ProPlayerPrefs Editor Documentation

Description

ProPlayerPrefs Editor is a Unity plugin that provides an advanced interface to edit your Unity in-game PlayerPrefs. In addition, it provides advanced encryption and decryption of string values and an in-game class to read and write these values, providing protection of your valuable in-game data from prying eyes.

ProPlayerPrefs Editor contains the following advanced features:

- Editing of Float, Integer and String PlayerPrefs values in both Edit and Play mode
- Auto refresh of values in Play mode on all supported platforms
- Encryption settings stored in convenient Asset files that can be shared between Edit and Play mode
- Simple Play mode component for encrypting and decrypting setting values
- Encryption and decryption of individual string values within the Editor
- Filtering on Key Types, Keys Names and Values
- Full Editor platform support for Windows and Mac systems

ProPlayerPrefs Editor is the most advanced component of its kind available.

Email support requests to support@gigabytesol.com or post to the following Unity Forum thread:

http://forum.unity3d.com/threads/introducing-pro-player-prefs-editor.375628/

Encryption

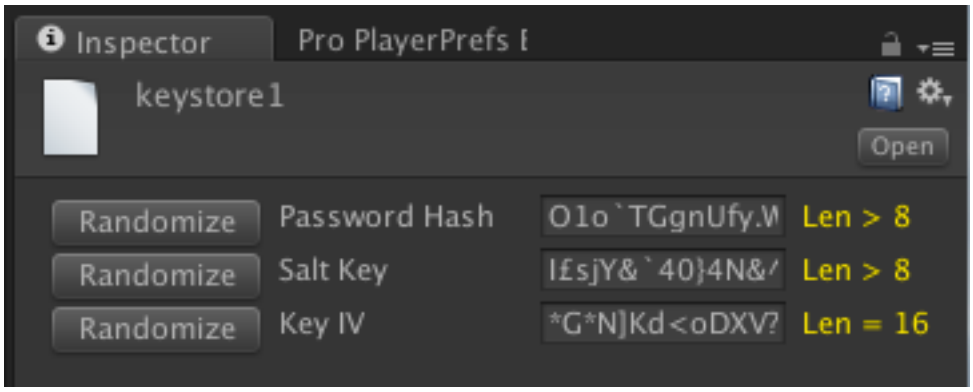
ProPlayerPrefs Editor contains advanced encryption features to keep your PlayerPrefs data safe from analysis and tampering. The encryption used is based upon the .NET RijndaelManaged class with a key length of 256 bits. The encryption features require the following unique properties to encrypt and decrypt string values:

- Password Hash – A string composed of at least 9 characters. The string should be as complex as possible.
- Salt Key – A string composed of at least 9 characters which is cryptographically combined with the Password Hash to make a 256 bit symmetric encryption key.
- Key IV – An string composed of exactly 16 characters which is used to initialize the encryption algorithm.

