



# Mobile Messages for Campaign

---

PRODUCT MANUAL | August 2023

# 1 Foreword

## Copyright

The contents of this manual cover material copyrighted by Marigold. Marigold reserves all intellectual property rights on the manual, which should be treated as confidential information as defined under the agreed upon software licence/lease terms and conditions.

The use and distribution of this manual is strictly limited to authorised users of the Marigold Interactive Marketing Software (hereafter the “Software”) and can only be used for the purpose of using the Software under the agreed upon software licence/lease terms and conditions. Upon termination of the right to use the Software, this manual and any copies made must either be returned to Marigold or be destroyed, at the latest two weeks after the right to use the Software has ended.

With the exception of the first sentence of the previous paragraph, no part of this manual may be reprinted or reproduced or distributed or utilised in any form or by any electronic, mechanical or other means, not known or hereafter invented, included photocopying and recording, or in any information storage or retrieval or distribution system, without the prior permission in writing from Marigold.

Marigold will not be responsible or liable for any accidental or inevitable damage that may result from unauthorised access or modifications.

User is aware that this manual may contain errors or inaccuracies and that it may be revised without advance notice. This manual is updated frequently.

Marigold welcomes any recommendations or suggestions regarding the manual, as it helps to continuously improve the quality of our products and manuals.

## 2 Table of Contents

<b>1</b>	<b>Foreword</b>	<b>2</b>
<b>2</b>	<b>Table of Contents</b>	<b>3</b>
<b>3</b>	<b>Introduction</b>	<b>5</b>
3.1	What can you do with Mobile?	5
3.1.1	Push notifications	5
3.1.2	In app messages	8
3.1.3	In-app content	8
3.2	Types of content available for mobile	9
<b>4</b>	<b>How does Mobile Push work?</b>	<b>11</b>
<b>5</b>	<b>Details on how messages are sent and data is retrieved</b>	<b>12</b>
5.1	Push notifications	12
5.2	In-app messages	12
5.3	In-app Content	13
<b>6</b>	<b>Push Notification functionality</b>	<b>15</b>
6.1	Push banners	16
<b>7</b>	<b>What is needed to send mobile messages</b>	<b>17</b>
7.1	Mobile app	17
7.2	Dedicated Mobile interface	17
7.3	Mobile API	17
7.4	Dedicated lists in Campaign	18
7.5	What events are available	19
<b>8</b>	<b>Installation and Set-up</b>	<b>23</b>
8.1	What do you need to know before starting	23
8.2	Create or obtain a mobile app	23
8.3	Create mobile app content containers	24
8.4	Register the mobile app (optional)	25
8.5	Include an opt-out event (optional)	25
8.6	Create a dedicated Mobile interface in Campaign	25
<b>9</b>	<b>Configuring the Push component in a journey</b>	<b>28</b>

9.1	Push Personalization	33
9.2	Links in Push notifications	34
9.3	Advanced Push properties	36
<b>10</b>	<b>Events generated by the Mobile push component</b>	<b>38</b>
<b>11</b>	<b>Using Mobile push component in a journey</b>	<b>40</b>
11.1	Push notification for a limited audience	40
11.2	In-app message	43
11.3	In app content	46
<b>12</b>	<b>Mobile push reporting</b>	<b>48</b>
<b>13</b>	<b>Q&amp;A</b>	<b>49</b>
<b>14</b>	<b>Addendum: The IOS Certificat</b>	<b>51</b>

## 3 Introduction

### 3.1 WHAT CAN YOU DO WITH MOBILE?

With Mobile you can send mobile messages to a mobile device which have your app installed. These messages are sent from Campaign journeys. You can target all contacts that have your app installed.

Interactions on these messages are measured and are used to take specific action such as sending emails. The app can track numerous events and return the data to Campaign where it can be used for filtering, segmentation or steer a journey.

#### What type of messages can be sent ?

The Mobile solution allows to send following types of messages

- ⇒ Push messages
- ⇒ In-app messages
- ⇒ In-app content

#### 3.1.1 PUSH NOTIFICATIONS

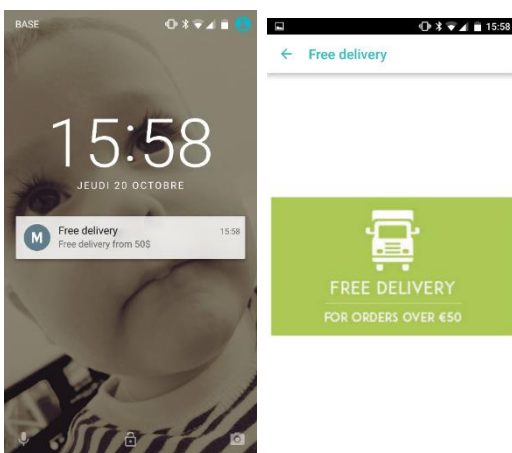
**Push notifications** can be used to keep your customers informed with relevant information or important updates, even when your app is running in the background or is inactive. Push notifications can ensure that relevant content is delivered on time, which brings value for a user and convertible actions for you.

Push messages usually appear on the device lock screen. A notice pops up when there is a new message and the message is available in the message center. From there it can be clicked and opened in the actual app. This must be permitted on the device. So nothing needs to be provided in the app by the app developer. Permission is granted in the device settings, so it depends on the OS, iOS or Android. And the contact can choose to activate the message center.

The user must have opted in to receive push messages. The opt-in is stored in Campaign.

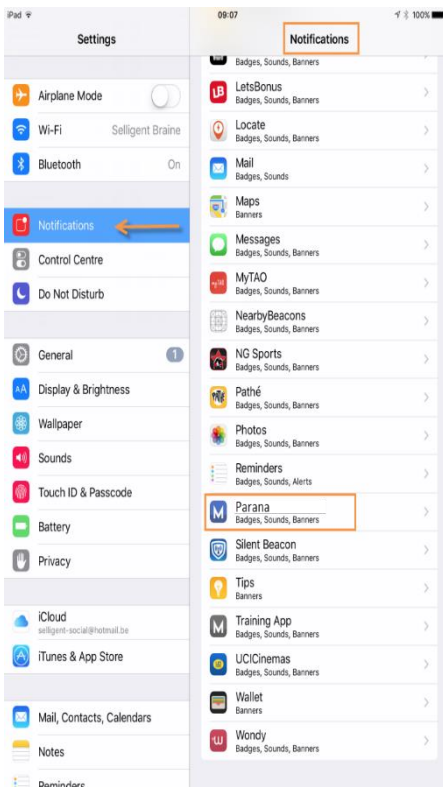
When Push messaging is configured, you can define the message **banners** and the **actual messages** in the Push component in Campaign journeys. The mobile API will make sure these banners are sent and displayed on the device.

Example: On the left, you can see the banner: an alert there is a new message. The user is not in the app at this moment and this banner pops up on the lock screen. When you click it, the app is opened and the actual message is displayed.



### 3.1.1.1 ACTIVATE PUSH NOTIFICATION ON IOS

From the Settings of the device, go to Notifications and select the installed app (eg. Parana)

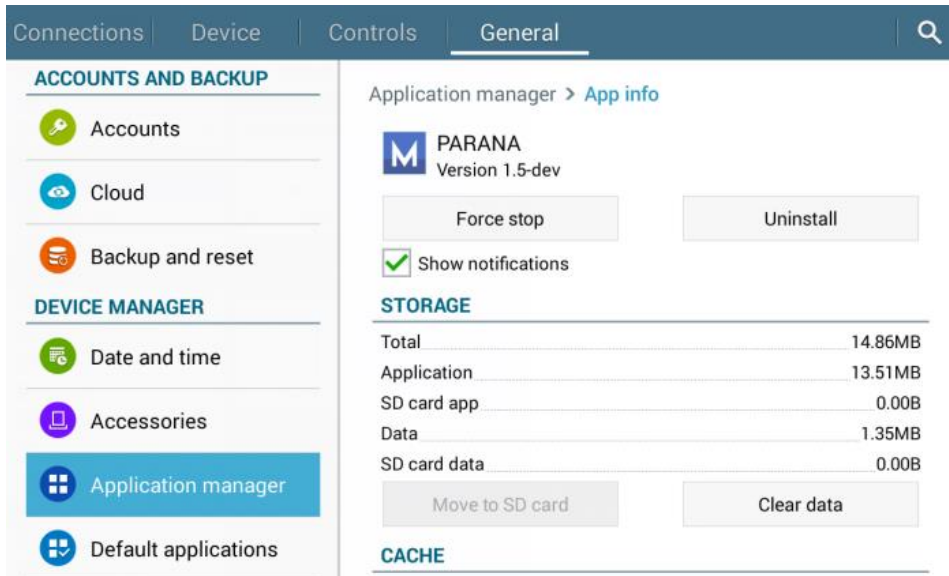


In the settings for the Parana app activate push messages and indicate if these should arrive in the message center.



### 3.1.1.2 ACTIVATE PUSH NOTIFICATION ON ANDROID

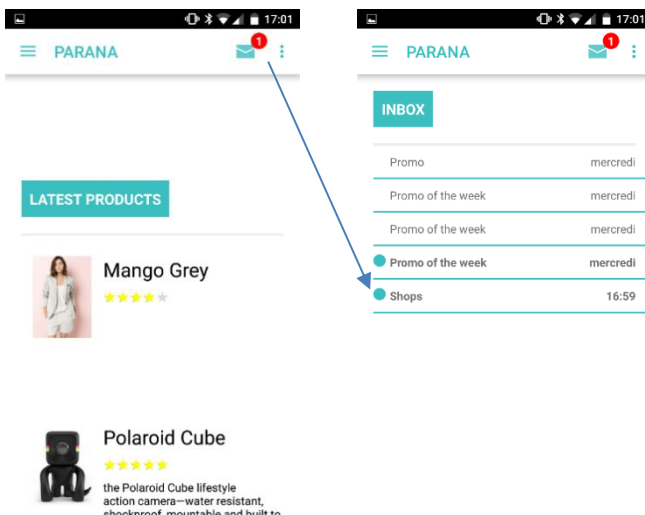
Go to the general tab/ Application Mgr. Select your app and activate the Notifications.



### 3.1.2 IN APP MESSAGES

In addition, **in-app messages** are sent to users that are currently using the app. If the contact is not in the app the messages are not received and can even expire. When the expiration date was reached and the message was not sent to the app it is deleted. Unlike push messages, no permissions need to be granted on the device itself. Contacts are opted in for these in-app messages automatically. The app developer can create an opt-in/out functionality in the app, which is also stored in Campaign.

Example: On the left the contact is in the app and sees a new message. When he clicks the message icon, he gets an overview of all messages and can then open the message itself.

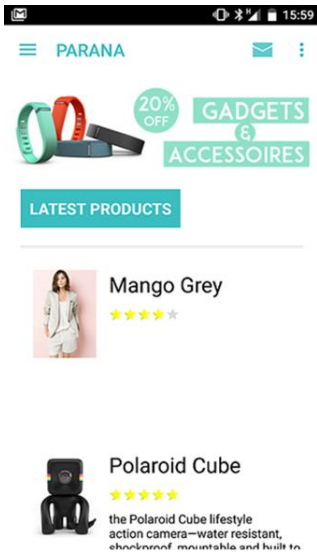


### 3.1.3 IN-APP CONTENT

Personalized content that is displayed in the app itself in dedicated locations in the app. This can be for instance a banner, or some other html content.



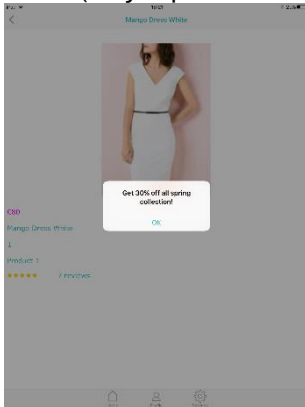
Example: Here the contact is in the app. In the banner of the app an image 'Gadgets and accessories' is displayed, that is explicitly added as in-app content at that location in the app. That content will expire after some time



### 3.2 TYPES OF CONTENT AVAILABLE FOR MOBILE

The following types of message content can be sent.

- Alerts (only in push notifications)



- HTML



- Website URLs

- Images



- Maps with markers (only in push notifications)



The following links can be used in a message:

- Make a phone call directly to the defined number
- Send an SMS: opens the default sms application to send a sms to the defined number
- Open mail client: opens the default email application to send an email to the defined email address (you@company.com)
- Open the browser: opens the default browser with a defined url
- Open another app: opens a defined application such as Facebook, Twitter, etc (fb//)
- Open a method in the app
- Open the store: opens the store at a defined location(iTunes or Market) to download a defined application. In IOS this button does not have a parameter linked to it. For Android you need to enter a parameter that corresponds to the namespace for the app (eg. For Twitter would be com.twitter.android). The app developer needs to know how the app is called in the Market.
- Close notification button: Closes the notification message and stores this click event in the Events table
- Responses with keyboard/files: closes the notification and shows the keyboard and text area. When text is entered, a Send button is available. The entered response is send via the web service and stored in the Events table in Campaign.
- Response with image: there are two links in the app, one to take a picture and the other one to select a file. Interactions on these links are not tracked.
- Passbook

Technical Note: The feedback on sending of push notifications is stored in the Interfaces table. If the message was sent correctly, if an error returns etc, is stored and can be used in journeys to retarget. The statuses stored are in progress, succeeded, failed.

## 4 How does Mobile Push work?

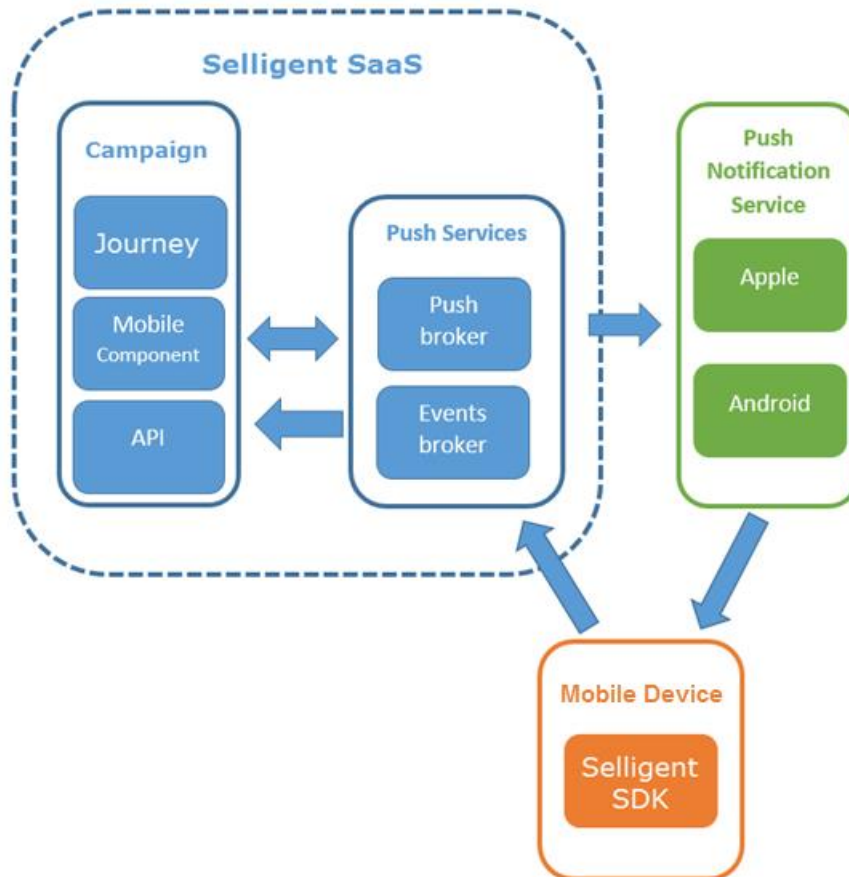
The following is the process for mobile push:

- ⇒ The contact **installs** the mobile app.
- ⇒ When the contact **launches** the app, a dedicated event is automatically called in the SDK, and device-related data is sent (device token, device ID, platform, etc.) and stored in a dedicated **Devices** list.
- ⇒ When the contact **registers** in the app, a record is created in a linked audience list. Also, a check is made to verify if the device already exists in the **Devices** table. If it does, nothing happens besides an update of the LASTMODIFIED\_DT field.
- ⇒ When the contact **registers** in the app, another event is called and if the contact does not exist in the Devices table, a record is created in the **Audience** list storing at least the email of the contact and creating a link with the **Devices** list. If a contact with that email is detected, the record is updated. Note: registration is not mandatory, but allows to identify the device and link it to a contact.
- ⇒ When the contact **logs in** to the application, the linked audience list is updated with any additional information such as email, language, gender, etc. In addition, the contact's status as logged in/out and registered/unregistered is also stored in the Devices table. The contact is now a known contact.
- ⇒ If multiple contacts use the same device, when a contact logs on, the device is linked to the contact's record. When a second contact launches the app, and logs in on the same device, the device is then linked to the second contact's record.
- ⇒ After the user has installed the app and exists in Campaign's Devices table, a dedicated Push component can be used in a journey to send the contact push notifications as well as in-app messages.
- ⇒ When the Push notification message is received, opened, or clicked, this information can be used in a journey to send other messages to the contact or take other actions.
- ⇒ In addition, it is possible to use custom events with custom parameters and store and use this information in Campaign in the **Events table**.

## 5 Details on how messages are sent and data is retrieved

### 5.1 PUSH NOTIFICATIONS

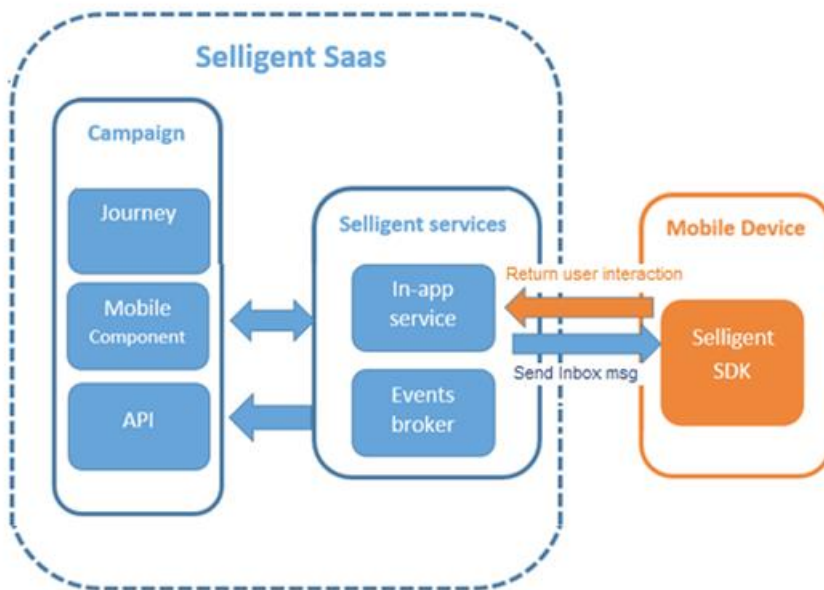
The way push notifications are sent is slightly different than for in-app messages. **When a journey with a Push notification** is launched, the messages are pushed to the Services (Push broker) and, from there, sent to Android or iOS services, which then send them to the devices. This happens in real-time. The customer interaction with the message on the mobile device is tracked by the SDK, which sends the information back to the services (Events broker).



