Google Sheets For Unity

January, 2019.

Latest version of this guide at:
http://bit.ly/devGuide

GSFU Unity3D.com forum thread:
http://bit.ly/forumThread

Store Page:
http://bit.ly/GSForU

API Reference:
http://bit.ly/APIRef

C Y B E R I A N S O F T W A R E
http://CyberianSoftware.com.ar
support@cyberiansoftware.com.ar

Developer Guide

Introduction

Principles

Using Google Sheets for Unity [GSFU] enables Unity3D developers to work with Google Drive.

While offering a simple and flexible solution, GSFU avoids introducing any unnecessary hassle. GSFU is totally unobstrusive to your development process, your game, and can improve your workflow in infinite ways.

Free from third party libraries, exotic dependencies, live OAUth authentications, or credentials. Just by using simple and local API calls to a webservice, your project will be able to save, retrieve, update and delete data from Google Drive, at runtime or design time.

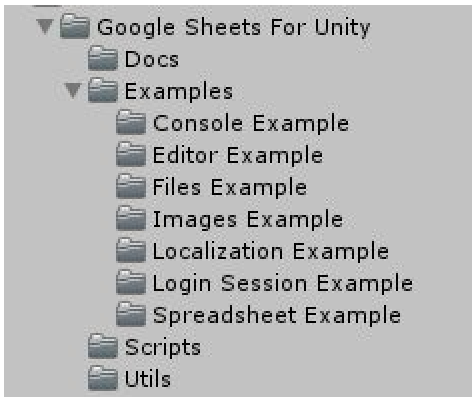
The web service is a script developed using Google Apps Script, which is a javascript based language, that implements some of the APIs for Google online services. Its conveniently installed in your Google Drive.

The source code for both extremes of the connection, Unity and Google, are included, which means optimal flexibility. Both sides can be extended as the needs may require, working with the Google Apps Script API, and the provided Unity API, which is very simple on its structure and approach to queries and responses management.

Package Contents

Unboxing GSFU

Once imported into Unity, the package contents will be visible:



```
▼ Google Sheets For Unity
    Docs
  ▼ Examples
      Console Example
      Editor Example
      Files Example
      Images Example
      Localization Example
      Login Session Example
      Spreadsheet Example
    Scripts
    Utils
```

Fig. 1

The core components of the asset are located at the "**Scripts**" and "**Utils**" folders.

The "**Examples**" folder includes a series of scripts and scenes for demo purposes, divided in folders by the topic they cover.

Note that **for the examples to work**, you need to complete some data fields on the ConnectionData asset: the webapp url, password and spreadsheet id. For more details on this, read next chapter, "Basic Usage".

The "**Docs**" folder includes a PDF version of this document.

Basic Usage
Understanding the principles

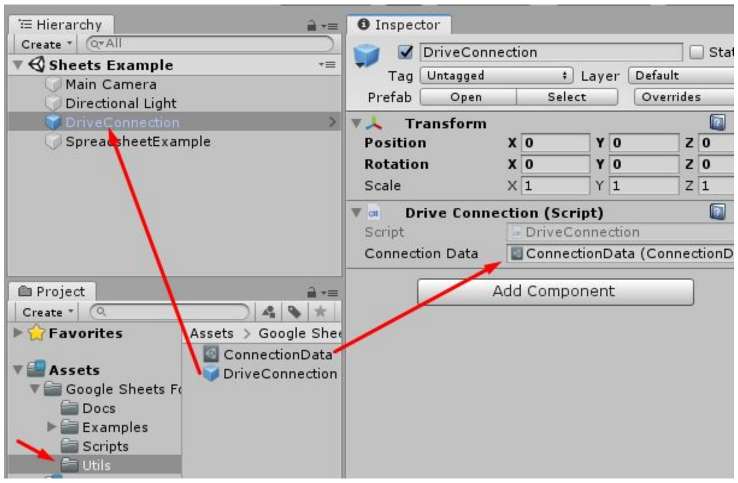
This chapter covers the use of the Google Sheets For Unity. To understand how to setup Google Sheets For Unity correctly, please read chapter entitled "Deployment".

The Google Sheets For Unity principle lies on using a technology mashup to simplify the connection between Unity & Google Drive. This way, we overcome the need to login into Google services using OAuth, and all the hassle, delays and latency that the signin process and the work to keep the session alive introduce.

While the underlying idea is to provide a foundation where to build upon, using the asset as a starting point, GSFU offers an out of the box API for doing basic CRUD operations from a Unity project, to handle spreadsheets, files and folders on Google Drive.

