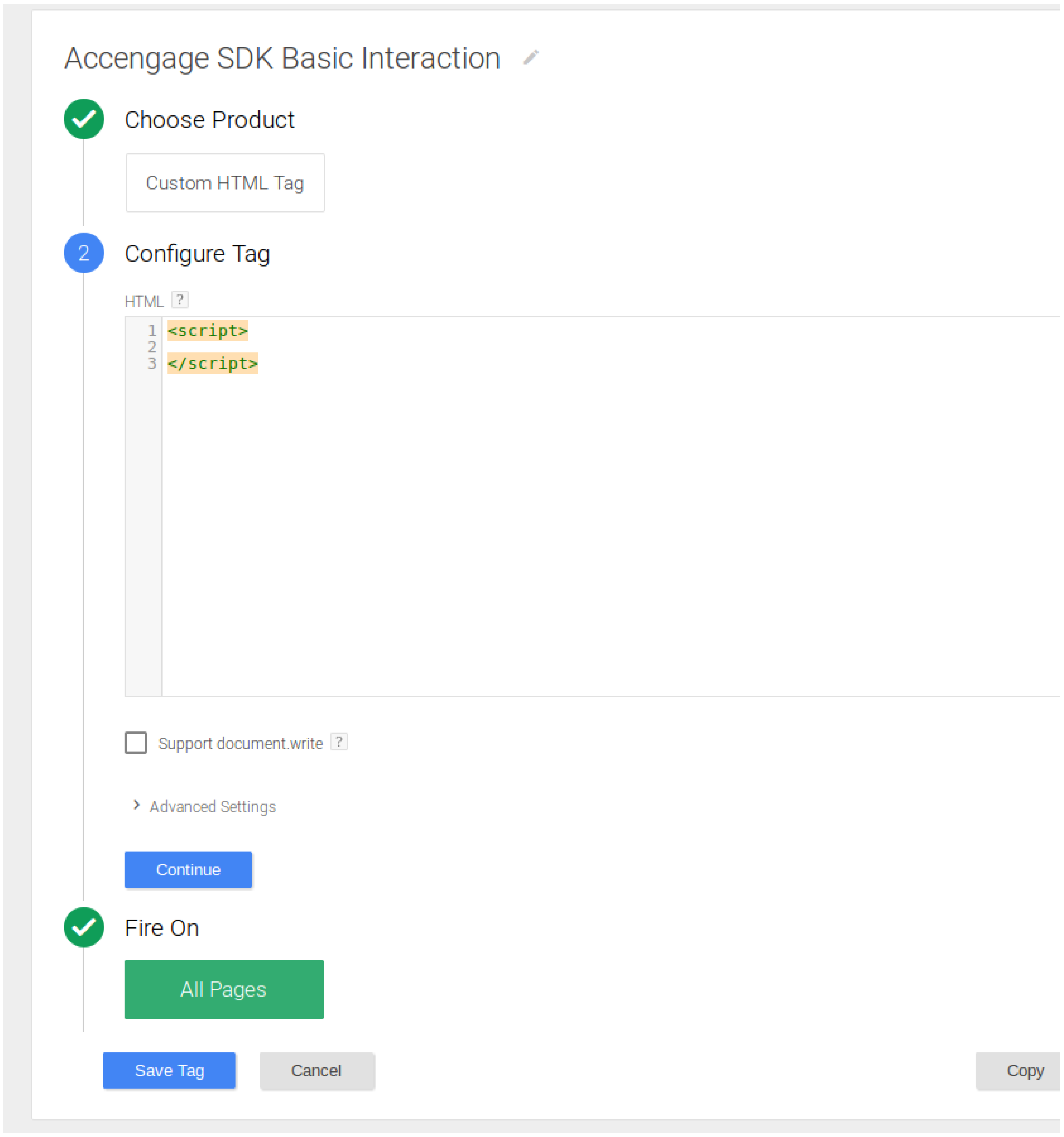
In the " Configure Tag ", you have to paste the **JavaScript snippet code** (*see description here*) wrapped within HTML </script> tags. Then choose the pages where the **Tag** will be launched.