### 5.2 IN-APP MESSAGES

In-app messages are sent to the services where they arrive in a queue waiting to be retrieved. These messages are not pushed to devices. The SDK retrieves In-app messages at predefined intervals (not in real-time). The interval is configurable within the app itself.

Note: It is also possible to define the maximum number of messages kept on the server. Notify your Services team who will modify that for you, if required.



When in-app messages are retrieved by the end user, the app developer decides how they are handled. They can be opened directly or added to the user’s inbox.

Push and In-app messages have a **mandatory expiration date**. When the date has been reached, if the message has not been retrieved, it is deleted.

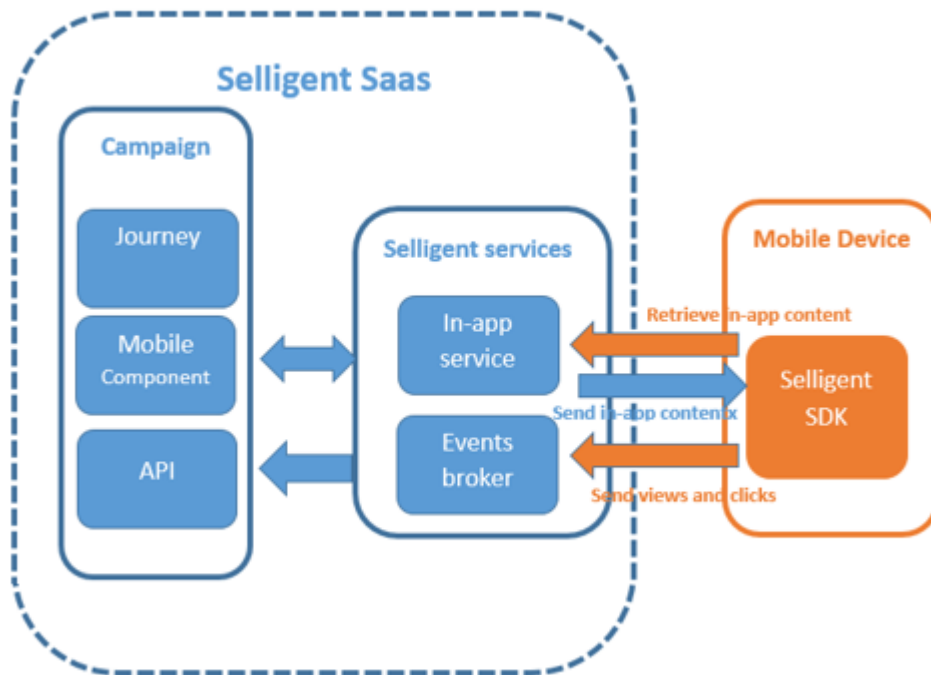
When no expiration date is chosen, it’s set to ‘today + 2 months’ by default.

Expressions can also be used in the expiration date field (eg.: `~(DATEADD('dd',7,GETDATE()))~`)

Note: When the same app is installed on different devices, an opt-out on one device will not result in an automatic opt-out on the other devices. If this is the behavior you want, it must be handled in the journey. Resetting the device results in losing the device ID.

## 5.3 IN-APP CONTENT

In-app content is defined as personalized content that is displayed in the app itself such as banners, prize notifications, or other HTML content.



A single app can have multiple containers that display in-app content. Each container can contain an unlimited number of in-app contents of type HTML. For images and URLs there can only be one content displayed in a container.

Note: If there is no content available to display in the container, the container can be hidden. It is the app developer who decides what happens when there is no content (Details in the technical documentation)

Containers and content are attributed a **category**. This way only content of category “A” will be displayed in a container of category “A.” (The category of a container is defined by the developer of the app. Refer to the technical documentation for more information).

Note: Only one type of content can be displayed in a container. Either HTML, images or a URL is displayed, not a combination of these. If this rule is not respected there will be no support when problems arise. The app developer needs to communicate the category and type of content that is expected and the journey designer needs to take this into account.

A container is also characterized by the **number of in-app contents** it can contain. This applies to HTML content only. If there is only one allowed, only the last one retrieved is displayed. If multiple content is allowed, these are displayed one after the other but always with the last one retrieved at the top.

In-app content can also contain links, but only a maximum of two is recommended, or else it would take up too much space (iOS recommendation).

Note that in-app content navigation is not allowed. Links in the in-app content can not be used to navigate to the in-app content itself.

In-app content can be displayed during one session only or over multiple sessions until new content is available. So for one-session content, as long as the user does not close the app but simply navigates pages in the app, the same content will be displayed.

Note: it is possible to send a push banner to inform the user new content is available. To do this you need a second push component to send the message. It is not automatically done when creating the in-app content.

The configuration of in-app content is defined in the Push component as is the case for push messages and in-app notifications.

Some examples: If you would have an app with multiple containers, each one with its proper in-app content, you need to create a journey with multiple push components, create the in-app content for each of these components, assign a category to the content and to the same for the containers in the app. The right in-app content will be displayed in the right container. In case you have an app with a container in which multiple HTML content should be displayed, you need again to create a journey with multiple push components, define the HTML in-app content in each of these components, assign a category to this content which corresponds to the category of the container.

Technical note:

For in-app content, following tags are supported in the HTML:

- \* br
- \* p
- \* div
- \* strong
- \* b
- \* em
- \* cite
- \* dfn
- \* i
- \* big
- \* small
- \* font
- \* blockquote
- \* tt
- \* a
- \* u
- \* sup
- \* sub
- \* h1 to h6
- \* img (limited to an url used as a reference to a resource present in the app and the code to retrieve this resource must be implemented in an ImageGetter object. )
- \* font but only the color and face attribute

Other tags can be supported via the implementation of a TagHandler object. Styling is not supported at all

## 6 Push Notification functionality

The following is an overview of what is supported in Push notifications.

- **Background notification is supported for iOS:** The app user can de-activate push notifications when the app is running in the background. This setting is defined in the system settings. By default, background mode is activated. This mode is required for the use of in-app messages.

Examples of background mode are continuing to play audio when the app is closed, continuing to receive location updates, etc.

- **Multiple app support:** A single Push component can be used by multiple apps. A customer with multiple brands can have a proper app per brand and use push notifications for each one of these brands. allows linking the different apps to one Push component. The customer has the choice between using the same lists for all brands or using distinct lists per brand on one and the same environment.

To be able to support multiple apps, the app ID will be stored in the Devices table. This way we can have a separate entry per app and per device in the devices table.

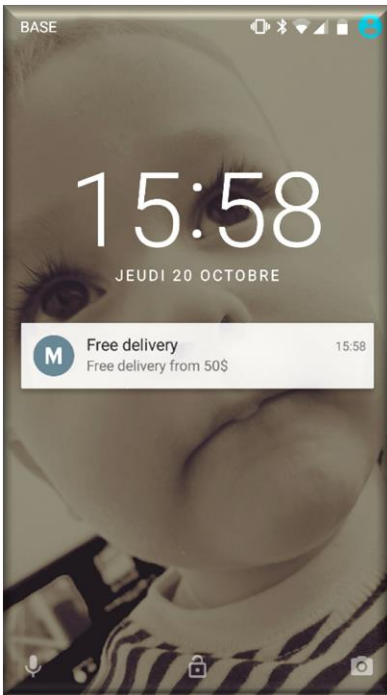
- **Queuing:** Because several apps can use the same service, queuing is used to maximize performance. Multiple threads are used and two queuing systems are employed: Azure or SQL.

Note: The number of push notifications that currently can be sent is around 3000 per minute. In the next release we target a minimum of 20000 per minute.

## 6.1 PUSH BANNERS

After the Push connection is configured and fields are selected, you can define messages and the message banners in the Push component in your Campaign journey. Using the Mobile API, you can make sure these banners are sent and displayed on the device.

**Example:** On the mobile device screen shown below, the left side shows the push notification labeled “**Free delivery**”. Even though the user is not currently logged into the app, this banner pops up on their lock screen. When the user taps the message, the app is opened (as shown on the right) and the actual message is displayed.





## 7 What is needed to send mobile messages

### 7.1 MOBILE APP

We need a **mobile app**. The customer either already has an app or needs to develop one. In any case, the SDK needs to be included in the app. Marigold provides an SDK for IOS and for Android.

What is an **SDK** ?

A software development kit (SDK) is a set of software development tools to create applications for a certain software platform or computer system. To enrich applications with advanced functionalities, advertisements, push notifications and more, most app developers implement specific software development kits.

The SDK contains a series of standard methods/events that you can use to send information from the app to the Marigold platform and store the data in Campaign lists. There are standard methods/ events with standard fields but the customer could extend this with custom events and custom fields that are not by default included. (Marigold provides detailed technical documentation on how to include the SDK in the app)

An example of an event is a *Login* in the app. This event needs to be called if you want to track whenever a user logs in into the application. One of the possible SDK fields available for the *Login* event is the date and time of login.

It is up to the app developer to call the right methods at the right location in the app and pass on the right parameters to the platform. (An interaction between the Campaign developer and the app developer is required to know what events and data needs to be included in the app.)

Beside the methods and fields that can be called in the app, the app will also have to be configured to receive **in-app content**. Therefore, **containers** need to be created in the app, at the location where the content should go. (Containers are in fact specific locations in the app. These locations are given a specific name and are identified by the term 'Container'. The name of the container must be used when defining the in-app content in the Mobile module). This also needs to be discussed between campaign developers and the app dev. (how these containers will be called and where they should be placed)

Next, a **registration** form can be provided by the app developer. It is not mandatory but if you want to be able to link a device to an actual contact, the device user should register himself. When the corresponding event is called in the app, the data is sent to the platform and a link between the device and a contact in the audience list can be made. For this, the email address of the contact can be used or the user id or even an external id. (every contact can have multiple devices linked to him.)

Lastly, the app developer also needs to know if the contact **can opt out for in-app messages**. If this is the case, the app developer should include this option in the app.

### 7.2 DEDICATED MOBILE INTERFACE

In addition, to be able to create a journey with a push component, an **interface** is required.

An interface allows interaction between the platform and external software. There are 2 types: file-based and plug-in based interfaces. For Mobile a plug-in interface is used. These use a custom DLL **script** called a "plug-in" to manage interactions with the external software. This Mobile interface uses the **Mobile plugin (dll)**, a specific plug-in created by Marigold that predefines information and events that can be used in Campaign journeys, related to the SDK used to develop the (external) app.

For instance, has the message been sent? Is an in-app message treated? When these events are configured in the interface they can be used in the journey.

### 7.3 MOBILE API

The Mobile API is used to communicate between Campaign and the app. This API is installed on the Campaign environment by Marigold employee.

## 7.4 DEDICATED LISTS IN CAMPAIGN

To use push messages in a journey, you must start by creating several lists in Campaign.

Campaign uses a series of tables for mobile push to store information on the devices used, the contact and the events triggered by the contacts. The customer can decide on the table names to use and where these should be stored in the Campaign database.

- **Audience** list – Stores known contacts. In this example this list is named “Parana Users”. Usually the Audience list already exists in Campaign as it is also used for other communications. The audience list can contain whatever fields are required to store information on the contact.
- **Devices** list – Stores mobile device data and contains one record for each device. This list contains all device data (*deviceid*, *devicetoken*, *platform*, etc.) and information on specific user interaction events such as login, logout, register and unregister.

For Mobile Push, the Devices list is the master contact list that will be used in a journey. The reason for this is that mobile messages are sent to a specific device, not to a contact. Contacts can have multiple devices, but messages are always intended for one device. When a contact first launches the app, this list is automatically updated. Afterwards, the list is updated whenever changes occur.

The following is a description of some of the fields in the **Devices** table:

- **Optout:** This field is set to 1 if Pushoptout and IAMoptout are both set to 1. The moment either push messages or in-app messages are re-activated this value will be set to 0 again.
- **Platform:** Indicates the device operating system: value 2= Android or 1 = iOS.
- **Device\_UID:** Unique ID of the device.
- **Device\_token:** Technical field based on the device-id and the application ID.
- **Pushoptout:** Field is either 0 (opted in) or 1 (opted out). When a contact installs the app, they accept or decline notifications. If they accept, this field is set to 0. If they decline, this field is set to 1. If the user later blocks the notifications, the next time the app is launched, this field is set to 1.
- **IAMoptout:** Opt-out value for in-app messages: 0 = opted in; 1 = opted out.
- **App\_ID:** The ID of the app. This allows storing multiple apps in the same Devices list.

The above list of fields is updated automatically by the event Setinfo, which is called when the app is launched. These fields are by default there but there are more standard fields that are used by the Setinfo event. They are however not matched by default. These additional standard fields can be added to the user list or to the devices list or their profile extensions.

Here is an overview of other Standard fields:

- SEF\_DEVIDE\_NEW\_ID
- SEF\_DEVICE\_TYPE
- SEF\_COUNTRY
- SEF\_SDK\_VERSION
- SEF\_SYSTEM\_VERSION
- SEF\_TIMEZONE
- SEF\_APPLICATION\_KEY
- SEF\_IDENTITY

- **Events** list – a data list containing the information for the events activated and stored. Only the numeric value of the event is stored. The events by default stored in this table need to be activated explicitly when configuring the app in the Mobile Module. The fields by default matched in this list are:

SEF\_TTYPE: The type of event, expressed as a numeric value

SEF\_LIST\_DEVICE\_ID

SEF\_IS\_IAM: is this an in-app message

SEPUSH\_ID: the id of the push message that triggered the event

SEF\_CREATED\_DT: the date of the event

Following type of events are stored

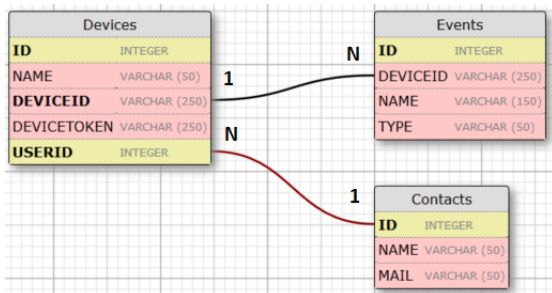
3	PushReceived
4	PushOpened
5	ClickButton
8	UserCustomEvent
10	ClickButtonCancel
12	MediaEvent

- **Profile extension** – Created by Marigold for the User and the Devices lists for custom fields. These profile extensions keep the master lists clean, making them faster to access. The profile extension is created automatically when the setup is done in the Mobile module. It will have the name of the Devices list + EXTMOBILE.

We recommend storing all non-standard matched fields in these profile extensions. These can either be standard SDK fields that are not matched by default or custom-defined fields. An example of a non-standard field could be address fields used in the registration of the contact. These are not in the SDK by default, but can be added if required.

The **Devices** list has relationships with the other lists as follows:

- **Many-on-1 (N:1)** to the Audience list. A contact can have multiple devices.
- **1-on-Many (1:N)** to the Events list. A device has multiple events (buttons pushed, app logins...)
- **One-to-One (1:1)** to the profile extension for custom fields, which is also linked to the Audience list.



## 7.5 WHAT EVENTS ARE AVAILABLE

To be able to define what you need, you need to know what is by default available as an event in the Mobile SDK.

An overview of all standard events:

- **PushReceived**: the message was received on the device
- **PushOpened**: the message has been opened
- **ClickButton**: a click on a button in the message
- **UserCustom**: a custom defined event. This can be anything that you need and is not provided by default
- **ClickButtonCancel**: the Cancel button in a message is clicked
- **Mediaevent**: refers to an event where a contact answers with a file attached. For instance, when the contact takes a picture and sends this.

All of the above events need to be activated explicitly (this is done by the Marigold administrator configuring the app). However once activated data is stored automatically in the **Events** table

For each one of these events a series of standard fields is available:

All standard SDK fields start with SEF\_. These fields will then be matched with fields in the list.

At PushReceived/PushOpened

- SEF\_TYPE (type of event, expresses as a numeric value)
- SEF\_LIST\_DEVICE\_ID (ID of the device)
- SEF\_IS\_IAM (is this a in-app message)
- SEF\_PUSHID (the id of the push)
- SEF\_CREATED\_DT (the date of creation)
- SEF\_APPLICATION\_KEY (the app key)
- SEF\_IS\_IAC (is this an in-app content)
- SEF\_BTN\_ID (the id of the button)

At ClickButton/ClickButtonCancel

- SEF\_TYPE
- SEF\_LIST\_DEVICE\_ID
- SEF\_IS\_IAM
- SEF\_PUSHID
- SEF\_CREATED\_DT
- SEF\_APPLICATION\_KEY
- SEF\_BTN\_LABEL
- SEF\_IS\_IAC
- SEF\_BTN\_ID

At UserCustomEvent

- SEF\_TYPE
- SEF\_LIST\_DEVICE\_ID
- SEF\_CREATED\_DT
- SEF\_APPLICATION\_KEY

- SEF\_IS\_IAC
- SEF\_BTN\_ID

#### At MediaEvent

- SEF\_TYPE
- SEF\_LIST\_DEVICE\_ID
- SEF\_IS\_IAM
- SEF\_PUSHID
- SEF\_CREATED\_DT
- SEF\_APPLICATION\_KEY
- SEF\_BTN\_LABEL
- SEF\_MEDIA\_TEXT (the text replied)
- SEF\_IS\_IAC
- SEF\_BTN\_ID

The following event is always executed. The app developer does not need to do anything for this event:

- **Setinfo:** this event is executed when the app is launched on the device. The SDK will automatically send information to Campaign and store that data in the Devices list

Following is a list of all the fields that are used by this event. The fields in green are by default matched with the Devices table. The other fields in black can be matched either in the profile extension of the Devices/Contacts list, the Devices list itself or the Contacts list. Never in the Events list.

- SEF\_DEVICE\_UID
- SEF\_DEVICE\_TOKEN
- SEF\_PLATFORM
- SEF\_PUSHOPTOUT
- SEF\_IAM\_OPTOUT
- SEF\_OPTOUT
- SEF\_APPLICATION\_ID
- SEF\_DEVICE\_NEW\_ID (used to store the new Device ID in case it changes (eg contact swaps devices))
- SEF\_DEVICE\_TYPE
- SEF\_COUNTRY
- SEF\_SDK\_VERSION
- SEF\_SYSTEM\_VERSION
- SEF\_TIMEZONE
- SEF\_APPLICATION\_KEY

- SEF\_IDENTITY
  - Then there are the events *Login, Logout, Register; Unregister* which return data to Campaign when a contact logs in or out, registers or unregisters from the app.

