iOS framework integration

ARENDSTRAAT 23 | POSTBUS 434 | T +31 (0) 35 64 66 300
1223 RE HILVERSUM | 1200 AK HILVERSUM | F +31 (0) 35 64 66 301
THE NETHERLANDS | THE NETHERLANDS | WWW.SPILGAMES.COM

Table of Contents

Purpose

- Build native application for iOS.
- Integrate them on Spil Games portals to benefit from their feature set.

Audience

This manual targets developers who build native applications for iOS and who want to use the features and services available through the Spil Games portal platform.

We assume you are familiar with the following concepts and tasks:

- Setting up an Xcode project.
- Requesting Apple certificates.
- Managing Apple certificates.

Get the package

You can download the latest version of the iOS framework package. It contains the following components:

- **Spil.framework**
- *Native sample*
- **Spil.bundle**
- **Spil Integration Helper.app**
- *Framework documentation* (if you prefer, you can read this documentation online).

⌦ **Spil.framework**: required if the application is *pure native*.

⌦ **Spil.bundle**: required if the application uses *Ads*.

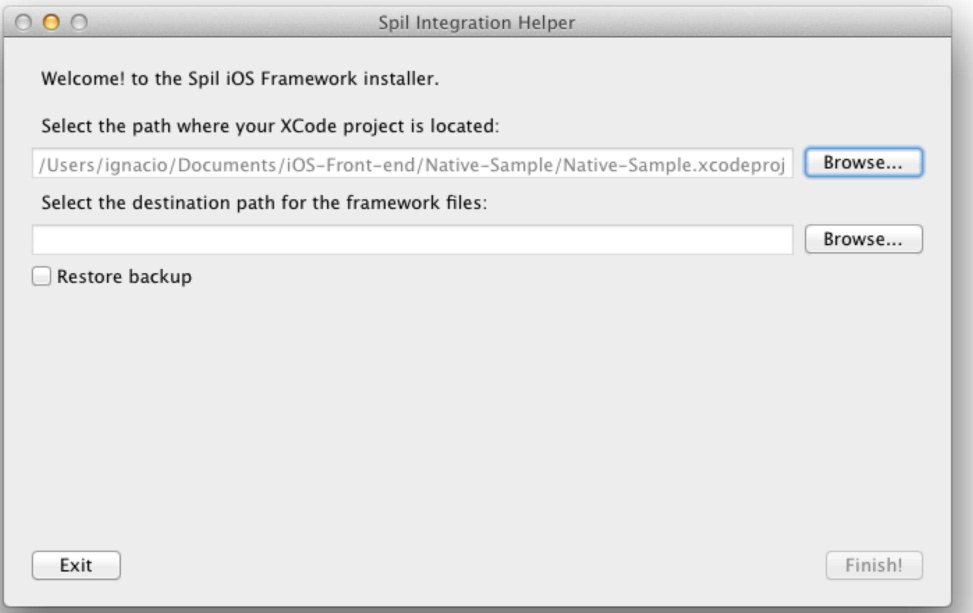
Pure native applications

Settings

To set up your Xcode project to use Spil Games iOS Framework:

- Run **Spil Integration Helper.app**, under the tools/builds folder.
- Follow the instructions displayed on the dialog window.

Select the location of your existing Xcode project:



Spil Integration Helper

Welcome! to the Spil iOS Framework installer.

