

# Local Notifications Extension

## Contents

<b>Extension's Features</b>	<b>2</b>
<b>Setup</b>	<b>3</b>
<b>Quick Start Guide</b>	<b>4</b>
Requesting Permissions (iOS)	4
Creating/Cancel Notification	5
<b>Functions</b>	<b>7</b>
Permission Functions (iOS only)	7
LocalPushNotification_iOS_Permission_Status()	7
LocalPushNotification_iOS_Permission_Request()	7
General Functions	8
LocalPushNotification_Create(uid, seconds, title, message, data)	8
LocalPushNotification_Cancel(uid)	8
<b>Constants</b>	<b>10</b>
Permission Status	10

## Extension's Features

- Create and cancel local notifications
- Request for permission and check its status on iOS

## Setup

The Local Notifications Extension doesn't require any initial setup or configuration.

## Quick Start Guide

This section aims to deliver an easy and simple set of examples that should help you migrate your project from a previous version of the Local Notifications extension to its updated version. Local Notifications used to be part of the runtime and all the functions were named “**push\_\***”; these were now renamed to “**LocalPushNotification\_\***”.

### Requesting Permissions (iOS)

The first thing to look for when creating a GMS project using local notifications on iOS is requesting permissions. This is required and must be granted in order for the application to be able to use local notifications.

#### [Sample]

```
// In this extension requests for permissions require only a function call.  
LocalPushNotification_iOS_Permission_Request();
```

This function will display the request prompt and trigger a callback Social Async Event when the task is completed, where the `async_load` map would contain a “**type**” key with the value “LocalPushNotification\_iOS\_Permission\_Request”, a “**success**” key that can have a value of 1 or 0 (depending on whether or not the task was successful) and a “**value**” boolean with information on whether or not the request was granted.

## Creating/Cancel Notification

### [Function Mappings]

Even though not all old functions map perfectly into the new API, the following list should get you started with some of the changes (check the section below for more details):

- **LocalPushNotification\_Create(...)** ⇔ `push_local_notification(...)`
  - triggers the event of type “Notification\_Local” when notified ([more details](#))
- **LocalPushNotification\_Cancel(...)** ⇔ `push_cancel_local_notification(...)`

### [Old Version]

```
// In the previous version you would create a local push notification using:  
push_local_notification(fireTime, title, message, data);
```

This function wouldn't return any value but would trigger a callback Push Notification Async Event, where the `async_load` map would contain information regarding the push notification.

For canceling the local notification you would need to iterate through all the existing queued notifications until you find yours and cancel it, following the code below:

```
// First we would need to create a map to pass into the function.  
var map = ds_map_create();  
  
// We loop while result is valid  
var result = push_get_first_local_notification(map);  
while (result >= 0) {  
    // Check the map data  
    if (map[? "data"] == "some_value") {  
        // Cancel the notification  
        push_cancel_local_notification(result);  
    }  
    var result = push_get_next_local_notification(map);  
}
```

### [New Version]

```
// In the new version you can use the following code to create a new local  
// notification.  
LocalPushNotification_Create(uid, timeSeconds, title, message, data);
```

The main difference you can spot is that we are able to give an unique identification string to the notification, but the other parameters are the same. This function will not return any value but will trigger a callback Push Notification Async Event, where the `async_load` map will contain information regarding the push notification.

For canceling the local notification you just need to use the previously declared notification unique identifier:

```
LocalPushNotification_Cancel (uid);
```

Now the notification is no longer queued and won't trigger the finish callback.

# Functions

## Permission Functions (iOS only)

`LocalPushNotification_iOS_Permission_Status()`

Description: This function requests for the permission status. Note that it will not request the user for notification permission, instead it just requests the OS for the current permission status. For requesting permission use '[LocalPushNotification iOS Permission Request](#)'. This function doesn't return any value but it triggers an Async Push Notification Event callback when the task is completed.

Returns: N/A

Triggers: Social Async Event

type: "LocalPushNotification\_iOS\_Permission\_Status"

success: *{boolean}* whether or not the task succeeded.

value: [{Permission Status}](#) the current notification permission status.

`LocalPushNotification_iOS_Permission_Request()`

Description: This function requests permission to show notifications to the user; this is required under iOS and if no permission is given no notifications will be received. This function doesn't return any value but it triggers an Async Push Notification Event callback when the task is completed.

Returns: N/A

Triggers: Social Async Event

type: "LocalPushNotification\_iOS\_Permission\_Request"

success: *{boolean}* whether or not the task succeeded.

value: *{boolean}* whether or not the permission has been granted.

## General Functions

`LocalPushNotification_Create(uid, seconds, title, message, data)`

Description: This function creates a new push notification that will trigger after a given amount of time as passed with a given unique identifier. This unique identification string can be whatever the developer wants and is used to access the notification afterwards (cancel it). Notifications work at an OS level meaning they will still fire if your application is in the background or closed. This function triggers an Async Push Notification Event callback when the task is completed.

Params:

**uid** *{string}*: The notification's unique identification string.

**seconds** *{double}*: The number of seconds the notification should be delayed by.

**title** *{string}*: The title shown in the OS notification.

**message** *{string}*: The body text shown in the OS notification.

**data** *{string}*: The data string to be attached to the notification. Note that this string can be anything including an array/struct as long as it is in string format.

Returns: N/A

Triggers: Push Notification Async Event

type: *{string}* "Notification\_Local"

id: *{string}* the notification's unique identification string.

title: *{string}* the notification's title shown in the OS.

message: *{string}* the notification's body shown in the OS.

data: *{string}* the data string passed into the notification creation function.

`LocalPushNotification_Cancel(uid)`

Description: This function will cancel (unregister) a previously registered notification, so the notification will not trigger its callback anymore.

Params:

**uid** *{string}*: The notification's unique identification string.

Returns: N/A



## Constants

### Permission Status

LocalPushNotification\_iOS\_Permission\_Status\_Authorized: "Authorized"

LocalPushNotification\_iOS\_Permission\_Status\_Denied: "Denied"

LocalPushNotification\_iOS\_Permission\_Status\_NotDetermined: "NotDetermined"