**SLywnow Basic**

[What is that? 1](#_Toc29347)

[Save system 1](#_Toc3150)

[How it works 1](#_Toc8184)

[Usage 1](#_Toc25386)

[GUI 3](#_Toc32368)

[Time savers 5](#_Toc8121)

[FilesSet 5](#_Toc15744)

[FastFind 11](#_Toc3701)

[EasyDo 11](#_Toc20264)

[JSON works 13](#_Toc6919)

[FastJSONTests 13](#_Toc21361)

[JsonHelper 13](#_Toc17491)

[UI works 14](#_Toc6336)

[UIEditor 14](#_Toc1517)

[Attributes 17](#_Toc8186)

[Button Attributes 17](#_Toc2626)

[Show/Hide Attributes 17](#_Toc19085)

# **What is that?**

SLywnow basic is main library for any SLywnow’s assets, like SLM, AutoLang e.tc. To use it type using SLywnow in your script. Asset also include [ZipUtil](https://github.com/tsubaki/UnityZip" \o "ZipUtil).

# **Save system**

### **How it works**

SaveSystemAlt is my alternative to PlayerPerfs, that have no limits of size, have multifiles system, have GUI inside Unity and can save Arrays and Lists.

### **Usage**

StartWork(int i=0) - run SaveSystemAlt by some index (different indexes mean different save files)

UseDebug(bool value) - show debug messages

bool IsWorking() - is SaveSystemAlt working?

SetString(string key, string value) - save some string with some key in current session, to write it into file use SaveUpdatesNotClose() or StopWorkAndClose()

SetInt(string key, int value) - save some int with some key in current session, to write it into file use SaveUpdatesNotClose() or StopWorkAndClose()

SetFloat(string key, float value) - save some float with some key in current session, to write it into file use SaveUpdatesNotClose() or StopWorkAndClose()

SetBool(string key, bool value) - save some bool with some key in current session, to write it into file use SaveUpdatesNotClose() or StopWorkAndClose()

SetArray<T>(string key, T[] array) - save some array of any types with some key in current session, to write it into file use SaveUpdatesNotClose() or StopWorkAndClose()

string GetString(string key, string def=null,bool fromanytype=false) - return string from key, if key not found will return **def** value, use **fromanytype** to get value from any types, not only from strings

int GetInt(string key, int def = 0) - return int from key, if key not found will return **def** value

float GetFloat(string key, float def = 0) - return float from key, if key not found will return **def** value

bool GetBool(string key, bool def = false) - return bool from key, if key not found will return **def** value

T[] GetArray<T>(string key) - return array of any type from key, if key not found will return **def** value

SetValueToUndefined(string key) - change key type to Undefined, Undefined type can be reads/writes from any types (by converted to string)

SetValueToSomeType(string key, SaveSystemSL.SSLTpe type) - change value type to another

bool HasKey(string key) - is key exist in current session?

RenameKey(string key, string newName) - rename some key in current session

DeleteKey(string key) - delete some key in current session

DeleteAll(string key, bool withFile = false) - delete all keys in current session

int IsIndex() - return current session’s index (from StartWork)

StopWorkAndClose() - stop current session and write all changes to file

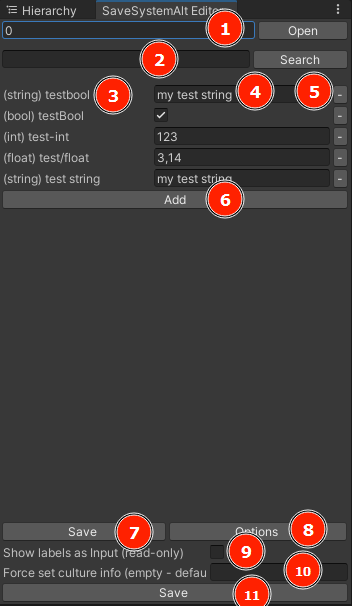
SaveUpdatesNotClose() - write all changes to file but not stop current session

OutputSSAData GetData(string key) - convert some key’s data to OutputSSAData class. Use it to move keys between indexes

WriteData(OutputSSAData input) - add some key from OutputSSAData to current session. Use it to move keys between indexes

### **GUI**

To open GUI press SLywnow/Save System Alt Editor. This editor works only in non-play mode.