These events are tracked always, however only when there's a matching field for these events they will be stored.

Following is a list of default fields that are available for these events.

At Login

- SEF\_USER\_ID
- SEF\_FIRST\_LOGIN
- SEF\_LAST\_LOGIN
- SEF\_LOGIN\_COUNT
- SEF\_LOGGED
- SEF\_MAIL

At Logout

- SEF\_USER\_ID

At Registration/Unregistration

- SEF\_USER\_ID
- SEF\_REGISTER
- SEF\_MAIL

Ex: This means that if you want to store the last login date of the contact for your app, that you need to add the field SEF\_LAST\_LOGIN to the field matching.

- Last, there are the **custom events**. These events need to be defined entirely from scratch. As well as the custom fields that are used by these events. (The app developer needs to know which event and what data to be able to send the information to Campaign from the app)

EX: the standard *Login* event only stores the data of the last login. If you want a history of dates of login you will have to create a custom event *Login*, that has a custom field 'Date' and add this custom field to the Events table (with Field matching in the Mobile module). You also need to create the field in the Events list.

## 8 Installation and Set-up

The installation is performed by the Marigold team. They will execute the required steps to install the services and platform. However, the customer has two actions to take:

- Create the app using the SDK
- Provide information to Marigold on:
  - Tables to use
  - The folder where tables should be stored
  - Fields to match
  - Events to track

### 8.1 WHAT DO YOU NEED TO KNOW BEFORE STARTING

It is important to define who needs to know what before setting the environment.

The app developer needs to know.

- which events need to be called and what data needs to be passed on.
- If custom events are required, when they need to be called and what custom fields are to be linked to those events
- If containers need to be created in the app to show in-app content, where and how they are named.
- If a registration form is required
- If opt-out needs to be provided in the app for in-app messages

Note that the app developer must also handle specific items. For instance, if you want to track registration, the app developer must provide in the app a Registration form that when submitted calls the Register event and passes on data.

The same goes for a Login to the app. The login process is handled by the app developer. If you do not want a login that is fine as well.

Note: Marigold provides all required documentation to set up the app using the SDK. This documentation exists for IOS and Android.

The Campaign system administrator needs to know

- Technical details on the app: a **ClientPublicKey** and a **SelligentPrivateKey** are communicated to the customer. => comes from Marigold
- Technical details on the Campaign environment, such as API urls and keys => comes from Marigold
- What lists should be created in Campaign
- What fields should be created for each one of these lists

The Campaign developer needs to know

- How the interface is called to be able to use it in a journey
- The names of the lists to use

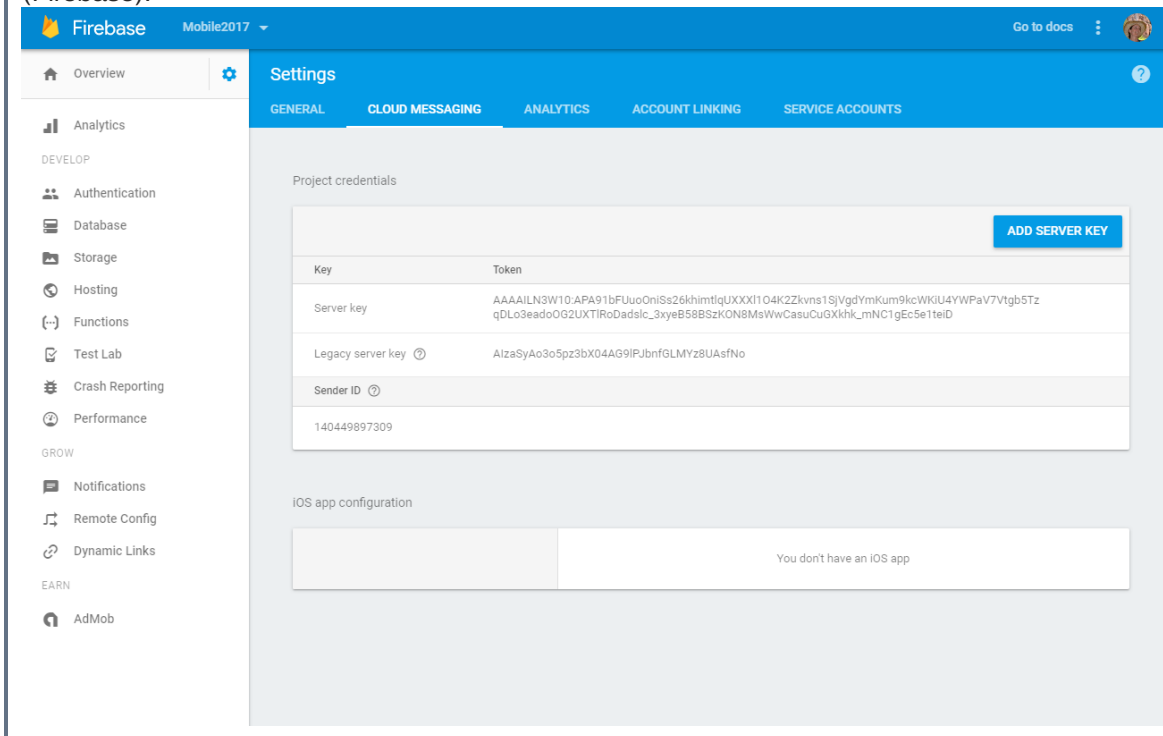
### 8.2 CREATE OR OBTAIN A MOBILE APP

To send a mobile message, you need to obtain or create a **mobile app**. Your app must also include the Software Development Kit (SDK). Marigold provides an SDK for iOS and for Android. Marigold provides detailed technical documentation on how to include the SDK in the app.

Technical note.

The IOS certificate and secret need to be provided to Marigold. For more information on how to get this, check out the [addendum](#).

For Android apps, the server key must be provided. (This is available from the Google admin console (Firebase)).



The SDK contains a series of standard methods/events that you can use to send information from the app to the platform and store the data in Campaign lists. There are standard methods/ events with standard fields, but the customer could extend this with custom events and fields.

An example of an event is a *Login* in the app. This event must be called if you want to track whenever a user logs in into the application. One of the possible SDK fields available for the *Login* event is the DATE/TIME of login.

It is up to the app developer to call the right methods at the right location in the app and pass on the correct parameters to the platform. (An interaction between the Campaign developer and the app developer is required to know what events and data needs to be included in the app.)

**Note:** When using the SDK, the developer must specify a URL to connect to the service **PushEvents**. For more details, contact your Support representative. This URL always has the following structure:

`<URL of install>+"Mobilepush/api/"`

### 8.3 CREATE MOBILE APP CONTENT CONTAINERS

Your app must also to configured to receive **in-app content**. Therefore, content **containers** must be created in the app at the location where the content will appear.

These content containers are each given a specific name and are identified by the term 'Container'. The name of the container is then used when defining the in-app content in the Mobile module. The names and locations of the containers must be discussed between Campaign developers and the app developers.



## 8.4 REGISTER THE MOBILE APP (OPTIONAL)

Next, you will want to create a way to register the mobile app user so that they are associated with their mobile device in Campaign. To do this, you must provide a registration form to the app developer, who will use it to create “Registration” event in the app.

When the user registers in the app, they provide an email address and name. The registration data is captured and sent to the platform and a link between the device and the mobile user is stored in an audience list based on their email address. Each contact may be linked to multiple devices.

## 8.5 INCLUDE AN OPT-OUT EVENT (OPTIONAL)

The app developer also needs to know if the contact can opt out (decline or refuse to accept) for in-app messages. If this is the case, the app developer should include this option in the app.

Campaign also provides dedicated opt-outs for both push notifications and in-app messages. If the user opts out, push messages are deactivated in the settings of the device. However, in-app messages can still be retrieved and shown in the inbox.

Regardless, the app developer can provide an opt-out option within their app, but not on the device. When the opt-out value=0, the device is opted in. If the value=1, the device is opted out. When the value=2, the device ID has been modified.

## 8.6 CREATE A DEDICATED MOBILE INTERFACE IN CAMPAIGN

To be able to create a journey with a Push component, a custom interface component must be created between the platform and external software.

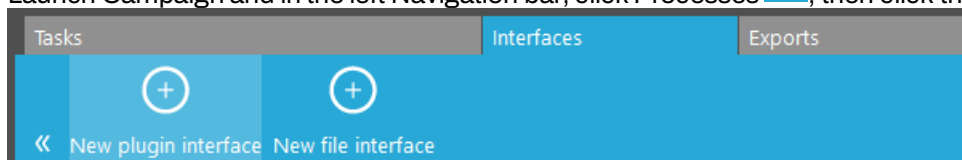
Campaign’s Mobile Push uses a custom DLL script “plug-in” to manage interactions with the external software. The **Mobile plugin (dll)** predefines information and events that can be used in Campaign journeys, depending on the SDK used to develop the external mobile app.

Depending on these variables and events, an interface component can have different input fields for the variables, and different triggers (arrows in a journey) for the events.

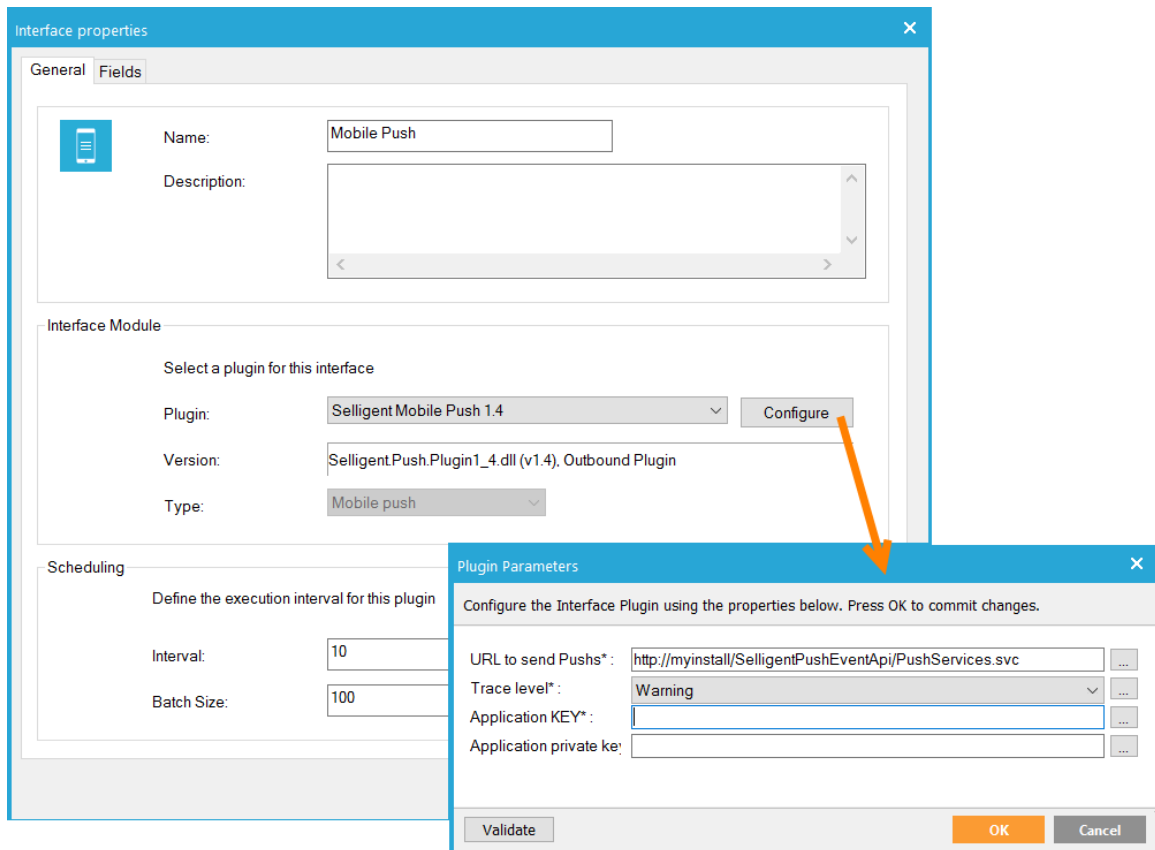
For instance, queries can be executed to determine if an in-app message has been sent or if a response has been received from a push notification or other message. When these events are configured in the interface, they can be used in a journey.

**To create a plugin:**

1. Launch Campaign and in the left Navigation bar, click Processes , then click the **Interface** tab.



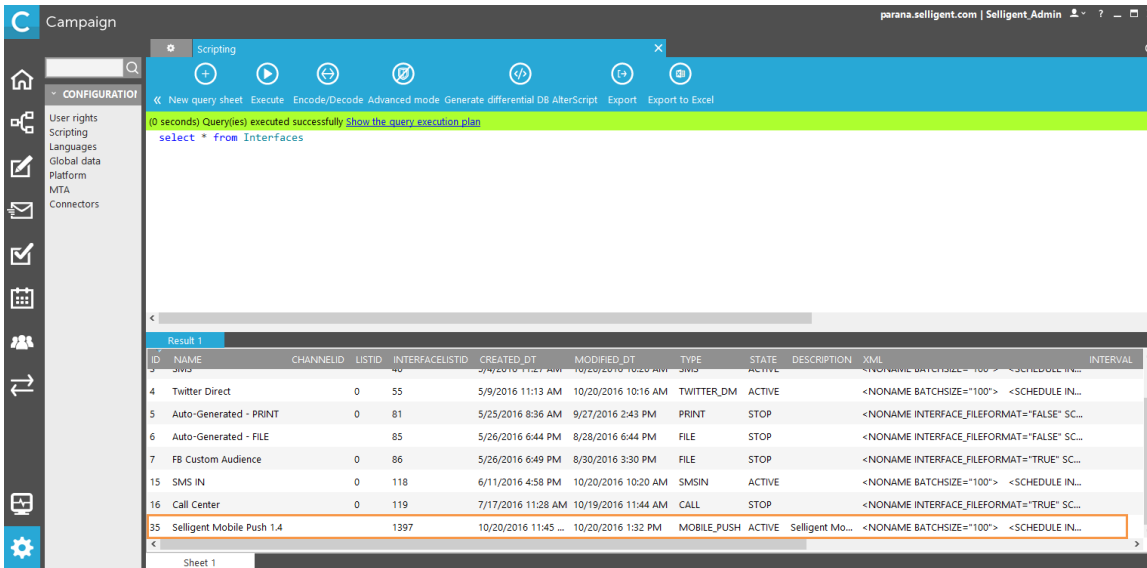
2. Click **New plugin interface**.
3. In the **Plugin** field, select the **Mobile Push** plugin.
4. Click **Configure**.



5. Specify the **URL** for the push service.  
**Example:** <Selligent\_install\_folder>/SelligentPushEventApi/PushServices.svc
6. Select the **Trace Level** at which you want to log messages. Note that the Debug level should only be used in test mode
7. Enter the **Application Key** and **Application private key** provided by Marigold.
8. Click **OK**.

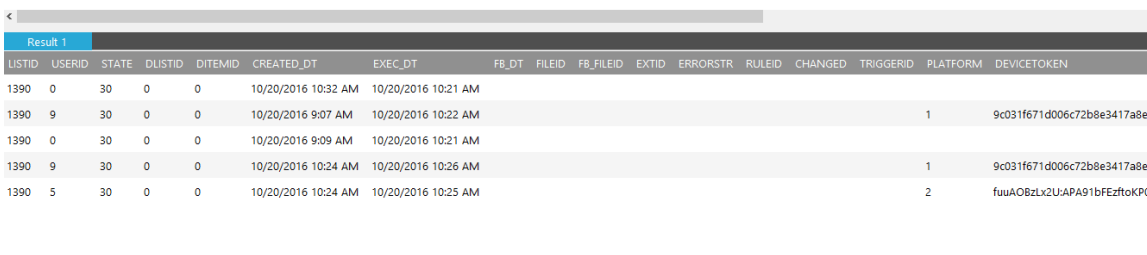
When a new interface is created, a record is created in the INTERFACES table and an ID is attributed to this interface:

To check this, go to Configuration/Scripting and enter the following: select \* from INTERFACES

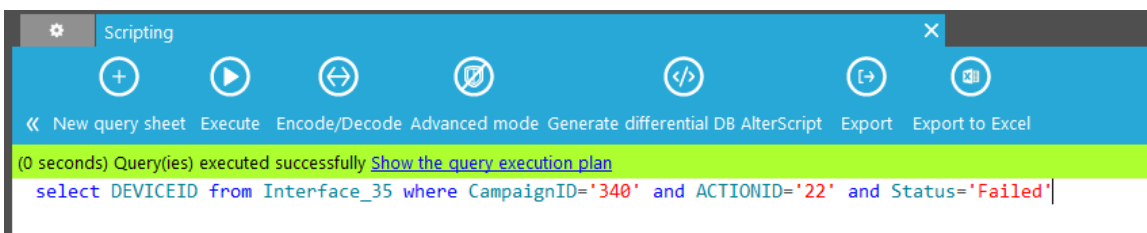


For every interface a dedicated table is created holding all the fields that the plugin provides in addition to the standard Interfaces fields. This table will store a record for every time the interface is executed. You have a view on the status of the interface (Failed, Sent, processing, treated) for each of the devices to which the message is sent, the platform that was used (IOS/Android), if the device is opted out, the campaign in which the interface is used, etc

In the Scripting section enter: `select * from INTERFACE_35` => 35 is the id of the mobile push interface



For example: you can use the status field to create a selection of devices for which the message was not sent (failed). Filter on the Campaign ID and action ID to select the right message and use `status=failed`



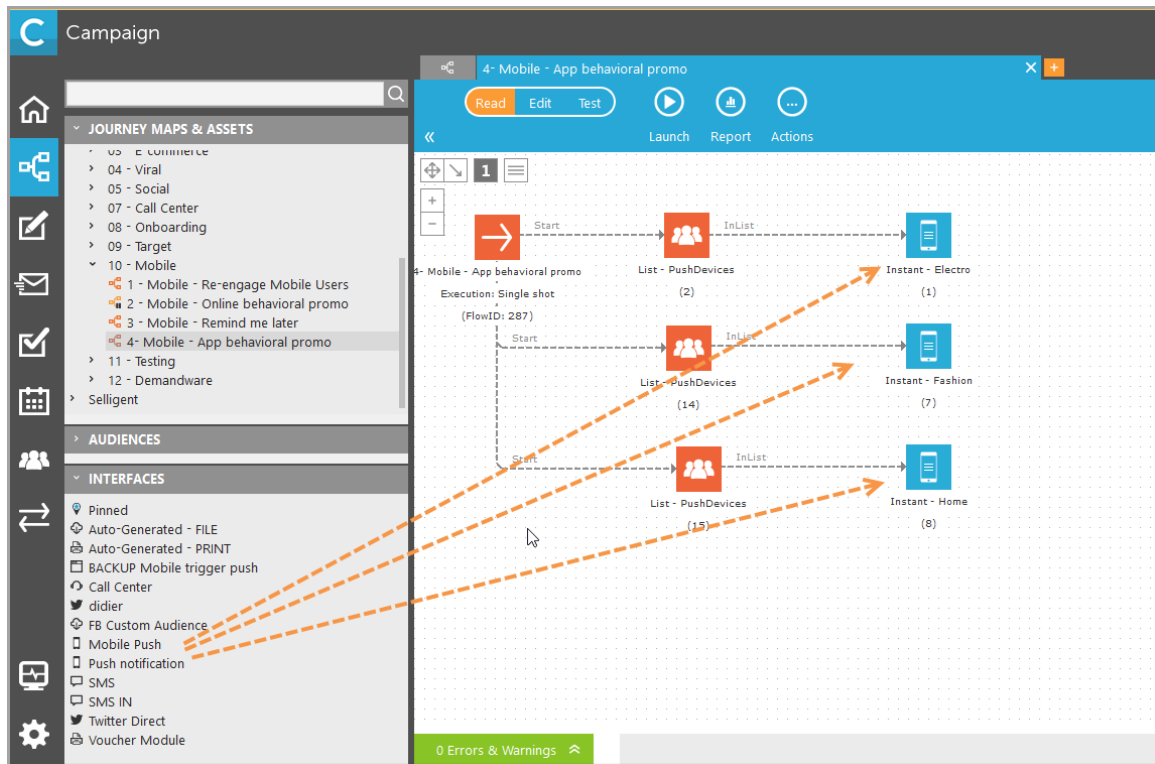
`select DEVICEID from Interface_35 where CampaignID='340' and ACTIONID='22' and Status='Failed'`

## 9 Configuring the Push component in a journey

When sending push messages from a journey, the Devices list is used as the audience list. The reason for this is that one contact can have multiple devices and if we use the contacts audience list, Campaign does not know to which device the message should be sent as there can be more than one.