The client side use is quite simple:

1. Drag the drive connection prefab into the scene.
2. Make sure the prefab has a ConnectionData asset assigned.



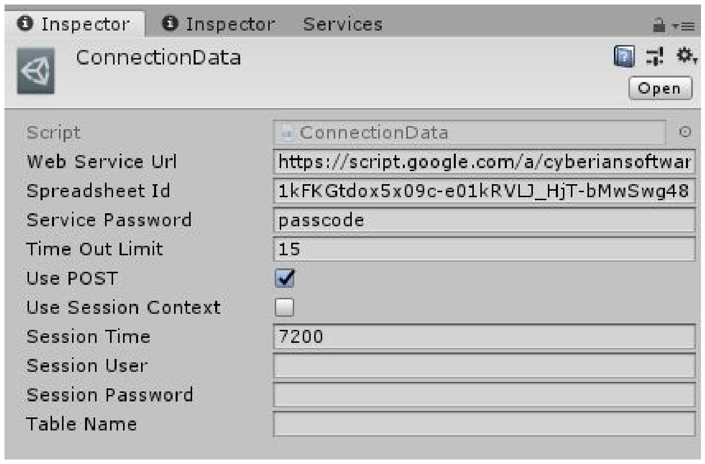
DriveConnection
Tag Untagged Layer Default
Prefab Open Select Overrides
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Drive Connection (Script)
Script DriveConnection
Connection Data ConnectionData (ConnectionD

Fig. 2.

GSFU is ready to be used, is all set for you to implement your mechanics using the `Drive` class API, whithin the `GoogleSheetsForUnity` namespace. You can see the API reference online here.

The **ConnectionData** asset serves as container for all the data needed to make connections to the online webservice.

You can have many different connections, represented on multiple ConnectionData assets. For this purpose, you can create new project elements by duplicating the included asset (CTRL+D in Unity), or by using the Create menú, as shown in the picture below.



ConnectionData

Script	ConnectionData
Web Service Url	https://script.google.com/a/cyberiansoftwar
Spreadsheet Id	1kFKGtdox5x09c-e01kRVLJ_HjT-bMwSwg48
Service Password	passcode
Time Out Limit	15
Use POST	✓
Use Session Context	
Session Time	7200
Session User	
Session Password	
Table Name	

Fig. 3. The ConnectionData Asset.

- **Web Service Url** - string value indicating the url of the published webapp. See chapter Deployment, for more details.

- **Spreadsheet Id** - string value to be set to the ID key of the spreadsheet to be used.

> To get the spreadsheet id, simply open the spreadsheet and watch at the URL. It will look like the example below, in which the key is “*abc123456*”.
>
> *https://docs.google.com/spreadsheets/d/**abc123456**/edit#gid=671966417*

- **Service Password** - string value with the password established in the web service script.

- **Time Out Limit** - float value that indicates, the time in seconds allowed for the connection to perform its complete cycle, before canceling it.

- **Use POST** - bool value, true by default, indicating the type of request to be created (POST or GET). Use POST unleast you have some concrete limitation.

- **Use Session Context** - bool value. Instead of stateless independent queries that use a password each time, the queries will depend on a session context for authorization. If you need to track users activity, this is will keep record of logins. See Login Session Example for more info on how to use sessions.

- **Session Time** - integer value. The session duration time, in seconds. The minimum recommended is 7200 seconds (2 hours) and the maximum possible is 21600 seconds (6 hours). After this time has passed, the session will close itself.

- **Session User** - string value. Username for the login session.

- **Session Password** - string value. Password for the login session.

- **Table Name** - string value. The name of the table where the credentials data is holded.

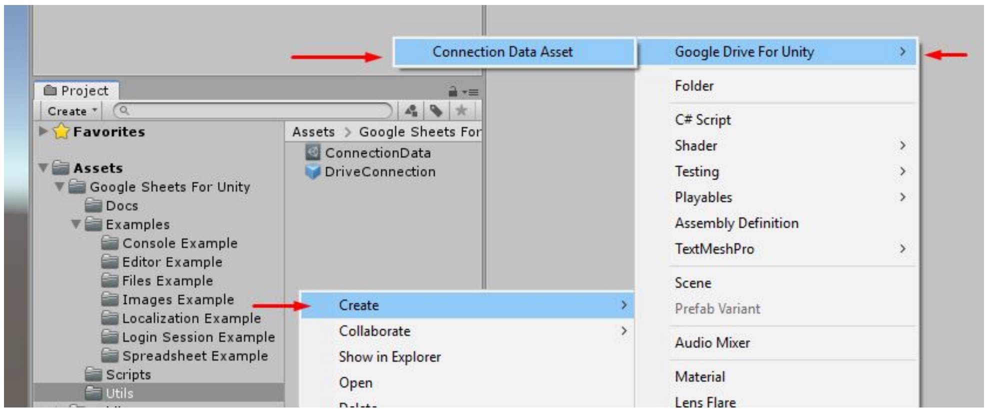


Fig. 4. Creating a new ConnectionData object through the Create context menu.