1. Index number, press “Open” to load this
2. Search field
3. Name and type of key
4. Value of key
5. Delete key
6. Add new key
7. Save all changes
8. Show/Hide options
9. Show labels of keys as input (only for copy, you can’t edit them here)
10. Set custom CultureInfo, leave it empty for InvariantCulture. Use it if your CultureInfo is different or if you use some other CultureInfo in the game
11. Save and hide options

# **Time savers**

### **FilesSet**

##### **LoadByte**

Loads bytes array of some file  
  
byte[] LoadByte(string path, string name, string format, bool datapath = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**

byte[] LoadByte(string path, bool datapath = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**

##### **SaveStream**

SaveStream(string path, string name, string format, string[] saves, bool datapath = false, bool add = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**saves** - array of strings to write  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**add** - just add new lines, don’t rewrite file

SaveStream(string path, string[] saves, bool datapath = false, bool add = false)  
**path** - full path to file with format  
  
**saves** - array of strings to write  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**add** - just add new lines, don’t rewrite file

SaveStream(string path, string name, string format, string save, bool datapath = false, bool add = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**save** - string to write  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**add** - just add new lines, don’t rewrite file

SaveStream(string path, string save, bool datapath = false, bool add = false)  
  
**path** - full path to file with format  
  
**save** - string to write  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**add** - just add new lines, don’t rewrite file

##### **LoadStream**

Loads data from file in string format. Can return data as array or single string with \n as new lines  
  
string[] LoadStream(string path, string name, string format, bool datapath = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**

string[] LoadStream(string path, bool datapath = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**

string LoadStream(string path, string name, string format, bool datapath = false, bool onlyoneline = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**onlyoneline** - load only first line

string LoadStream(string path, bool datapath = false, bool onlyoneline = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**onlyoneline** - load only first line

##### **LoadSprite**

Load any image file and convert it into Sprite with full rect and center pivot  
  
Sprite LoadSprite(string path, string name, string format, bool datapath = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**

Sprite LoadSprite(string path, bool datapath = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**

##### **SaveTexture**

Save Texture2D as file  
  
SaveTexture(Texture2D input, string path, TextureType format, bool datapath = false)  
  
**input** - imput texture that you want to save  
  
**path** - full path to file without format  
  
**format** - save file in .png or .jpg format?  
  
**datapath** - add Application.persistentDataPath before **path**

##### **CheckFile**

Checks is file or directory exist by some path  
  
bool CheckFile(string path, string name, string format, bool datapath = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**

bool CheckFile(string path, bool datapath = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**

bool CheckDirectory(string path, bool datapath = false)  
  
**path** - full path to directory  
  
**datapath** - add Application.persistentDataPath before **path**

##### **DelStream**

Delete some file  
DelStream(string path, string name, string format, bool datapath = false, bool dirtoo = false, bool forse = false)  
  
**path** - path to directory with file  
  
**name** - name of file without format  
  
**format** - format of file  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**dirtoo** - do you want delete directory where file exist too?  
  
**forse** - delete directory even if it’s contain another files

DelStream(string path, bool datapath = false, bool dirtoo = false, bool forse = false)  
  
**path** - full path to file with format  
  
**datapath** - add Application.persistentDataPath before **path**  
  
**dirtoo** - do you want delete directory where file exist too?  
  
**forse** - delete directory even if it’s contain another files

##### **Other**

string[] GetFilesFromdirectories(string path, string format, bool datapath = false, TypeOfGet type = TypeOfGet.Files) - return string array with info about files/directories in path.  
  
**format** - format filter, leave it empty to all  
  
**datapath** - add Application.persistentDataPath before path  
  
**type**:

* All - return all files and directories in path with full path
* Files - return all files in path with full path
* Directory - return all directories in path with full path
* NamesOfFiles - return all names of files in path without full path, based on Path.GetFileNameWithoutExtension
* NamesOfFilesWithFormat - return all names of files in path with formats (indexes will be same as in NamesOfFiles)
* Formats - return all formats in this directories (indexes will be same as in NamesOfFiles)
* NamesOfDirectories - return all names of directories without full path

CreateDirectory(string path, bool datapath = false) - create directory by path, can create multiple nested directories  
  
**path** - full path to directory  
  
**datapath** - add Application.persistentDataPath before **path**

RenameFile(string path, string oldName, string newName) - rename some file by path  
  
**path** - path to directory with file  
  
**oldName** - current name of file with format  
  
**newName** - new name of file with format

CopyFullDirectory(string sourceDirectory, string targetDirectory) - copy all files from one directory to another

string[] ConcatArray(string[] array1, string[] array2) - merge 2 arrays

### **FastFind**

GameObject InCords(Vector3 position, bool first, string tag = null, string[] blocktags = null) - return GameObject in some position  
  
