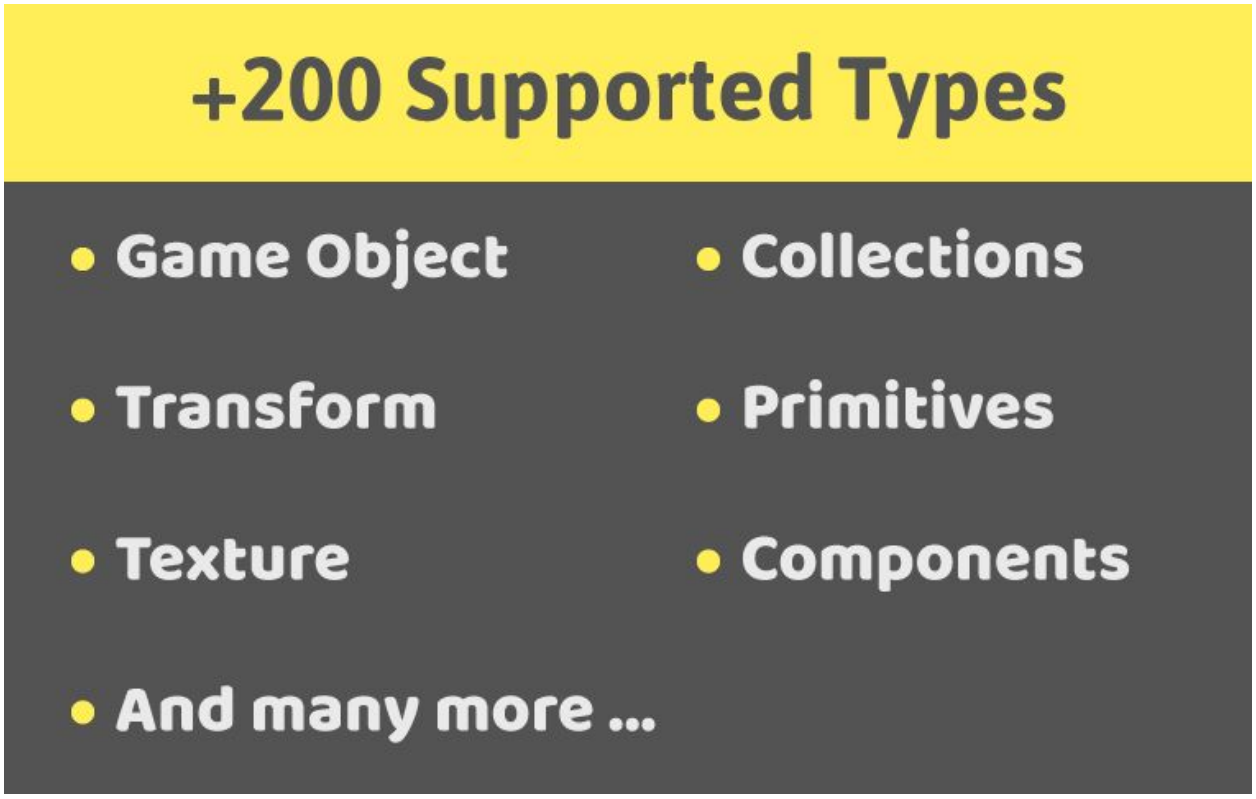
Save Game Pro

A Complete and Powerful Save Game Solution

Introduction

Save Game Pro is a complete, powerful and feature-rich save game solution for Unity (Game Engine) that lets you save everything such as Components, Data Types, GameObjects including Custom Data Types at everywhere including Web & Cloud, Local Storage, PlayerPrefs, Database.
Save Game Pro has an Elegant and Complete API for controlling saved data, for example, by using the API you can check if a data exists or you can Retrieve the saved files list and show them to user.

