CSave: Installation, Configuration, and Operation

HS Software

1. CSave Package Contents

CSave includes several useful programs:
CSaveGUI - ncurses-based UNIX GUI
CSaveMenu - CSave Main Menu **(start here)**
SaveToProto - Convert saves to protobuf format
ProtoToSave - Convert edited protobuf format files to BL3 saves
ProfileToProto - Convert profile saves to protobuf format
ProtoToProfile - Convert edited protobuf format profile to BL3 saves
SaveGenerate - Protobuf format save editor
ProfileGenerate - Protobuf format profile editor
SaveConvert - Convert BL3 saves to other platforms, and change the bitmask
LookupTables - required files for item editing.
Scripts - CSave script files, mostly for SaveGenerate.

2. Installation

2.1. Prebuilt Builds - Windows (Cygwin-compiled)

This is the recommended way to get CSave up and running. You need to do two things: First, download everything in the build_win64 folder and put all of those files into a folder somewhere. Then, create a folder named "LookupTables" in the folder where you put the CSave executables. Download everything from the LookupTables folder on GitHub and copy them into your LookupTables folder. It is MUCH easier just to use GitHub's "download zip" function to get CSave. That way, you can drag-and-drop the files to anywhere you want. Please note that CSave *will not function correctly* if you do not download the LookupTables properly. Next, open up a command prompt, and use the "cd" command to navigate to where your CSave folder is. It is easiest to open the CSave folder on Windows Explorer, click up top where the path is displayed, and copy and paste the path after the "cd" command on your command prompt.

2.2. Prebuilt Builds - Windows (MSVC-compiled)

As an alternative to the Cygwin-built CSave binaries, you may also use the MSVC-compiled version of CSave (MSVC = Microsoft (R) Visual Studio (TM) C/C++). Follow the instructions above, and be sure to download the LookupTables as well. The MSVC version of the CSave binaries is functionally equalivent to the "standard" Cygwin binaries.

2.3. Prebuilt Builds - OS/2 version 2+

CSave supports OS/2 2.0 and newer as valid OSes to run itself on. First, go to the CSave GitHub page and either download the files in the "build_os2v2" folder and copy those over to your OS/2 system. Please note that there are minor issues running CSave on an SMP OS/2 system. Crashes have been reported on OS/2 2.11 SMP, OS/2 Warp 3.0 SMP, and OS/2 Warp 4.52 (W4 kernel). Remember to also download the LookupTables fileset as well, or you will not be able to edit items at all. Yes, I know good and well that

nobody’s actually going to use CSave on OS/2, it’s mainly a compile verification step, as if it can be made to work on OS/2, it’ll run on pretty much anything.

2.4. Compiling - Unix, Linux, BSD, and Windows (Cygwin)

The first thing you need to do is set up your build environment - install protobuf, gcc, pkg-config, make, and git. After you’ve installed everything, retrieve the source code for protobuf-c by executing "git clone https://github.com/protobuf-c/protobuf-c.git". Next, enter the source code directory (cd protobuf-c) and execute "./autogen.sh", "./configure", "make -j4", and "make install". If the shell script isn’t executable, do a "chmod +x autogen.sh" before attempting to compile. Once you have successfully installed protobuf-c, return to your home directory - execute "cd" - and download the CSave source code with "git clone https://github.com/HackerSmacker/CSave.git". Enter the CSave source code directory with "cd CSave" and compile it with "make". Continue to the Operations section. If you ran into issues compiling something, here are some notes:

1 If you are on a true C89-only system, the best thing you can do is substitute stdint.h for inttypes.h in the protobuf-c.h header as well as the FileTranslator.h header. I could add in extra #define checks, but those are unnecessary.

2. The CSave GUI can be made to work with older X/Open Curses libraries, from about 1998 and up. Anything older will simply not work, or you will need to jump through a lot of hoops.

3. You will probably need to use GCC. While CSave can be made to compile with C89 compilers (for instance, the default ones on UNIX System V), you might run into strange issues.