The Accengage Web SDK is now integrating in your website. You can also add a **priority** to be sure that the snippet will be available as soon as possible. To do so, go to the *Configure Tag* section of the form, then *Advanced Settings > Tag Firing Priority*.

Create basic interactions

The aim: use the SDK and create some commands (e.g. an update device info), in your pages.
No change in the content of your pages is required. You must create two tags: one containing the snippet, the other one containing the commands you want.
.

While the previous **snippet integration Tag** has been created, we will now create another Custom HTML Tag:



Accengage SDK Basic Interaction

Choose Product: Custom HTML Tag

Configure Tag

HTML

```
<script>

</script>
```

“ Accengage SDK Basic Interaction ” Custom HTML Tag

In this example, the newly Tag will be fired in every pages, you can also configure it in the **Fire On** section of the tag.
Within it, you will then use the library by creating "commands". You can write a command to show the optinisation alert, create an update device info, etc. Please visit our API reference to check the available commands.

Section 6 - Troubleshooting

6.1 Optin method through my own domain

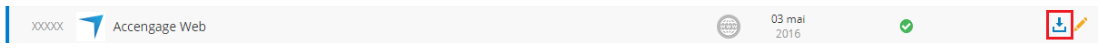
What are the prerequisites to use the new opt-in process ?

First of all, before integrating this opt-in process, be sure that you have correctly integrated our SDK by following the steps described in our documentation.

Already done it ? Let's start !

1. You have to download a zip file from our back-office (Settings > Manage application >) and upload the following files to your website root direction :
    a. manifest.json
    b. acc_sw.js
    c. acc_ww.js

    Example :

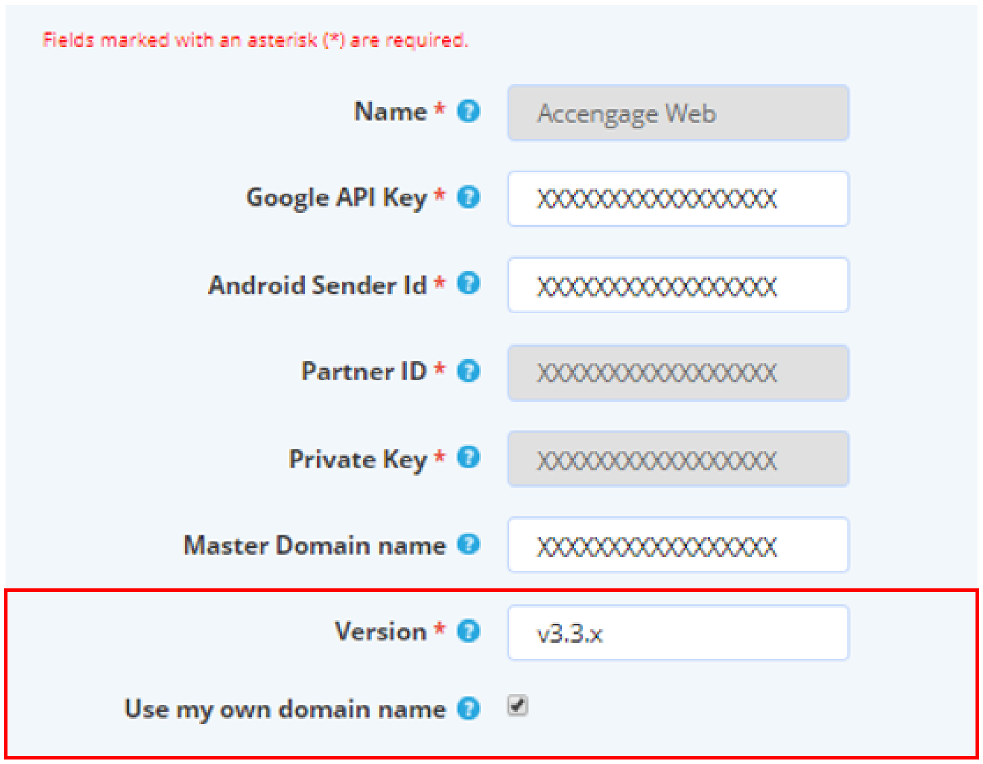
    

2. You have to reference "manifest.json" in all your website pages by including the following tag in the <head> of the pages :
    a. <link rel="manifest" href"/manifest.json">