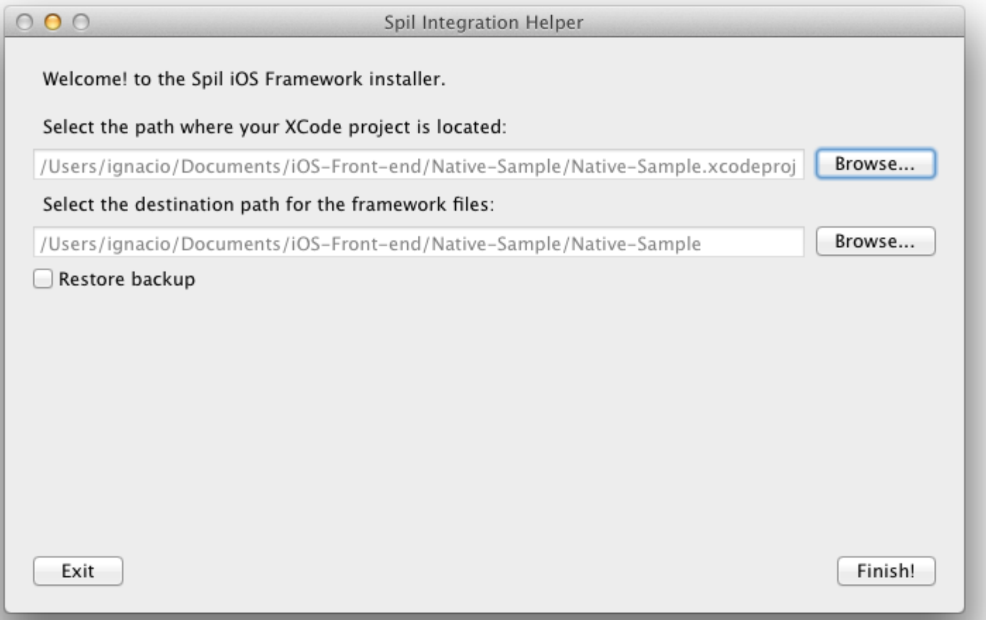
Select the path where your XCode project is located:

/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample.xcodeproj

Select the destination path for the framework files:

Restore backup

Then select the destination folder where you want to copy the files:



Welcome! to the Spil iOS Framework installer.

Select the path where your XCode project is located:

/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample.xcodeproj

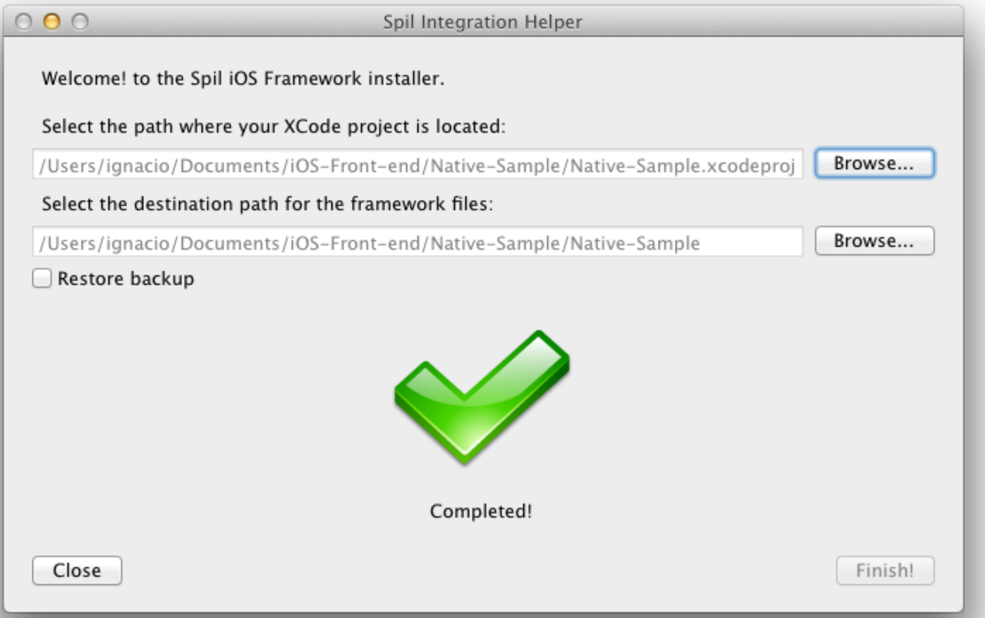
Select the destination path for the framework files:

/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample

Click **Finish!** to copy the files to the selected target location.

When the operation completes correctly, the following scenario should result:

- The Xcode project is configured with all required dependencies.
- A top-level group is created: **Spil**. You can move this group on the project tree later on to fit it within your project layout.