4. CSave will not work on big-endian machines. It just won’t.

5. You may run into issues compiling protobuf-c. You might want to compile protoc-c with a C++11 compiler, compile the protobuf files, then send those over to your old machine. **If you edit protobuf-c.h and FileTranslator.h, replacing stdint.h with inttypes.h, it should† compile.**

2.5. Compiling - Windows (Native/MSVC)

First, you need to compile protobuf-c and add it to your install’s ”Lib and ”Include and make sure they are accessible by the compliler. From there, use a GNU make like this: "make -f Makefile.vc". This will compile CSave using the Visual C++ toolchain. You will need to edit Makefile.vc to tell the linker where the protobuf-c library is.

2.6. Compiling - OS/2 and other platforms (Watcom C/C++ 11)

First, compile protobuf-c and install the single header it comes with. Then, edit Makefile.watcom and make sure SYSTEM = os2v2. os2v2 is OS/2 version 2 and newer. From there, type "make -f Makefile.watcom" to cross-compile. You will need to edit Makefile.watcom to tell the linker where to find the protobuf-c library.

2.7. Compiling - OS/2 (EMX/ArcaOS Packages)

Follow the Unix installation instructions written above. The same process should apply. You might need to edit the Makefile to get it to properly compile. It should be noted that I have never actually tested this.

---

† I should probably clarify here, but it’s aside the point. Some machines that are 32-bit will completely lack a 64-bit integer type, which can cause serious issues. For instance, if you’re on a HP PA-RISC 1.1 system, you will probably need to use "unsigned long". If you are on a native 64-bit system like a DEC Alpha AXP running OSF/1, then 64-bit integer types will be available in your "inttypes.h" header. While this is incredibly kuldgey, it is enough to allow you to edit BL3 saves on the oldest systems possible.

2.8. Compiling - OpenVMS 8.4 (DEC C/C++)

The GUI will not compile. Everything else should compile with a new enough DEC C. VAX/VMS is not very well supported, and you might run into serious issues.

3. Operation

3.1. Quick start with CSaveMenu

For new users, it is highly recommended that you start with CSaveMenu. You can run it directly from the folder that you installed CSave to - on any platform. The precompiled builds of CSave should run on any OS.

CSaveMenu is a question-and-answer driven application, so be sure to read each prompt and check this manual carefully if it asks you to.

3.2. Quick start

Start by running "SaveToProto XXX.sav" and then "SaveUnpack XXX.sav.proto" to get information about a save file. You can then use "SaveGenerate XXX.sav.proto" and then "ProtoToSave XXX.proto" to get a usable file. If you have CSaveGUI installed, please just use that.

3.3. Converting saves

The included utilities "SaveToProto" and "ProtoToSave" to convert to and from "binary format" (i.e. a normal save file) and "protobuf format" (the data actually in the save file) with ease. These programs take one parameter each - the name of the save file.
To convert a save to protobuf format: "SaveToProto original.sav"
To edit the file: "SaveGenerate original.sav.proto edited.proto"
To convert from protobuf format to binary format: "ProtoToSave edited.proto original.sav"

Likewise, you must also convert your profile saves. To acomplish this, use these programs:
To convert a profile to protobuf format: "ProfileToProto profile.sav"
To edit the file: "ProfileGenereate profile.sav.proto profedited.sav.proto"
To convert from protobuf format to binary format: "ProtoToProfile profedited.sav.proto profile.sav"

If you want to convert to and from console saves, simply put the "profile code" number as the last command line argument. For SaveToProto, enter something like this: "SaveToProto 2.sav 3" (3 is the platform code for PS4 saves). For ProtoToSave, enter something like this: "ProtoToSave 2.sav.proto 2.sav 3"

3.4. Converting saves across different platforms

The "SaveConvert" program can convert, for instance, a PS4 save to a PC save. It manipulates the header and re-encodes the save data. Use this program with care, as it is basically a shortcut to running SaveToProto and ProtoToSave, except this program manipulates the headers to make the game not clobber over your save. Of course, you can also use a series of other CSave commands to manually convert saves without using SaveConvert. SaveConvert is useful for converting a large amount of saves in batch - just be sure to specify the correct platform numbers.