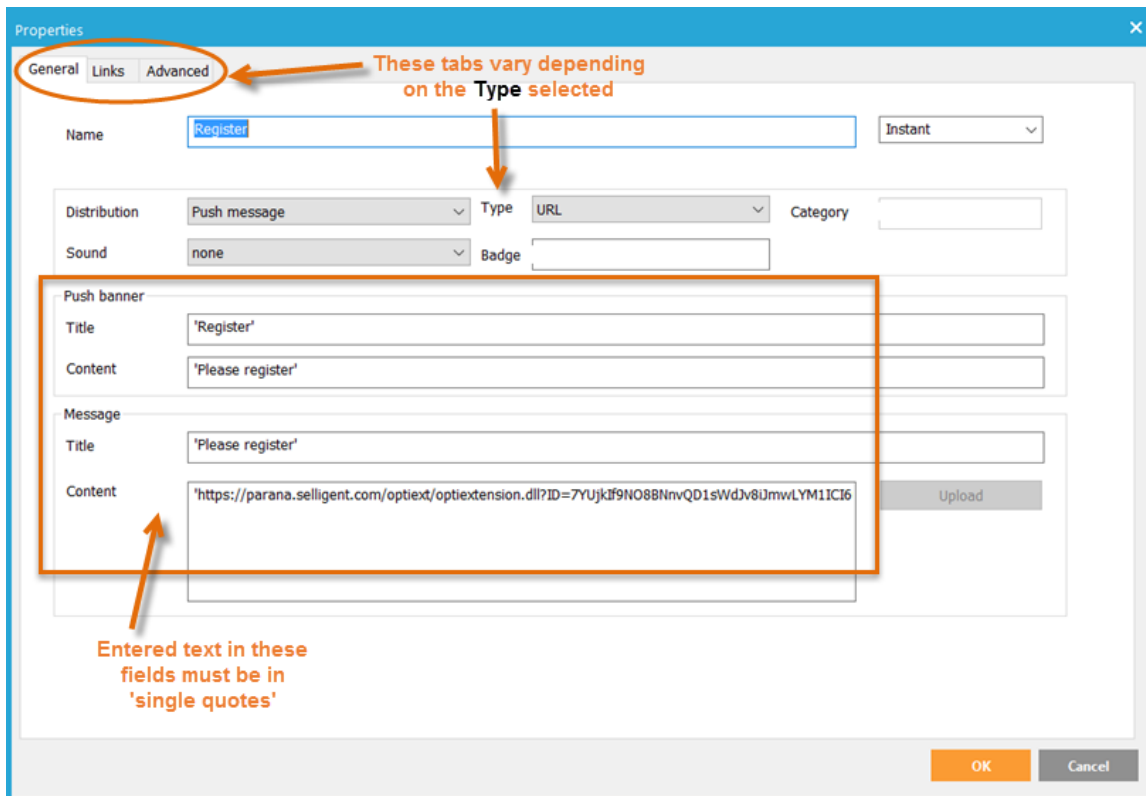
To create a journey using Mobile push:

1. Create a new journey and add the **Devices** audience list from the **Audiences** tree on the left.
2. Drag and drop the **Mobile Push** component from the **Interfaces** tree on the left.



3. Configure the Mobile Push component as follows:

**NOTE:** All values entered in the **Push banner** and **Message** fields need to be placed in single quotes.



Properties

General Links Advanced

These tabs vary depending on the Type selected

Name: Register Instant: Instant

Distribution: Push message Type: URL Category:

Sound: none Badge:

Push banner

Title: 'Register'

Content: 'Please register'

Message

Title: 'Please register'

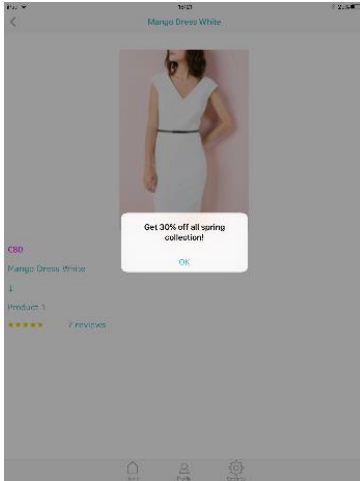
Content: 'https://parana.selligent.com/optiext/optiextension.dll?ID=7YUjkf9NO88NnvQD1sWdJv8UmwLYM11C16' Upload

Entered text in these fields must be in 'single quotes'

OK Cancel

Specify the following parameters:

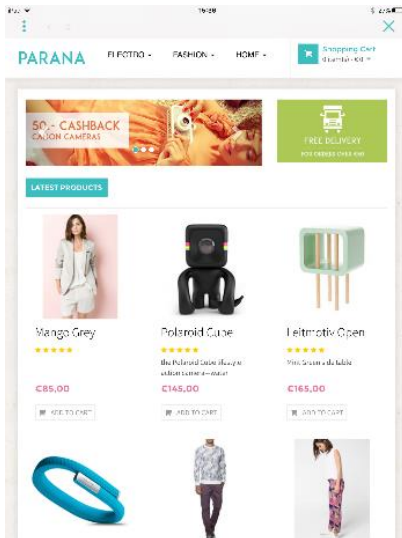
- **Name:** Enter a unique name for the Mobile Push component.
- **Instant:** Choose from the drop-down menu whether activation of this component in the journey will be **Instant** or **Scheduled**.
- **Distribution:** This setting determines how the message is sent.
  - **Push notification:** Send push notification.
  - **In-app message:** Send an in-app message.
  - **In-app message if Push unavailable:** Send a push but when not available, send an in-app message.
  - **In-app content:** Content to be displayed in a container in the app.
- **Type:** This is the type of message that will be sent. Not all types are available for all distributions. The available types are:
  - **Alert:** (Not available for in-app content) This option sends a simple text notification that is no more than 500 characters. Text entry must be surrounded in single quotes. To use personalized fields, add them with the plus sign. **Example:** 'Hello'+MASTER.FIRSTNAME



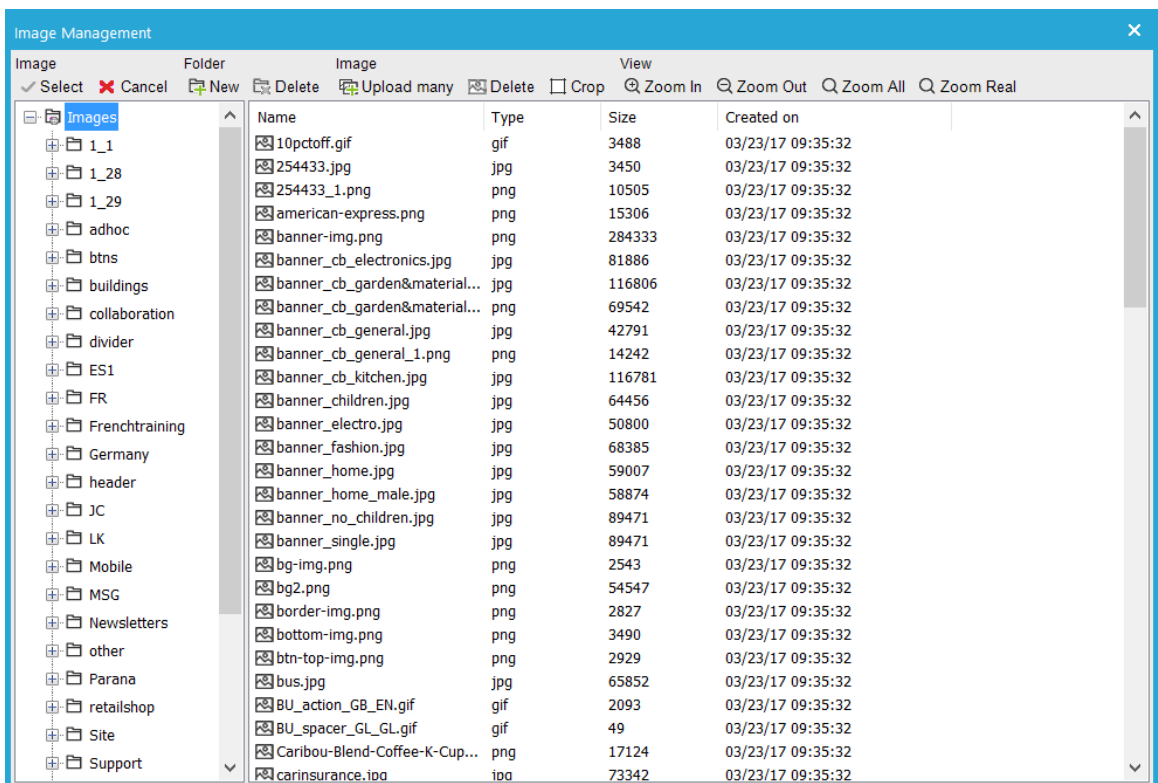
- **HTML (not for in-app content or in-app message):** With this option, you can specify website content containing text (max 500 characters), links, images or any other HTML5 elements. HTML notifications are recommended for message formatting.



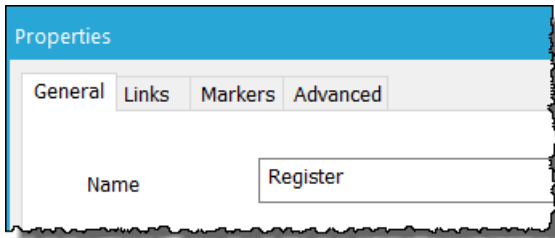
- **URL:** With this option, you must enter the URL of a website page in the Message Content box. The content of the referenced website is displayed in the notification window and the app user can browse the web page.



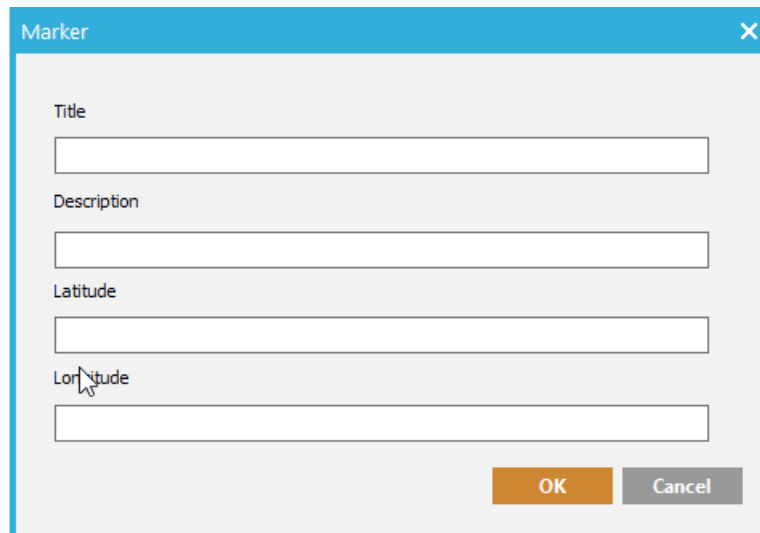
- **Image:** When this option is selected, you can click the **Upload** button (under Message) to select an image from the Image Management window (shown below), which gives you access to the images in Campaign. Or you can enter the URL of an image on a web page. When an image notification is sent, the URL of the image is sent and the image is rendered when the notification is displayed.



- **Map** (Not available for in-app content) With this option, markers (map pins) must be defined to display on the map. At least one marker must be defined. When you select this type, an additional **Markers** tab is added to the dialog.



Click the **Add marker** button to start adding markers.



The **Title** and **Description** that you enter for the marker will be displayed in the information window when the user clicks on the marker on the map.

Note: A maximum of 100 markers may be defined.





- **Page in another journey:** This allows to jump to a page in another journey and is available for push notifications and in-app content.
- **Category:** Mandatory, but only applicable for in-app content. By setting a category for the in-app content, you can display only content of the specified category in a container. For example, you could create a container that only displays content of category “Home.”
- **Sound:** Specify the sound played when the push notification is received.
- **Badge:** On iOS devices only, when a notification is received, a badge is displayed on the app icon. The number on the badge indicates the number of notifications received and not yet opened. The number in the badge is always incremented by the number in the “badge” property of the notification. Opening the app resets the badge.
- **Push Banner:** Only used for push messages.
  - **Title:** Title displayed when a push message is received in the notification center or in the lock screen (depending on iOS setting).
  - **Content:** The content of the notification in the notification center.
- **Message:** Used for push, in-app messages, and in-app content.
  - **Title:** The title of the message that will be displayed in the application inbox or in the app container.
  - **Content:** The content of the message or in-app content.
- **Upload:** This button allows you to upload an image from Campaign.

## 9.1 PUSH PERSONALIZATION

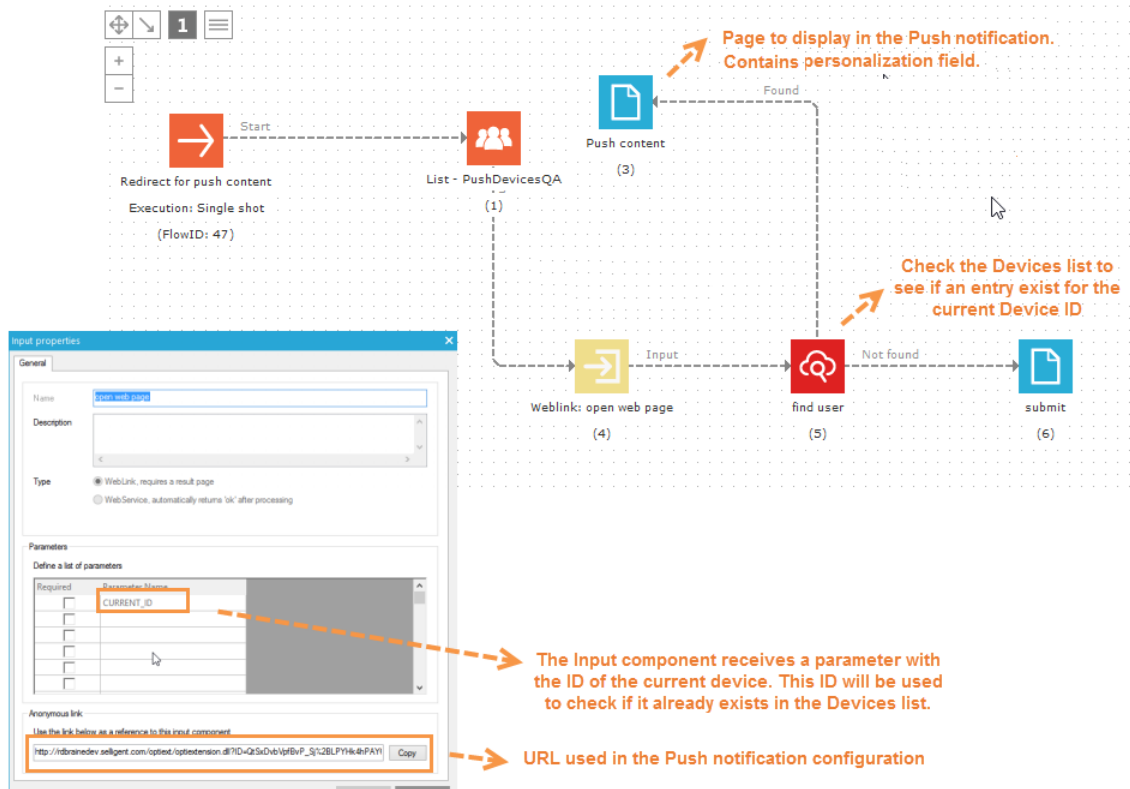
You can personalize a page displayed in the push message (for example, the contact’s address and first name) by using two journeys.

- Create a journey that contains a **Push** component of type URL, with the URL pointing to the input component of the journey containing the web page. This URL requires a parameter to pass to the input component. **For example:**

```
http://[install]/optiext/optiextension.dll?ID=Ji6eQccmuP59n4Ge99K38ssUESqVdCbAr%2B%2B66CnOI1Bs
zIJ8vkX5Oqb602Sprz7C1hcbcaXjV3&CURRENT_ID=MASTER.ID)
```

- Create a second journey with an **Input** component that expects the ID of the current device, and perform a lookup in the Devices table and then display the page with the personalization.
- The URL used in the Push component will then point to the URL available from the Input component in this journey, completed with the parameter.

