Google Sheets To Unity

V 0.9.6

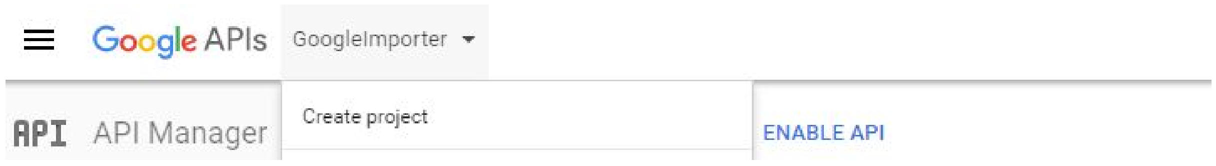
Intro

Google sheets to unity is a 2 way system for importing and exporting data from google sheets to unity in both the editor and runtime (providing an internet connection is established).It can be used as to add a more data driven workflow to your unity project when updating and storing all information on a spreadsheet and then importing this information into Unity.

Initial Setup

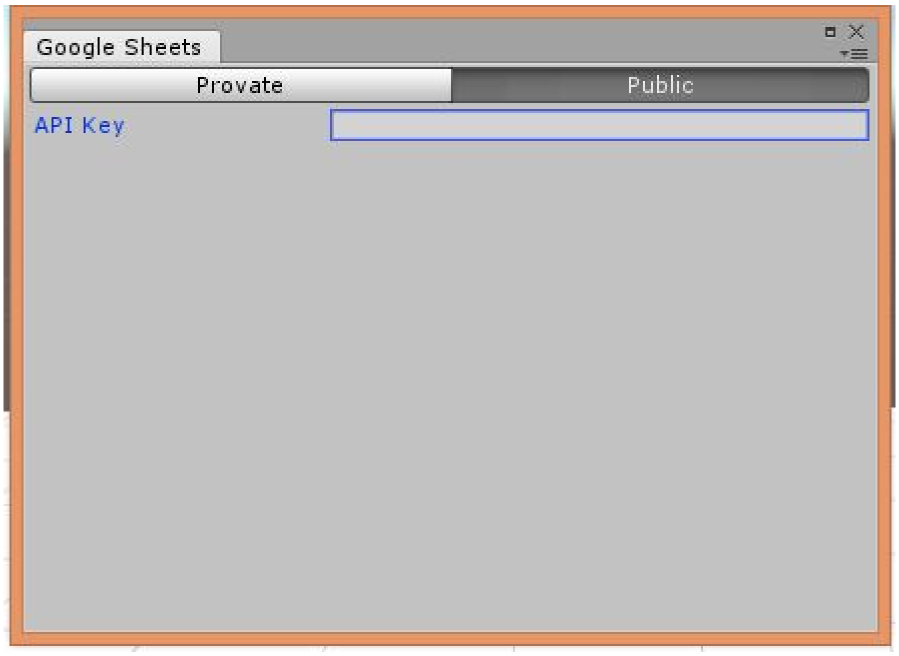
Public Sheets (Read only)

1.After setting you sheet to be public you need to visit the Google Developer Console(Console.developers.google.com) and create a new project. This can take a few minutes to create the project.

