spilgames

iOS framework integration

spilgames

ARENDSTRAAT 23
1223 RE HILVERSUM
THE NETHERLANDS

POSTBUS 434
1200 AK HILVERSUM
THE NETHERLANDS

T +31 (0) 35 64 66 300
F +31 (0) 35 64 66 301
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 obtain the iOS framework package from Spil Games. It contains the following components:

- **Spil.framework**
- *Native sample*
- **Spil.bundle**
- **Spil Integration Helper.app**
- *Framework documentation*.

⌦ **Spil.framework**: required if the application is *pure native or* Unity-based.

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

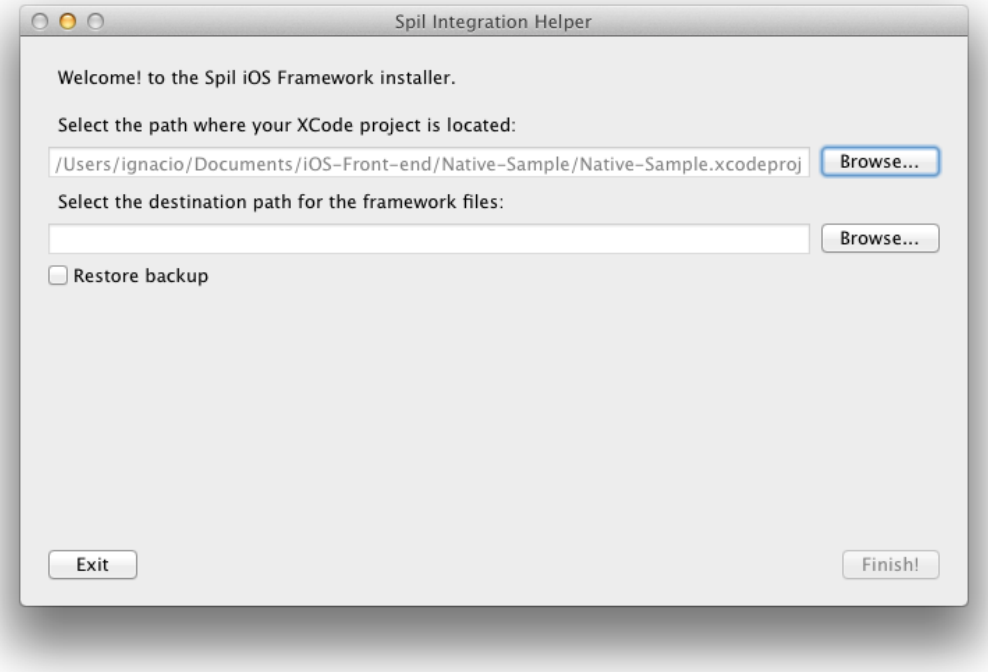
Pure native applications

Settings

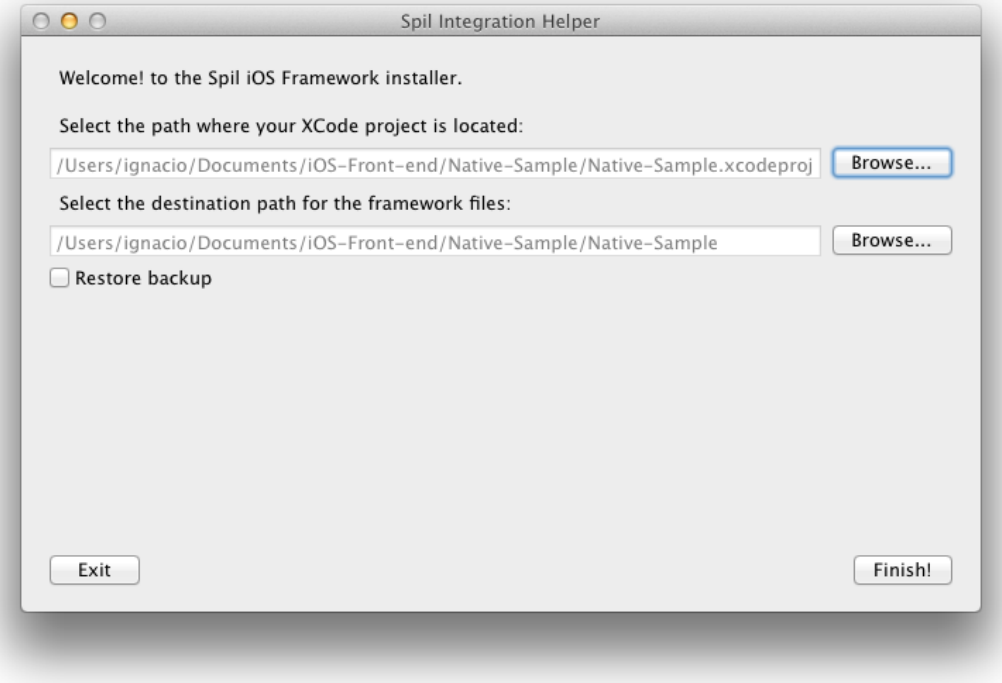
To set up your Xcode project to use Spil Games iOS Framework:

- 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:



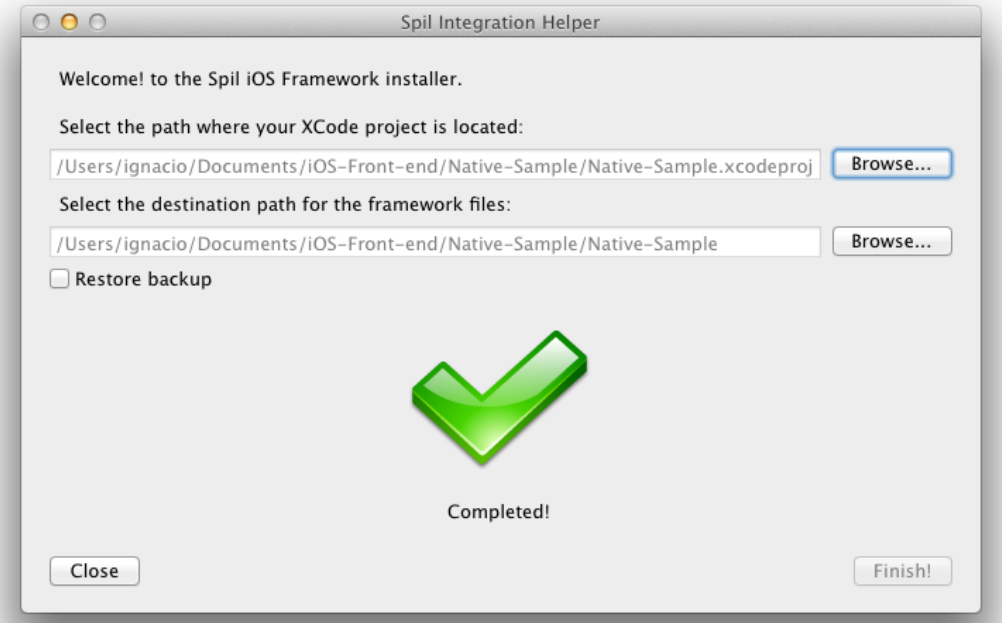
Then select the destination folder where you want to copy the files:



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 under **Spil**. You can move this group on the project tree later on to fit it within your project layout.



- 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. You can define custom objects in this JSON file.

We encourage using the standard JSON format, although the old JSON format is still supported.

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

After initializing the Spil object, you can set the delegates for the following sub-systems:

- App Settings,
- Ads, and
- A/B testing.

The last step is optional.

To do so, implement the protocols `AppSettingDelegate`, `AdsDelegate` and/or `ABTestDelegate` in the appropriate classes. You can choose these classes based on your convenience. In the example below, they are implemented in the `AppDelegate` class.