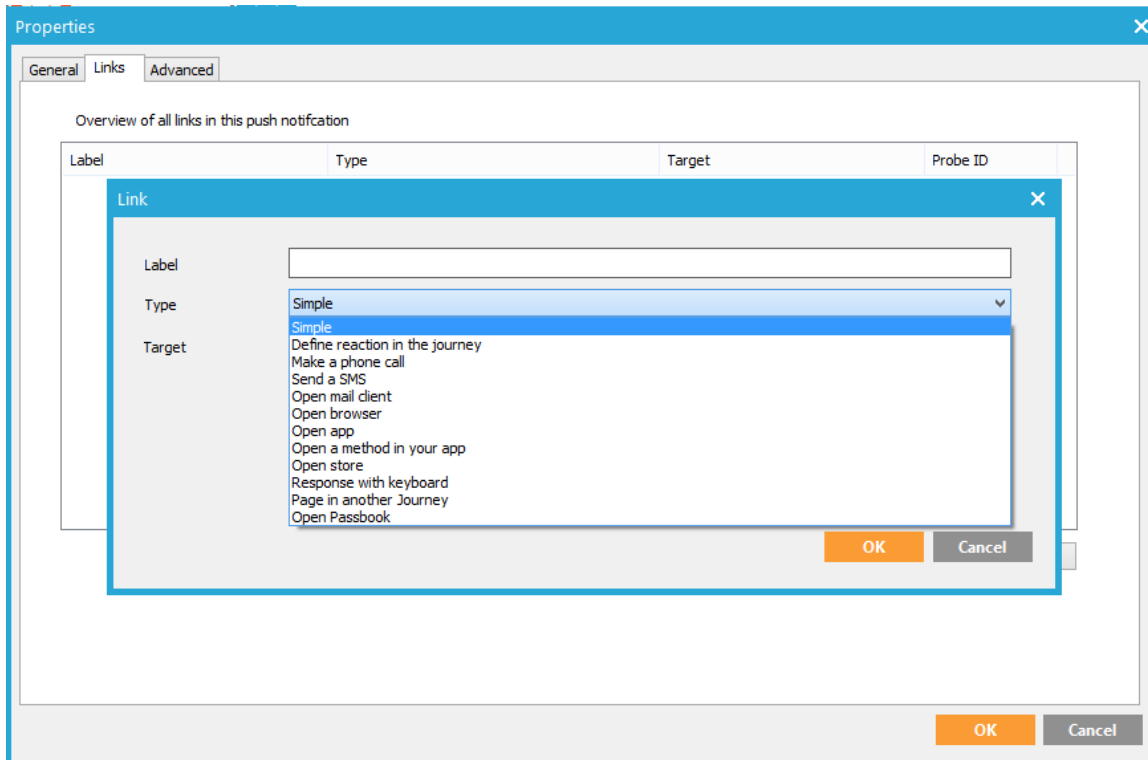
**Example:**



## 9.2 LINKS IN PUSH NOTIFICATIONS

On the Push notification Properties screen, you can also add hyperlinks to notification push messages.

A maximum of three links can be added to push and in-app messages. For in-app content, only two links are recommended and, for images in in-app content, only one link is recommended by iOS.



For each link, specify a **Label**, **Type** and **Target**.

The following link types are available:

- **Simple:** This is a simple hyperlink to a website URL.
- **Define reaction in the journey:** When this link is clicked, the journey redirects to another journey to send an email, etc. When the user clicks the link in the notification, it will close the notification.

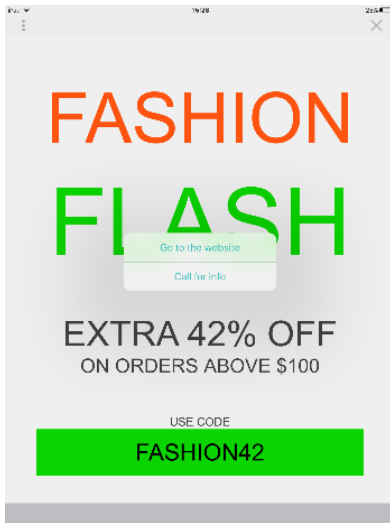
**Technical Note:** Currently, for technical reasons, when using this link, the Push component is followed by a Page component.

- **Make a phone call:** Allows the app user to make a phone call to the defined phone number.
- **Send a SMS:** Opens the default SMS application and sends an SMS text message to the indicated phone number.
- **Open mail client:** Opens the default email application to send an email to the indicated email address.
- **Open browser:** Opens the default browser on a defined URL.
- **Open app:** Opens a defined application such as Facebook, Twitter, etc.

**Technical Note:** The syntax of the target differs depending on the mobile device operating system. Currently, no automatic conversion is made to the correct syntax for iOS or Android. It is recommended that you create two separate Push components dedicated to each OS.

- **Open a method in your app:** Opens an API method in the application.
- **Open store:** Opens the store application (iTune or Android Market) to download a defined application.
- **Response (not applicable for in-app content):** a response is sent with a hardcoded reply.
- **Response with keyboard:** (not applicable for in-app content) Closes the notification and shows the keyboard and text area. When text is entered, a Send button is available. The entered response is send to the web service and stored in the table Events.
- **Response with picture** (not applicable for in-app content): there are two links in the app, one to take a picture and the other one to select a file. The interaction on these links is not tracked. An example of use of this could be a contest, where people need to send a picture holding the brand's product.

**Example with links:**



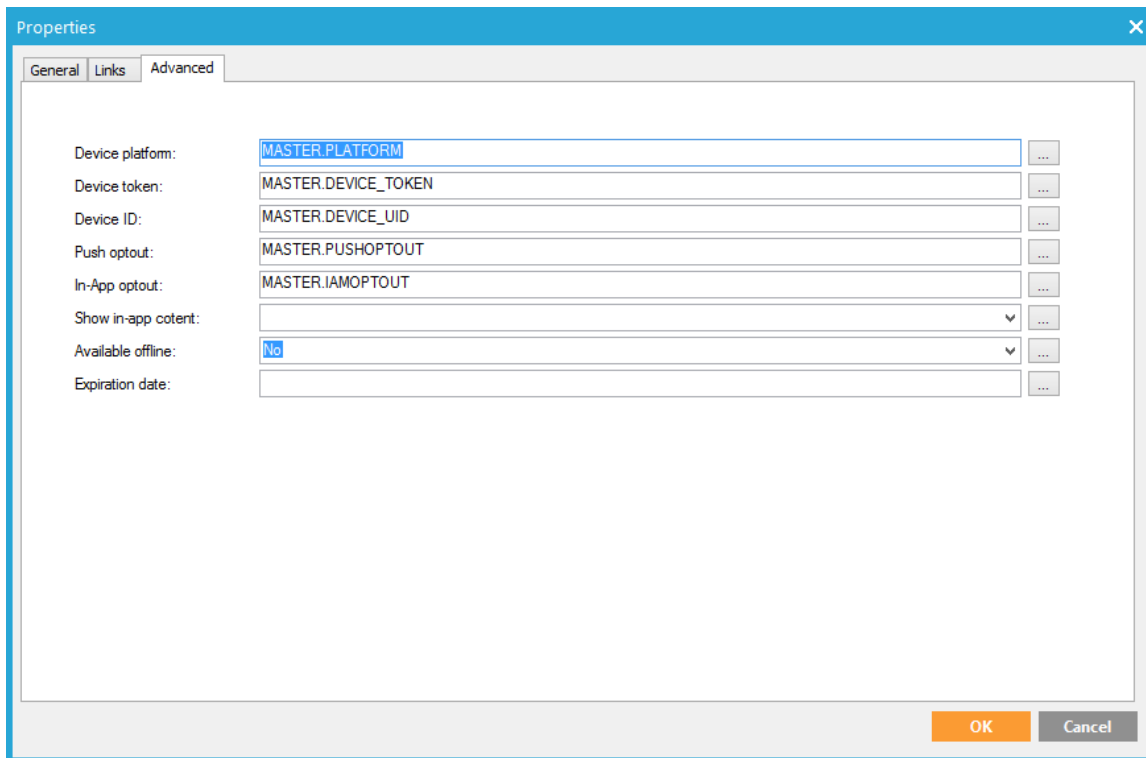
- **Page in another journey:** Opens the page in the browser
- **Passbook:** Opens the passbook file at the given URL (eg. airline boarding pass, movie tickets, etc). The passbook file must already exist. Campaign nearly provides the possibility to open it. When the button is clicked the notification is closed and the passbook is opened.

Note: depending on the selected Passbook provider personalization fields can be added. The field values are sent via a dynamic URL. Note2: Passbook formats are different depending on the OS. Make sure to target the right devices when sending a passbook.

Note: Links that are added to the push message behave like sensors in the message that send the contact to an external location. They can be used as events in a journey to move the contact further along the journey.

### 9.3 ADVANCED PUSH PROPERTIES

On the Push Notification **Properties** screen, you can click **Advanced** tab to configure the mobile device IDs and other parameters.



- **Device Platform:** Sets the platform (IOS or Android) to which the message is sent. This information is available in the audience list (e.g. DEVICES.PLATFOME)
- **Device token:** Defines the specific device to which the push notification should be sent. This information is available in the audience list (DEVICES.DEVICE\_TOKEN). The device token is a combination of the device ID and the app. When multiple apps are installed on the device using the same SDK, the device token will allow differentiating between the different apps. The device token is only used to send push notifications.
- **Device ID:** Field allowing retrieval of the device ID of the target (MASTER.DEVICE\_ID). This is the ID of the device that is used to send in-app messages.
- **Expiration date:** This applies to push and in-app messages and sets the expiration date after which non-retrieved messages are no longer delivered.

The expiration date is mandatory and by default set to 'today + 2 months'.

Expressions can also be used in the expiration date field (eg.: `~(DATEADD('dd',7,GETDATE()))~`)

- **Available offline:** Stores the in-app content locally once it has been viewed.
- **Show in-app content one time only:** If set to true, the content will only be shown during one session. If the user leaves the app and comes back again, the content is no longer available. If the option is not set to true, the content remains available until there is new content available on the server for that container.

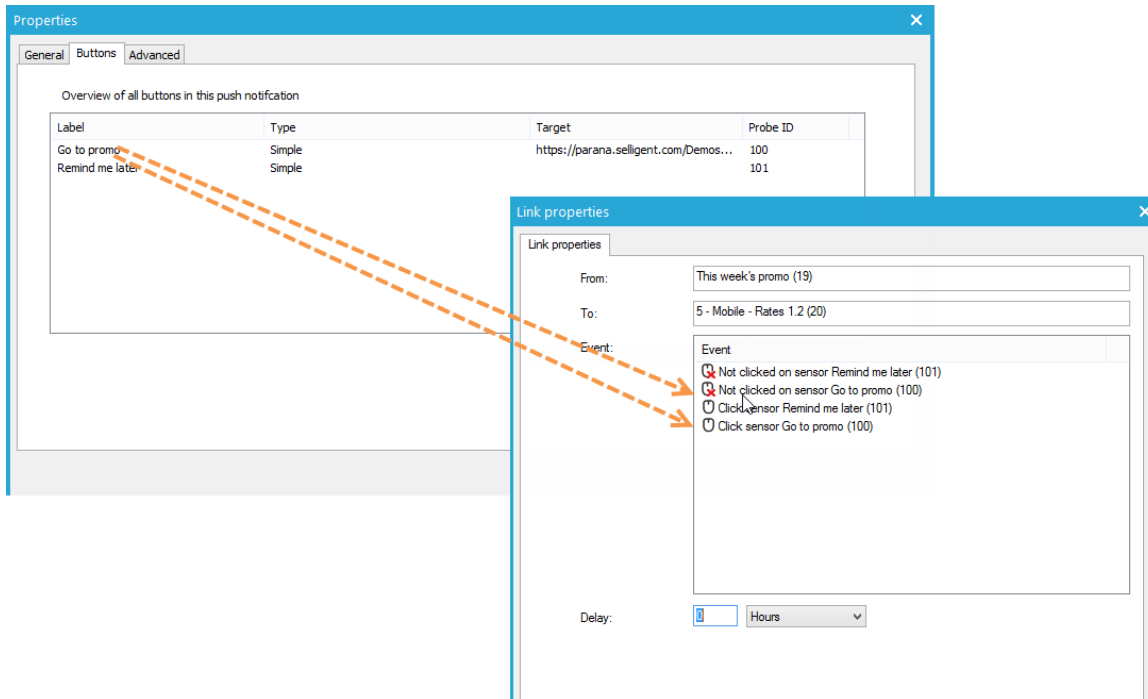
## 10 Events generated by the Mobile push component

The Mobile Push component generates a series of events, based on the links created in the message.

User interaction with these links can be used to push the contact further along in the journey.

### Example:

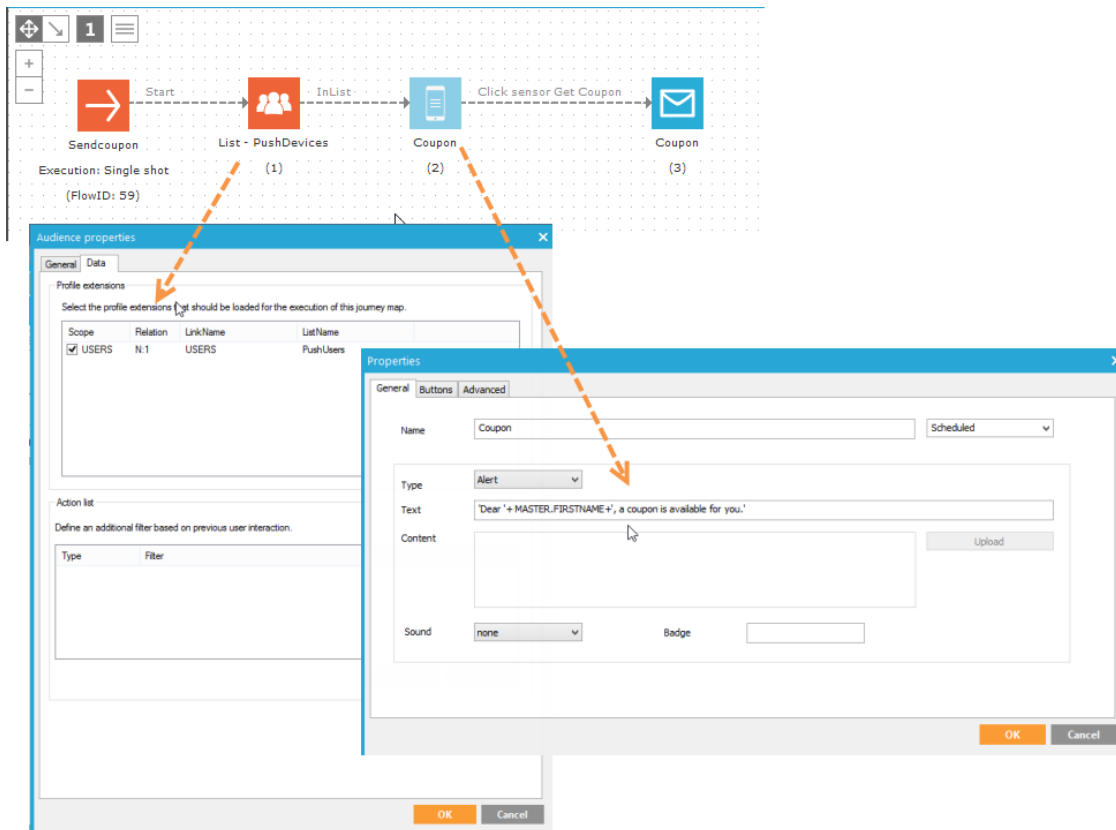
In the image below, the links 'Go to Promo' and 'Remind me Later' are simple links. The interaction on these links can be used as an event in the journey:



It is also possible to use the default events for a message, such as *Send*, *View*, *Undeliverable* and *After delivery*, as an event in the journey.

### Example:

In the journey in the image below, when the user receives the push message, a link is available to request the coupon. When the user clicks the link, an email is sent with the coupon.



The screenshot displays a journey map with the following steps: Start (Sendcoupon), InList (List - PushDevices), Click sensor Get Coupon, and Coupon. Below the journey map, two configuration windows are open:

- Audience properties:** Shows profile extensions for 'USERS' with a relation of 'N:1' and link name 'USERS'. The list name is 'PushUsers'.
- Properties:** Shows the configuration for the 'Coupon' step. The name is 'Coupon', type is 'Alert', and the text is 'Dear '+ MASTER.FIRSTNAME+', a coupon is available for you.'.

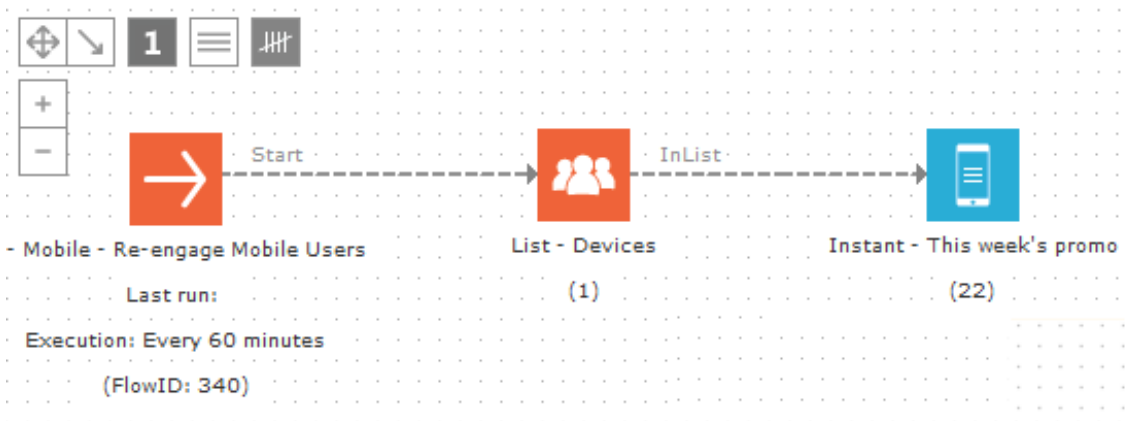
Note: Since the journey uses the Devices for an audience list, you need to enable the use of linked lists in the Audience component/Data tab to be able to personalize the message and send an email.

## 11 Using Mobile push component in a journey

When sending push messages from a journey, the Devices list is used as the audience list. This is because one contact can have multiple devices and if we use the contacts audience list, Campaign does not know to which device the message should be sent as there can be more than one.

### 11.1 PUSH NOTIFICATION FOR A LIMITED AUDIENCE

Following is an example of a push notification sent to a limited audience:



The filter applied to the Devices list will only target specific devices. The purpose is to relaunch users to use the app.