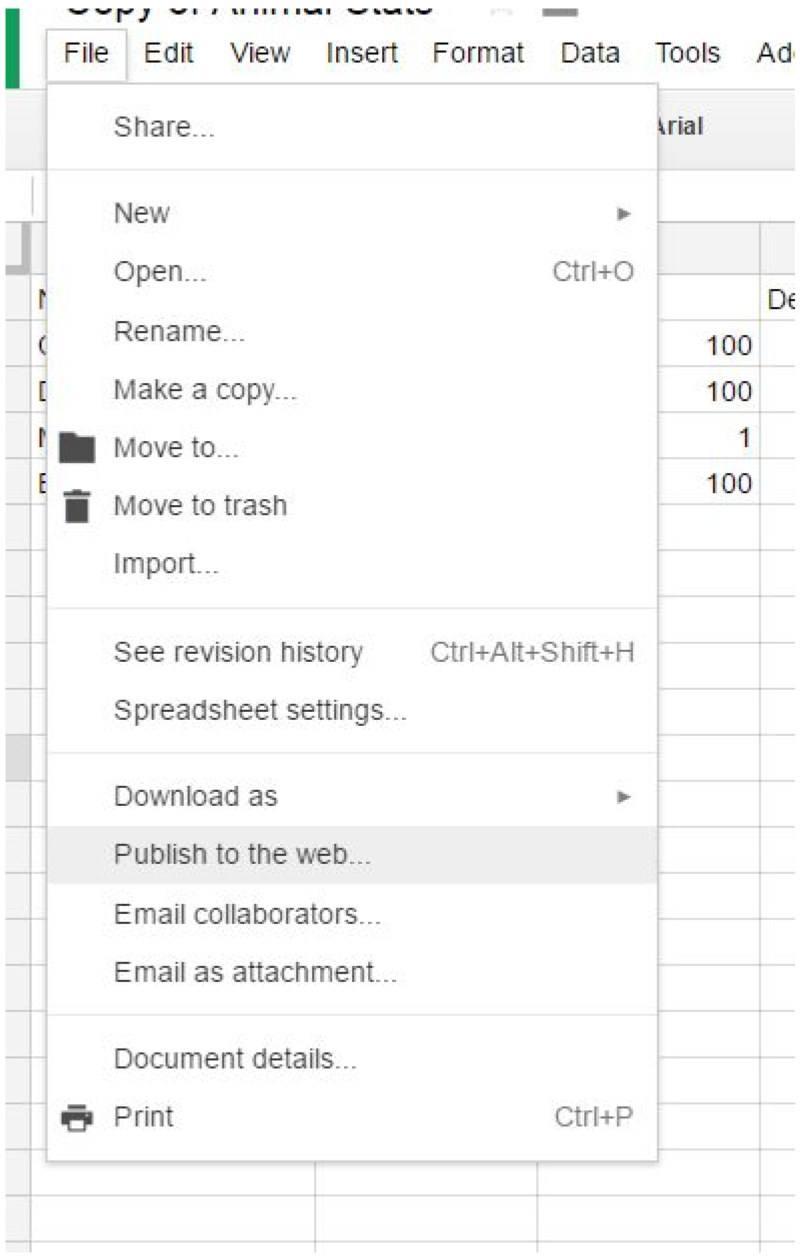


2. Click on the Credentials button on the left had side and click Create credentials. From the dropdown list select API key.

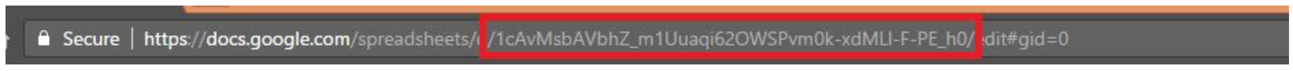
3 Copy the new generated API key into the api key text box on the google sheets to unity config menu.



4. You need to enable your spreadsheet to be published. This is done by clicking File > Publish to the Web and then clicking “Publish” in the opening popup. By Doing this the sheet can be readable without authenticating yourself but can only be used to read data.



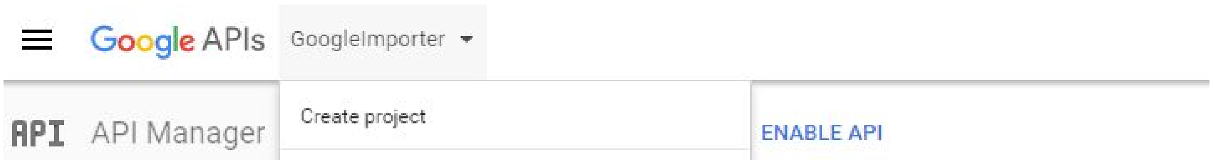
Each Spreadsheet has a unique ID that can be found in the web url for that sheet as shown below. This is the spreadsheet id needed to import the data.


Secure | https://docs.google.com/spreadsheets/d/1cAvMsbAVbhZ_m1Uuaqi62OWSPvm0k-xdMLI-F-PE_h0/edit#gid=0

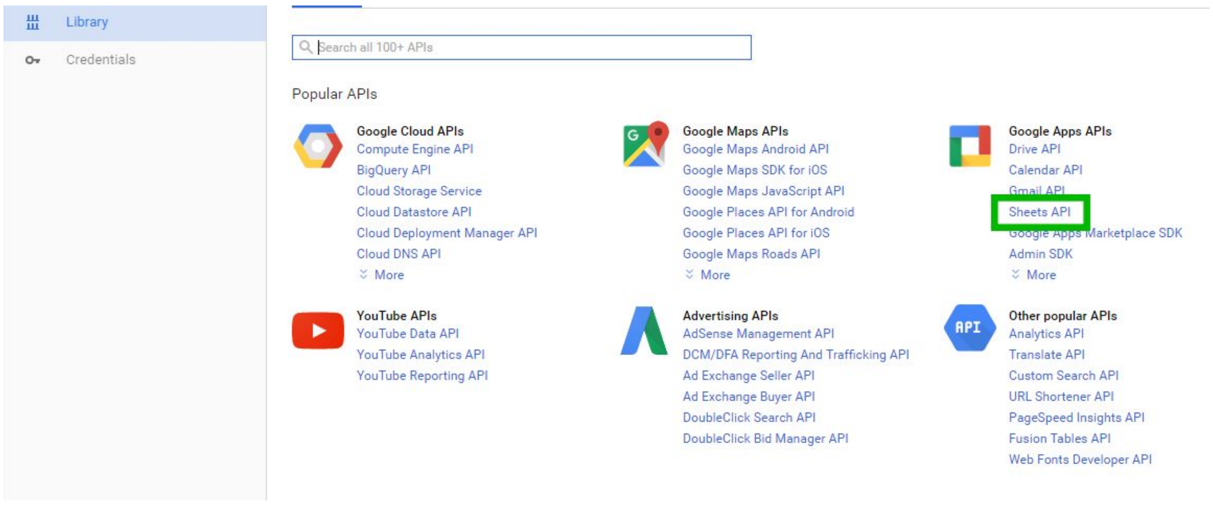
Private Sheets(Read and Write)

After adding Google Sheets to Unity into unity there are a few steps that need to be set up in order to link your unity project with google.