    > **Important**
    > Be careful, if you already have a file named "manifest.json" in your website, you'll have to merge the lines from "manifest.json" into your existing file

3. If you're migrating from an older version of our SDK and once the steps described above are complete, activate the new opt-in process in the Accengage Dashboard :
    a. Log in into the Accengage Dashboard
    b. Go to "Settings > Manage Application" and click on "Edit"
    c. Then, change your SDK version for the new one : 3.3.x or higher
    d. Enable the checkbox called "Use my own domaiin name"
    e. Click on save

    Example :



Fields marked with an asterisk (*) are required.

Name *	Accengage Web
Google API Key *	XXXXXXXXXXXXXXXXXX
Android Sender Id *	XXXXXXXXXXXXXXXXXX
Partner ID *	XXXXXXXXXXXXXXXXXX
Private Key *	XXXXXXXXXXXXXXXXXX
Master Domain name	XXXXXXXXXXXXXXXXXX
Version *	v3.3.x
Use my own domain name	✓

What is the purpose of the files available in the Zip file ?

Browsers need those files to keep the push web working properly.

When using the historical optin method (through the Accengage's domain or a custom one), the files are hosted in the Accengage's databse.

Now, as the optin process is proceed directly on your domain those files must be hosted in your own website. If they are not, the optin process will not work.

Is the new opt-in process available for all users ?

No, this new opt-in process is available only for users navigating in "HTTPs".

This means that your website must be available in HTTPs if you want to use this opt-in method.

If the user navigates in your website through "HTTP", then he'll have the same opt-in process as before. In otherwise he'll have the Accengage landing page before the browser's native permission request.

What happens if my website is avaialble both in HTTP and HTTPs ?

User navigating through HTTP will have the same process as before, meaning by the Accengage landing page.

Users navigating through HTTPs will have the new opt-in process.

Can I update my SDK version without using the new opt-in process ?

As said in our first answer, to activate the new opt-in method you have to enable it in the Accengage dashboard.

This said, if you don't want to use this new process, keep the checkbox disabled. However we highly recommend you to use it as it can accelerate a lot your users engagement.

Note that this new opt-in process is by default disabled.

Can I have duplicated users between ones optin in HTTP and ones optin in HTTPs for the same website ?

Our SDK can recognize if a user is optin HTTP or HTTPs.

This means that a user opt-in in HTTP and navigating on your website through HTTPs will not be ask to opt-in again.

Same thing for the ones opt-in in HTTPs and navigating through your website in HTTP.

However and as before, if the user deletes his navigation data, the SDK will not be able to recognize him meaning that your website will possibly ask him to opt-in once again even if he has replied before the navigation data deletion.

Can I use buttons, specific pages, interstitials etc... to display the native the native pop to the user ?

Yes, you can ask the opt-in to the user when he clicks on a specific button, consults a specific page or when an interstitials appears by using a new SDK method called : `requestBrowserPermission().`

This method can be called the same way as the other SDK methods :

`ACC.push(['push:requestBrowserPermission']).`

Depending on which opt-in process you're using (the new one or the Accengage landing page), your website will propose the right process to the user :

- If the user is navigating through HTTPs and if the new process is implemented in your website and activated in the Accengage dashboard, the SDK will display the browser's native permission
- If the user is navigating through HTTP, the SDK will use the Accengage landing page which will call the browser's native permission

As a user of the Accengage dashboard, do I still have one database per application ?

Yes, this new opt-in process does not affect the databases and you'll still have one database per application in the Accengage dashboard.

6.2 The user is not yet opt-in

Unable to render {include} The included page could not be found.

6.3 The user is already opt-in

6.2.1 Default browser setting changes

What happens if the user does not allow third-party cookies ?

The SDK will not be able to run for the specific user. The latter will not be registered in the user database, nor will he receive push notifications.