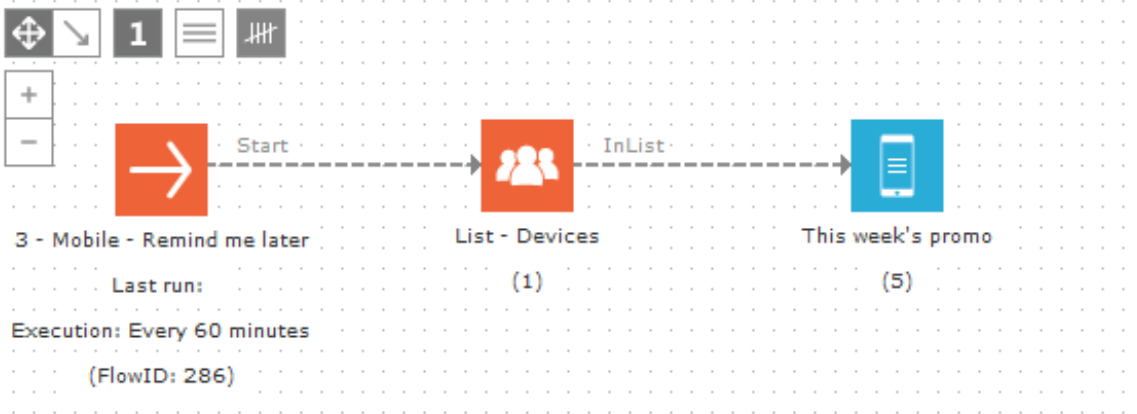
In this example, we assume that a user that logged on only once more than a week ago is a new user and it might be interesting to relaunch this user. Users that have connected multiple times in the past but haven't logged on for a month are considered as users that have 'abandoned' the app. The first part of the condition checks for users currently not registered or logged on.

```
SQL REGISTER = 0
OR
LOGGED = 0
OR
LOGIN_COUNT = 1
AND
LAST_LOGIN < 7DD
OR
LOGIN_COUNT > 1
AND
LAST_LOGIN < 1MM
```

The Push notification itself is a text message with a *Remind me later* link. For contacts clicking this link, an event is triggered and stored in the Events list which allows us to use that information and target them in another journey.

The following is an example of a triggered journey using the information stored on the events triggered on a device:

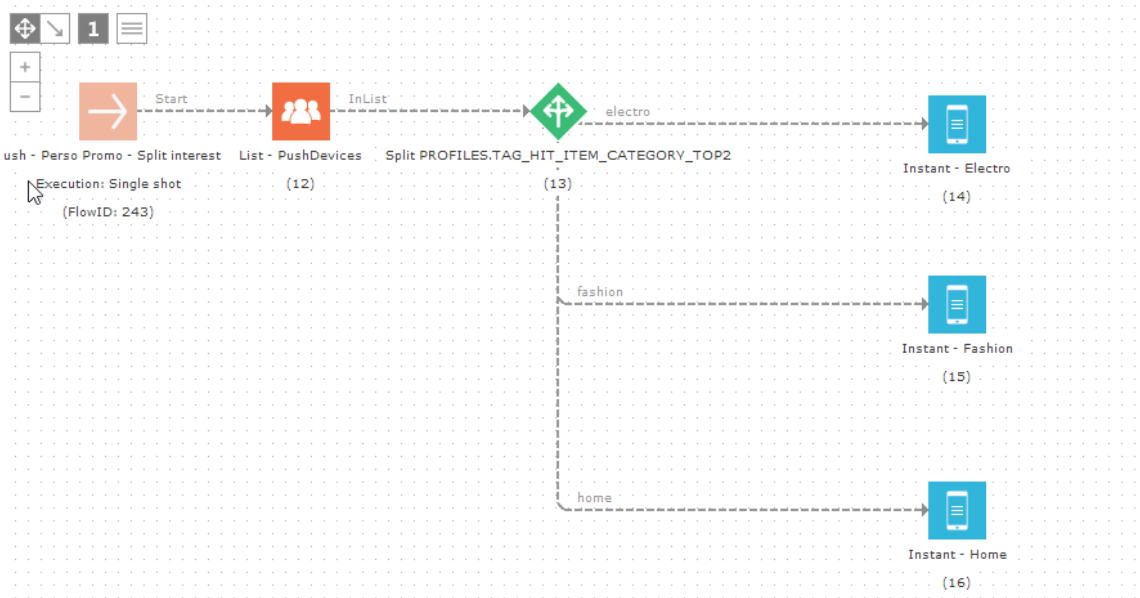




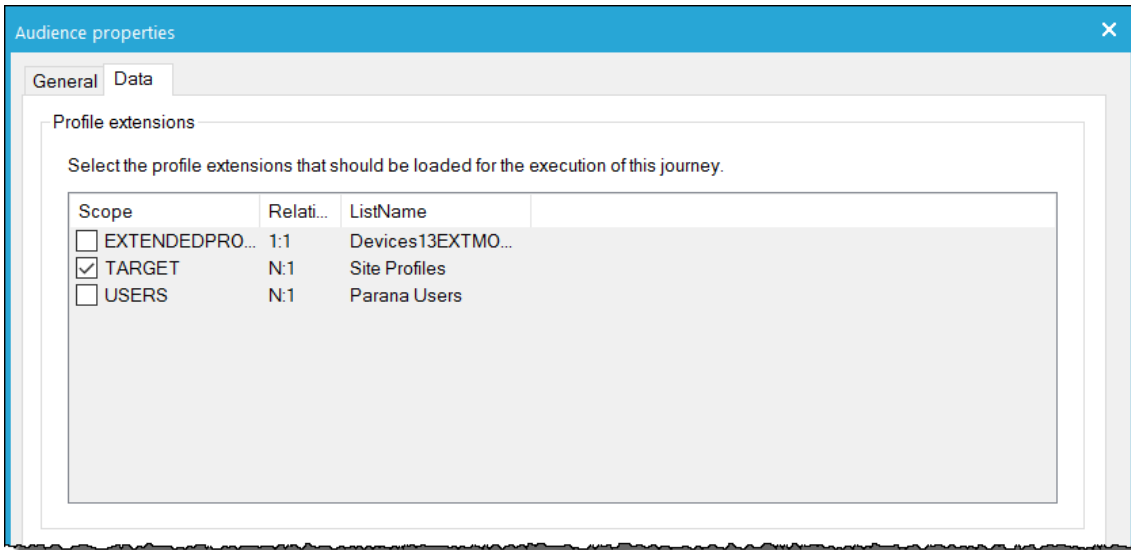
The audience is filtered based on the following filter: only contacts that pressed the button 'Remind me later' in the last two days are targeted:

```
Include where exists PushEvents
TYPE = 'ClickButton'
AND
BTNLABEL = 'Remind me later'
AND
CREATED_DT < -2DD
```

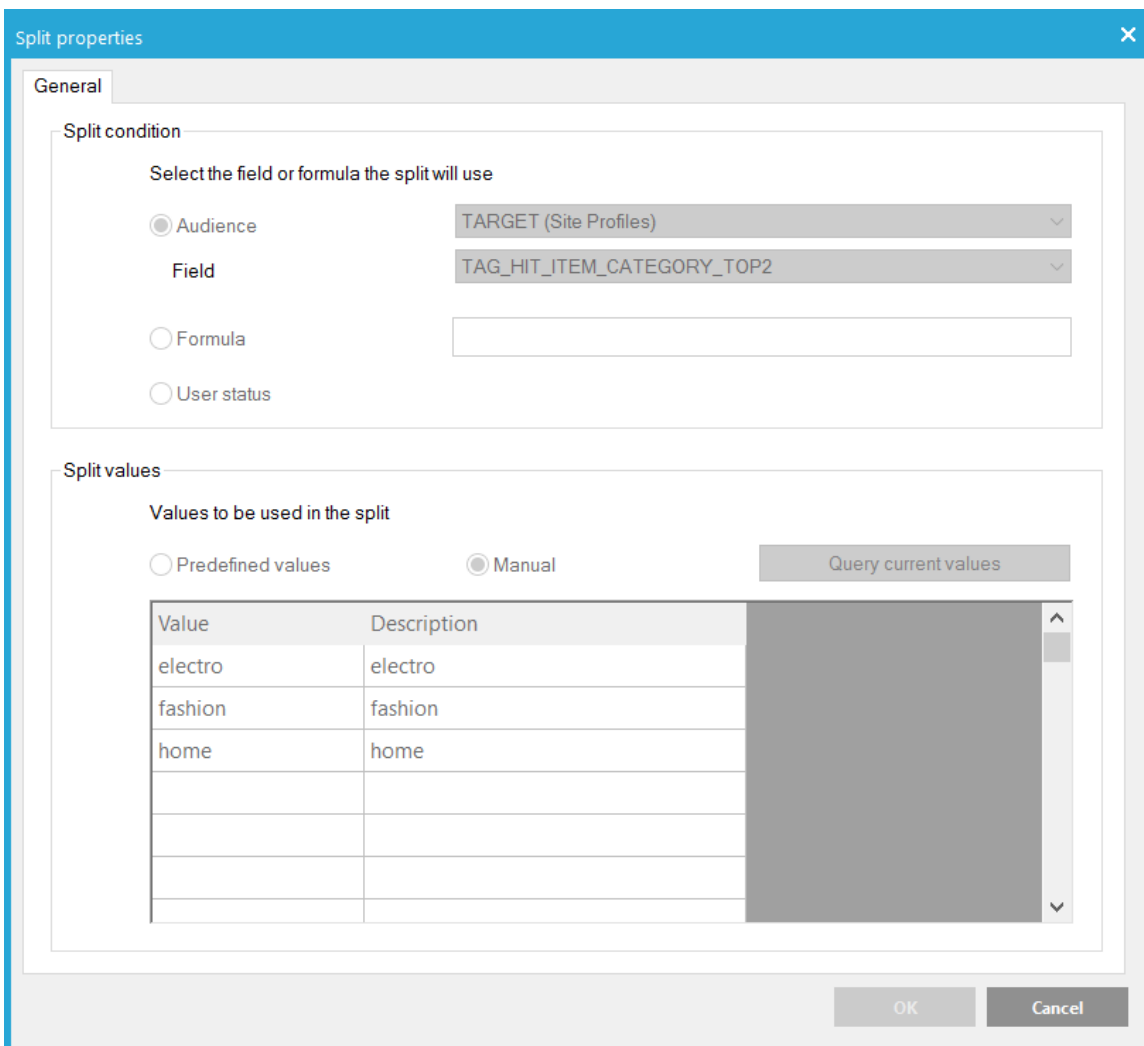
Another example of the use of the Push notification, after a Split component where a check is made on information in the user profile of the contact.



In this case, the user's profile in Site is directly linked to the Devices list and the audience list in the journey is set to enable the use of the target profiles:



The Split component will then use a field in this Site profile to detect points of interest and send notifications in correspondence with their interest:

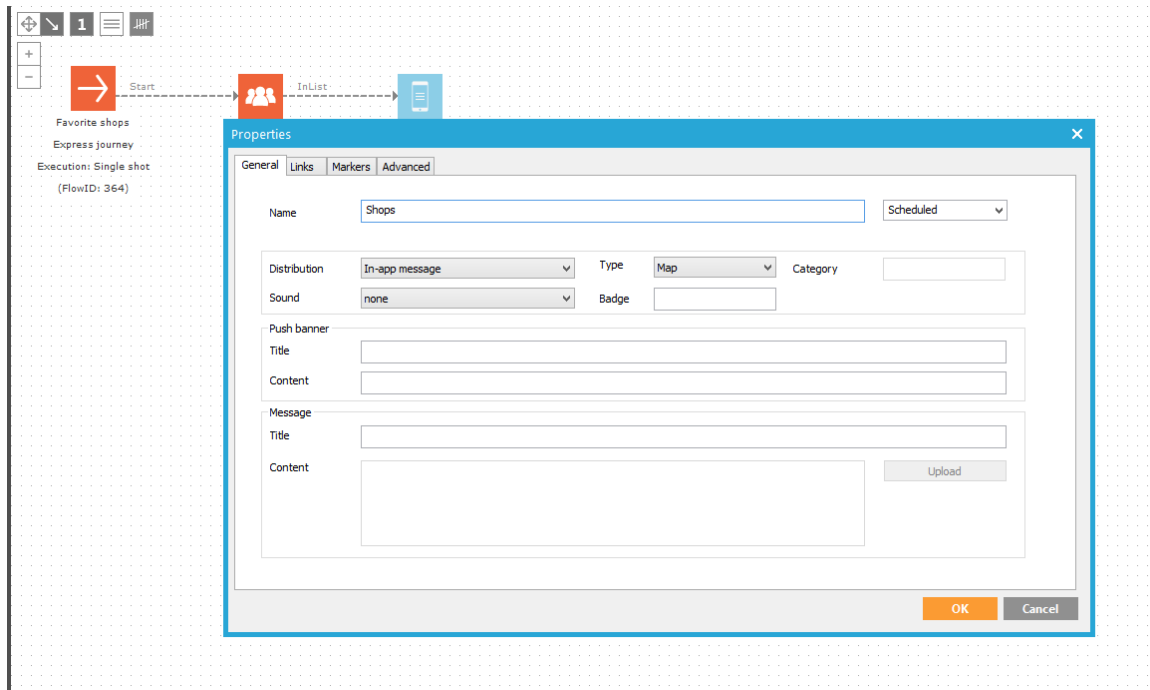


## 11.2 IN-APP MESSAGE

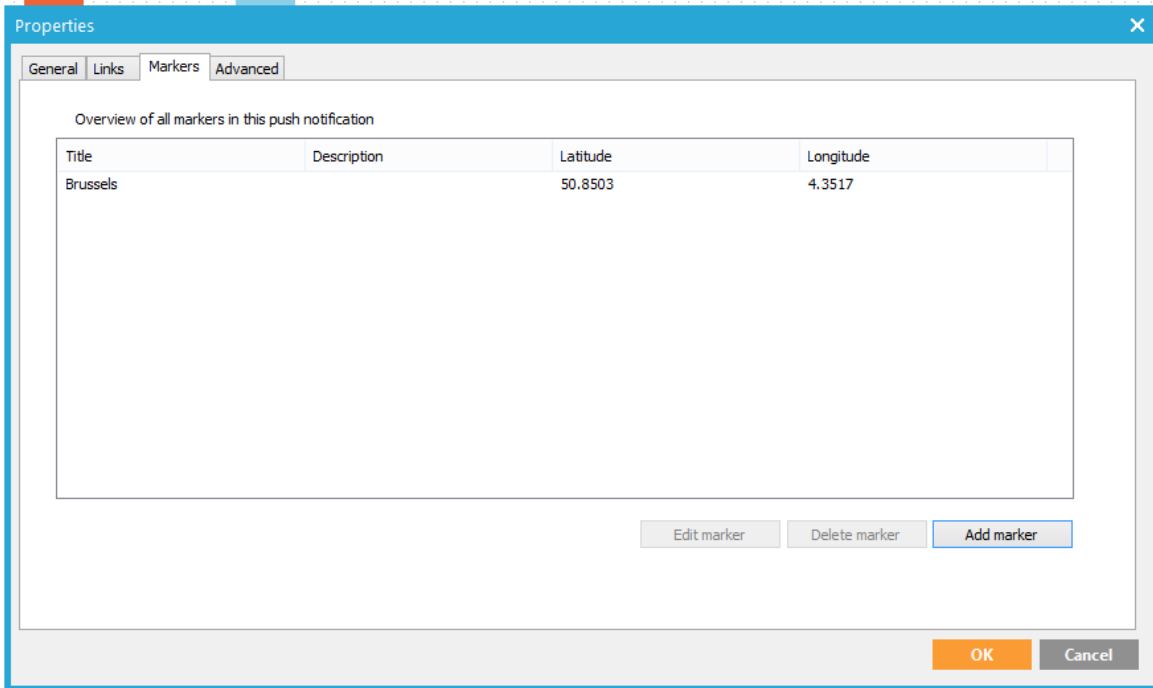
This example will show the app user a map with the shops. In this case, it is an in-app message and is only displayed when the contact is in the app.

When the in-app message is sent, it is marked as treated.

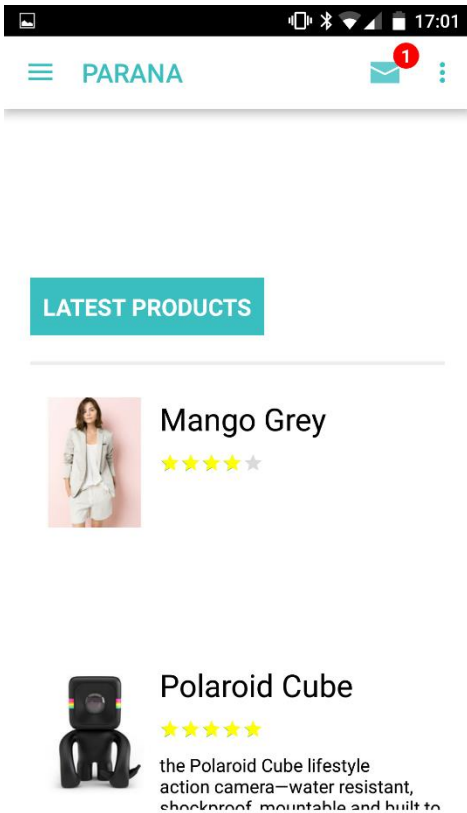
1. Create a journey 'Favorite shops' and add the devices audience and mobile push component as we have already done in this example
2. Open the properties of the push component.
3. Select in-app message and map as type. The markers tab is added.



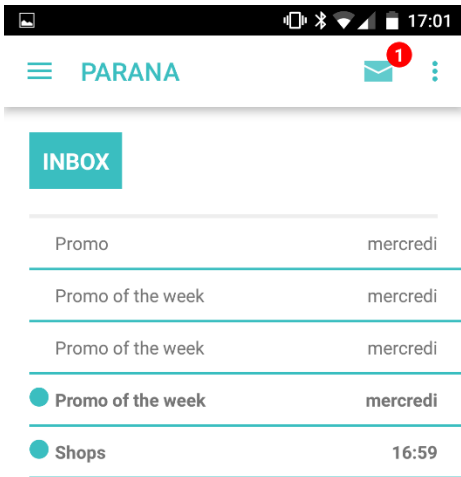
4. Add first a title and content for the message.
5. Next on the markers tab, add the latitude and longitude of each one of the locations. Use single quotes everywhere.



