# iOS framework integration

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

# 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 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 these components:

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

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

⌧ **Spil.unitypackage**: required if the application is only Unity-based.

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

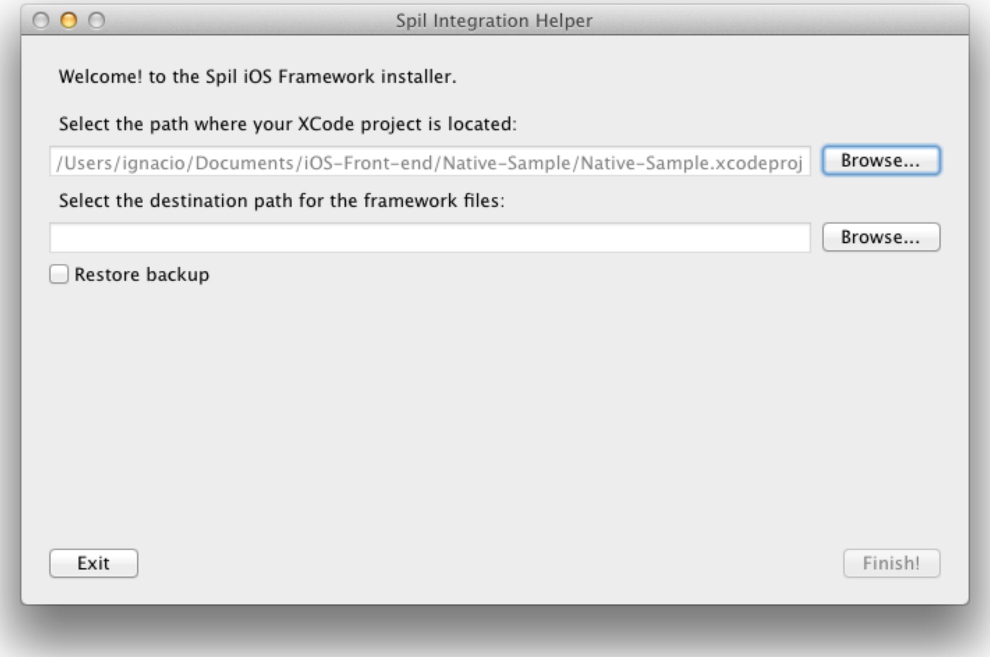
# Pure native applications

## Settings

To set up your Xcode project to use Spil Games iOSFramework:

- Run **Spil Integration Helper.app**
- Follow the instructions displayed on the dialog window.

Select the location of your existing Xcode project:



Welcome! to the Spil iOS Framework installer.

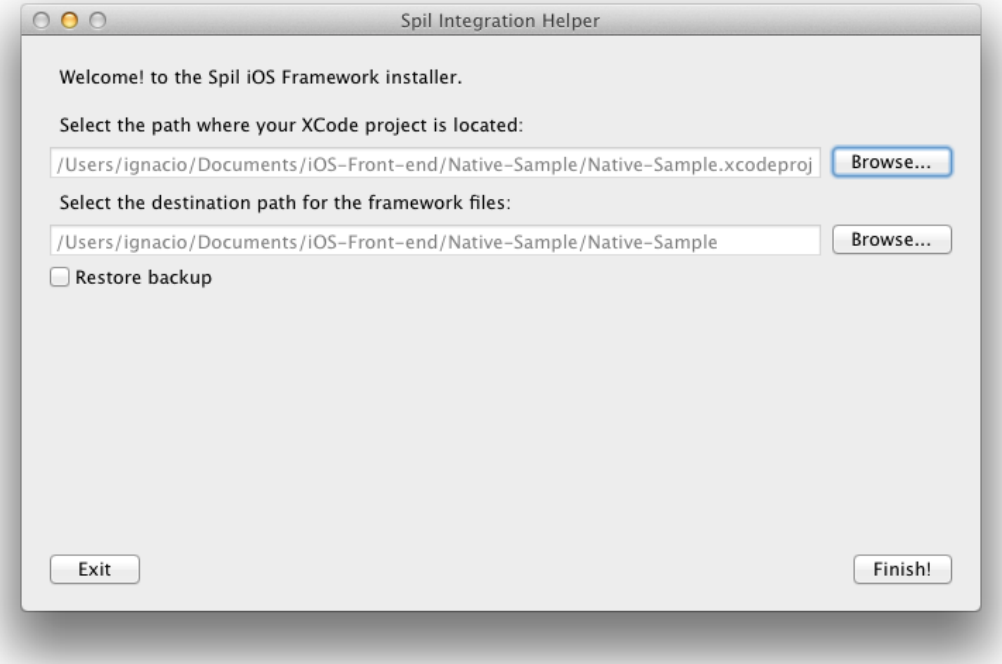
Select the path where your XCode project is located:

/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:



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:

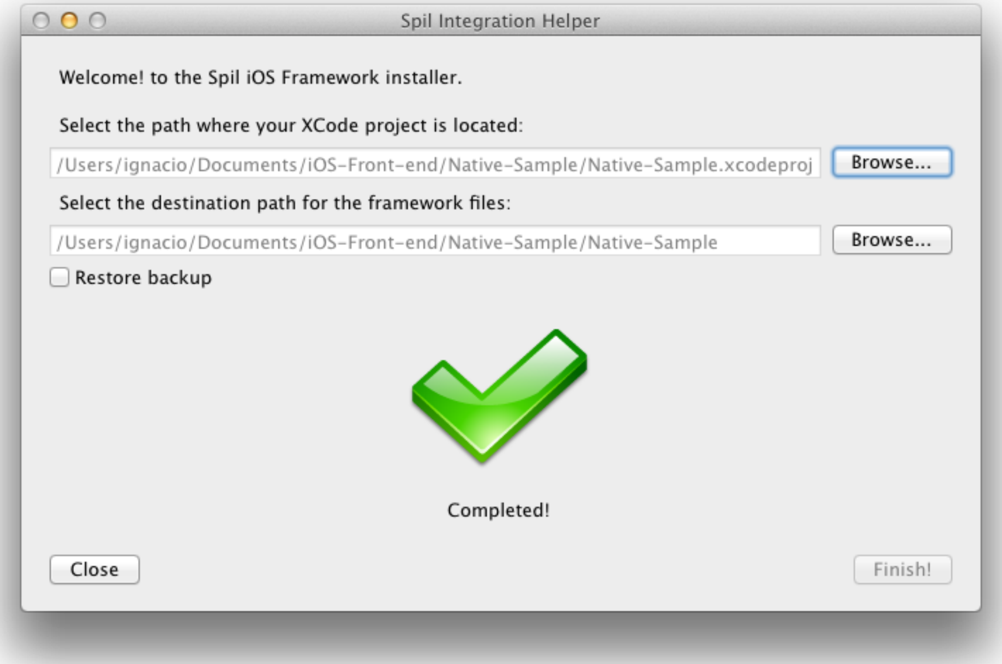
/Users/ignacio/Documents/iOS-Front-end/Native-Sample/Native-Sample

□ Restore backup

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

Moreover:

- 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.



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

- 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 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. The old JSON format is still supported, but we encourage the usage of the standard JSON format from now on.