Editor Use

Besides working with Google Spreadsheets at runtime, from a game or app, you can also utilize GSFU as a development tool from the Unity environment itself.

For this purpose, you need to make sure the `DriveConnectionEditor` script, which has a serialized field for setting a **ConnectionData** asset, has the correct or intended asset assigned to it, and then you simply implement your mechanics using the same API than the one used for the runtime applications!

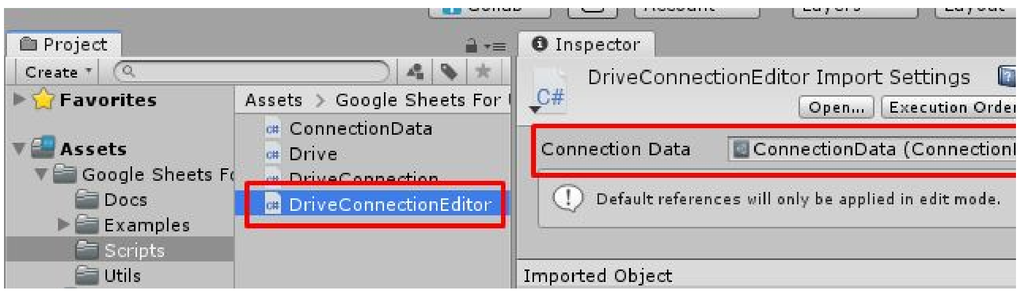


Fig. 5. DriveConnectionEditor script is selected on the project view, and the ConnectionData field can be seen.

Deployment

Server Side

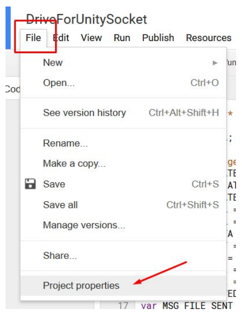
First step in setting up Google Sheets For Unity, is getting a copy of the web service script. It was developed using Google Apps Script, and is conveniently stored and deployed on your Google Drive. You can grab your copy from this link:

> http://bit.ly/webappScript
> *(Clicking on the link should automatically create a copy of the script).*

Second step is setting your webservice password (is "*passcode*" by default).

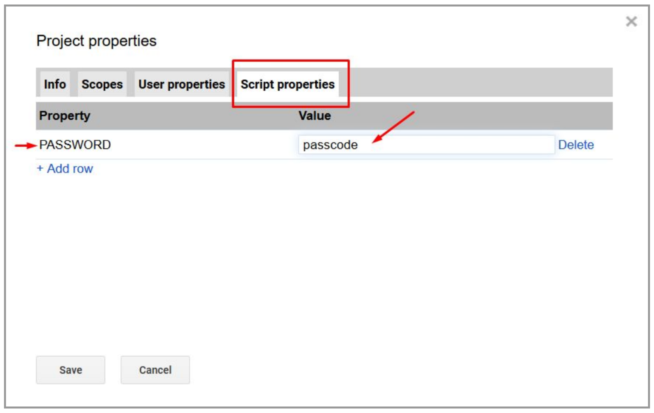
Open your script file, by double clicking on the file at Google Drive. If it's the first time, it will guide you to associate the script files to the Drive Script Editor.

1. Go to Menu File -> Project Properties.
2. Go to Script Properties tab, and click on ""passcode" value to change it.



```
var MSG_FILE_SENT
```

Fig. 6. (Step 1)



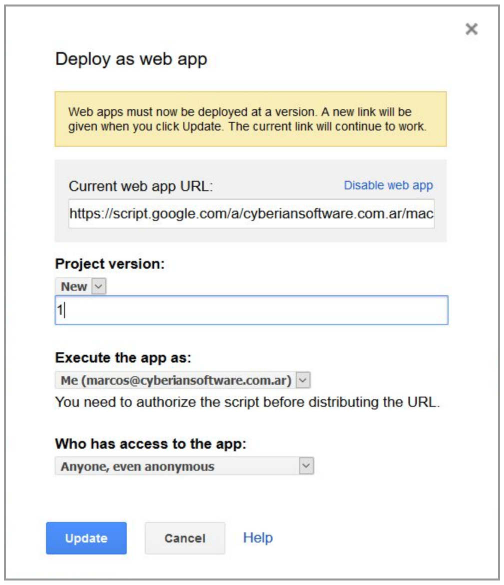
Property	Value	
PASSWORD	passcode	Delete

Fig. 7. (Step 2)

The value you set there is the string to be used in the ConnectionData asset, in the "Service Password" field.