6. Close the dialog and save the journey.
7. When the message is sent it looks like follows in the app:



8. When the message is opened all inbox messages are listed and the one sent last is at the bottom



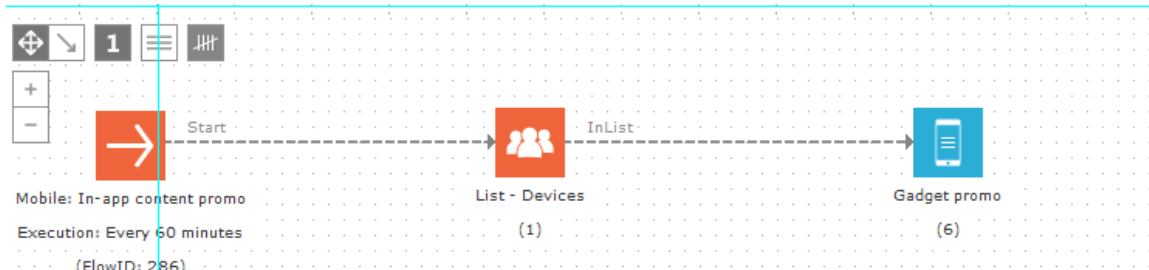
9. Open the message. The map with a marker on brussels is displayed.



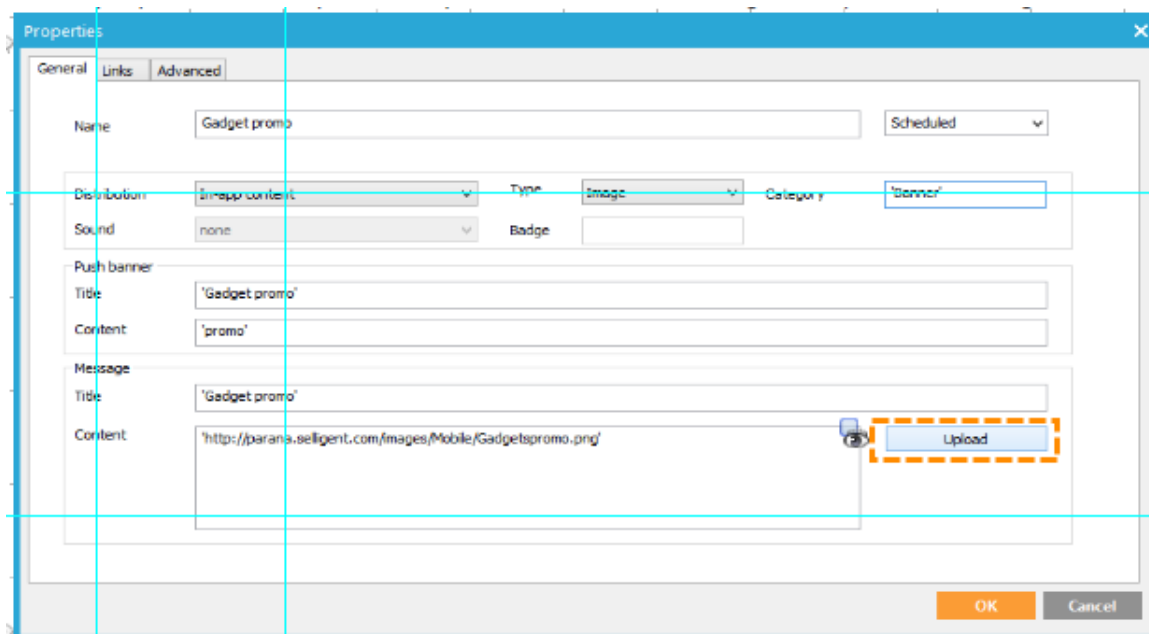
## 11.3 IN APP CONTENT

With this example we want to show you how specific content can be added to the mobile app. The app developer will define in the app containers where this content will be placed. These containers have a name and this name is what will be used when configuring the content. In our example, the container in the app is called 'Banner'.

1. So start by creating a journey using the Devices list and the Push component.



2. In the properties of the push component, set the following and upload the image

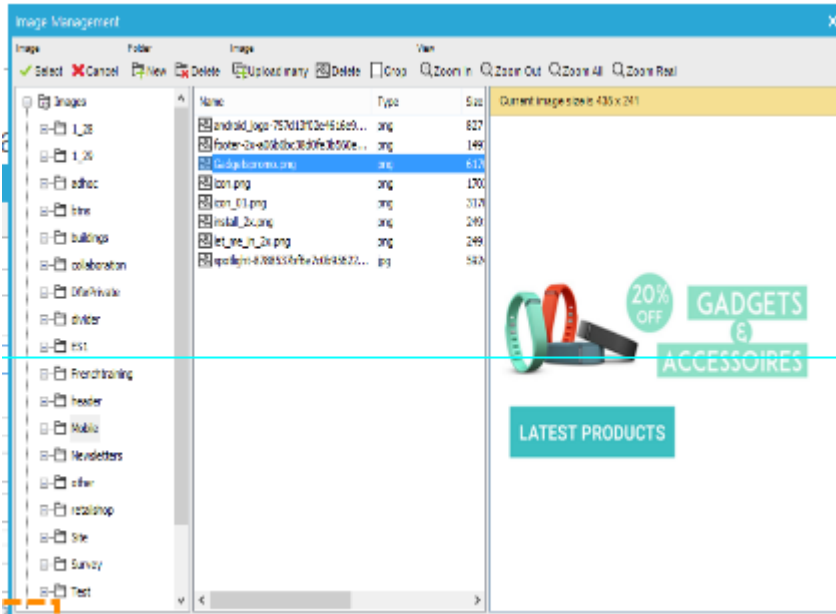


3. **Distribution** is set to in-app content of type image.

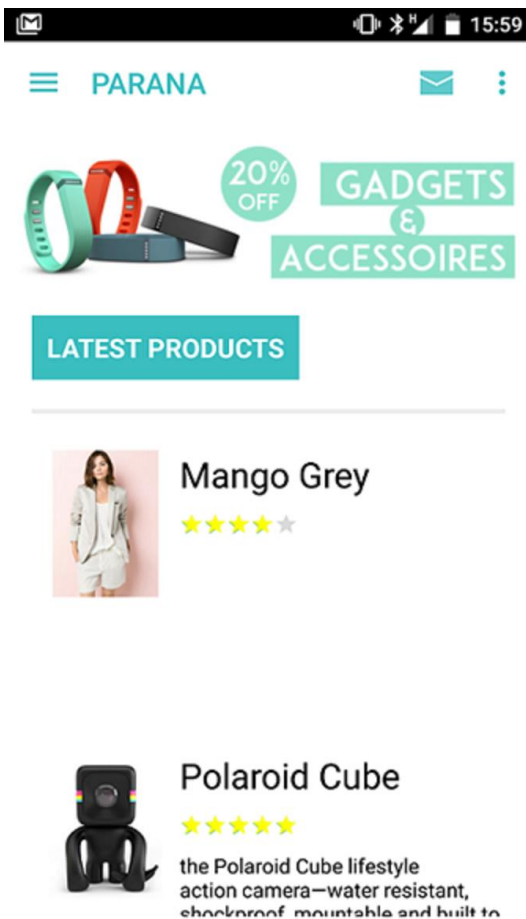
4. The **execution** is set to scheduled. In this case the journey scheduling options are taken into account. (If you would want to send this as an instant message, similar to an instant email, you can set it to instant and it will be sent after a contact click event.)

5. The location in the app where the image must be displayed is defined by the **Category** field. The name of the category here must correspond to the name of the used container in the app.

6. **Sound** can be used to display a sound when a message is received and **Badge** works only on IOS to set the number of unread messages in the app icon.



When the journey is executed, the selected image will be shown in the app, in the Banner section



## 12 Mobile push reporting

Reporting on Mobile Push is available from the standard Interface reports. To access them, go to the journey containing the Mobile Push interface and select the interface from the tree on the left.



## 13 Q&A

### **Is Mobile push also available for in-house customers?**

Yes, it is available also for in-house customers, but the Mobile Push Services will be deployed on installations for all customers (Saas and In-house). From a security point of view, this means that the necessary ports must be opened on the customer's installation.

### **Do we have reporting on the mobile interactions?**

Mobile push component is an interface component. The mobile push reports can be found in the portal with the other interface reports. Status field in the interface table can show if the push has been sent, in error, or in process. In addition, all events related to the Mobile SDK are stored in the "PushEvents" table (push received, push open, click button, user custom events).

### **Can we track the user behavior within the app and use that info in Campaign?**

Yes, custom events can be created in the app, allowing to track the user behavior and send the information back to the platform to enrich the profiles.

Each custom event can be stored at different levels depending on how the environment has been set up. Information can be stored in the PushEvents tables or in one of the two audience lists containing the devices and the users.

### **Is Mobile push delivered free of charge?**

Sending the actual push message is free, but the Mobile solution is charged in the ES platform. Charges are based on the number of contacts.

### **Is it possible to be redirected from a push message to a specific section in the app?**

Yes, a specific button is available to open the app on a specific section.

### **Is it possible to track if an app user visits a webpage from a notification message?**

Yes, the standard Campaign mechanism can be used to track if a user visits the page. This mechanism is based on the tracking functionality of Campaign.

### **Is there multi-language support?**

When messages need to be sent in different languages, that needs to be handled in the journey itself by creating the message in the desired language. The language of the recipient can be based on the language of the device, which is available by default. A custom property can be added for the language of the app.

### **When using notifications of type URL, where must the URL be located?**

The moment the specified URL is accessible from the environment, its content is shown. It is of no importance where the URL is located on the Campaign server or somewhere else.

**Can we send notifications to anonymous users?**

The only thing required to be able to send notifications is a device-token and the device-id. The device-token is required to be able to send messages. The device-id is required to receive events.

**Can we send notifications to users that have not registered or logged on?**

Yes. All users that have installed the app can receive messages from Campaign.

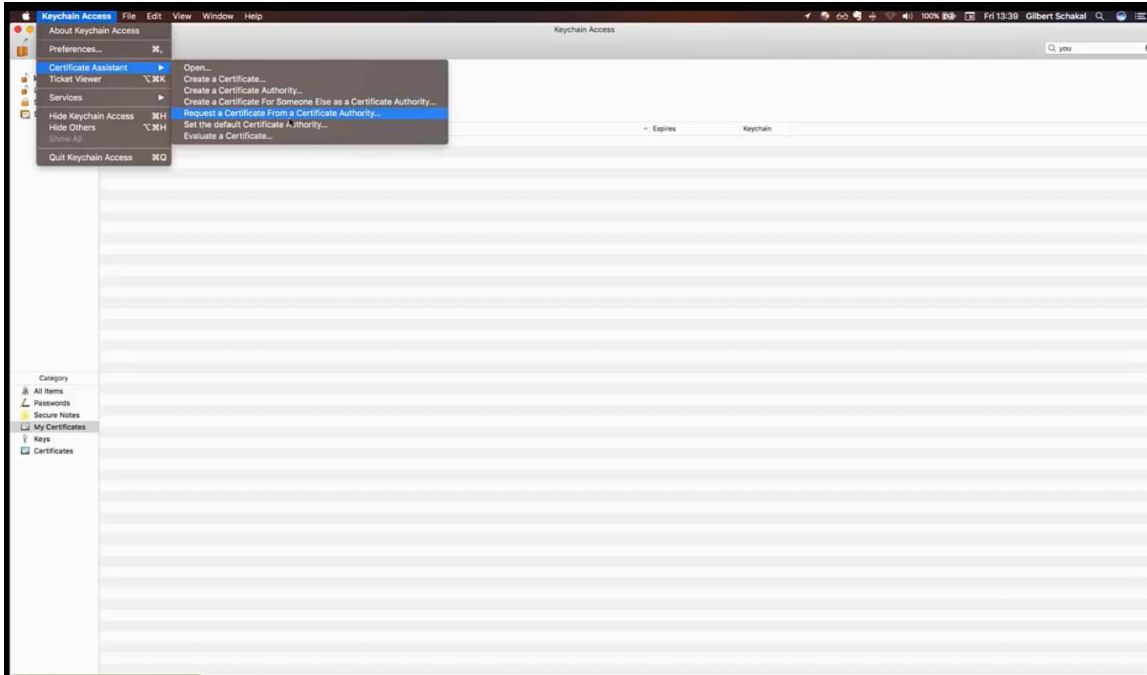
**Is it possible to measure the open rate of a push message?**

All events linked to the notification are stored in Campaign: push open, click button.

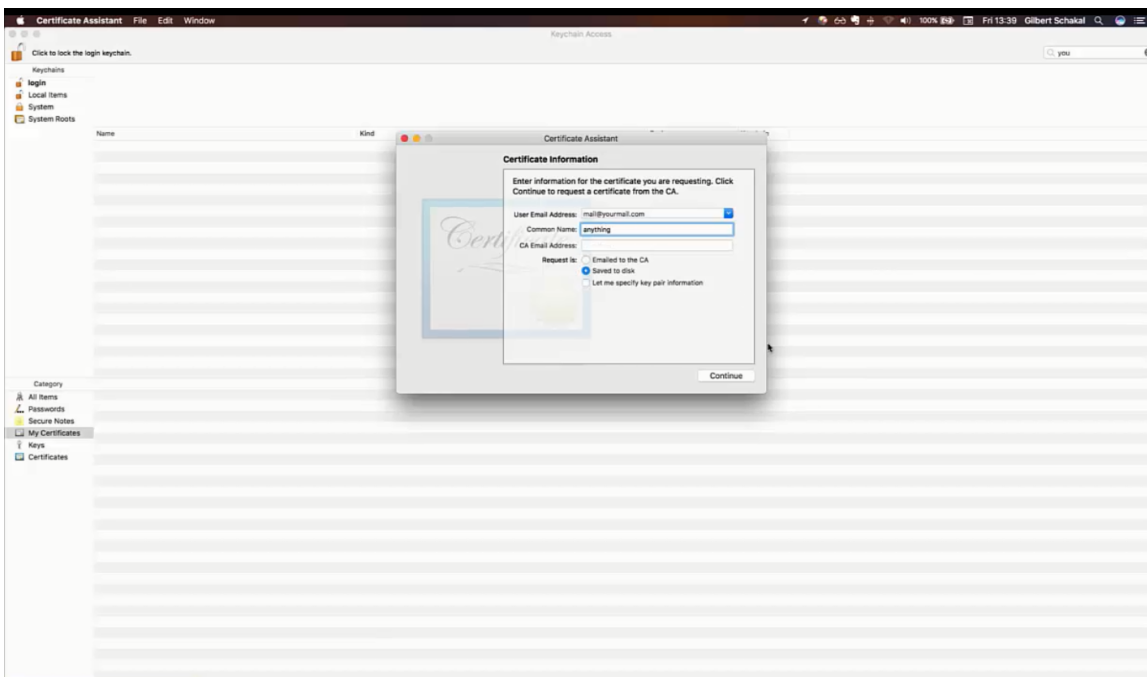
## 14 Addendum: The IOS Certificat

The first step is to **create the certification Request file**.

1. To do this, launch the KeyChain application on your Apple computer.
2. In the top menu bar, click KeyChain Access, then click Certificate Assistant, and select “Request a certificate from a Certificate authority”.

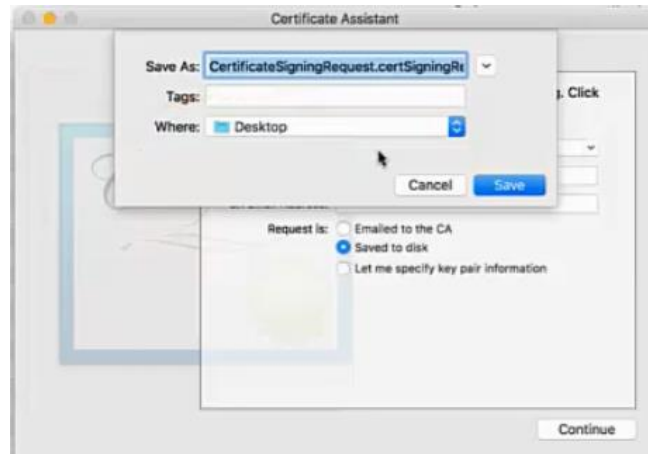


3. In the dialog that is displayed, enter some email data (this can be whatever you want) and make sure to indicate where the file must be saved. We will save it to disk.



4. Click **Continue**.

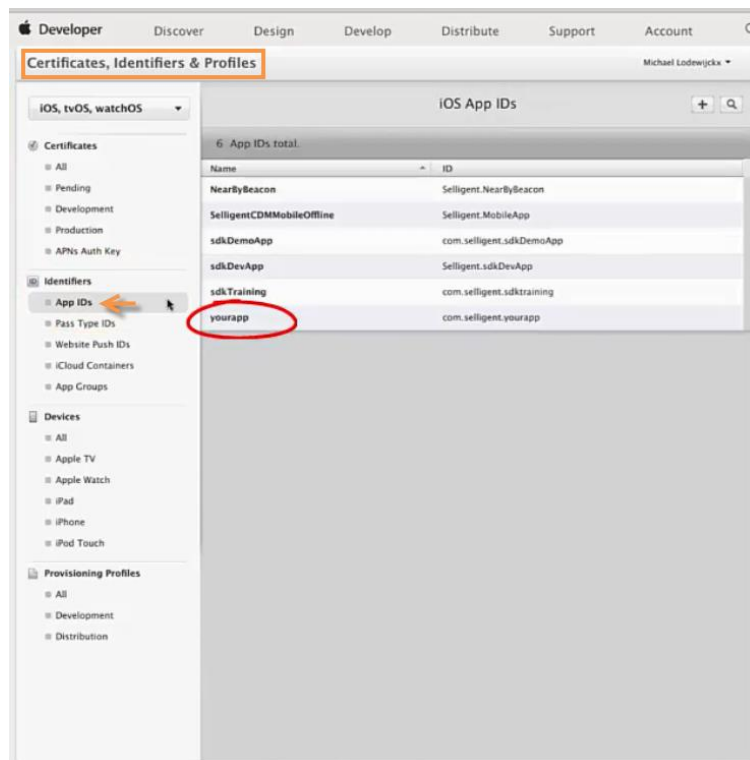
5. In the second dialog, provide a name for this file and use the extension dot .CertSigningRequest. Indicate where the file must be saved. We will save it to our Desktop here.