3.5. Running scripts

There are several example scripts in the "Scripts" folder included with the distribution. Please inspect these files and realize that they are actually just a series of commands you pipe into SaveGenerate or ProfileGenerate. To "run" a script file, use a command like this: "SaveGenerate input.sav.proto output.sav.proto

< scriptFile.txt". Please note that your command shell might have different command syntax. Make sure that "SaveGenerate" and/or "ProfileGenerate" are in your system path, or are in the same folder as your scripts are stored. Also note that the Windows command processor and PowerShell are not case-sensitive.

3.6. Editing saves

3.7. Comprehensive List of ProfileGenerate commands

3.7.1. set goldenkeys

This command sets how many Golden Keys you have. It will ask for how many you want.

3.7.2. set diamondkeys

This command works exactly like the command that sets golden keys, except this one will prompt you for how many Diamond Keys you want.

3.7.3. set vckeys

This command will allow you to get Vault Card reward keys. This does not affect the chests at all, you will need to use "set vaultcardchests" for that.

3.7.4. set vcid

This command will set which Vault Card is currently active. Usually, this is 1, but as the game continues to be updated, this value can be changed. Please note that setting it to some unknown high value will result in weirdness.

3.7.5. set vcdayseed

This command will set the seed for the current day. Since you will need to run this command often, it might be advisable to use a Windows scheduled task or a UNIX cron job to run a script over your profile save and force the seed to be something unique every day. The Vault Card index counter starts from zero, so card 1 is actually 0.

3.7.6. set vcweekseed

This command functions exactly like "set vcdayseed" but it sets the reward seed for the week. As before, you might want to use a cron job or a scheduled task to automate this. Like other commands, the index starts at zero.

3.7.7. set vcrewardcardid

This command is "dangerous" and should be used with care. This command will allow you to manipulate the card ID associated with a Vault Card rewards object. This value does not affect much, but you should not change it unless you really know what you’re doing, as setting it to something invalid might result in a corrupted game save or a crashing game.

3.7.8. set vcexp

This command will set how much experience is on a certain Vault Card. It will prompt for the card to modify (again, the first card is ID 0) and how much experience you want to put on it.

3.7.9. set vcchests

This command will set how many openable chests there are. Specify the card ID and the number of openable chests.

3.7.10. set vcchestsopened

There isn't much point to this command, but it allows you to set how many Vault Card reward chests you have opened.

3.7.11. set vcusedkeys

This command sets how many keys you have used. This is the opposite of "set vckeys". This has similar relevance to the "set vcchestsopened" command, as in, it's basically useless.

3.7.12. unlock all

Unlocks *everything.* Please note that this could cause some problems. Also, **ABSOLUTELY UNDER NO CIRCUMSTANCES USE THIS COMMAND UNLESS YOU HAVE MET THE LICENSE REQUIREMENTS. IT IS ILLEGAL TO USE THIS COMMAND TO OBTAIN CONTENT WHICH YOU DID NOT PAY FOR. I ASSUME NO LIABILITY IF YOU USE THIS.**

3.7.13. set gtokens

This command can be used to set how many unredeemed Guardian Rank tokens you have. Specify any number.

3.7.14. set grank

Use this command to set your Guardian Rank level (the one that appears at the bottom of the screen). Use this command with caution, as this is a deprecated field.

3.7.15. set gexp

Set the *old* Guardian Rank experience level. Please note that this value is no longer used.

3.7.16. set gnewexp

Set the Guardian Rank experience level. Use this instead of the "set gexp" command as that field is no longer used by the game.

3.7.17. set gseed

Set the Guardian Rank random rewards seed. Please enter it in integer form. ProfileUnpack will output that value in integer form. If you are given a hexadecimal-format seed, use a calculator to convert it to decimal and enter it.