Chrome (desktop)

Chrome (mobile)

6.2.2 Concerning the Native Notification Permission

What happens if the user blocks all notifications in Chrome's settings ?

In this case, the user won't be able to subscribe to push notification. The use of the SDK method **push:isCompliant** will apply an *on Error* callback, telling that the “**User has denied notification permission**”.

Chrome (desktop)

Chrome (mobile)

On the optout-to-optin page, what happens if the user closes the browser's native permission ?

The user is opt-out and not registered in the Accengage database. But, the native permission can be re-displayed.

On the optout-to-optin page, what happens if the user blocks the browser's native permission ?

The user is opt-out and not registered in the Accengage database. His response to the permission will be visible in Chrome’s settings with the “block” status. He must change his permission to allow or delete it in order to re-display the native permission.

6.2.3 Concerning the Accengage HTML Alert

What happens if the user refuses the Accengage HTML Alert ?

The user is opt-out and not registered in the Accengage database. But the device ID is nevertheless registered in the browser's database.

What happends if the user closes the Accengage HTML Alert ?

The user is opt-out. If he clicks on the close box or does not interact with the alert, then, at the next opening of the page, the option “reAskingDelay” will be questioned in order to identify if the alert needs to be re-displayed or not.

Section 7 - JS Doc

7.1 Core plugin

core:updateDeviceInfo

Use this command whenever you want to update the current user's information. Some fields, that we use internally, are filtered.

Argument	Required	Type	Description
options	yes	<object>	a list of fields <string> with value <string|number>
callbacks	optional	<object>	• "onSuccess" : <void> • "onError" : <string> label error

For example

```
ACC.push([
 "core:updateDeviceInfo",
 {
  "field1": "value1",
  "field2": "value2"
 }
]);
```

core:getDeviceID

Use this command to retrieve the current user **device ID**.

Argument	Required	Type	Description
options	not used		
callbacks	yes	<object>	• "onSuccess" : <string> device ID • "onError" : <string> label error while retrieving the user device ID

For example

```
ACC.push([
 "core:getDeviceID",
 null,
 {
  "onSuccess": function(deviceID) {
   console.log( "user device ID : " + deviceID );
  }
 }
]);
```

core:isCompliantWithPushPlugin

Use this command to know if the user is **compliant with the Push Plugin**

Argument	Required	Type	Description
options	not used		
callbacks	yes	<object>	• "onSuccess" : <void> • "onError" : <void>

For example

```
ACC.push([
 "core:isCompliantWithPushPlugin",
 null,
 {
  "onSuccess": function() {
   console.log( "this user is compliant with the push plugin !" );
  },
  "onError": function() {
   console.log( "this user is NOT compliant with the push plugin" );
  }
 }
]);
```

core:addCustomListeners

Use this command to register your custom listeners to an **event** triggered by the library

Argument	Required	Type	Description
options	yes	<object>	a list of event names <string> with listener <function>. Available **Core plugin** events : • "sdkLoaded" : <object> this event is retroactive, i.e. if you use this command after the library has been loaded, your provided listener will be triggered. It is also a one time event. It returned a object containing the set configuration of your application
callbacks	optional	<object>	• "onSuccess" : <array> returning the events that have been listened • "onError" : <string> label expliciting the reason why no events have been listened

For example

```
ACC.push([
 "core:addCustomListeners",
 {
  "sdkLoaded": function(config) {
   console.log( "SDK is loaded !", config );
  }
 }
]);
```

7.2 Push plugin

push:showAlert

Use this command when you want to show the **HTML alert** to the user.

Argument	Required	Type	Description
options	optional	<object>	a new set of parameters that can override the configuration of your application (see the "alert configuration" section to check all the possibilities)

callbacks	optional	<object>	• "onSuccess" : <void> • "onError" : <string> label error

Example

```
ACC.push(["push:showAlert"]);

/*********************************/

// or, let's say you have a configuration so that every alert are injected
with a specific content
// for this command execution, you can provide other contents (or any other
options !):
var params = {
 theme: "modern",
 contents: {
  misc: "a unique content can be now display!"
 }
};
ACC.push(["push:showAlert", params]);
```

push:launchLandingOnClick

Use this command to bind one or more *DOM Anchor Elements* (**</a>** tag) to launch the **optout-to-optin landing page** on click.