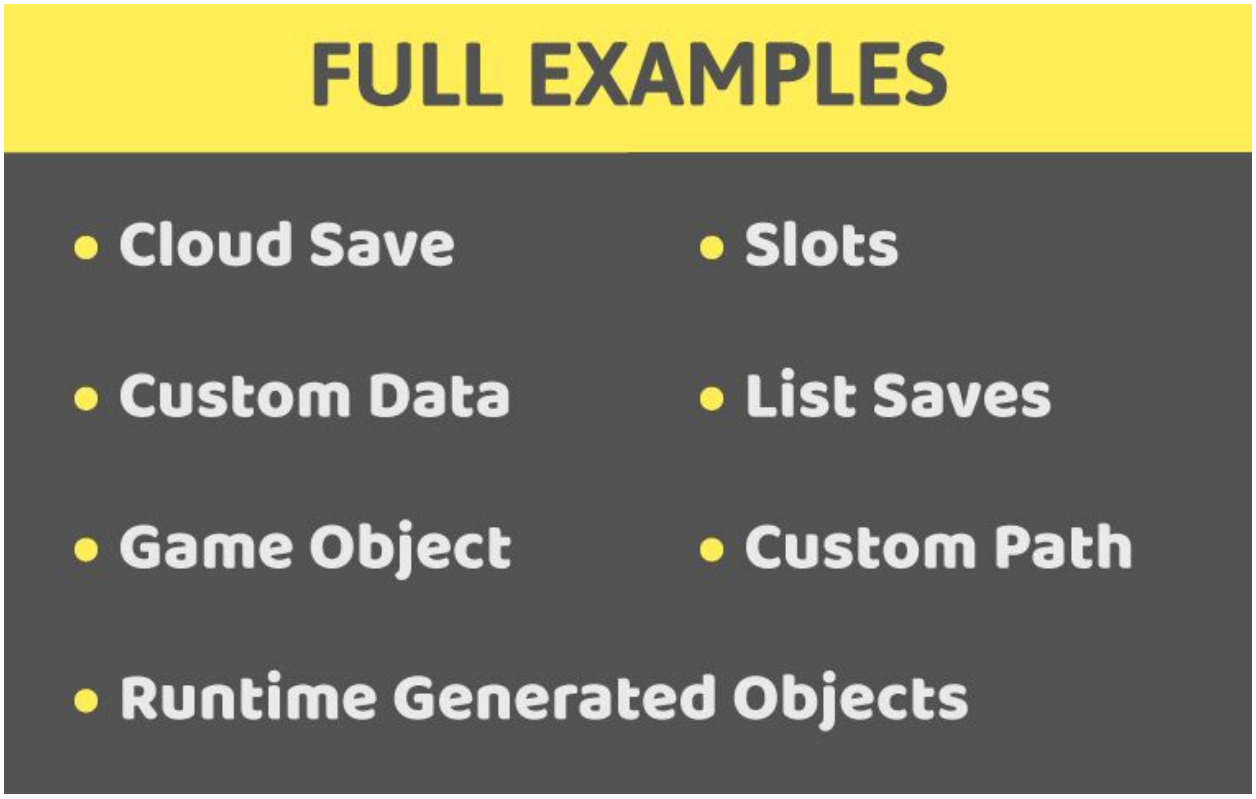
Supported Types



Save Game Pro supports more than +200 Types built-in and you can add your own extra types manually or by using Type Creator. (description is available at below)
The Supported Types including:

- Almost All Components, such as BoxCollider, Rigidbody, MeshRenderer, Camera, Transform.
- GameObject, the Whole GameObject will be saved, that means all Components including All Childs.
- Transform with Hierarchy, that means the transforms are saved by their root parent and they will be loaded like that.
- Primitives, such as Integer, String, Boolean, …
- Collections, Almost all collections supported, such as Dictionary, List, LinkedList, Multi-Dimensional Arrays (there are no limitation), Stack, Queue, HashSet, …
- Data Types, Almost all Data Types such as Vector3, Vector2, Vector4, Mesh, …