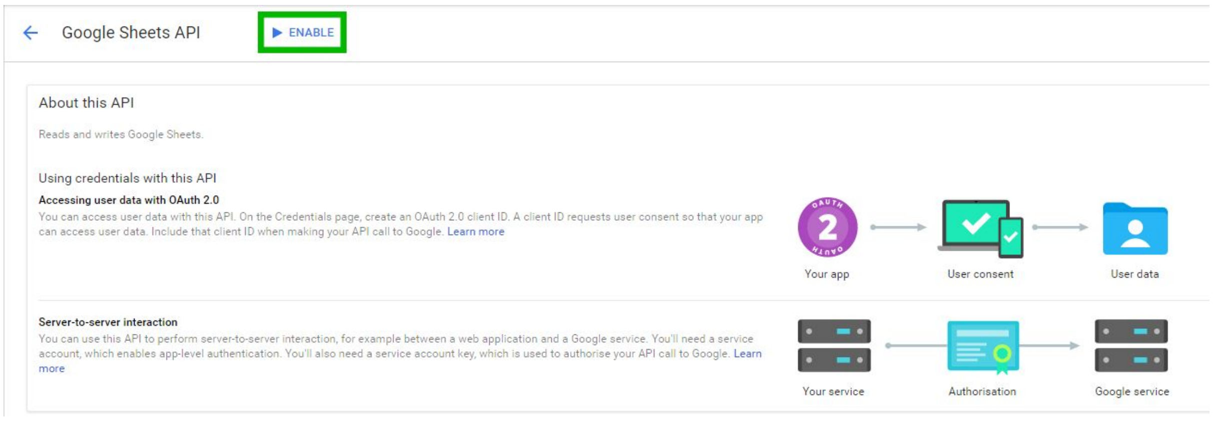
1. Login or create an account with google developer console (Console.developers.google.com) and create a new project. This can take a few minutes to create the project.



2. Once the project is created click on the “Library” button on the left hand side and either find the “Google Sheets API” link in the “Google Apps APIs” section, or use the search box to location quickly.



3. On the google Sheets API page click “ENABLE”.



About this API

Reads and writes Google Sheets.

