Edge Filesystem Reference Guide

by XGASOFT

1. Introduction **3**
2. Buy Now **3**
3. Download PDF **3**
4. Changelog **3**
5. Reference Guide **4**
5.1. The Game Timer **4**
5.1.1. edgefs_init_timer **5**
5.1.2. edgefs_get_timer **5**
5.2. Saving **5**
5.2.1. edgefs_save_date **5**
5.2.2. edgefs_save_time **6**
5.2.3. edgefs_save_timeplayed **6**
5.2.4. edgefs_save_var **6**
5.2.5. edgefs_save_screenshot_temp **7**
5.2.6. edgefs_save_screenshot **7**
5.3. Loading **7**
5.3.1. edgefs_load_date **7**
5.3.2. edgefs_load_time **8**
5.3.3. edgefs_load_timeplayed **8**
5.3.4. edgefs_load_var **8**
5.3.5. edgefs_load_screenshot **8**
5.4. edgefs_delete **9**
5.5. Functions **9**
5.5.1. file_encrypted **9**
5.5.2. file_text_encrypt **9**
5.5.3. file_text_decrypt **9**
6. End-User License Agreement ("EULA") **10**

1. Welcome to Edge Filesystem

Thank you for choosing Edge Engine Filesystem. **Edge Engine** is a fully cross-platform, modular framework built to augment Game Maker Studio with pre-made code and assets that serve as the foundation for a wide variety of game genres. All Edge Engine modules feature creative, human-readable code with helpful notations throughout, making them both powerful and easy to use.

Edge Filesystem (or Edge FS) is a powerful solution for saving and loading data that makes storing virtually any type of variable content as simple as naming a file to write to. Save data can be encoded and encrypted to prevent tampering with the flip of a switch, and Edge FS even has special functions just for recording the date, time, and amount of time invested into a given save file.

In this guide you will learn how to save and load data with Edge FS, as well as how to properly implement timestamps and use the new `file_*` functions Edge FS adds to GML.

2. Buy Now(https://marketplace.yoyogames.com/assets/2488/save-filesystem-edge-engine)

3. Download PDF(https://docs.xgasoft.com/wp-content/uploads/sites/2/edge-filesystem-reference-guide-4.pdf)

4. What's New

v1.2.1

• Updated documentation to new format

v1.2

• Added support for high-level encryption for `ds_map` data structures

v1.1

• Updated syntax to combine filename and extension into a single argument

• New permanent price reduction—33% off previous price!

v1.0

• Initial release

5. Overview

Giving users the ability to preserve their activities and restore them later is a fundamental necessity of many different types of programs, yet often an overlooked challenge for new programmers. With Edge FS it's easier than ever, but as always, there's no magic button that does all the work for you, the developer. Effective use of Edge FS will require effective use of both custom and built-in variables. Because these variables will differ greatly from developer to developer and project to project, there's no instant, one-size-fits-all solution. However, with Edge FS in your toolbelt it's fast and simple to save your data *one variable at a time* and later load it all back to where it belongs. *What* you save is up to you. *How* you save it is Edge FS.

For example, one of the most basic building blocks of creating a save file is storing the last room the player was in. With Edge FS, this is boiled down into one simple line of code:

```
edgefs_save_var(room, ”room”, ”my_save.dat”, true);
```

But what if you want to have a separate room for your save menu? In this case, the 'what' changes slightly, because you want to save the room the *player was last in*, not the save menu room itself. Fortunately, there are numerous ways to go about solving this problem.

Let's begin with a variable. We'll call it `global.last_room`. To store the room we want, all we have to do is run the code `global.last_room = room;` while the desired room is active. This can happen in a number of places—the player object, the room creation code, or even an independent object just for this singular purpose. So long as the code is *only run in game rooms* and not in menu rooms it will keep a running record of the last *game* room the player was in, menus excluded. With such a variable in place, we can now easily save the room we want from any other room where the variable is not redefined. The code remains mostly the same, but with one key difference: we save our *variable* as the room rather than the room itself. Like so:

```
edgefs_save_var(global.last_room, ”room”, ”my_save.dat”, true);
```

And now that we have our room saved, we can easily load it again at any time with the `room_goto` command and `edgefs_load_var`:

```
room_goto(edgefs_load_var(“room”, ”my_save.dat”));
```

Therefore, to take full advantage of what Edge FS has to offer, you'll want to gain a solid understanding of how both GameMaker Studio and Edge FS functions work. For the latter, refer to the rest of this reference guide.