# Coding

In your `UIApplicationDelegate` implementation, include the following:

```
#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. This method allows creating 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 App Settings, Ads and A/B testing sub-systems. 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 these 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
    //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 the **spilgames_default_settings.json** file in the bundle or from the latest version retrieved from the server. The remaining features (i.e. Ads, and A/B testing) they are disabled.

Your delegate should look like this:

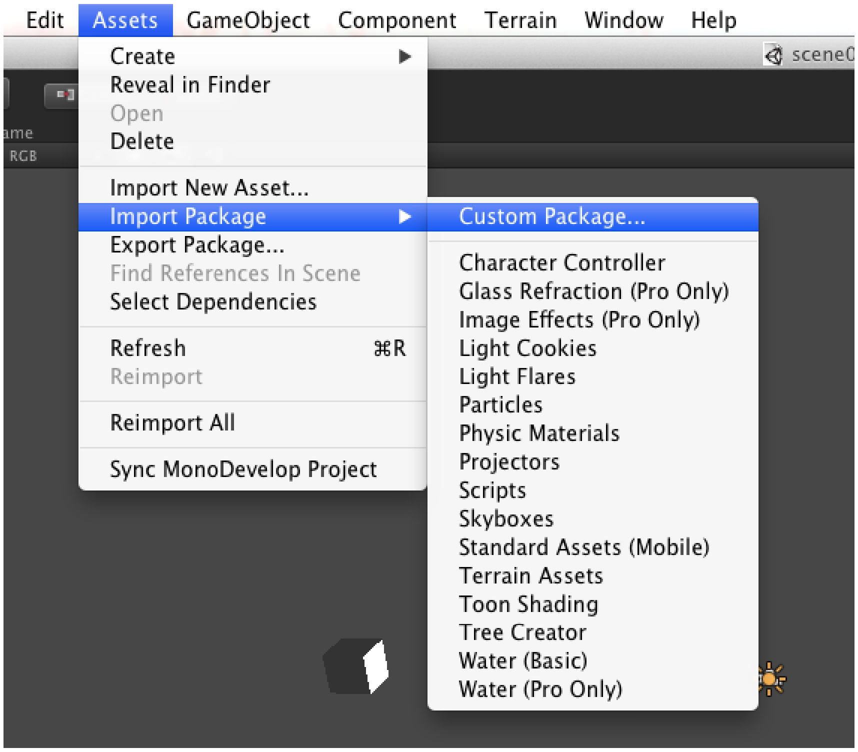
```
//App settings delegate
-(void) appSettingsDidLoad:(NSDictionary*)as{
       NSLog(@"%@",as);
}
-(void) appSettingsDidFailWithError:(NSError*)error{
       NSLog(@"error!: %@", error);
}
-(void) appSettingsDidStartDownload{
       NSLog(@"Download started, maybe trigger something?");
}

//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 on the documentation folder in the package.

# Unity-based applications

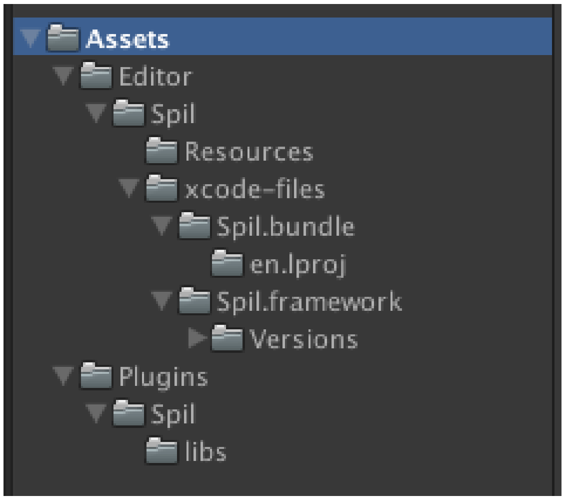
## Settings

1. After setting up your Unity project, import **spil.unitypackage**, as shown below:



2. In the **spil.unitypackage** there is a folder with:
   - The Unity plug-in
   - spilgames_default_settings.json
   - Post processing plugin
   - XCode files (Spil.bundle and Spil.framework)