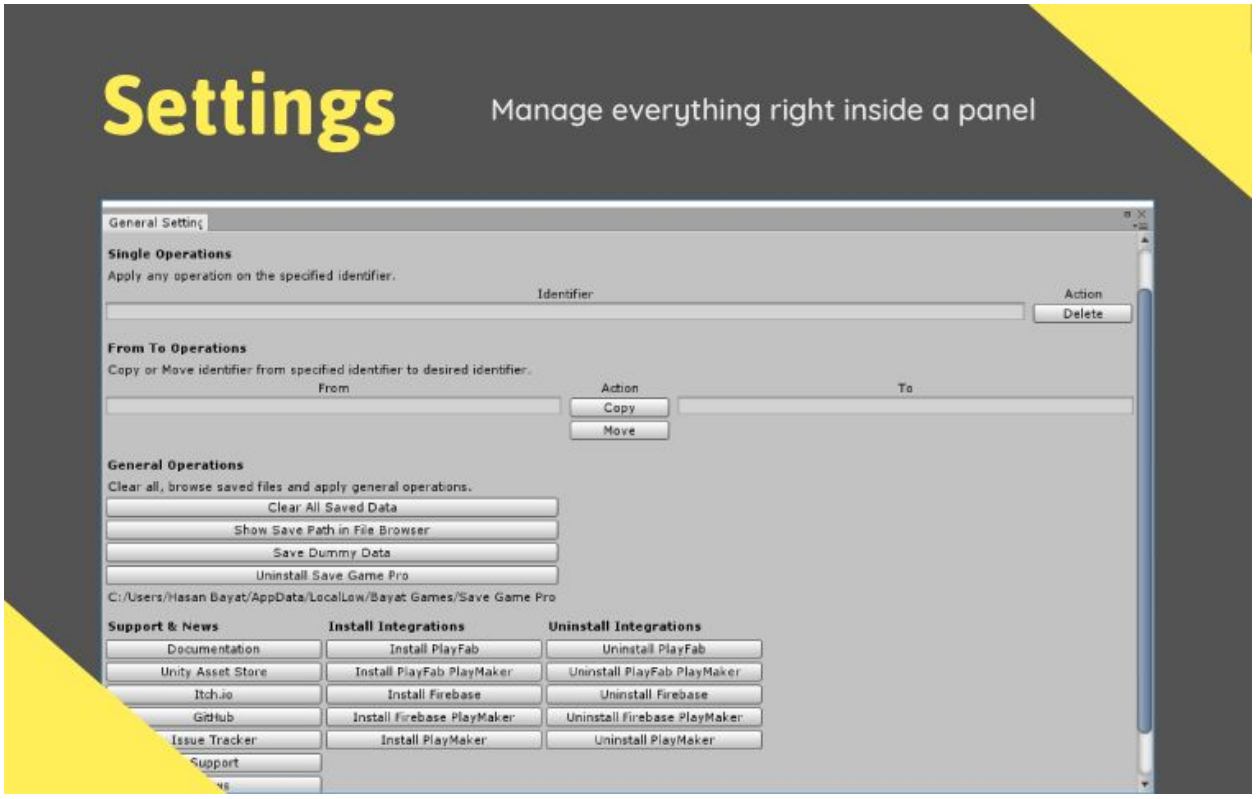
Full Examples



Complete set of Examples included to help you get started easier and faster. Also, each integration comes with it's own examples.
The Example included are:

- Cloud Saving
- Custom Path Saving
- Listing Saved Files
- Saving Collections
- Saving Custom Data
- Saving Game Object
- Saving Runtime Generated Objects
- Saving Simple Data
- Saving Slots
- Saving Texture2D
- Saving Transform

General Settings



Configure, Modify and Apply General operations on Save Game Pro right inside a panel and Install/Uninstall integrations easily by simple clicks:


General Operations
Clear all, browse saved files and apply general operations.
Clear All Saved Data
Show Save Path in File Browser
Save Dummy Data
Uninstall Save Game Pro
C:/Users/Hasan Bayat/AppData/LocalLow/Bayat Games/Save Game Pro
Apply General Operations
Install & Uninstall Integrations
Install Integrations
Install PlayFab
Install PlayFab PlayMaker
Install Firebase
Install Firebase PlayMaker
Install PlayMaker
Uninstall Integrations
Uninstall PlayFab
Uninstall PlayFab PlayMaker
Uninstall Firebase
Uninstall Firebase PlayMaker
Uninstall PlayMaker

PlayMaker Integration

Save Game Pro integrates with PlayMaker fully and completely by adding Save Game Pro API methods as Actions to PlayMaker. Also, each integration includes the PlayMaker Actions as well.