**position** - position of GameObject  
  
**first** - return first object on this position, Iinstead of last GameObject  
  
**tag** - search GameObject with tag  
  
**blocktags** - ignore GameObject with tags

GameObject[] AllInCords(Vector3 position, string tag = null, string[] blocktags = null) - return all GameObjects in some position  
  
**position** - position of GameObjects  
  
**tag** - search GameObjects with tag  
  
**blocktags** - ignore GameObjects with tags

string GetDefaultPath() - returns path to “Documents” folder in any systems, include android (return sdcard folder)

Transform FindChild(Transform parent, string name) - find child in some transform, check all subparents too, in contrast UnityEngine’s solution

### **EasyDo**

Texture2D byteToTexture(byte[] input) - convert some bytes to Texture2D

string[] UIMultiLineToStringArray(string input, string enter = "\n") - split a string into an array with string as separator  
  
**enter** - separator

string StringArrayToUIMultiLine(string[] input, string enter = "\n") - combine array into string with some separator  
  
**enter** - separator

Color MoveToColor(Color from, Color to, float speed, byte r = 255, byte g = 255, byte b = 255) - smooth move from one color to another. Use this in Update or FixedUpdate  
  
**from** - current color  
  
**to** - color you want to see  
  
**speed** - speed of canges  
  
**r** - coefficient of change of the red channel  
  
**g** - coefficient of change of the gree channel  
  
**b** - coefficient of change of the blue channel

Color MoveToColorWithAlpha(Color from, Color to, float speed, byte r = 255, byte g = 255, byte b = 255, byte a = 255) - smooth move from one color to another. Use this in Update or FixedUpdate  
  
**from** - current color  
  
**to** - color you want to see  
  
**speed** - speed of canges  
  
**r** - coefficient of change of the red channel  
  
**g** - coefficient of change of the gree channel  
  
**b** - coefficient of change of the blue channel  
  
**a** - coefficient of change of the alpha channel

Color SetPositionColor(Color from, Color to, float position) - get color between 2 colors with some position  
  
**from** - color1  
  
**to** - color2  
  
**position** - position between colors

T[] JSONtoArray<T>(string json) - convert some JSON to array of some type

Shuffle<T>(this List<T> list) - random shuffle of elements in the list

Swap<T>(this List<T> list, int i, int j) - swap two elements in the list

# **JSON works**

### **FastJSONTests**

DateTime getTime(string from = "http://worldtimeapi.org/api/timezone/Europe/Moscow") - Get current time from web  
  
**from** - your api source

string getIp(string from = "http://worldtimeapi.org/api/timezone/Europe/Moscow") - Get user’s ip from web  
  
**from** - your api source

### **JsonHelper**

T[] FromJson<T>(string json) - convert some JSON to array of some type

string ToJson<T>(T[] array, bool prettyPrint=false) - convert some array of some type to JSON

Move<T>(this List<T> list, int i, int j) - move element from one position to another in the list

List<T> Clone<T>(this List<T> list) - create copy for some list

Swap<T>(this List<T> list, int i, int j) - swap two elements in the list

# **UI works**

### **UIEditor**

##### **Sprite generators**

Sprite GetSpriteWithColor(Color color, int width = 1, int height = 1, float pivotX = 0.5f, float pivotY = 0.5f) - generate sprite with some color  
  
**color** - color you want to see  
  
**width** - width of future sprite  
  
**height** - height of future sprite  
  
**pivotX** - pivotX of future sprite  
  
**pivotY** - pivotY of future sprite

Texture2D GetTextureWithColor(Color color, int width = 1, int height = 1) - Generate Texture2D with some color  
  
**color** - Color you want to see  
  
**width** - width of future texture  
  
**height** - height of future texture

Sprite GetSpriteWithGradient(Gradient gradient, bool Xdirection, int width = 1, int height = 1, float pivotX = 0.5f, float pivotY = 0.5f) - Generate sprite with some gradient  
  
**gradient** - gradient you want to see  
  
**Xdirection** - Is axis of gradient X?  
  
**width** - width of future sprite  
  
**height** - height of future sprite  
  
**pivotX** - pivotX of future sprite  
  
**pivotY** - pivotY of future sprite

Sprite GetSpriteWithGradient2D(Gradient gradientX, Gradient gradientY, int width = 1, int height = 1, float pivotX = 0.5f, float pivotY = 0.5f) - Generate sprite with some gradients in both axises  
  