Using credentials with this API

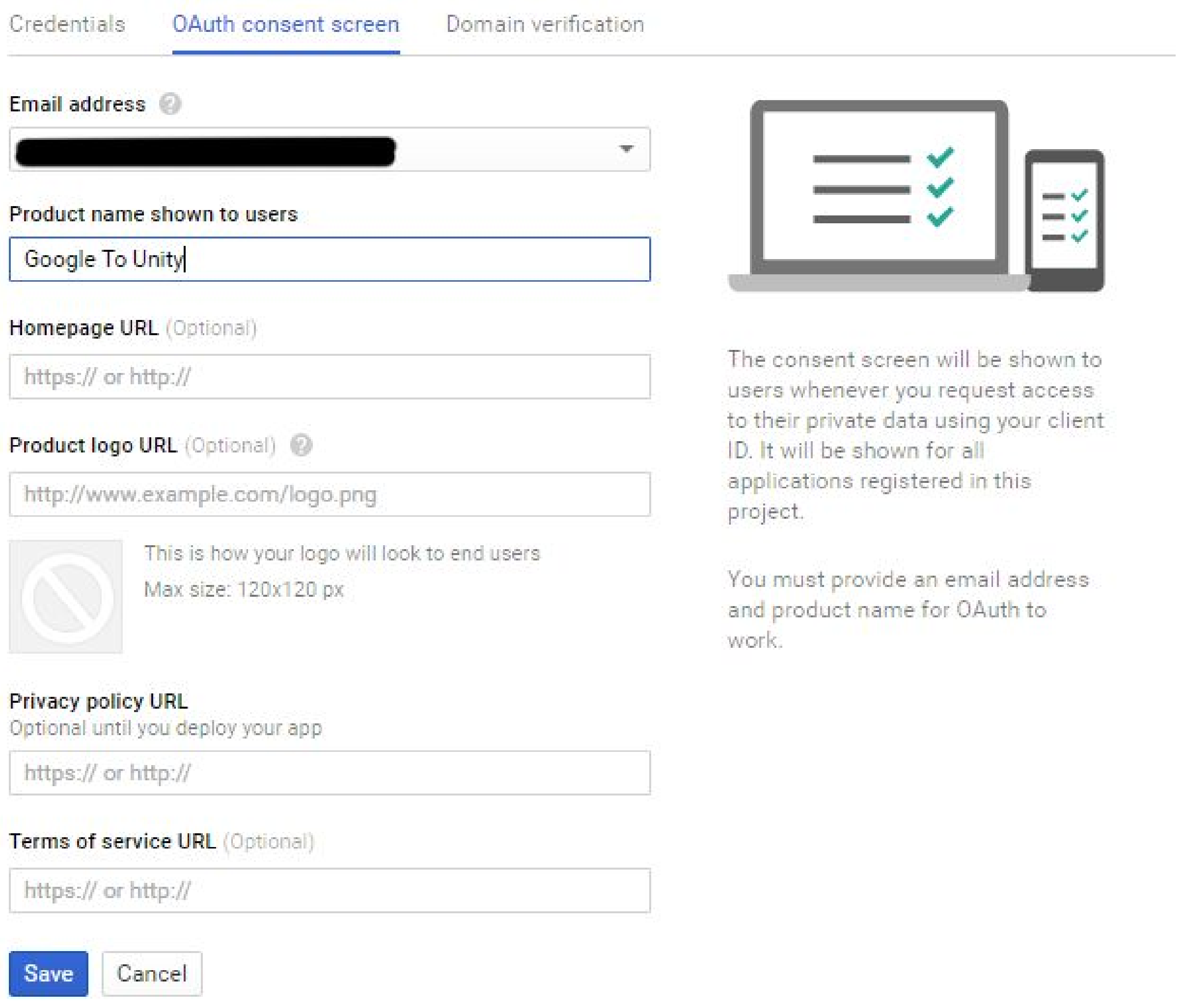
Accessing user data with OAuth 2.0

You can access user data with this API. On the Credentials page, create an OAuth 2.0 client ID. A client ID requests user consent so that your app can access user data. Include that client ID when making your API call to Google. Learn more

Server-to-server interaction

You can use this API to perform server-to-server interaction, for example between a web application and a Google service. You'll need a service account, which enables app-level authentication. You'll also need a service account key, which is used to authorise your API call to Google. Learn more

4. Navigate to the “Credentials” page on the left hand side and then to the “OAuth consent Screen” and select an email address for your product and enter a product name and click save.



Email address

[redacted]

Product name shown to users

Google To Unity

Homepage URL (Optional)

https:// or http://

Product logo URL (Optional)

http://www.example.com/logo.png

This is how your logo will look to end users

Max size: 120x120 px

Privacy policy URL

Optional until you deploy your app

https:// or http://

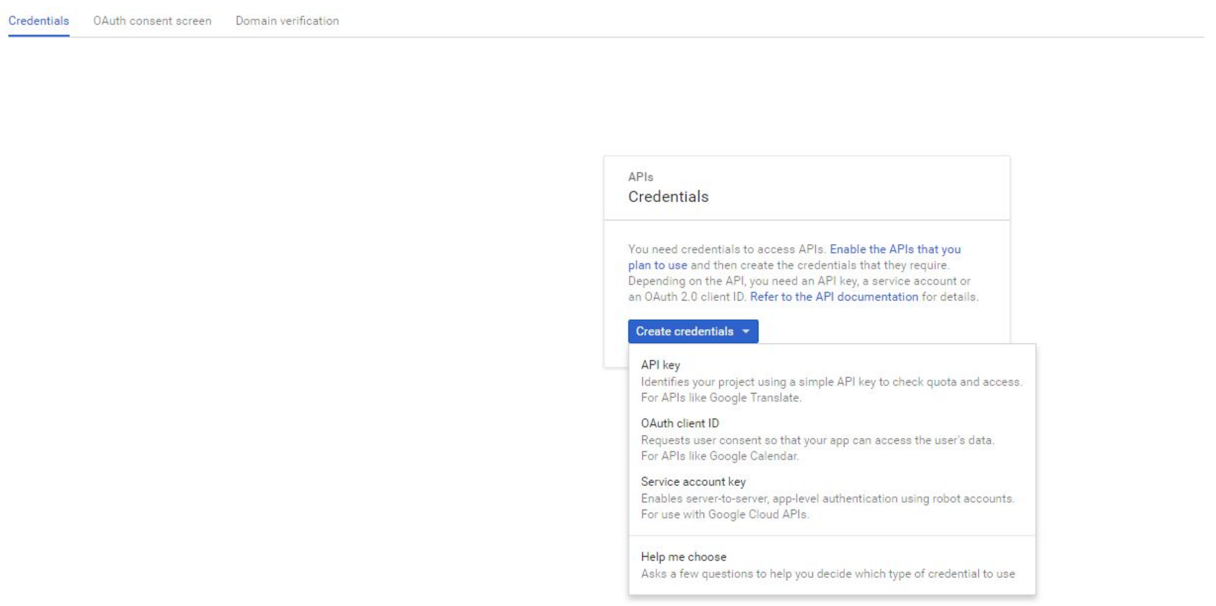
Terms of service URL (Optional)

https:// or http://

The consent screen will be shown to users whenever you request access to their private data using your client ID. It will be shown for all applications registered in this project.