5.1. The Game Timer

To save accurate time stamps with the total time played across all saved and loaded sessions, Edge FS employs a single custom object called `obj_timer`. This object requires only a pair of tiny scripts to function, but it is vital that it be placed in the *very first* room of your game project and that the 'persistent' box is checked in the object editor. This enables `obj_timer` to exist in every game room and silently track the total time played in your game. It should also be recreated upon performing a load operation. For example:

```
if !instance_exists(obj_timer) {

   instance_create(0, 0, obj_timer);

}
```

Note that this process is only necessary for persistent objects, and if no timestamp is desired, `obj_timer` can be ignored altogether.

5.1.1. edgefs_init_timer()

```
edgefs_init_timer();
```

Initializes the game timer with the count set to the current session time, in microseconds.

This script should be run in the Create event of a dedicated game timer object. While it will attempt to compensate for any delays, this object should still be run first thing in the game and set to 'persistent' for accurate readings across all rooms.

5.1.2. edgefs_get_timer()

```
edgefs_get_timer();
```

Increments the game timer initialized with `edgefs_init_timer`, in microseconds. Should be run in the Step event of a dedicated game timer object.

This script will adapt to any room speed and account for lag to save accurate readings. The running object for this script should be the first instance created in the game and set to 'persistent' for keeping time across all game rooms.

5.2. Saving

Saving data in Edge FS comes in two parts: saving variable data, and saving a timestamp marking when that data was last saved. Because different projects may have different needs, timestamp data is split into three separate scripts for both saving and loading. This allows for saving only one or two types of timestamp data and likewise loading back only one or two figures or another. If no timestamp is desired, these three scripts can be ignored altogether.

Note that by default on Windows platforms saved files are stored under `%localappdata%GAME_NAMEefs_data`. While any extension can be used for save files, output files will always be readable by standard text editors. This can be useful when debugging, as you will be able to read exactly what data has been saved and where. Naturally, encrypted output files will not be *human*-readable, even if they are technically readable by text editors.

5.2.1. edgefs_save_date(fname, encrypt)


```
edgefs_save_date(“mysave.sav”, true);
```

Saves the local date in MM/DD/YYYY format to the specified save file, optionally encrypting the file when complete. The **fname** parameter specifies the filename **including the extension** to be saved to, as a string (enclosed in quotes). Secondarily, the **encrypt** parameter is a true/false value that will either enable or disable encrypting the output file to prevent tampering by users. It is recommended to set this value to ‘true’ except when debugging.

> Note: The default date format in this script can easily be altered by editing the script and changing the order in which the built-in `current_day`, `current_month`, and `current_year` variables appear under the comment *`//Write timestamp data`*.

5.2.2. edgefs_save_time(fname, encrypt)

```
edgefs_save_time(“mysave.sav”, true);
```

Saves the local time in HH:MM format to the specified save file, optionally encrypting the file when complete. The **fname** parameter specifies the filename **including the extension** to be saved to, as a string (enclosed in quotes). Secondarily, the **encrypt** parameter is a true/false value that will either enable or disable encrypting the output file to prevent tampering by users. It is recommended to set this value to ‘true’ except when debugging.

5.2.3. edgefs_save_timeplayed(fname, encrypt)

```
edgefs_save_timeplayed(“mysave.sav”, true);
```

Saves the total time played in microseconds to the specified save file, optionally encrypting the file when complete. ‘Total time played’ in this case refers to the combined time spent in the current session plus the time spent in any loaded previous session. The **fname** parameter specifies the filename **including the extension** to be saved to, as a string (enclosed in quotes). Secondarily, the **encrypt** parameter is a true/false value that will either enable or disable encrypting the output file to prevent tampering by users. It is recommended to set this value to ‘true’ except when debugging.

Note that the functionality of this script depends on the existence of a dedicated game timer object (such as the included `obj_timer`) running the `edgefs_init_timer` and `edgefs_get_timer` scripts.

5.2.4. edgefs_save_var(variable, key, fname, encrypt)

```
edgefs_save_var(myvar, “myvar”, “savefile.sav”, true);
```

Saves the data contained in the input variable to the specified save file, optionally encrypting the file when complete. This data is identified in two steps: first, the **variable** parameter refers the variable to saved itself, and second, the **key** parameter refers to what the variable will be saved *as* within the save file. The key parameter must be input as a string (enclosed in quotes). Typically, both parameters should be named the same, but this is not required—it is only recommended to simplify the loading process later, as the name of the data being loaded will match the variable it is being loaded back into. The **fname** parameter specifies the filename **including the extension** to be saved to, as a string (enclosed in quotes). And finally, the **encrypt** parameter is a true/false value that will either enable or disable encrypting the output file to prevent tampering by users. It is recommended to set this value to ‘true’ except when debugging.