6. Click Save and then click Done.

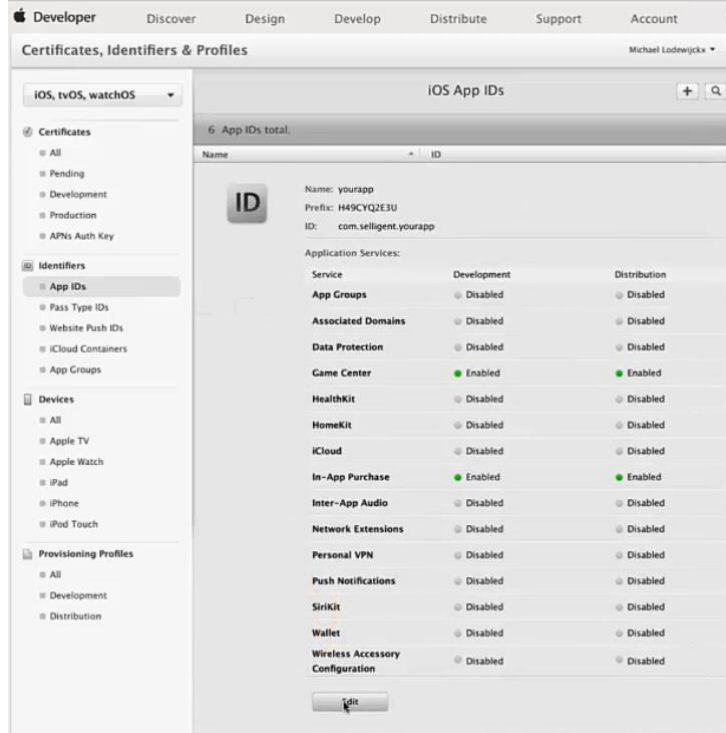
The next step is **to create the actual certificate.**

1. Go to the apple developer site, which is (<http://developer.apple.com>).
2. Log in to your developer account and go to Certificates, Identifiers & Profiles.
3. Under Identifiers, click App IDs and then look for your app.

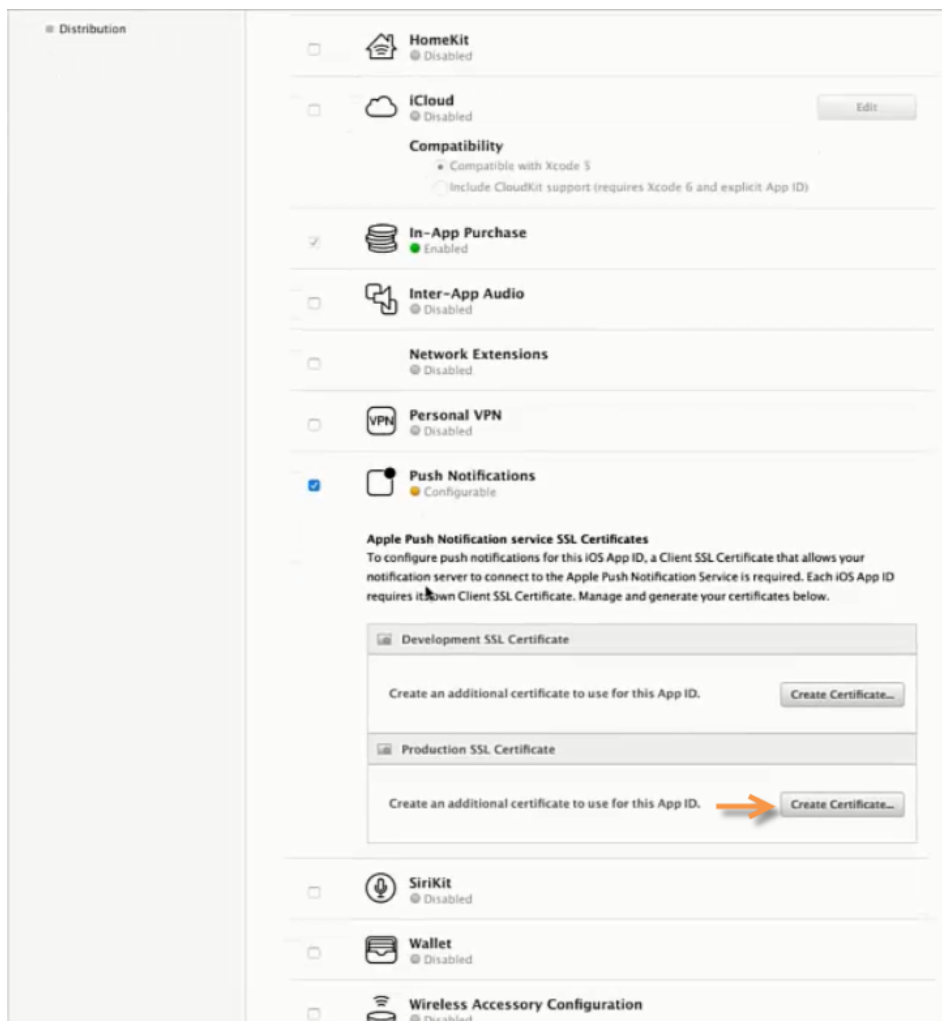


Ours is called 'yourapp'.

4. Click it and click Edit.



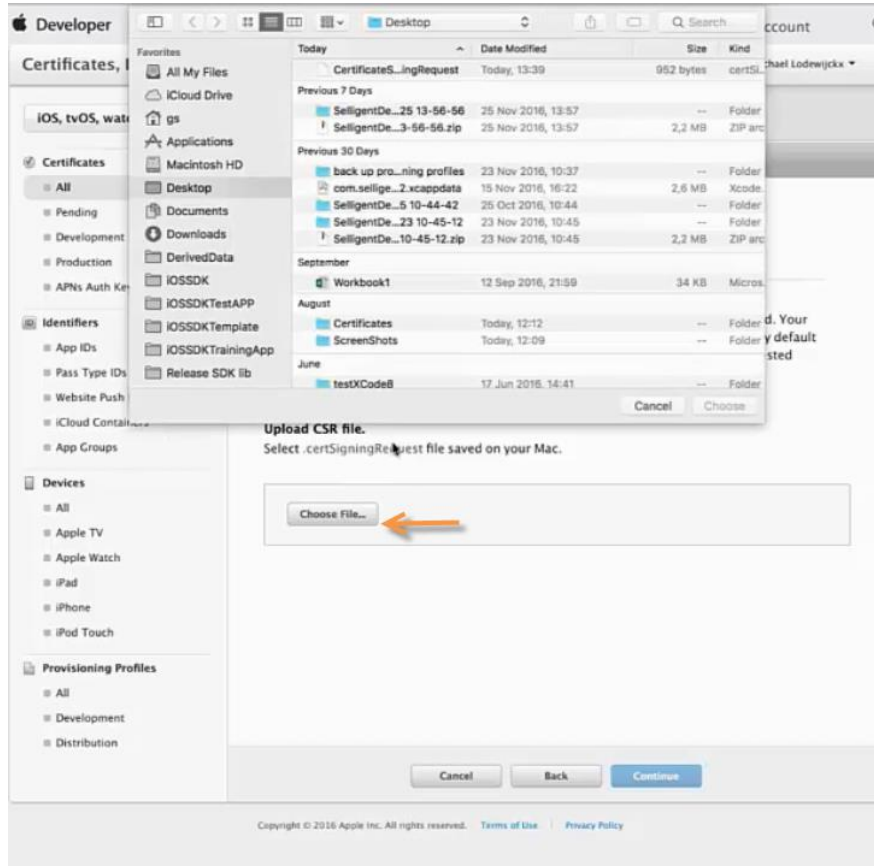
5. Make sure to check 'Push notifications':



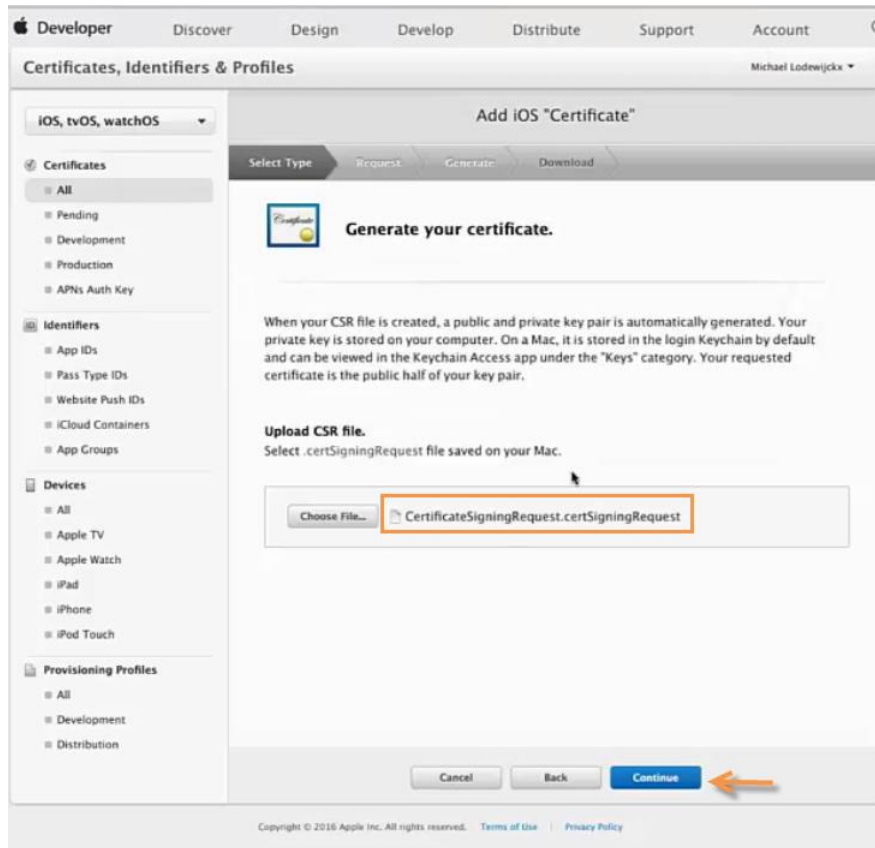
6. Next, click 'Create certificate'.

7. Then click Continue.

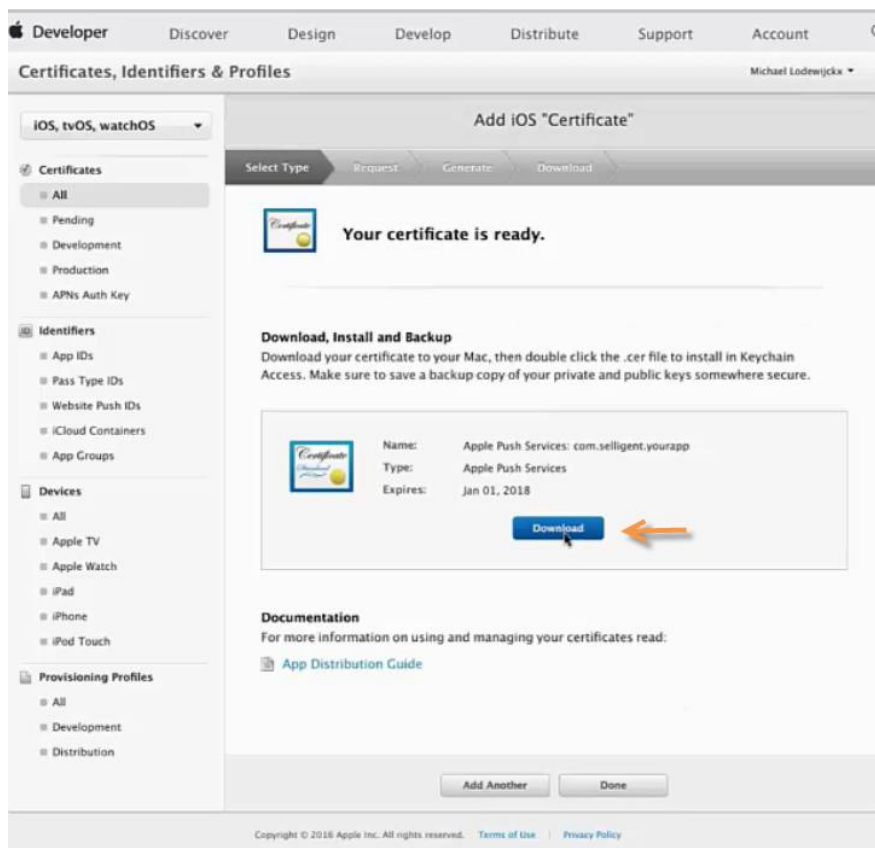
8. Next, you need to select the CSR file that we generated previously using the KeyChain tool. Select 'Choose file' and navigate to the Desktop to locate the file.



The name of the file is now displayed.

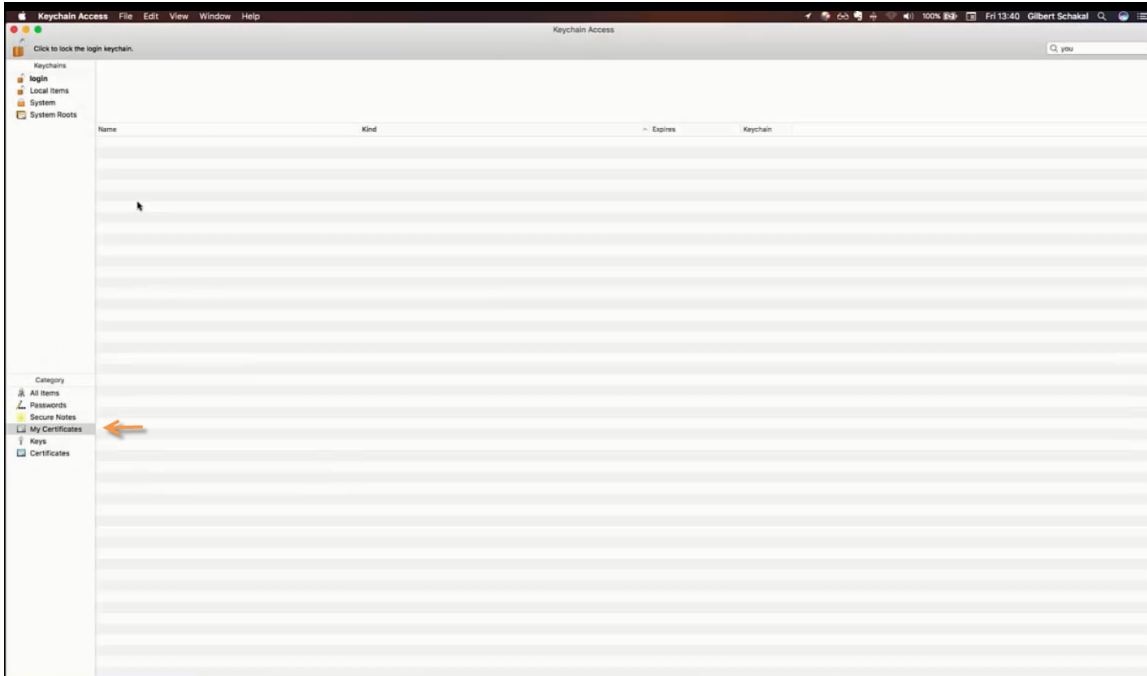


9. Click Continue and then click Download



10. When it's complete, click Done. The certificate is now available for the Certificates entry in the KeyChain application.

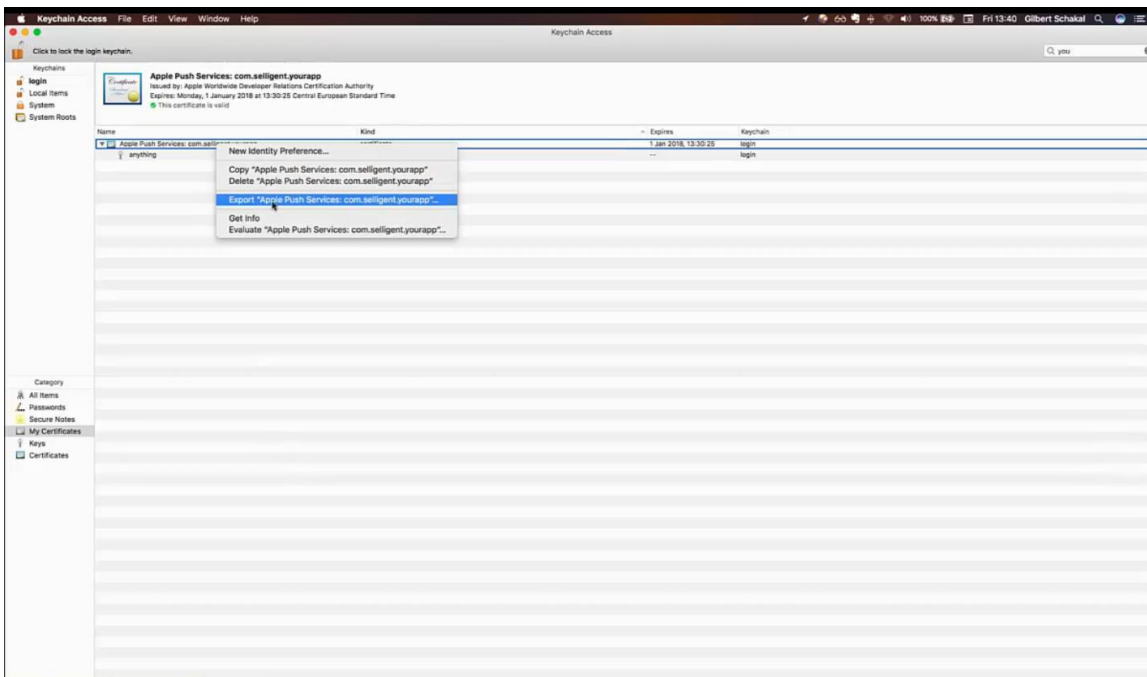
11. Next, return to the KeyChain Access application and Under Category, click the 'My certificates' entry. From here we will export the certificate.



12. From the folder where the certificate was downloaded, double-click the file. It will automatically be added here to the KeyChain Certificate folder.

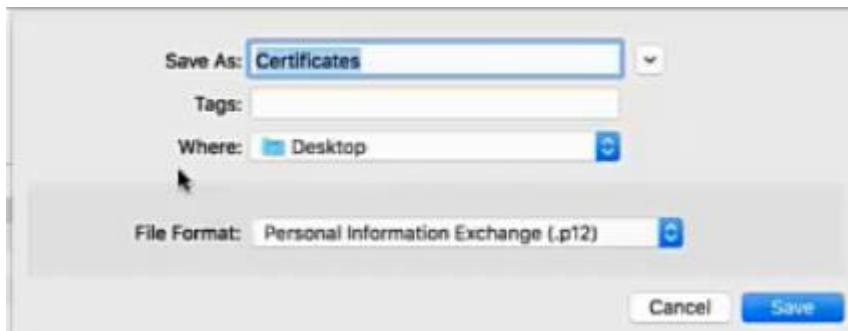
13. Once the certificate is displayed, make sure to expand the folder.

14. Right-click only the certificate (which is the first line, don't include the private key) and select from the menu 'Export Apple Push services....'





15. Give the export file a name and again indicate where the file must be exported to (the Desktop, in our case).



16. Enter a password for this certificate and click OK.

17. Next, click Allow to export the file.

The certificate is now ready to use.