You must provide an email address and product name for OAuth to work.

5. Still on the “Credentials” page now navigate to the “Credentials” tab, click on the “Create credentials” dropdown followed by selecting “OAuth client ID”



Credentials

You need credentials to access APIs. Enable the APIs that you plan to use and then create the credentials that they require. Depending on the API, you need an API key, a service account or an OAuth 2.0 client ID. Refer to the API documentation for details.

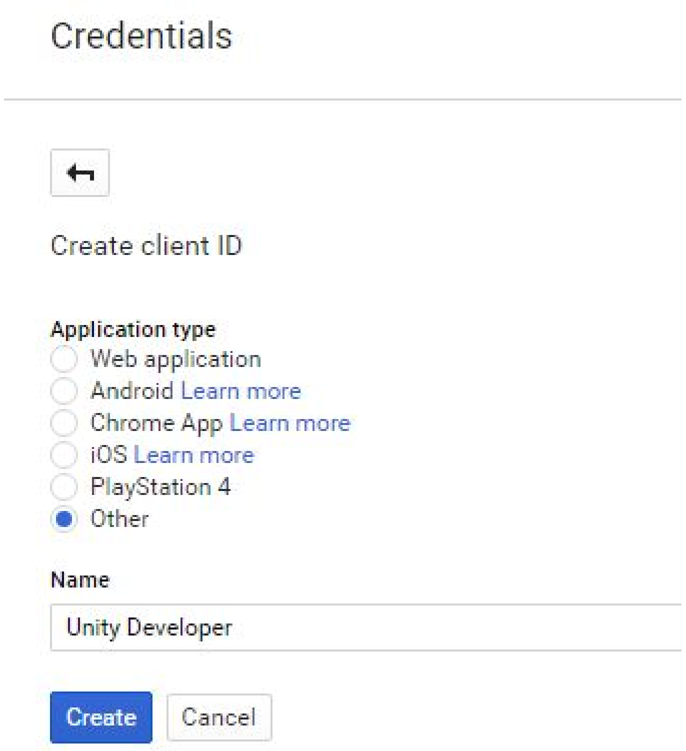
API key
Identifies your project using a simple API key to check quota and access. For APIs like Google Translate.

OAuth client ID
Requests user consent so that your app can access the user's data. For APIs like Google Calendar.

Service account key
Enables server-to-server, app-level authentication using robot accounts. For use with Google Cloud APIs.

Help me choose
Asks a few questions to help you decide which type of credential to use

6. Select “Other” from the list and end “Unity Developer” in the name field that appears and press create

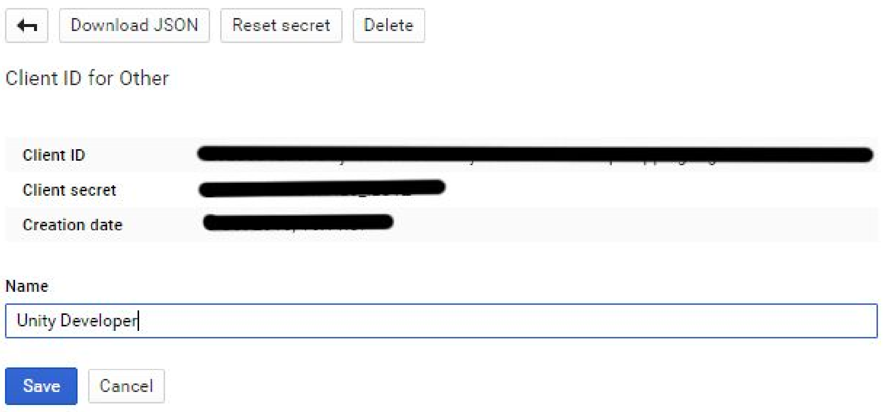

Credentials

Create client ID

Application type
- Web application
- Android Learn more
- Chrome App Learn more
- iOS Learn more
- PlayStation 4
- [✓ Other]

Name
Unity Developer

7. Your client ID and client Secret will now appear. (You can click on the created credential to get these in the future if needed)