3.7.18. set greward

Set the parameters of a Guardian Rank reward. This will prompt you for the class path to a reward, you will need to obtain a list of these from the game files. For info on how to get at this data, please look for an article named "Accessing Borderlands 3 Data" on the BLCmods or BLCM GitHub pages.

3.8. Comprehensive List of SaveGenerate commands

3.8.1. quit, exit

Exit from SaveGenerate. The output file will be generated and saved. Please now run ProtoToSave to pack the save file into a BL3 binary format save.

3.8.2. set name

Set the player's preferred name. It will prompt for a string.

3.8.3. set class

Sets the player's class. Takes an integer. 0 is Amara, 1 is FL4K, 2 is Moze, and 3 is Zane.

3.8.4. set sdu

Sets SDU values. Iterates through each SDU and prompts for a new level. Press Enter to use the previous value, or specify a blank line if using a script file. If you are using a script file and you have too many blank lines, the editor will ignore them. If you don't have enough blank lines, the editor will set that SDU to zero.

3.8.5. set mayhemlevel

Set the Mayhem Mode level for any playthrough you want. It will first prompt for the Mayhem level you want, then it will prompt for what playthrough you want to update. Playthrough 0 is NVMH, and 1 is TVHM.

3.8.6. set expoints

Set the total amount of experience points. This does not mean set the level - setting the EXP level affects the level because you need a certain amount of EXP to clear that level. Prompts for an integer value.

3.8.7. set level

Sets the player level by getting the level, and setting the EXP to the minimum required to clear that level. Prompts for an integer. The max accepted value is 80, although this will not be accepted by the game, and will instead drop you down to the current level cap.

3.8.8. set quest

This command will prompt the user for three things: first, the quest path. Use "SaveUnpack name.proto | grep CSAV001MSN" to find the quest. Copy and paste the class path for the mission into the editor. When prompted for the playthrough, enter 0 for NVHM or 1 for TVHM. Next, enter the quest state like this: 0 is Not Started, 1 is Active, 2 is Completed, 3 is Failed, and 4 is Unknown. Do not enter 4, your

game will most likely crash.

3.8.9. set guardianrank

This feature is currently not implemented. Check back later for an update.

3.8.10. set money

Set how much money you have. Takes an integer.

3.8.11. set eridium

Sets how much Eridium you have. Takes an integer.

3.8.12. unlock skilltree

Enable the selection of all skills on the tree. Does not coorespond to how many skill points you have.

3.8.13. set skillpoints

Sets how many skill points you have. Takes an integer - there does not appear to be a cap on this value.

3.8.14. set challenge

Modifies a challenge. "Challenges" includes crew challenges, and those challenges that pop up on the left side of your screen every now and then (especially during a new playthrough). Challenges are shared between playthroughs, so it will not prompt if you want to search NVHM or TVHM. It will first prompt you for what challenge you want. Enter the class path of the challenge (remember to use SaveUnpack to find them). Then, enter a completion state (1 for completed and 0 for uncompleted).

3.9. Comprehensive List of ItemLookupTest Commands

3.9.1. findtable

Allows you to look up an item table whose name you specify. For instance, BPInvPart_SM_TED_C (or any of the files included in the LookupTables directory). This is the command you can use to verify your CSave installation.

3.9.2. getindex

Extracts out an item class path. The format for the input is the "index" in a table of an item. This is the same number that is encoded in the BL3 serial format data. You can use this command to quickly look up what an item’s index is if you manually examine an inventory serial.

3.9.3. findindex

This command takes a class path to an object and returns its index. This command calls the same function that the item editing code uses to manipulate stored items. If this command does not properly work, you have encountered a serious issue and you won’t be able to edit items with CSave.