Spil Integration Helper

Welcome! to the Spil iOS Framework installer.

Select the path where your XCode project is located:

/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample.xcodeproj

Select the destination path for the framework files:

/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample

☐ Restore backup

Completed!

- If this operation does not work as expected, your project will be restored automatically: no parts of your original project will be altered or damaged.
- If the project integration completes correctly, but your code does not compile afterwards, you can restore it by selecting again the project file, and then by checking the **Restore backup** checkbox.

Open the selected Xcode project: the top-level group in the structure tree is called **Spil**. This group contains 3 files:

⌧ **Spil.framework**

⌧ **spilgames_default_settings.json**

⌧ **Spil.bundle**

spilgames_default_settings.json is the default settings file, in case the application cannot connect to the server to download them. In this JSON file you can define custom objects using the standard JSON format.

Coding

In your `UIApplicationDelegate` implementation, include the following line:

```
#import <Spil/Spil.h>
```

Now you can create the Spil object:

You create it in the `application:didFinishLaunchingWithOptions:` method using the `spilWithAppID:token:configs:` method. With this method you can create an instance that is directly linked to your environment in the Spil Games service platform.

Parameters

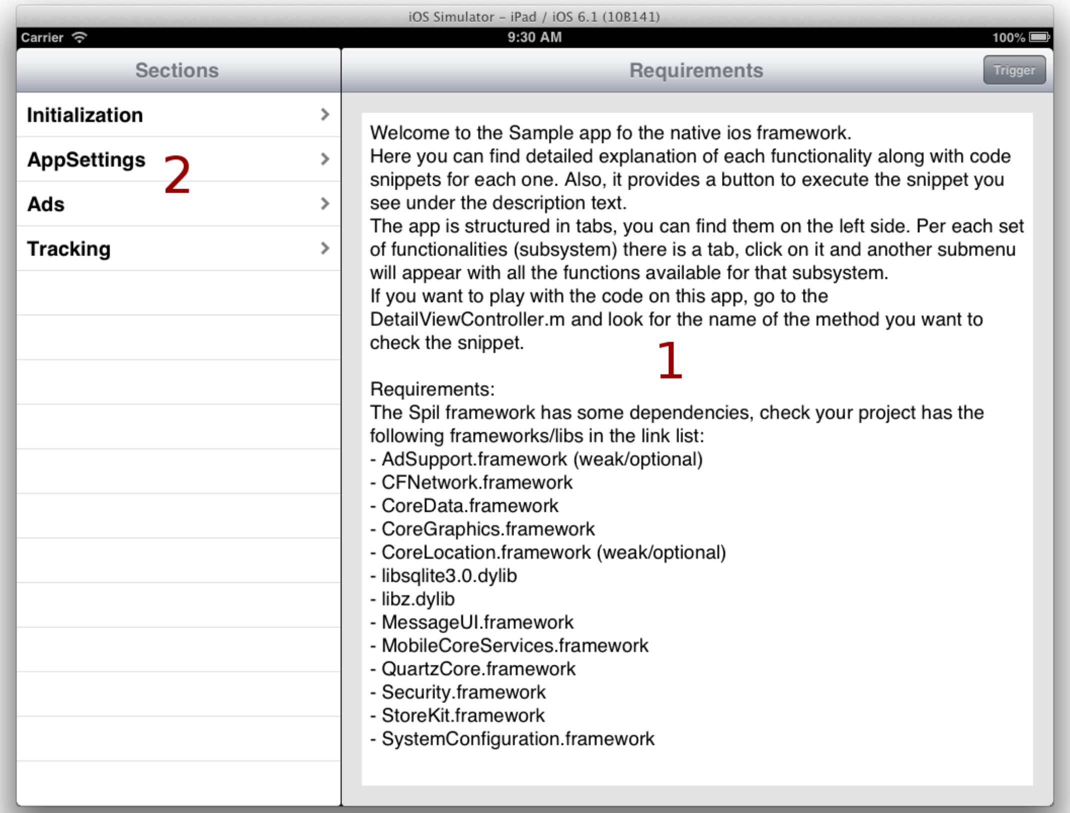
Initialize the environment with the `Initialize` method.