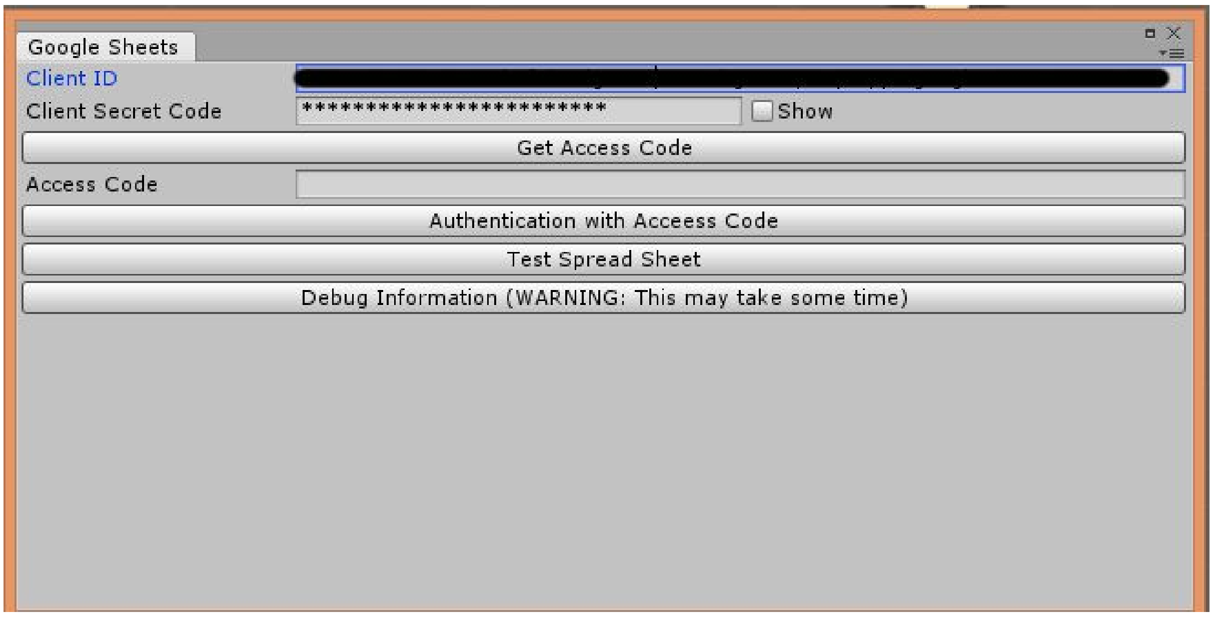


Client ID for Other

Client ID	[redacted]
Client secret	[redacted]
Creation date	[redacted]

Name
Unity Developer

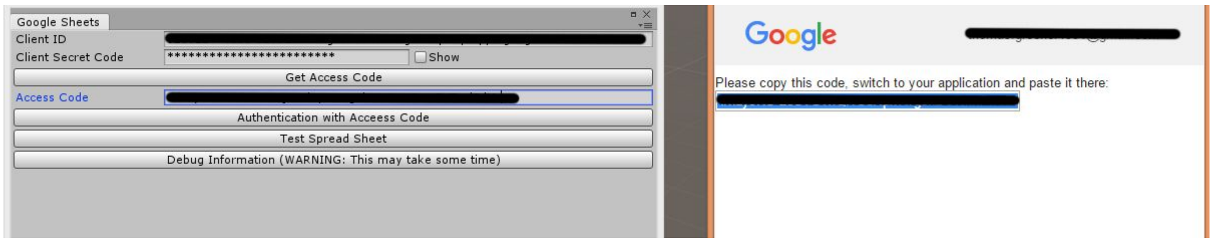
8. In Unity open the Google Sheets to Unity window (Window > Google Sheets To Unity)



Client ID: [redacted]

Client Secret Code: ************************

9. Copy the Client ID and the Client Secret to the relevant fields and click get access code.
10. A web page will now open where you will be asked to log into your Google account that the spreadsheet is available on (note documents that are shared with your account will still show). Follow the steps on the webpage and this will present you with an access Code, copy this code into the unity project "Access Code" field and click "Authentication with Access Code". Unity will log out Validation successful if validation was a success.



Client ID: [redacted]

Client Secret Code: ************************

Access Code: [redacted]

[redacted]

Please copy this code, switch to your application and paste it there:

[redacted]

11. If you wish to view all sheets and worksheets that your login has provided you access to click the "Debug Information" button. This will take a few minutes but will display all sheets on your google account has access to (the more sheets the longer this will take)
12. You are now set up with Google Sheets to Unity. Refer to the usage section for code references.

Example Projet (Animal Stats)

Intro

GS2U comes with a quick example project to demonstrate import and exporting data from a spreadsheet. This example will show you how to update assets with values from a spreadsheet and how to send new data to the spreadsheet.

NOTE: Prior to running the example scene you must have Linked Google sheets to your Unity as described in the initial setup section.

After you have your project linked with Google open the “Animal Stats Google Sheets Link” and select File > Make A Copy and rename the copy to “Animal Sheets”. This will create you a copy of the google sheet used in the example and allow you to access and modify it.

Example 1 - Updating Stats

Navigate you the example files (Default : Assets > Google Sheets To Unity > Animal Example) and open up the “Animal Stats” folder In here are 3 created animal scriptable objects with some basic information for health, attack and defence. By clicking on one of these items you can see all the information that the animal holds. Each of these animals has a custom inspector on them that gives the user access to an “Update” button. Clicking this button will update the values from Google Sheets. If you modify the values on google sheets and click update the new information will populate the relevant fields.