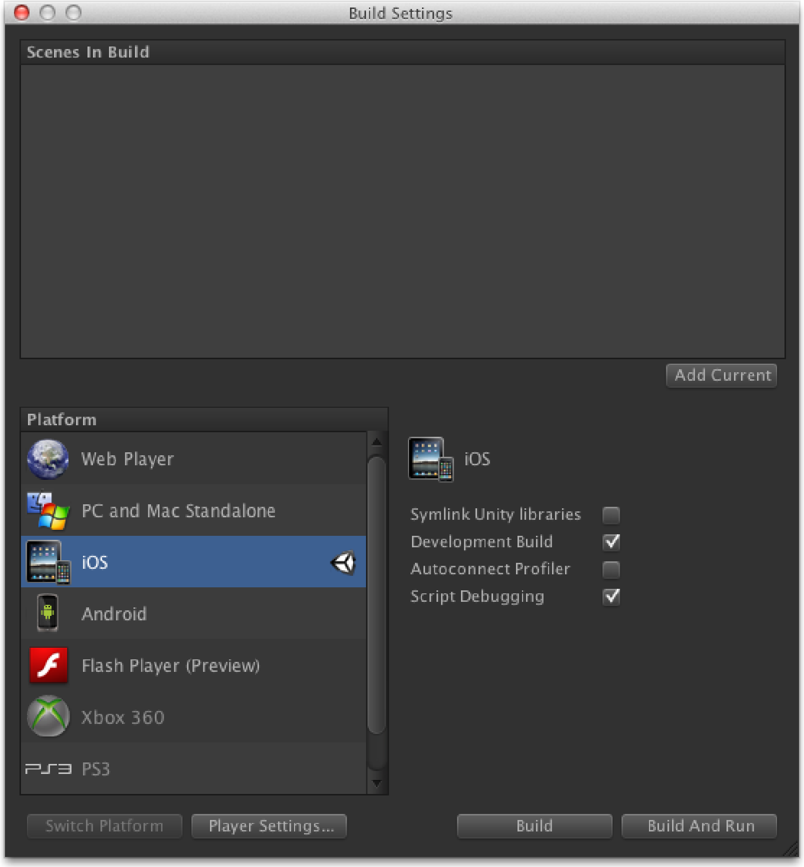
Now your project should look like this:



```
Assets
  Editor
    Spil
      Resources
      xcode-files
        Spil.bundle
          en.lproj
        Spil.framework
          Versions
  Plugins
    Spil
      libs
```