**gradientX** - gradient on X axis  
  
**gradientY** - gradient on Y axis  
  
**width** - width of future sprite  
  
**height** - height of future sprite  
  
**pivotX** - pivotX of future sprite  
  
**pivotY** - pivotY of future sprite

##### **Dropdown generators**

Fills dropdown by something, use it to create drowdown in realtime  
**dropdown** - output dropdown link  
  
**enter** - input dropdown

FillDropDownByTextList(out Dropdown dropdown, List<string> strings, Dropdown enter) - Fill dropdown by texts  
  
**strings** - strings list

FillDropDownBySpriteList(out Dropdown dropdown, List<Sprite> sprites, Dropdown enter) - Fill dropdown by sprites  
  
**sprites** - sprites list

FillDropDownBySpriteAndTextList(out Dropdown dropdown, List<Sprite> sprites, List<string> strings, Dropdown enter) - Fill dropdown by sprites and texts  
  
**strings** - strings list  
  
**sprites** - sprites list  
  
Lists must have same size

FillDropDownByColorList(out Dropdown dropdown, List<Color> colors, Dropdown enter) - Create dropdown with colors (usefull for color choice in game)  
  
**colors** - colors list

FillDropDownByColorAndTextList(out Dropdown dropdown, List<Color> colors, List<string> strings, Dropdown enter) - Create dropdown with colors (usefull for color choice in game) and fill it by texts  
  
**colors** - colors list  
  
**strings** - strings list  
  
Lists must have same size

##### **For Unity Events**

Allows you to process data from string, created for using in Unity Events inside editor  
**input** - input string that stored data  
  
**pos** - position of data in string  
  
**space** - separator of line  
  
**def** - default value

string GetStringUIEvent(string input, int pos, char space, string def = null)

int GetIntUIEvent(string input, int pos, char space, int def = 0)

float GetFloatUIEvent(string input, int pos, char space, float def = 0)

bool GetBoolUIEvent(string input, int pos, char space, bool def = false)

Example:  
  
Input string: 1 true myEvent  
  
To get data use:

1. GetIntUIEvent(str, 0, ' ')
2. GetBoolUIEvent(str, 1, ' ')
3. GetStringUIEvent(str, 2, ' ')

# **Attributes**

### **Button Attributes**

Allows you to call the function directly from the editor outside of the play mode. Usefull for some editor-only functions (don’t forget about #if UNITY\_EDITOR) and tests  
  
Based on [EasyButtons](https://github.com/madsbangh/EasyButtons), see usage here

### **Show/Hide Attributes**

Helps you to show/hide fields in the editor without writing CustomEditor for script  
Exept [ShowFromMultiple] all this attributes have same options:  
  
**propertyName** - name of propertyName, recomended to use nameof() here  
  
**checkval** - value to check  
  
**inverse** - inverse check, show if false and hide if true  
  
All types checks:  
  
[ShowFromBool(string propertyName, bool checkval=true, bool inverse=false)]  
  
[ShowFromInt(string propertyName, int checkval, bool inverse=false)]  
  
[ShowFromFloat(string propertyName, float checkval, bool inverse=false)]  
  
[ShowFromEnum(string propertyName, int checkval, bool inverse=false)] - type index in enum as value  
  
[ShowFromString(string propertyName, string checkval, bool inverse=false)]  
  
[ShowFromObjectNotNull(string propertyName, isNull=false)] - check is any object field null or not

##### **Multiple checks**

You can check multiple values using [ShowFromMultiple]. Use it to check few values of some property or many few property with same value, or many property and values.  
  
**string[] propertyName/string propertyName** - name of property or properties (in array) to check, if you use only one property then all values will be checks only with it, if many then indexes must match  
**string[] vals/string vals** - values of property or properties (in array) to check, if you use only one value then all properties will be checks with it, if many then indexes must match  
**string[] types/string types** - types of properties, if all properties has same type then enter only it, if not then enter all types and indexes must match  
**Mode** - mode of checking:

* and - requires all parameters to pass check
* or - requires that only one parameter pass check  
    
    
    
  Examples:  
    
  [ShowFromMultiple("QteType", new string[2] { "3", "4" }, "enum", ShowFromMultipleAttribute.mode.or)] - show field if parameter QteType in 3rd ot 4th position  
    
  [ShowFromMultiple(new string[2] { nameof(movePattern), nameof(lookAtThePlayer) }, new string[2] { "3", "true" }, new string[2] { "enum", "bool" }, ShowFromMultipleAttribute.mode.and)] - show field if movePattern parameter is in 3rd position and lookAtThePlayer parameter is true