This script supports saving real and string variables, 1D and 2D arrays, all types of data structure, and surfaces.

Note: As of version 1.2, this script supports high-level encryption for `ds_maps` using the built-in `ds_map_secure_save` function. This function uses a much more secure form of encoding than base64 and is good for storing sensitive information such as passwords or in-app purchase data. However, it also means files containing an encrypted `ds_map` should **not** be decrypted with the `file_text_decrypt` script! This will cause `edgefs_load_var` to attempt to load the encrypted `ds_map` as already decrypted when it in fact uses a different type of encryption altogether, causing the game to crash.

5.2.5. edgefs_save_screenshot_temp()

```
edgefs_save_screenshot_temp();
```

Takes a screenshot and saves it as a temporary file to later be saved permanently using `edgefs_save_screenshot`. This allows `edgefs_save_screenshot` to record screenshots taken in other rooms than the one it is being run in. To prevent data buildup, only one temporary screenshot can be stored at a time—if this script is run multiple times, the previous screenshot will be overwritten. The temporary screenshot will be deleted when saved permanently.

5.2.6. edgefs_save_screenshot(fname, use_temp)

```
edgefs_save_screenshot("mysave.sav", true);
```

Takes a screenshot and saves it to be *associated with* the specified save file as a preview image for that saved data. This script differs from other Edge FS save scripts in that encryption is not supported, therefore while the second parameter is still a true/false value, `edgefs_save_screenshot` shares only one parameter in common with the rest of Edge FS: the **fname** parameter, which specifies the filename **including the extension** to be saved to, as a string (enclosed in quotes). The unique **use_temp** parameter is a true/false value that enables or disables saving a temporary screenshot created with `edgefs_save_screenshot_temp` rather than taking a new screenshot. If 'use_temp' is set to 'true' but no temporary screenshot is found, a new one will be taken instead regardless.

Note that unlike other Edge FS scripts, `edgefs_save_screenshot` does not write data directly to the filename specified. Instead the screenshot will be stored in a separate data folder associated with the specified save file. Screenshots taken with this script, then, are not intended to be used for general purposes, but specifically as thumbnails for displaying on save/load menu screens. To save general purpose screenshots, use the `screen_save` and `screen_save_part` functions built-in to Game Maker Studio.

5.3. Loading

Loading data in Edge FS follows a very similar process as saving, but with a few key differences. First and foremost, all loading scripts return *data*, meaning they must be run in conjunction with a variable to be returned into. In other words, rather than just be run as 'script(X)', they must be run as 'variable = script(X)'. Second, loading scripts detect encryption automatically, therefore the encryption parameters of save file scripts are absent here.

5.3.1. edgefs_load_date(fname)

```
date = edgefs_load_date("mysave.sav");
```

Loads the local date in MM/DD/YYYY format from the specified save file and returns it as a string. The **fname** parameter specifies the filename **including the extension** to be loaded, as a string (enclosed in quotes). If the

save file or saved data does not exist it will be created with default values. See the Loading overview for more information on loading data with this script.

5.3.2. edgefs_load_time(fname)

```
time = edgefs_load_time(“mysave.sav”);
```

Loads the local time in HH:MM format from the specified save file and returns it as a string. The **fname** parameter specifies the filename **including the extension** to be loaded, as a string (enclosed in quotes). If the save file or saved data does not exist it will be created with default values. See the Loading overview for more information on loading data with this script.

5.3.3. edgefs_load_timeplayed(fname, set_time)

```
timeplayed = edgefs_load_timeplayed(“mysave.sav”, false);
```

Loads the total time played in microseconds from the specified save file and returns it as a string with the format HH:MM:SS. ‘Total time played’ in this case refers to only the time spent in the loaded previous session.
The **fname** parameter specifies the filename **including the extension** to be loaded, as a string (enclosed in quotes). If the save file or saved data does not exist it will be created with default values. Secondarily, the unique **set_time** parameter is a true/false value that enables or disables setting the game’s current session time to that of the loaded save file. This is made optional to allow drawing the total time played as a string without applying it to the game (e.g. for displaying time played as part of the label on a save slot in a save menu). Note however that upon performing a full load operation this value must be set to ‘true’ for an accurate total time played count. See the Loading overview for more information on loading data with this script.

Note that the functionality of this script depends on the existence of a dedicated game timer object (such as the included `obj_timer`) running the `edgefs_init_timer` and `edgefs_get_timer` scripts.

