

readme.txt

IdStorage Manager v1.1 by Chilly willy

=====

This is my latest attempt to make dealing with the idstorage keys easier, as well as provide extra features. Currently, this app will dump all existing keys, verify the keys against files on the memory stick, and display individual keys in hex or ascii.

Installation

=====

To install IdStorage Manager on a PSP with custom firmware, make a directory in the GAME (or GAME150 for custom firmwares) directory called idstoragemgr (actually, this name doesn't matter at all) and copy EBOOT.PBP, idstorage.prx, and the empty keys directory into it.

Usage

=====

To use Id Storage Manager, put any key files you wish to verify or write into the idstoragemgr/keys directory. These files must have a file name of the form 0x0041.bin, where the "0041" is the key number in hexadecimal. If you dump the keys on the PSP, the files will be dumped to that same directory in the format described. If you don't understand what I mean, just dump the keys and look at the resulting filenames.

Run IdStorage Manager from the XMB like any other homebrew. You'll see a title screen for a few seconds, then be taken to the status screen. Note that the status screen hasn't been done yet.

From the status screen, you can perform the other IdStorage Manager functions by pressing the appropriate button. Button mappings are printed at the top of the display. Press the Circle button to exit back to the XMB. Press the Square button to dump the keys to your memstick. Press the Triangle button to view individual keys. Finally, press the Cross button to verify and fix the keys using files on the memstick.

While viewing the individual keys, press the Triangle button to switch the view between hexadecimal and ASCII. Press the Circle button to return to the status screen. Press Right/Up/RTrigger to advance 1/16/256 keys, and Left/Down/LTrigger to go back 1/16/256 keys. Although it says press the Cross button to edit, that isn't done yet so it does nothing.

While verifying keys, it will print which key is being checked and whether it passes or fails the comparison against the file on the memstick. If it fails, it asks you to press the Circle button to skip over this key (hence leaving it alone), or to press the Cross button to write the contents of the file to the IdStorage key. Once it is done verifying, it will return to the status screen. If you used this to fix keys, you can press the Cross button again to verify the keys, ensuring the fixed keys were written and pass the verification this time.

Let's say you wished to restore a PSP back to stock. You would run this app on a stock PSP and dump the keys. You would then run this app on the PSP you wish to restore using the keys from the stock PSP. Any keys that are different will ask you to correct them. If you did something like this, you had better be sure you have a firmware installed that will work with the stock keys! For example, if you have a TA-082 with 1.50 (and maybe the 3.xx custom firmware) and you restored the keys back to stock, the 1.50 firmware would fail to load on a restart. You have been warned!

Warning

=====

Although I've tried to make this program bug-free and safe to use, it has the possibility of bricking your PSP if you don't use it correctly. Viewing and dumping keys should never cause a problem, but verifying/fixing keys

readme.txt

can brick the PSP easily if you don't know what you are doing. Even then, there is a remote possibility the PSP could be bricked by fixing keys. Nothing is 100% safe. One thing I've done here is not check the battery level... it takes less than a minute to write keys, so I don't think it's really needed. However, if you run on batteries, you need to make sure they are charged to a decent level before fixing keys! You have been warned again! I might add a battery check into later versions... I'll have to see what people have to say about this issue. Any other operation by IdStorage Manager should be safe regardless of the battery level. If your battery goes dead while dumping keys, the worst that happens is you get one corrupted file on the memstick.

Acknowledgements

=====

This program was made possible by the efforts of many people. I'd like to thank Stapol, Mathieulh, harleyg, JasOnuk, and Dark_Alex. If anyone else thinks they deserve some credit, let me know. :)