Cross Platform

Save Game Pro supports almost all platforms that Unity supports, we have tested the Save Game Pro examples in:

- Standalone (Windows, Mac, Linux)
- Android (and it should work well on iOS)
- Samsung TV

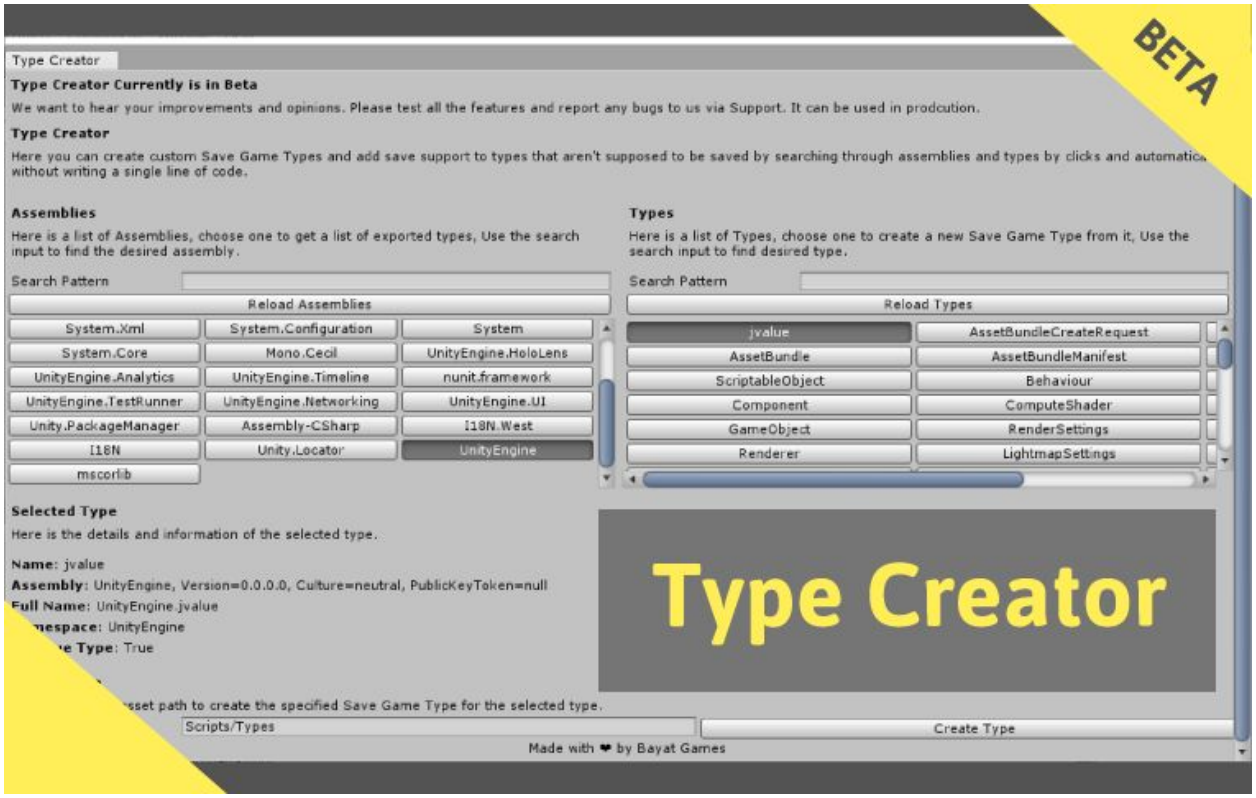
But we know it should work as excepted on **All Unity Platforms**.