To set the proper delegates to the Spil object, call the following methods:

```
[spil getSettings:delegateImplementation],

[spil getAds:delegateImplementation] and
```

`[spil getABTest:delegateImplementation]`, respectively.

Now your method should look like this:

```
-(BOOL) application:(UIApplication *)application
     didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Your initialization code goes here
    // Start Spil object initialization
    Spil* spil = [Spil spilWithAppID:@"<spil-app-id>"
                                                token:@"<spil-auth-token>"
                                              configs:@{
              SG_ENVIRONMENT_KEY:SG_ENVIRONMENT_LIVE_VALUE,
              SG_TRACKING_ID_KEY:@"<tracking-app-ids> ",
    }];
    // End Spil object initialization
    [spil getSettings:self];
    [spil getAds:self];
    [spil getABTest:self]; // Optional

    return YES;
}
```

After initializing the Spil object, the settings for the framework features are downloaded. Depending on the configured delegates, the appropriate actions are triggered.

If no Internet connection is available, the app settings are retrieved from:

- Either the **spilgames_default_settings.json** file in the bundle, or
- The latest version retrieved from the server.

The remaining features (i.e. Ads, and A/B testing) are disabled.

Your delegate should look like this:

```
//App settings delegate
-(void) appSettingsDidLoad:(NSDictionary*)as{
       NSLog(@"%@",as);
}
-(void) appSettingsDidFailWithError:(NSError*)error{
       NSLog(@"error!: %@", error);
}
-(void) appSettingsDidStartDownload{
      NSLog(@"Download started, maybe trigger something?");
}

//Ads delegate
-(void) adDidStart{
       NSLog(@"ads system started");
}
-(void) adDidFailToStart:(NSError*)error{
       NSLog(@"ads system failed: %@",error);
}
-(void) adWillAppear{
       NSLog(@"ad will appear");
}
-(void) adDidAppear{
       NSLog(@"ad appeared");
}
```

```
-(void) adDidFailToAppear:(NSError*)error{
        NSLog(@"ads failed to appear: %@",error);
}

-(void) adPopupDidDismiss{
        NSLog(@"ad dismissed");
}
-(void) adMoreGamesWillAppear{
        NSLog(@"more games will appear");
}
-(void) adMoreGamesDidAppear{
        NSLog(@"more games appeared");
}
-(void) adMoreGamesDidFailToAppear:(NSError*)error{
        NSLog(@"more games failed to appear: %@",error);
}
-(void) adMoreGamesDidDismiss{
        NSLog(@"more games were dismissed");
}

//A/B Test delegate
-(void) abtestSessionDidStart{
       NSLog(@"ab test session started");
       // It's a good idea to retrieve the test info as soon as possible
       [[Spil sharedInstance] abtestGetTestDiff];
}
-(void) abtestSessionDidEnd{
}
-(void) abtestSessionDiffReceived:(NSArray*)diffs{
       NSLog(@"info received! %@",diffs);
       // Parse the diffs.
}
```

You can find explanations of these methods in the Doxygen-generated documentation, available in the documentation folder included with the package.

JSON format (Deprecated)

App Settings require additional metadata in the JSON format to render the front-end admin. This makes the format slightly more verbose than usual.

Below is the BNF definition of the format.

```
<JSON> ::= { <pair> [, <pair>]* }

<pair> ::= "key": <obj>

<obj> ::=  { "type":"string", "value": <string> } |

           { "type":"number", "value": [+|-]<number> }  |

           { "type":"bool", "value": true|false } |

           { "type":"array", "value": <obj> }
```

Sample format

```
{

       "color":{"type":"number","value":"0x00ff00"},

       "rotation":{

              "type":"array",

              "value":{

                           "x":{"type":"number","value":"0"},

                           "y":{"type":"number","value":"0"},

                           "z":{"type":"number","value":"0"}

              }

       },

       "hide";{"type":"bool","value": false}

}
```

Note: when the delegate/listener is called back, you need to define the following data in both native and Unity-based apps to correctly parse the settings:

- Format
- Name
- Types.

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, available on GitHub: 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.