Third step: the script **deployment as webapp**. To do it, please follow these steps:

1. Save a new version of the script by selecting **File > Manage Versions**, then **Save New Version**.
2. Select **Publish > Deploy as web app**.
3. Under **Project version**, select the version you just saved.
4. Under **Execute the app as**, select your account (the developer's).
5. Under **Who has access to the app**, select "*Anyone, even anonymous*$^{(*)}$".
6. Click **Deploy** (In further deploys, the button will say "Update") .



Deploy as web app

Web apps must now be deployed at a version. A new link will be given when you click Update. The current link will continue to work.

Current web app URL: https://script.google.com/a/cyberiansoftware.com.ar/mac

Project version: New 1

Execute the app as: Me (marcos@cyberiansoftware.com.ar)

You need to authorize the script before distributing the URL.

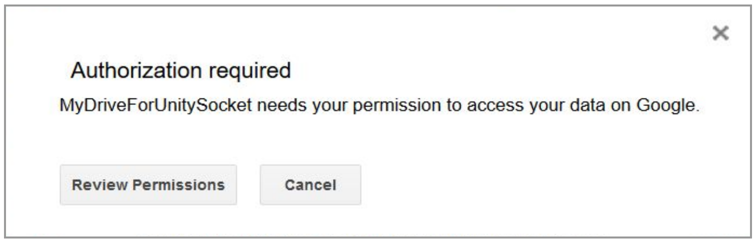
Who has access to the app: Anyone, even anonymous

Fig. 8. Deploy Dialog.

(*) If "*Anyone, even anonymous*" is not listed as an option, you need to check your Google Apps Administrator panel, there is a policy option for sharing the docs outside the organization. You can read more about Google Apps sharing policy here: https://support.google.com/a/answer/60781

Fourth step (only the first time): Granting access rights.

In order for the web app to use the Google Script API, you need to grant it permissions to run. Usually, the authorization request displays after deploying your script for the first time (inmediately after clicking "Deploy"):



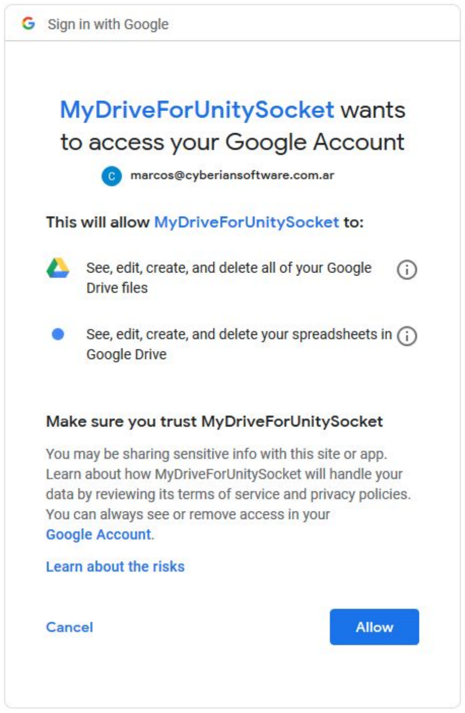
Authorization required

MyDriveForUnitySocket needs your permission to access your data on Google.

Fig. 9. Authorization Request.

You need to click on "Review Permissions".

A new Google authorization Dialog will show up, just click on “**Allow**”:


Sign in with Google

MyDriveForUnitySocket wants to access your Google Account

marcos@cyberiansoftware.com.ar

This will allow MyDriveForUnitySocket to:

See, edit, create, and delete all of your Google Drive files

See, edit, create, and delete your spreadsheets in Google Drive

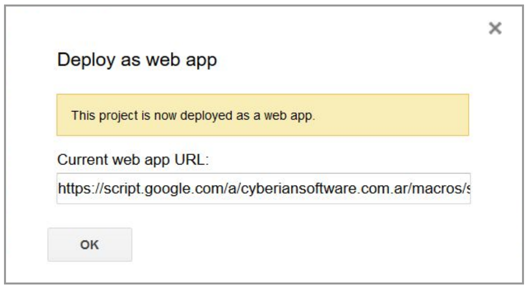
Make sure you trust MyDriveForUnitySocket

You may be sharing sensitive info with this site or app. Learn about how MyDriveForUnitySocket will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Learn about the risks

Fig. 10. Script permissions review.

Once you click “**Allow”**, you'll see a new dialog with a message indicating that your scriptt has been successfully deployed as a web app:


Deploy as web app

This project is now deployed as a web app.

Current web app URL:

https://script.google.com/a/cyberiansoftware.com.ar/macros/s

Fig. 11. “Script Deployed” dialog.

The above dialog provides the URL for your app, copy & paste on Unity, in the **“Web Service Url”** field of the ConnectionData asset.

Done!

Important for the script customization!

If you make changes to your script, you will have to repeat the revision and deployment process then, or your changes will not go live. For further info, see https://developers.google.com/apps-script