If you wish to update multiple fields at once there is an editor script available to do this. Under the menu “Update > Update All Animals” (the editor script for this is available in the Editor folder of the example project. This will update all animals known to the animal container (This is found in the resource folder of the example project). As well as updating all the animals that exist in the project it will also create new assets for animal it can not locate in the project. For example the default spreadsheet contains an entry for “Bear” although you could create a new Animal asset and click the update button clicking the update all will automatically create this asset for you.

Example 2 - Updating Stats at runtime

Navigate and open the “Animal Example” scene. This scene contains 3 cubes each with a different animal assigned to them. If you press play the gui above the animals shows the stats for that animal. These are initially the values that were available when the user pressed play. If you wish to update these values at runtime each cube has a script called “animalExampleHolder.cs” attached to them. Marking the “Update on Play” field to true will result in the data being updated when the user pressed play for that particular animal.
Attached to the “Animals” gameobject there is a script called “animalManager.cs”. This also contains a “Update On Play” field however when this one is ticked it will update all animals known in the “Container” (this points to the animal container stored in resources) when the program stats instead of just a singular animal.

NOTE: Updating lots of information during runtime can be costly however so it is recommended that this is used sparingly and would recommend doing this process during a loading screen or similar system. It also would require the user to have an active internet connection during the runtime process in order to collect the data.

Example 3 - Sending Data To Google Sheets

GS2U is also capable of populating a sheet from inside the editor. In this example we will create a new animal called “Shark” and send the information to the sheet for future editing purposes. Navigate to the Animal Stats folder and copy one of the existing animals. Rename the asset to “Shark” and modify the animals information to whatever you want. When you're happy click the “Add” button on the animal and it will appear in google sheets along with the rest of your animals.

Useage

Both private and public worksheets operate in the same way once the WorksheetData has been loaded however they are loaded in slightly different ways.

In the current Version public spreadsheets perform in the background and will return a result when completed. This feature will be in the private sheets in a future update.

Public Spreadsheets

Creating a Spreadsheet Manager

The public Spreadsheet Manager is the core class for Google Sheet to Unity for accessing public spreadsheets

```
PublicSpreadSheetManager publicSpreadSheetManager

Void Start()
{
 publicSpreadSheetManager= new PublicSpreadSheetManager();
}
```

Loading a worksheet

Once A public spreadsheet manager has been created it can be used to load a worksheet. You need the spreadsheets id from the web url (example :docs.google.com/spreadsheets/d/13b-lOdfRG-VCvoE0zmxITS8958Sx1adLdI0x-oF4RBY/edit#gid =0). Additionally a worksheet, to and from cells (“A1”,”F4), doing this will make the top row the title fields, and a callback to process the data

```
public void LoadPublicWorksheet(string spreadsheetID, OnSpreedSheetLoaded callback = null);
public void LoadPublicWorksheet(string spreadsheetID, string worksheetName, OnSpreedSheetLoaded
callback = null);
public void LoadPublicWorksheet(string spreadsheetID, string from, string to, OnSpreedSheetLoaded
callback = null);
public void LoadPublicWorksheet(string spreadsheetID, string worksheetName, string from, string
to, OnSpreedSheetLoaded callback = null);
```

example:

```
WorksheetData data = manager.LoadPublicWorkSheet("1GVXeyWCz0tCjyqE1GWJoayj92rx4a_hu4nQbYmW_PkE",
"Sheet2",”A2”,”B7”, Callback);
```

Processing a Public worksheet

The data from a public worksheet comes in as a dictionary and can be quickly searched from using the row and column titles for a direct search or and for each to loop through the dictionary. Public spreadsheets can also support multiple fields within the same row title. As such the data needs to be followed by an index within that row. If the row only contains 1 set of data this is always [0] otherwise it is the array index.

E.g

```
Debug.Log(manager.WorkSheetData["Dog"].data["Health"][0]);
```

Or

```
foreach (string s in m.WorkSheetData["Badger"].data["Items"])
{
    Debug.Log(s);
}
```

Private Spreadsheets

Creating a Spreadsheet Manager

The Spreadsheet Manager is the core class for Google Sheet to Unity for accessing private spreadsheets for reading and writing

```
SpreadSheetManager spreadSheetManager

Void Start()
{
 spreadSheetManager = new SpreadSheetManager();
}
```

Loading A Spreadsheet

Once a spreadsheet manager is created it can load a spreadsheet into memory by either using the sheets name (if 2 sheets have the same name it will return the one created first) or its unique id from google, this is obtained on the google sheets web address, and is the highlighted part in this example:
docs.google.com/spreadsheets/d/13b-lOdfRG-VCvoE0zmxITS8958Sx1adLdI0x-oF4RBY/edit#gid=0)

```
GS2U_SpreadSheet spreadsheetFromName = spreadSheetManager.LoadSpreadSheet("Google Sheets
To Unity Test");
GS2U_SpreadSheet spreadsheetFromID =
spreadSheetManager.LoadSpreadSheet("13b-lOdfRG-VCvoE0zmxITS8958Sx1adLdI0x-oF4RBY", true);
```