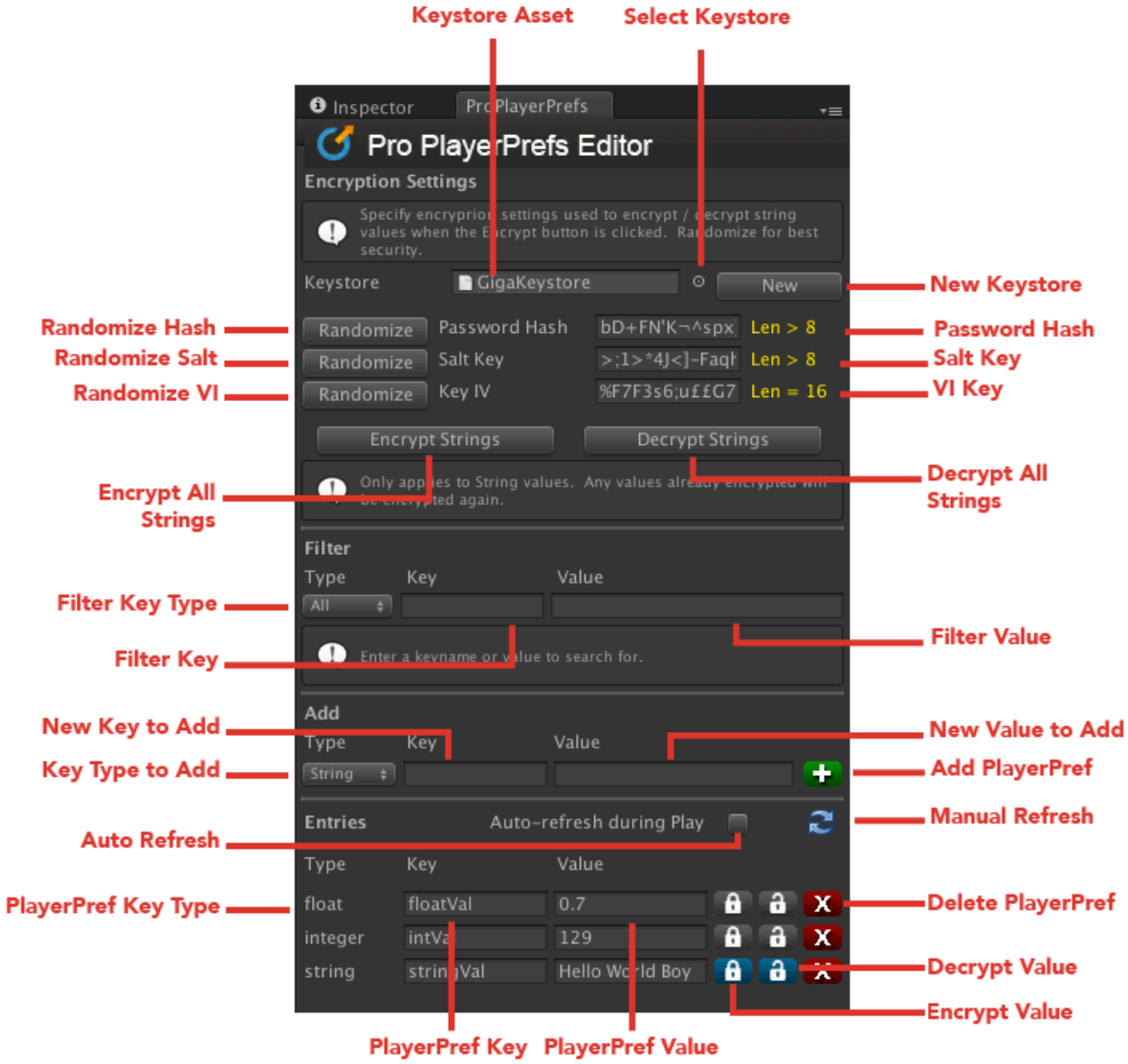
To make generation of the above values simple, and to ensure they remain complex, Pro PlayerPrefs has buttons to generate appropriate values at random for you. When a new Keystore Asset is first created, random complex values are setup automatically.

To keep your valuable encryption settings safe and to enable them to be used in your Game from Play Mode, ProPlayerPrefs Editor saves all of the above values in a single convenient Keystore asset which is stored in your Project Assets folder, and you can create as many of these Keystore assets as you wish. Because the Keystore settings are stored in an Asset rather than the Scene they can be easily re-used across different scenes and even different projects.

When you select a Keystore Asset within the Assets folder, you will be presented with the Keystore Inspector which is shown below:



Interface



- **Encryption Settings**
  - Keystore Asset – The currently selected Keystore Asset. The Keystore Asset stores all your encryption settings as an object in your project so it can be easily used by your game.
  - Select Keystore – Select the Keystore to use from your Project Folder. Will open a dialog showing all existing Keystore assets.
  - New Keystore – Create a new Keystore Asset with random Key settings and store it in your Project Folder.
  - Password Hash – The Password Hash string used to encrypt and decrypt string values. This string must be at least 9 characters long.
  - Salt Key – The Salt value which is combined with the Password Hash to generate a complex encryption key. The Salt Key must be at least 9 characters long.
  - Key IV – The Initialization Vector used to setup the encryption function. The IV must be exactly 16 characters in length.
  - Randomize Hash – Will generate a new complex Password Hash.

- Randomize Salt – Will generate a new complex Salt Key.
- Randomize IV – Will generate a new complex Key IV.
- Encrypt Strings – Will encrypt all PlayerPref strings. It will not change Float or Integer values. Warning: if values are already encrypted then they will become double encrypted.
- Decrypt Strings – Will decrypt all PlayerPref strings. It will not change Float or Integer values.

- **Filter**
  - Filter Key Type – Sets the kind of PlayerPref types which are filtered and displayed. Warning: If not set to All, only the specified Key types will be visible.
  - Filter Key – Filters Key Names by the string entered here.
  - Filter Value – Filters Key Values by the string entered here.
- **Add**
  - Key Type to Add – The type of PlayerPref you want to add which can be either String, Integer or Float.
  - New Key to Add – The name of the PlayerPref you want to add.
  - New Value to Add – The value of the PlayerPref you want to add
  - Add PlayerPref – This button will create a new PlayerPref entry using the Type, Name and Value specified.
- **Entries Editor**
  - Auto Refresh – Enables or Disables Auto-refresh during Play mode. If enabled, the values in the Main Editor will update in near real-time as they are changed in the game.*
  - Manual Refresh – Clicking this button will manually re-read the PlayerPrefs settings.
  - **For every PlayerPref entry, you will see the following fields and buttons**
    - PlayerPref Key Type – The Type of the PlayerPref entry, which can be String, Integer or Float.
    - PlayerPref Key – The PlayerPref Key name. You can edit this for existing entries.
    - PlayerPref Value – The current value of the PlayerPref. You can edit this for existing entries.
    - Encrypt Value – Encrypt the current value using the Keystore settings defined above. Only string values can be encrypted.
    - Decrypt Value – Decrypt the current value using the Keystore settings defined above. Only string values can be decrypted.
    - Delete PlayerPref – Delete the current PlayerPref entry.