5.3.4. edgefs_load_var(key, fname)

```
myvar = edgefs_load_var(“myvar”, “mysave.sav”);
```

Loads and returns the data contained in the input variable key from the specified save file. The variable **key** parameter refers to the name of the saved data as a string—typically matching the name of the variable it was originally saved from. The **fname** parameter specifies the filename **including the extension** to be loaded, as a string (enclosed in quotes). If the save file or saved data does not exist it will be created with default values. See the Loading overview for more information on loading data with this script.

This script supports loading real and string variables, 1D and 2D arrays, all types of data structure, and surfaces. Any necessary initialization of these data types is performed as part of the load operation.

5.3.5. edgefs_load_screenshot(fname)

```
screenshot = edgefs_load_screenshot(“mysave.sav”);
```

Loads a screenshot *associated with* the specified save file and returns it as a sprite. The **fname** parameter specifies the filename **including the extension** to be loaded, as a string (enclosed in quotes). See the Loading overview for more information on loading data with this script.

Warning: this script is potentially volatile and can result in memory leaks and reduced performance when misused. Care should be taken not to run this script continuously, and sprites loaded with this script should **always** be unloaded when no longer in use by running the built-in `sprite_delete` function. **Loaded screenshots that are not unloaded are liable to create a memory leak and crash the game!**

Note that unlike other Edge FS scripts, `edgefs_load_screenshot` does not load data directly from the filename specified. See `edgefs_save_screenshot` for more information.

5.4. edgefs_delete(fname)

```
edgefs_delete(“mysave.sav”);
```

Physically erases all saved data associated with the specified save file from the system storage. This includes variables as well as external data such as screenshots and surfaces. The **fname** parameter specifies the filename **including the extension** to be deleted, as a string (enclosed in quotes).

Use with caution! Naturally, running this script is irreversible!

5.5. Functions

On top of providing specialized, easy-to-use scripts designed for providing specific functionality, Edge FS also extends the basic functionality of GameMaker Studio with a few generic functions that fit in line with existing commands. These scripts can be used manually or even independently of Edge FS, but using them is not required in order to get the most out of Edge FS. Edge FS simply relies on these scripts for its core functionality and also leaves them available for advanced users to expand GameMaker Studio’s capabilities even further. While it is not necessary for you to use the following functions yourself, they must be present in order for Edge FS to function.

5.5.1. file_encrypted(fname)

```
if file_encrypted(working_directory + “file.txt”) == true { … }
```

Checks whether the given file is encrypted and returns either ‘true’ or ‘false’. The **fname** parameter refers to the full path of the input file **including the extension**, as a string (enclosed in quotes). If the input file does not exist, ‘false’ will be returned by default.

5.5.2. file_text_encrypt(fname)

```
file_text_encrypt(working_directory + “file.txt”);
```

Encrypts the given file using base64. The **fname** parameter refers to the full path of the input file **including the extension**, as a string (enclosed in quotes). Note that while virtually any file can be encrypted with this script, it is not guaranteed that non-text filetypes will successfully decrypt without breaking the file. If the input file is already encrypted no action will be taken.

5.5.3. file_text_decrypt(fname)

```
file_text_decrypt(working_directory + “file.txt”);
```


Decrypts the given file using base64. The **fname** parameter refers to the full path of the input file **including the extension**, as a string (enclosed in quotes). If the input file is already decrypted no action will be taken.

Note that this script should **not** be used on files containing an encrypted ds_map, as ds_maps are encrypted with a stronger method than base64, and attempting to load a decrypted file containing a still-encrypted ds_map will cause the game to crash!

6. End-User License Agreement ("EULA")

LAST UPDATED: 02/14/2019

We know that reading EULAs isn't very exciting, but this is important. Please take your time to review and ensure you understand the terms of this document before proceeding to use XGASOFT products in your own work.

If you have any questions or concerns about the terms outlined in this document, please feel free to contact us at contact@xgasoft.com or by visiting our Contact & Support page.

LICENSE AGREEMENT

This License Agreement (the "Agreement") is entered into by and between XGASOFT (the "Licensor"), and you (the "Licensee"). This agreement is legally binding, and becomes effective when you purchase and/or download a free product from XGASOFT or authorized third-party distributors. If you do not agree to the terms of this Agreement, do not purchase, download, or otherwise use XGASOFT products.

In order to accept this Agreement, you must be at least eighteen (18) years of age or whatever age is of legal majority in your country. Otherwise, you must obtain your parent's or legal guardian's approval and acceptance of this Agreement in your stead. XGASOFT accepts no liability for your failure to meet this requirement.

XGASOFT delivers content through authorized third-party distributors, each of which may require its own separate End-User License Agreement ("EULA"). XGASOFT accepts no liability for the terms of any third-party agreements, nor for your failure to meet them.