3.10. Comprehensive list of CSave message prefixes

CSAV001GEN - General information
CSAV001CLS - Player class information
CSAV001SKL - Skill points, XP, skills, and tree information
CSAV001SDU - SDU information
CSAV001VEH - Vehicle parts, loadouts, and configurations
CSAV001MSN - Missions/quests
CSAV001AMO - Ammo and grenades
CSAV001GRD - Guardian rank, level, perks, and rewards
CSAV001ROM - Crew quarters/bedroom information (including guns on the rack)
CSAV001ECH - ECHO logs
CSAV001FTM - Fast Travel machines: blacklisted, active, and reachable
CSAV001INV - Inventory: backpack and equipped
CSAV001CUS - Customizations: color, skin, emotes
CSAV001CHL - Challenge information
CSAV001ICL - Money (ICL means Inventory Category List)
CSAV001MHM - Mayhem Mode information
CSAVOO1ILT - Item Lookup Test program
CSAV001FIL - CSave file processing messages
CSAV001ABD - Abnormal End (crash)
CSAV001RWS - Read Write Save operations: loading and saving files
CSAV001CNV - Conversion functions
CSAV001RGN - Saved region data (possibly unused?)
CSAV001VCD - Vault Card Data

3.11. Platform Codes

These numbers are supposed to be entered as the last command line argument on SaveToProto, ProtoToSave, ProfileToProto, or ProtoToProfile.

1 PC save file - Steam
2 PC profile file - Steam
3 Ps4 save file
4 PS4 profile file
5 XB1 save file (NOT IMPLEMENTED)
6 XB1 profile file (NOT IMPLEMENTED)
7 PC save file - Epic Games
8 PC profile file - Epic Games
9 XBSX save file (NOT IMPLEMENTED)
10 XBSX profile file (NOT IMPLEMENTED)
11 PS5 save file (NOT IMPLEMENTED)
12 PS5 profile file (NOT IMPLEMENTED)

4. Other Programs

The CSave source code contains other programs, including:

1. ReadSaveHeader
2. ItemLookupTest

3. GenSaveHeader

These programs can be used to test CSave. ReadSaveHeader is especially useful for getting information about what platform a save originated on, while GenSaveHeader can make a save header to make completely new save files. Please note that using GenSaveHeader with SaveGenerate and then ProtoToSave is rather risky, as the game will fill in a lot of missing information. ItemLookupTest is used to get information about items, and is mainly present to test your installation's paths.

5. FAQ

What does CSAV001 mean? Well, it's the message prefix. CSave was originally intended to be part of one of my MVS (IBM mainframe) systems, and on that operating system, every message is prefixed by a message code. Ideally, if you came across an error message, you could go to the "Messages and Codes" manual for that program product, and you would flip through it looking for that message number, and it would tell you how to fix it. Since CSave would like to honor its MVS heritage, the message prefixes still persist in the latest CSave versions.

Can CSave export a PS4 save that I can put on a flash drive? No, and it never will. In order to produce an importable PS4 save, you will need to cryptographically sign it. Since I don't have those certificates required to sign a save and never will, you're on your own here.

Will CSave ever support XB1 saves? Probably not. I have no knowledge of how the XB1 save format works, and they're probably encrypted like they are on the PS4. Since I also am not going to have those certificates, you're on your own here too. Technically, CSave also does not support producing even the XB1 header, as nobody has been able to examine XB1 save files to investigate the header and bitmask.

Why does CSave support all these weird OSes? Why not? I figured that if I can get CSave to run on computers from the Bad Old Days, like VAX/VMS, Windows NT, OS/2, and VM/CMS, it should literally run on anything. This was mainly intended to improve portability and weed out bugs, as I have encountered several portability bugs in libprotobuf-c that have resulted in CSave only working on little-endian systems.

Why does CSave crash so much? Mainly because I simply don't have the time to throughly test CSave and weed out every possible bug. However, if you do encounter lots of crashes, please file a bug report on GitHub and I will get to it to the best of my ability.

6. Additional Help

If you need more help, please check out the UNIX manpages that have been added to the source repository. These are intended as a supplement to this manual, and to help you remember the commands' arguments should you forget. If you are encountering actual bugs, please file a bug report. If you have any further questions, you're on your own.