- `appID` and `authToken`: the first two parameters are provided by Spil Games when you get the package.
- `configs`: this is the last parameter. It is a dictionary, and it needs to include a number of defined keys (see Allowed keys further on in this document).

Delegates

Almost all the available services require a *delegate* to notify the app about specific actions or results. For further information about the delegates and for a more detailed description of the implementation, see the **Native-Sample** project: it's an iPad application that can run on the simulator; it provides explanations about the methods, their usage and the expected results. Moreover, you can modify the code and use it as a playground to experiment with the framework.

The most important file in the project for your experimentation is **DetailViewController.m**. This file includes a full implementation of the delegates; moreover, you can see the flow of the actions that take place during the operation.

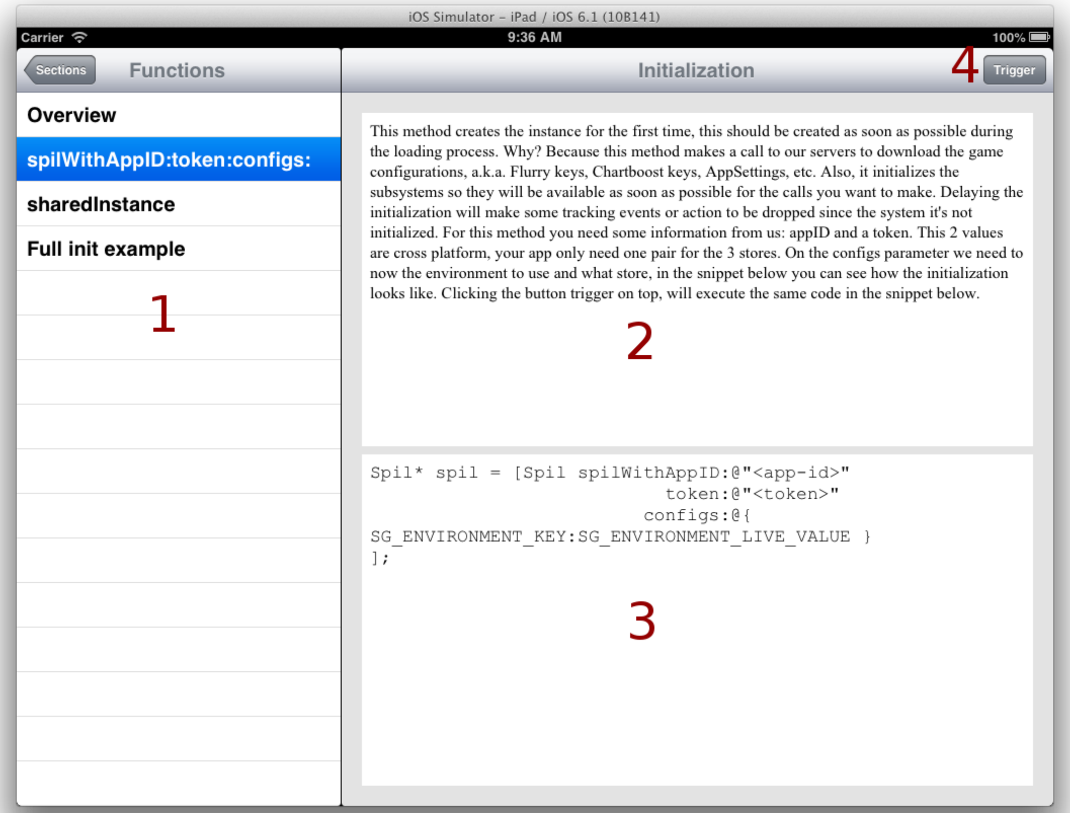


Welcome to the Sample app fo the native ios framework.
Here you can find detailed explanation of each functionality along with code snippets for each one. Also, it provides a button to execute the snippet you see under the description text.
The app is structured in tabs, you can find them on the left side. Per each set of functionalities (subsystem) there is a tab, click on it and another submenu will appear with all the functions available for that subsystem.
If you want to play with the code on this app, go to the DetailViewController.m and look for the name of the method you want to check the snippet.