Web & Cloud

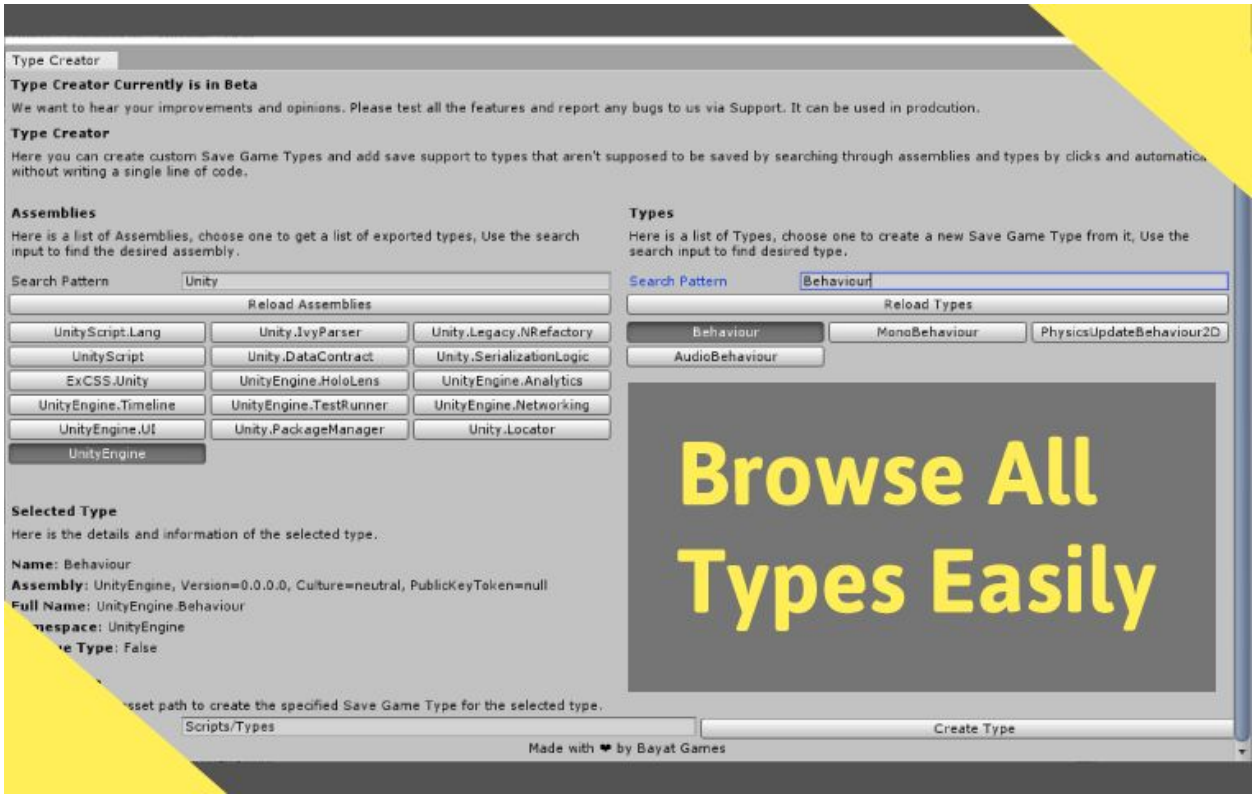
Save Game Pro integrates as well with your Cloud environment such as **PHP** and **Node.js** and supports most popular Database Engines, such as **MySQL** and **MongoDB**, Also, Save Game Pro integrates with **Firebase** and **PlayFab** as well to let you save your game data and sync it across devices.
Here is the list of custom Cloud integrations:

- PHP MySQL
- Node.js MySQL
- PHP MongoDB
- Node.js MongoDB

Custom Types (Type Creator)



Save Game Pro supports Custom Types, that means you can add serialization of Types that aren't supported by Save Game Pro, So you can create these Custom Types easily and automatically using **Type Creator**.
Type Creator provides an easy, fast and helpful interface for Creating and Browsing types, Also, Type Creator includes Search inputs to help you browse all types by searching for the desired Type.



Getting Started

Now, let us get started with Save Game Pro and learn how to use Save Game Pro to save and load the Game Data.
You have many choices for saving and loading the Game Data, you can save and load game data from local storage or Cloud, but we recommend you to use Cloud Save, because the Player data is stored on Cloud and not in the device that is owned by User, and also the User won't lost the data unless he lost the username and password, So you can power your game with Cloud saving and loading, but for the start, we recommend you to use the Local Storage.