*NOTE: If you want to edit the PlayerPref Key or Value during Play Mode, then you must temporarily turn-off Auto Refresh, otherwise the values coming back from the game with override your changes.

Play Mode Encryption / Decryption

Because the Keystore is an asset in your project, using it from your scripts couldn't be any simpler. Simply follow the steps below to access your Encrypted values:

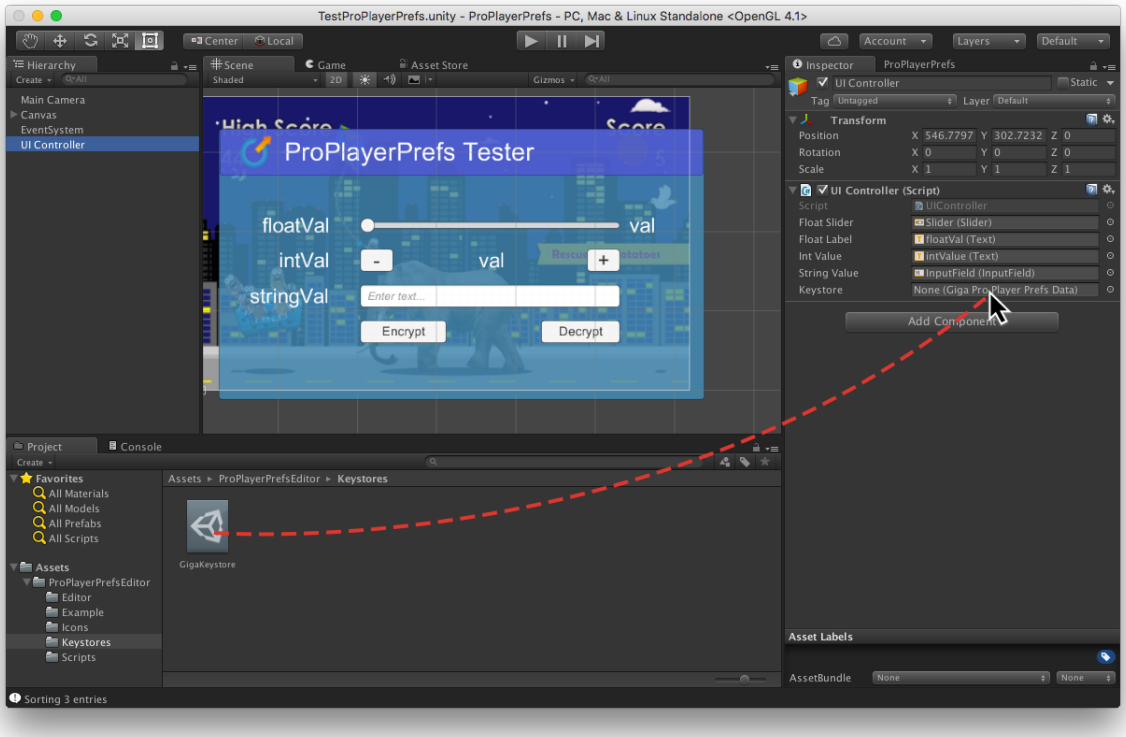
Add a Using directive to your script for GigabyteSol

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using GigabyteSol;
```

Add a public property of GigaProPlayerPrefsData to your script

```
public class UIController : MonoBehaviour {
    public Slider floatSlider;
    public Text floatLabel;
    public Text intValue;
    public InputField stringValue;
    public GigaProPlayerPrefsData keystore;
```

Drag your Keystore Asset onto the property you just defined



Call the keystore.Encrypt and keystore.Decrypt functions

```
public void Decrypt()
{
    if(keystore != null) {
        string s = keystore.Decrypt(stringValue.text);
        PlayerPrefs.SetString("stringVal", s);
        stringValue.text = s;
    } else {
        Debug.LogWarning("No keystore configured on UI Controller");
    }
}

public void Encrypt()
{
    if(keystore != null) {
        string s = keystore.Encrypt(stringValue.text);
        PlayerPrefs.SetString("stringVal", s);
        stringValue.text = s;
    } else {
        Debug.LogWarning("No keystore configured on UI Controller");
    }
}
```

All of the above is demonstrated in the Sample Scene called TestProPlayerPrefs in the following folder:

/Assets/ProPlayerPrefsEditor/Example/