Requirements:
The Spil framework has some dependencies, check your project has the following frameworks/libs in the link list:
- AdSupport.framework (weak/optional)
- CFNetwork.framework
- CoreData.framework
- CoreGraphics.framework
- CoreLocation.framework (weak/optional)
- libsqlite3.0.dylib
- libz.dylib
- MessageUI.framework
- MobileCoreServices.framework
- QuartzCore.framework
- Security.framework
- StoreKit.framework
- SystemConfiguration.framework

1. On the first screen, you see the system requirements for the framework (number *1* in the figure above.)
2. You can select different sections according to the functionality you are interested on, to inspect its functions (number *2* in the figure above).

When you select a section, you can see an overview of the corresponding functionality that is provided, and you can read an explanation of the available functions:



This method creates the instance for the first time, this should be created as soon as possible during the loading process. Why? Because this method makes a call to our servers to download the game configurations, a.k.a. Flurry keys, Chartboost keys, AppSettings, etc. Also, it initializes the subsystems so they will be available as soon as possible for the calls you want to make. Delaying the initialization will make some tracking events or action to be dropped since the system it's not initialized. For this method you need some information from us: appID and a token. This 2 values are cross platform, your app only need one pair for the 3 stores. On the configs parameter we need to now the environment to use and what store, in the snippet below you can see how the initialization looks like. Clicking the button trigger on top, will execute the same code in the snippet below.

```
Spil* spil = [Spil spilWithAppID:@"<app-id>"
                           token:@"<token>"
                         configs:@{
SG_ENVIRONMENT_KEY:SG_ENVIRONMENT_LIVE_VALUE }
];
```

1. On the left pane, select the function you are interested in.
2. On the upper right pane you can read the functional description of the method.
3. On the lower right pane there is a sample code snippet.
4. If you want, you can run the code snippet by clicking the **Trigger** button on the top right corner of the dialog window.

This app provides a simulation of the whole flow: *initialization, setup, use, notifications and error handling*.

Go to the section you want to check, select the corresponding function, then trigger the code snippet.

In your code, the flow should be similar to the following example:

- Initialize the **spil** object;
- `[[Spil sharedInstance] set<Subsystem>Delegate:<implementation-of-delegate>];`
- Make calls where appropriate, for example `[[Spil sharedInstance] <subsystem><Function>]`.

For example, this is the flow to show an interstitial ad:

- call: `spilWithAppID:token:configs:`
- call: `setAdsDelegate:`
- call: `adsNextInterstitial`.

Allowed keys

Key:	`SG_ENVIRONMENT_KEY`
Description:	sets the environment to work in: *development* or *production*.
Values:	`SG_ENVIRONMENT_DEV_VALUE`, `SG_ENVIRONMENT_LIVE_VALUE`
Mandatory:	YES

Key:	`SG_ENVIRONMENT_SETTINGS_URL_GET`
Description:	the URL the app settings are stored in. It must point to a JSON file.
Values:	a `NSURL` object.
Mandatory:	YES, if `SG_ENVIRONMENT_KEY` is set to `SG_ENVIRONMENT_DEV_VALUE`

Key:	`SG_APP_SETTINGS_POLL_TIME_KEY`
Description:	the refresh interval for the app settings. Values are in seconds. Used only if `SG_ENVIRONMENT_KEY` is set to `SG_ENVIRONMENT_DEV_VALUE` If no value is specified, the default setting is one (1) second.
Values:	float
Mandatory:	*No*

See also

- Doxygen-generated documentation: further explanations of the methods used to set the delegates for the App Settings and Ads sub-systems.
- Samples available on GitHub: further details about setup and configuration parameters for Unity-based application projects.