Save Game Pro has a feature-rich API that lets you control the saved Data easily and fast, The API methods are:

- Save
- Load
- Exists
- Delete
- Clear
- Move

- Copy
- GetFiles
- GetDirectories

The main methods are Save & Load, that they let you save and load your game data easily anytime, anywhere.
So let us describe each API method separately for more information.

Save

The Save method is used for saving the data using the specified identifier.
Here is a simple usage example:

```
SaveGame.Save ( "score", myPlayer.score );
```

In the example, the player score is saved at the **score** identifier.
Note: The identifier is the filename on file supported platforms such as Standalone and Mobile platforms.
Note: You can use folders in the identifier, check the example below:

```
SaveGame.Save ( "myPlayer/score", myPlayer.score );
```

Load

The Load method is used for retrieving the saved data using the specified identifier and the given type.
Here is a simple usage example:

```
int defaultScore = 0;
myPlayer.score = SaveGame.Load<int> ( "score", defaultScore );
```

In the example, the player score is loaded from **score** identifier using the specified type, and the type is Integer in this example.
Note: The default value is optional.

Exists

The Exists method is used to check if the specified identifier exists or not.
Here is simple usage example:

```
if ( SaveGame.Exists ( "score" ) ) {
	int defaultScore = 0;
	myPlayer.score = SaveGame.Load<int> ( "score", defaultScore );
} else {
	myPlayer.score = 0;
}
```

In the example, It first checks whether the **score** identifier exists or not, if exists then it loads data, if it not exists, it sets the player score to zero.
Note: You can use Default Values, and they work just same as the example above.

Delete

The Delete method is used for deleting the specified identifier.
Here is a simple usage example:

```
SaveGame.Delete ( "score" );
```

In the example, the **score** identifier will be removed from user storage.

Clear

The Clear method is used for clearing user data, that means the Clear method will remove all User saved data completely, so use it at your own risk.
Here is a simple usage example:

```
SaveGame.Clear ();
```

In the example, the **user saved data** will be removed completely.

Move

The Move method is used for moving or renaming identifiers.
Here is a simple usage example:

```
SaveGame.Move ( "score", "myPlayer.score" );
```

In the example, the **score** identifier will be moved or renamed to **myPlayer.score**.
Note: You can use Move method for renaming the identifiers and files.

Copy

The Copy method is used for copying identifiers.
Here is a simple usage example:

```
SaveGame.Copy ( "score", "score.backup" );
```

In the example, the **score** identifier will be copied to **score.backup**.
Note: You can use Copy method to backup saved data.

GetFiles

The GetFiles method is used for retrieving saved files from the specified directory or the root directory if no directory specified.
Here is a simple usage example:

```
FileInfo[] files = SaveGame.GetFiles ();
```

In the example, the saved files at root directory is returned, the root directory is Application.persistentDataPath.
Note: You can specify the directory to retrieve the files from.

GetDirectories

The GetDirectories method is used for retrieving the directories at the specified identifier.
Here is a simple usage example:
DirectoryInfo[] directories = SaveGame.GetDirectories ();
In the example, the directories at root directory is returned, the root directory is Application.persistentDataPath.
Note: You can specify the directory to retrieve the directories form.

Integrations

Save Game Pro has plenty of Integrations available, you can install them via **General Settings** or manually by importing packages from **BayatGames/SaveGamePro/Integrations** folder.
There are integrations available for Save Game Pro, currently we have 3 integration available:

1. PlayMaker
2. Firebase
3. PlayFab

PlayMaker

PlayMaker is a visual scripting tool for Unity that lets you create games without writing a line of code.
To install PlayMaker integration you need to Buy PlayMaker if you don't already own it.