Loading a worksheet

Using the selected spreadsheet you can now access a particular worksheet within that sheet using either the worksheets name or its number. NOTE: worksheet ID starts from 0

```
GS2U_Worksheet worksheetFromName = spreadsheetFromName.LoadWorkSheet("Demo Sheet");
GS2U_Worksheet worksheetFromSheetNumber = spreadsheetFromName.LoadWorkSheet(2);
```

Creating a new worksheet

Used to create a new worksheet and have access to it instantly if you wish to send data from the editor to a worksheet. Unless a size is specified the new spreadsheet will be 60 columns by 250 rows.

```
        GS2U_Worksheet newWorksheet = spreadsheetFromName.AddNewWorksheet("New WorkSheet");
        GS2U_Worksheet newWorksheet2 = spreadsheetFromName.AddNewWorksheet("New WorkSheet2",
100,100);
```

Get or Set a worksheets size

Used to get or set a worksheets size. NOTE: setting the size smaller than its current size will permanently delete these cells and any data stored in them.

```
        Vector2 worksheetSize = worksheet.GetWorksheetSize();
        Debug.Log("Colums " + worksheetSize.x + "  Rows: " + worksheetSize.y);

        worksheet.SetWorksheetSize(150, 150);
```

Adding additional rows and columns

Used to extend the size of a worksheet Columns

```
        worksheet.AddRowsAndColumns(10, 10);
```

Delete a Worksheet

Used to remove a worksheet from a the spreadsheet.

```
        worksheet.DeleteWorksheet();
```

Load all a worksheets information

Used to load all the information on a worksheet to cycle through data.

```
 WorksheetData worksheetInformation = worksheet.LoadAllWorksheetInformation();

            //Display results
            foreach (RowData row inworksheetInformation.rows) //loops through every row
            {
                //shows the title of the row (the value in column A), EG "Bear"
               Debug.Log(row.rowTitle);

               //loops through every cell in that row
                    foreach (CellData cell in row.cells)
                {
                        //shows the column title and the value in that cell.
                       //example output "Health 100";
                            Debug.Log(cell.cellColumTitle + "    " + cell..value);
                        }

            }
```

Get a List of all the column or row titles

Returns a list of all the column titles

```
        List<string> columnTitles = worksheet.GetColumnTitles();
        List<string> rowTitles = worksheet.GetRowTitles();
```

Edit a Rows Information

Edits a rows information based on either the rows title or its position in the table, if two rows have the same title it will modify the first instance of this row. Can be used to edit a whole row or just a value within that row. In the dictionary the first field is the name of the columns title and the second field is the new value, if a column's title does not exist the value is skipped.

Note: all keys must be without spaces or special characters e.g “Total Kills” would become “TotalKills”

```
worksheet.ModifyRowData("Bear", new Dictionary<string, string>
{
  {"Weight", "250" },
  {"TotalKills", "10"}
});

worksheet.ModifyRowData(5, new Dictionary<string, string>
{
 {"Weight", "250" },
});
```

Add new data row

Adds a new row of data at the next available position on the worksheet.In the dictionary the first field is the name of the columns title and the second field is the new value, if a columns title does not exist the value is skipped
Note: all keys must be without spaces or special characters e.g “Total Kills” would become “TotalKills”

```
worksheet.AddRowData(new Dictionary<string, string>
    {
      {"Weight", "250" },
      {"Size", "1.5m"}
    });
```

Delete a row of data

Deletes a row of data from the worksheet found by either the rows title or its position in the table, if two rows have the same title it will delete the first instance of this row.

```
worksheet.DeleteRowData("Bear");
worksheet.DeleteRowData(2);
```

Get data from a cell reference

Using a set cell reference to draw data. NOTE: the cell reference may change if rows are added or removed

```
CellData data = worksheet.GetCellData("B",7);
Debug.Log(data.value); //debug log the value of B7
```

Modify a cell's data from a cell reference

Using a cell reference to modify a particular cell’s data. NOTE: the cell reference may change if rows are added or removed

```
worksheet.ModifyCellData("B",7, "Shark"); // will change the value in B7 to be "Shark"
```

Video Tutorials

Setup : https://youtu.be/uTsOhS50H7Y?list=PLuNz4X8IvmwJNtg4KiTnh9R0aKPJQPGJz

Single Data Use :
https://youtu.be/T0R1SFKV_IA?list=PLuNz4X8IvmwJNtg4KiTnh9R0aKPJQPGJz

Multi Data Use : https://youtu.be/6liU2B4dMp4?list=PLuNz4X8IvmwJNtg4KiTnh9R0aKPJQPGJz

Updating in build:
https://youtu.be/yqUqXvClsGY?list=PLuNz4X8IvmwJNtg4KiTnh9R0aKPJQPGJz

Change Log

9.5.2

- Bug fix where public spreadsheets would not work as intended
- Unified version across unity versions