Argument	Required	Type	Description
options	yes	<array>	of <string> containing selected anchors to bind. The value must start with "#" or "." (ID or ClassName)
callbacks	optional	<object>	• "onSuccess" : <array> provided elements that have been utterly binded • "onError" <string> label error

Example

```
ACC.push([
 "push:launchLandingOnClick",
 [
  "#myLink",
  ".myOtherLinks"
 ]
]);
```

push:requestBrowserPermission

Use this command to bind one or more *DOM Anchor Elements* (**</a>** tag) to display the native push browser's permission (**from SDK 3.3.x**)

Argument	Required	Type	Description
options	yes	<array>	of <string> containing selected anchors to bind. The value must start with "#" or "." (ID or ClassName)
callbacks	optional	<object>	• "onSuccess" : <array> provided elements that have been utterly binded • "onError" <string> label error

Example

```
ACC.push([
 "push:requestBrowserPermission",
 [
  "#myLink",
  ".myOtherLinks"
 ]
]);
```

If you want to display the browser native permission once the user clicks on a specific link of your website instead of immediatly once the page is loaded, you must set a generic css identifier (ID) to your anchor <a>: accengageRequestBrowserPermission

The SDK will then check this ID when the page is loading:

- If there is no ID, browser native permission will be triggered immediatly once the page is loaded.
- if there is an ID, browser native permission will not be triggered automatically but when the user will perform a click on the link you set up.

Example

```
ACC.push([
 "push:requestBrowserPermission",
 [
  "#accengageRequestBrowserPermission",
 ]
]);
```

push:isOptin

Use this command to know if the user is optin to push. It can tell *why* the user is optout, e.g. if he has denied the native notification permission.

Argument	Required	Type	Description
options	not used		
callbacks	optional	<object>	• "onSuccess" : <object: status <string>, details <object>> user is optin • "onError" <object: status<string>, details <object>> user is optout

Example

```
ACC.push([
 "push:isOptin",
 null,
 {
  "onSuccess": function(res) {
   console.log( "this user is optin since " + res.details.firstOptinDate );
  },
  "onError": function(err) {
   var isHardOptout = (err.details.notificationPermission === 'denied');
   if (isHardOptout) {
    console.log( "this user is hard optout: notification permission has been
denied, we cannot ask him to be optin" );
   } else {
    console.log( "this user is soft optout: we can ask him to be optin" );
   }
  }
 }
]);
```

push:addCustomListeners

Use this command to register your custom listeners to an **event** triggered by the library

Argument	Required	Type	Description
options	yes	<object>	a list of event names <string> with listener <function>. Available **Push plugin** events : • "plugin:started" : <object> this event is retroactive, i.e. if you use this command after the library has been loaded, your provided listener will be triggered. It is also a one time event. It returned a object containing the set configuration of the plugin • "landingFeedback:optin" : <void> triggered when the **optout-to-optin landing page** returns that the user has granted permission and is now optin • "landingFeedback:softOptout" : <void> same thing, but triggered when the user has not granted the notification permission. In this case, the user can still be optin. • "landingFeedback:hardOptout" : <void> same this, but triggered when the user has denied the permission, hence he's definitely optout
callbacks	optional	<object>	• "onSuccess" : <array> returning the events that have been listened • "onError" : <string> label expliciting the reason why no events have been listened

Example

```
ACC.push([
 "push:addCustomListeners",
 {
  "plugin:started": function() {
   console.log( "the push plugin is now started" );
  },
  "landingFeedback:optin": function() {
   console.log( "this user is now optin" );
  },
  "landingFeedback:softOptout": function() {
   console.log( "this user is still optout, but can grant permission later" );
  },
  "landingFeedback:hardOptout": function() {
   console.log( "this user has denied permission and he is definitely optout"
);
  }
 }
]);
```

7.3 Track plugin

Each command related to the **Track Plugin** uses an **option** <object> and **callbacks** <object> (containing <function>).