Firebase

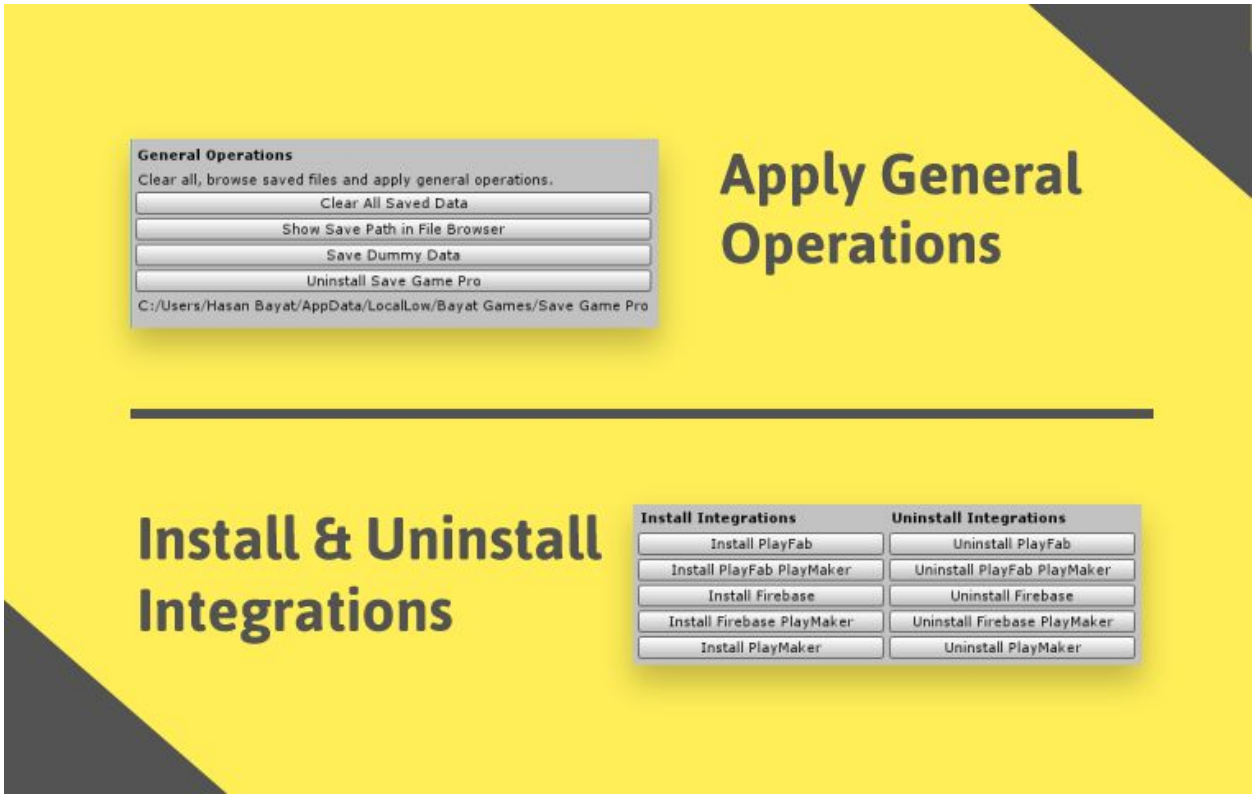
Firebase helps you build better mobile apps and grow your business.
To install Firebase integration you need to Install Firebase Unity SDK if you don't already installed it.

PlayFab

PlayFab is a backend platform for games, delivering powerful real-time tools and services for LiveOps.
To install PlayFab integration you need to Install PlayFab Unity SDK if you don't already installed it.

Automatic Installation

You can install integrations automatically using **General Settings**:



Just click on desired integration button to install it, and then if you want to install it, just click Uninstall button for the desired integration.

Manual Installation

Open your file browser and navigate to your current Unity project and open **Assets/BayatGames/SaveGamePro/Integrations** folder, now open each integration folder you want to install, the Firebase and PlayFab integrations have two unity package, one unity package for Save Game Pro integration and the second one for PlayMaker integration.

Thanks

Thank you for choosing and using Save Game Pro, we would be happy to hear your opinions and ideas.

Resources

Here is a list of some useful resources for you:

- Support and News

- Unity Asset Store
- Trello
- Save Game Pro on GitHub
- Save Game Pro on Unity Asset Store
- Save Game Pro Documentation
- Save Game Pro on Trello

Thanks for reading.

Made with 💛 by Bayat Games