## Generate the Xcode project

1. From the menu, select the **Build settings…** command.
2. Under **Platform**, select **iOS**, as shown below:



Build Settings

Scenes In Build

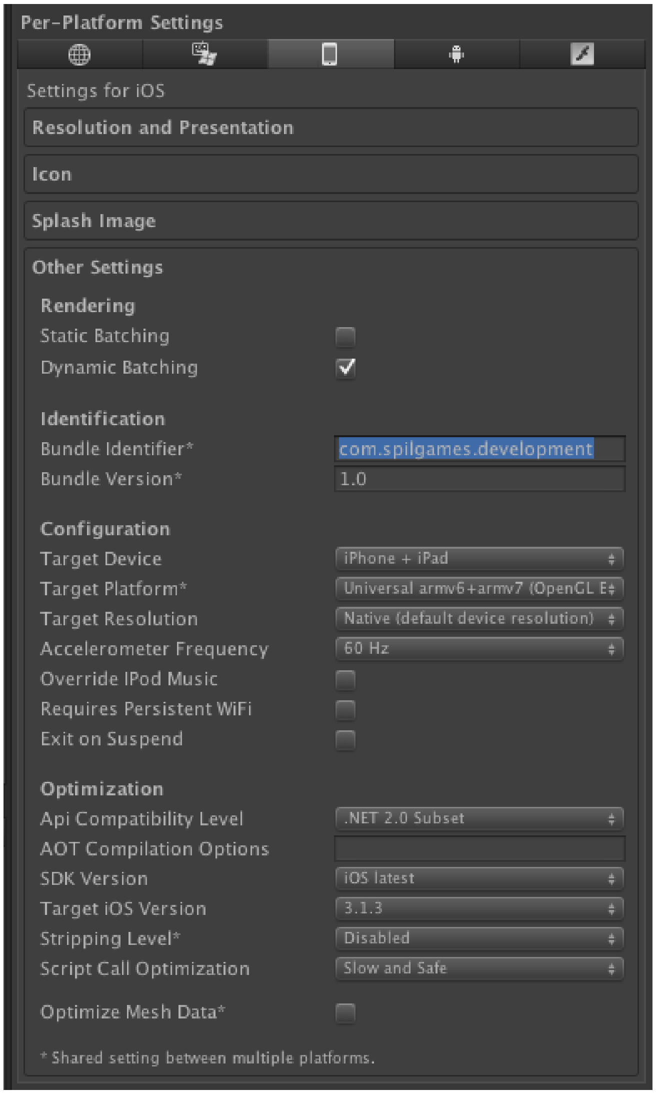
Platform

iOS

Symlink Unity libraries □
Development Build ☑
Autoconnect Profiler □
Script Debugging ☑

# Configure the certificates for your iOS application

1. Click the **Player settings…** button. A panel opens.
2. Set up the general fields like icons, orientation, preferred, and so on.
3. Select the target platforms, the available architectures, the target devices, and the iOS level.
4. In the **Bundle Identifier** field, set the name of the application you want to build, as shown below:



Per-Platform Settings

Settings for iOS

Resolution and Presentation

Icon

Splash Image

Other Settings

Rendering

Static Batching ☐

Dynamic Batching ☑

Identification

Bundle Identifier* com.spilgames.development

Bundle Version* 1.0

Configuration

Target Device iPhone + iPad

Target Platform* Universal armv6+armv7 (OpenGL E

Target Resolution Native (default device resolution)

Accelerometer Frequency 60 Hz

Override IPod Music ☐

Requires Persistent WiFi ☐

Exit on Suspend ☐

Optimization

Api Compatibility Level .NET 2.0 Subset

AOT Compilation Options

SDK Version iOS latest

Target iOS Version 3.1.3

Stripping Level* Disabled

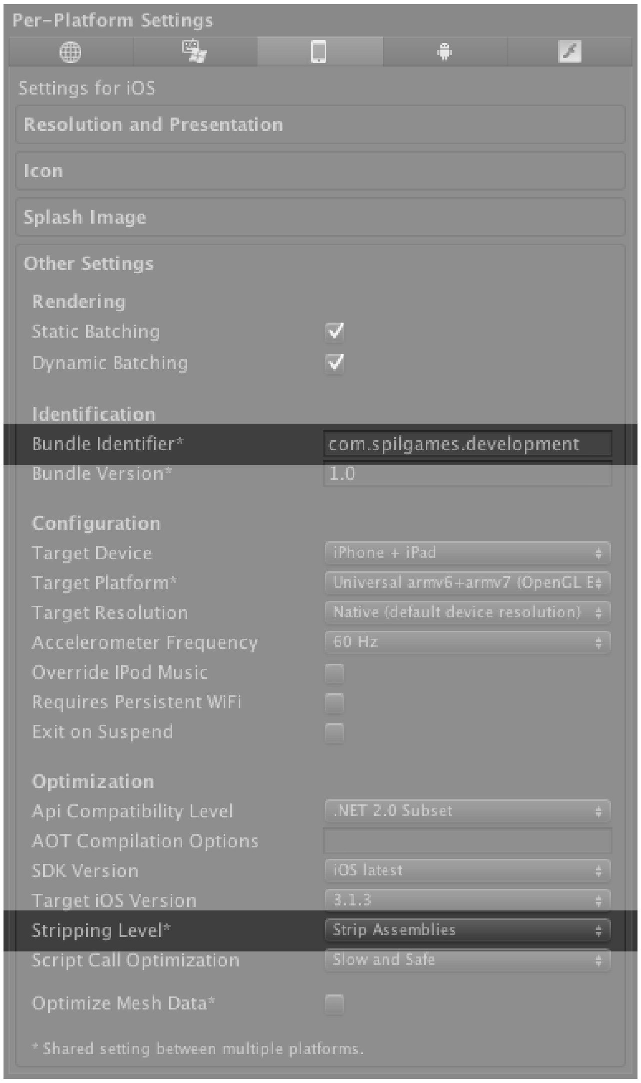
Script Call Optimization Slow and Safe

Optimize Mesh Data* ☐

* Shared setting between multiple platforms.