STANDARD LIFETIME LICENSE

This is a license, not a sale. XGASOFT retains ownership of all content (including but not limited to any copyright, trademarks, brand names, logos, software, images, animations, graphics, video, audio, music, text, and tutorials) comprising digital products and services offered by XGASOFT (the "Property"). All rights not expressly granted are reserved by XGASOFT.

Subject to your acceptance of the terms of this Agreement, XGASOFT grants you a worldwide, revocable, non-exclusive, non-transferable, and **perpetual** license to download, embed, and modify for your own purposes XGASOFT Property solely for incorporation with electronic applications and other interactive media, including both commercial and non-commercial works, wherever substantial value has been added by you.

Any source code included as part of XGASOFT Property must be compiled prior to redistribution as an incorporated work, whether for commercial or non-commercial purposes.

PATREON LIMITED LICENSE

When you register as a recurring financial supporter of XGASOFT through Patreon (Patreon, Inc.), XGASOFT may provide free access to XGASOFT Property as a reward, subject to the terms of each contribution tier. This is a privilege, not a right.

XGASOFT retains ownership of all content (including but not limited to any copyright, trademarks, brand names, logos, software, images, animations, graphics, video, audio, music, text, and tutorials) comprising digital products and services offered by XGASOFT (the “Property”). All rights not expressly granted are reserved by XGASOFT.

Subject to your acceptance of the terms of this Agreement, XGASOFT grants you a worldwide, revocable, non-exclusive, non-transferable, and **temporary** license to download, embed, and modify for your own purposes XGASOFT Property solely for incorporation with electronic applications and other interactive media, including both commercial and non-commercial works, wherever substantial value has been added by you.

Any source code included as part of XGASOFT Property must be compiled prior to redistribution as an incorporated work, whether for commercial or non-commercial purposes.

This license shall remain effective for the duration of your subscription to XGASOFT through Patreon. In the event that you cancel or reduce your contribution to a lower tier not qualifying for free access to XGASOFT Property, this license will be considered revoked and void for any and all public commercial and non-commercial activities. In order to continue using XGASOFT Property publicly, you must purchase a standard lifetime license.

This limitation shall not be applied retroactively, so that any existing, complete, and publicly available commercial and non-commercial properties using XGASOFT Property will not be considered in violation of this agreement.

SINGLE-USER

This Agreement grants one (1) user an applicable license to use XGASOFT Property on unlimited devices. This license may not be transferred, shared with, or sold to other users.

However, you, the Licensee, may use XGASOFT Property along with a team or company of collaborators wherever substantial value has been added by you.

This limitation does not extend a license to other users. For any works unrelated to you, collaborators must purchase separate licenses.

MODIFICATIONS

In accordance with the terms of this Agreement, you may freely modify, or alter the functionality of XGASOFT Property exclusively for your own use.

Modifying the Property will not terminate your license, however XGASOFT cannot guarantee the quality and functionality of modified versions of the Property, nor its compatibility with other products.

XGASOFT accepts no liability for any loss or damage incurred by the modified Property, and reserves the right to refuse technical support for the modified Property.

Modifications made to XGASOFT Property in no way represent a change of ownership of the Property.

You may not reverse-engineer XGASOFT Property for the purpose of commercial exploitation which may be in competition with XGASOFT.

MUTABILITY

License fees are determined for each product and service on a case-by-case basis, and XGASOFT reserves the right to change fees on the Property with or without prior notice.

XGASOFT reserves the right to modify, suspend, or terminate this Agreement, the Property, or any service to which it connects with or without prior notice and without liability to you, the Licensee.

LIABILITY

By using XGASOFT Property, you agree to indemnify and hold harmless XGASOFT, its employees, and agents from and against any and all claims (including third party claims), demands, actions, lawsuits, expenses (including attorney's fees) and damages (including indirect or consequential loss) resulting in any way from your use or reliance on XGASOFT Property, any breach of terms of this Agreement, or any other act of your own.

This limitation will survive and apply even in the event of termination of this Agreement.

GOVERNING LAW AND JURISDICTION

This Agreement shall be governed by and interpreted according to the laws of the United States of America and the State of Kansas.

If any provision of this Agreement is held to be unenforceable or invalid, such provision will be changed and interpreted to accomplish the objectives of such provision to the greatest extent possible under applicable law, and the remaining provisions will continue in full force and effect.

CONCLUSION

This document contains the whole agreement between XGASOFT and you, the Licensee, relating to the Property and licenses thereof and supersedes all prior Agreements, arrangements and understandings between both parties regarding XGASOFT Property and licenses.