For example

```
var options   = {
 "id": 1001
};
var callbacks = {
 "onSuccess": mySuccessFunc(),
 "onError": myErrorFunc( <string> )
};

ACC.push(["track:event", options, callbacks]);
```

The **options** <object> differ from each other. Each command has its own data to provided.

The **callbacks** <object> will tell if your command has been validated and will be treated. In this case the "onSuccess" provided function will be applied. Otherwise the "onError" function will provide you a <string> expliciting the error.

For the sake of clarity, we will only tell how to set your **options** <object> for each command (as the callbacks object is generic and straightforward to understand).

track:event

Options	Required	Type	Description

id	required	<numeric>	Please provide a valid number (NaN will not be a valid value) Please note that you can create **custom event** in the *Advanced Settings* section of our interface
details	optional	<object>	*key - value* set of data, used to details your **event**

For example

```
ACC.push(["track:event", {"id": 1001}]);
```

track:lead

Options	Required	Type	Description
lead	required	<string>	
value	required	<string>	you can also provide "now()" as the value, it will automatically replace the string as a formatted date

For example

```
ACC.push(["track:lead", {
 "lead": "foo",
 "value": "bar"
}]);
```

track:cart

Options	Required	Type	Description

item	required	<object>	The item <object> will respect this format:
			`{`
			` id: <string>`
			` price: <numeric>`
			` currency: <string>`
			` quantity: <numeric> [default: 1]`
			` label: <string> [optional]`
			` category: <stringg> [optional]`
			`}`

id	optional	<string>	The id of your **cart**, it could match the id provided in a **purchase** track event

For example

```
var myItem = {
 "id": "c22d2ea6-9184-11e7-abc4-cec278b6b50a",
 "price": 42,
 "currency": "EUR"
}

ACC.push(["track:cart", {
 "item": myItem
}]);
```

track:purchase

Options	Required	Type	Description
id	required	<string>	
price	required	<numeric>	please provide a valid number (NaN will not be a valid value)
currency	required	<string>	
items	optional	<array>	containing **item** <object> respecting this format: `{ id: <string> price: <numeric> currency: <string> quantity: <numeric> [default: 1] label: <string> [optional] category: <stringg> [optional] }`

For example

```
ACC.push(["track:purchase", {
 "id": "123456789",
 "price": 420,
 "currency": "EUR"
}]);
```

7.4 Collect plugin

collect:setOptinData

Use this command whenever you want to set the **optin data** value.

Argument	Required	Type	Description

options	yes	<object>	an object with the **optinData** parameter • "optinData" : <Boolean>
callbacks	optional	<object>	• "onSuccess" : <void> • "onError" : <string> label error

For example

```
ACC.push([
 "collect:setOptinData",
 {
  "optinData": true
 },
    {
        "onSuccess": () => console.log('Optin data has been set'),
        "onError": err => console.log(`An error happened ${err}`)
    }
]);
```

collect:getOptinData

Use this command to retrieve the **optin data** value

Argument	Required	Type	Description
options	not used		
callbacks	optional	<object>	• "onSuccess" : <object> an object containing the optinData value • "onError" : <string> label error

For example

```
ACC.push([
 "collect:getOptinData",
    {
        "onSuccess": res => console.log('Optin data value
${JSON.stringify(res)}'),
        "onError": err => console.log(`An error happened ${err}`)
    }
]);
```

7.5 Facebook Messenger Plugin

fbMessenger:generateButton

Use this command to generate a Facebook Messenger button. You can create buttons as many as you want.

Argument	Required	Type	Description
containerId	yes	<string>	the ID of your DOM Element that will contain the generated button

color	optional	<string>	a constant defining the color of the generated button. Available colors : • "blue" *default if not provided* • "white"
size	optional	<string>	a constant defining the size of the generated button. Available sizes : • "standard" • "large" *default if not provided* • "xlarge"
callbacks	optional	<object>	• "onSuccess": <void> • "onError": <string> label expliciting the reason why no button has been generated

For example

```
ACC.push([
 "fbMessenger:generateButton",
 {
  "containerId": "#myDivElement"
 }
]);
```