☑ **Note**: the application name must be *exactly the same* as the corresponding name in the provisioning profile.

5. Set **Stripping Level** to **Strip Assemblies**, as shown below:



Per-Platform Settings

Settings for iOS

Resolution and Presentation

Icon

Splash Image

Other Settings

Rendering

Static Batching ☑

Dynamic Batching ☑

Identification

Bundle Identifier* com.spilgames.development

Bundle Version* 1.0

Configuration

Target Device iPhone + iPad

Target Platform* Universal armv6+armv7 (OpenGL E

Target Resolution Native (default device resolution)

Accelerometer Frequency 60 Hz

Override IPod Music □

Requires Persistent WiFi □

Exit on Suspend □

Optimization

Api Compatibility Level .NET 2.0 Subset

AOT Compilation Options

SDK Version iOS latest

Target iOS Version 3.1.3

Stripping Level* Strip Assemblies

Script Call Optimization Slow and Safe

Optimize Mesh Data* □

* Shared setting between multiple platforms.

6. Select **Build**. This generates the Xcode project that will compile the final application.
7. In the open Xcode project, the following items should be included in the project:
    - **Spil.framework**
    - the resource bundle (**Spil.bundle**)
    - the **spilgames_default_settings.json** file.

    ✓ **Notes**:

    - Use **spilgames_default_settings.json** and the resource bundle as a reference to the corresponding files in the Unity folder.
    - Make sure the following frameworks are included as well, as shown below:
        - **UIKit.framework**
        - **QuartzCore.framework**

- **SystemConfiguration.framework**
- **CoreLocation.framework [weak link]**
- **libsqlite3.0.dylib**
- **CFNetwork.framework**
- **CoreGraphics.framework**
- **AdSupport.framework [weak link]**
- **AssetsLibrary.framework**
- **CoreVideo.framework**
- **CoreMedia.framework**
- **MobileCoreServices.framework**
- **StoreKit.framework [weak link]**
- **AVFoundation.framework**
- **CoreTelephony.framework**

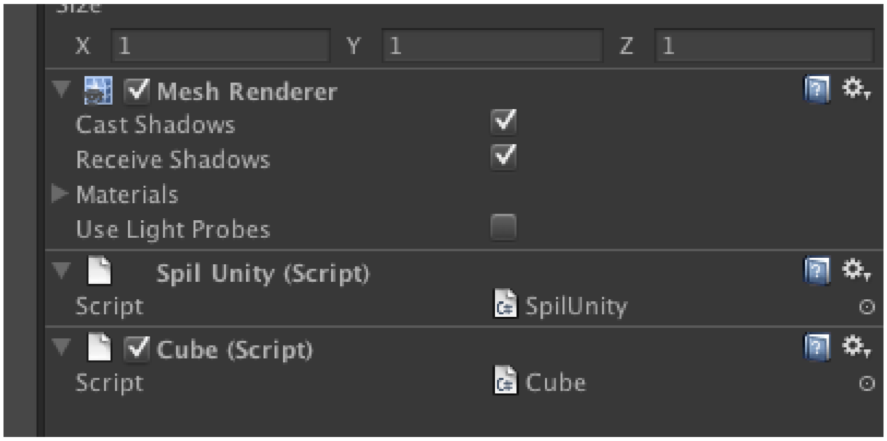
8. Set the flags for the compiler and specify them, if they are not defined:
   8.1 In the project **Build Settings** tab, under the **Setting** category, expand the **Other C Flags** node.
   8.2 Add the C flag `-mno-thumb` to the list.
   8.3 In the project **Build Settings** tab, under the **Setting** category, expand the **User-defined** node.
   8.4 Add the user-defined setting `GCC_THUMB_SUPPORT` and set its value to `NO`.

# Coding

1. Include the Spil namespace to your project and to the JSON parsing API. This works in the same way as in pure native applications:

   - `using Spil;`
   - `using LitJson;`

2. Add the **SpilUnity.cs** script to an object.



Size

X 1 Y 1 Z 1

☑ Mesh Renderer

Cast Shadows ☑

Receive Shadows ☑

Materials

Use Light Probes □

Spil Unity (Script)

Script SpilUnity

☑ Cube (Script)

Script Cube

The camera is an ideal object because it behaves like a singleton, but any other object will work as well.

## Parameters

Initialize the environment with the **`Initialize`** method in the same way as you do for pure native applications (see Pure native applications and Coding for pure native applications earlier in this document).

- **`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 struct, and it needs to include a number of defined keys. The keys are the same you use for pure native applications (see Allowed keys further on in this document).

The following is an example of configuration settings used to initialize the environment (see following page):

```
using Spil;
using LitJson;
public class Cube : MonoBehaviour, SpilAppSettingsListener, SpilAdsListener {
        SpilUnity instance;
        Vector3 rotation = Vector3.zero;
        // Use this for initialization
        void Start () {
                instance = (SpilUnity)GetComponent<SpilUnity>();
                SpilSettings configs;
                configs.SG_ENVIRONMENT_KEY = enviroment.SG_ENVIRONMENT_LIVE_VALUE;
                configs.SG_TRACKING_ID_KEY="<tracking-app-ids>";

                instance.Initialize("<spil-app-id>","<spil-auth-token>",configs);
                instance.GetSettings(this);
                instance.StartAds(this);
                instance.GetABTest(this);
        }
}
```

☑ **Notes**:

- You need to implement the `SpilAppSettingsListener` and `SpilAdsListener` and optionally `SpilABTestListener` interfaces. You can implement them wherever appropriate.
- You need to pass references to the Spil object using the `GetSettings(listenerImplementation)`, `GetAds(listenerImplementation)` and `GetABTest(listenerImplementation)` methods, respectively.

Refer to the following example:

```
//App settings listener
public void AppSettingsDidLoad(JsonData data){
        renderer.material.color = new Color(
(((int)data["color"]>>16)&0x000000ff)/255.0f,
        (((int)data["color"]>>8)&0x000000ff)/255.0f,
        (((int)data["color"])&0x000000ff)/255.0f);

        rotation = new Vector3((int)data["rotation"]["x"],
                               (int)data["rotation"]["y"],
                               (int)data["rotation"]["z"]);
}
public void AppSettingsDidFailWithError(string error){
        Debug.LogError(error);
}
//Ads listener
public void AdDidStart(){
        Debug.Log("started");
}
public void AdDidFailToStart(string error){
        Debug.LogError(error);
}
public void AdWillAppear(){
        Debug.Log("will appear");
}
public void AdDidAppear(){
        Debug.Log("appeared");
}
public void AdDidFailToAppear(string error){
        Debug.LogError(error);
}
public void AdPopupDidDismiss(){
        Debug.Log("popup was dismissed");
}

public void AdMoreGamesWillAppear(){
        Debug.Log("more games will appear");
}
public void AdMoreGamesDidAppear(){
        Debug.Log("more games appeared");
}
public void AdMoreGamesDidFailToAppear(string error){
        Debug.LogError(error);
}
public void AdMoreGamesDidDismiss(){
        Debug.Log("more games were dismissed");
}

public void ABTestSessionDidStart(){
        Debug.Log("A/B test session started");
        instance. ABTestGetTestDiff();
}
public void ABTestSessionDidEnd(){
        Debug.Log("A/B test session ended");
}
public void ABTestSessionDiffReceived(JsonData diffs){
        Debug.Log("A/B test differences received");
}
```

Now you can compile the project in Xcode and run it directly in the devices. The example shown above includes parsing the default settings provided to modify color and rotation of the object.

For further details, refer to Unity Sample available on package received.

# 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*

# 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.

# 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.
- Unity Sample available on GitHub: further details about setup and configuration parameters for Unity-based application projects.