

---

## NSZ

A compression/decompression script (with optional GUI) that allows user to compress/decompress Nintendo Switch dumps loselessly, thanks to zstd compression algorithm. The compressed file can be installed directly with supported NSW Homebrew Title Installers.

## Mirror

A Swiss mirror of this repository is maintained under <https://gitlab.nicobosshard.ch/nicoboss/nsz>  
This will be the new home in case GitHub ever takes down nsz. Please bookmark it.

## Legal

- This project does NOT incorporate any copyrighted material such as cryptographic keys. All keys must be provided by the user.
- This project does NOT circumvent any technological protection measures. The NSZ file format purposely keeps all technological protection measures in place.
- This project shall only be used for legally purchased games.
- This project is MIT licensed. Check LICENSE for more information.

## Installation:

There are several ways the install this tool. You can find details on installation for all of them below.

**You need to have a hactool compatible keys file in a suitable directory to use this tool.**

The keys file must be located as `prod.keys` file in `%USERPROFILE%/.switch/` (Windows)/`$HOME/.switch/` (UNIX) or `keys.txt` in the working directory. You must legally obtain your keys!

## Windows Builds

You can also use the Windows binaries. They do not require any external libraries to be installed and can be run without installing anything. You can find the binaries in the release page.

**Methods listed below requires you to have Python 3.6+ and pip3 installed.**

---

## PIP Package

Use the following command to install the console-only version:

```
pip3 install --upgrade nsz
```

Use the following command to install the GUI version:

```
pip3 install --upgrade nsz[gui]
```

## Android

1. Install “Pydroid 3” and the “Pydroid repository plugin” from the Play Store
2. Open “Pydroid 3” and navigate to “Pip”
3. Enter “nsz” and unselect “use prebuild” then press install
4. Navigate to “Terminal” to use the “nsz” command
5. The first time it will tell you where to copy your prod.keys which you should do using the “cp” command
6. Use any command line arguments you want like “nsz -D file.nsz” to decompress your game

## Running from source

The tool can also be run by cloning the repository, installing the requirements and then executing nsz using `python3 nsz.py`

Use the following command to install the console-only versions requirements:

```
pip3 install -r requirements.txt
```

Use the following command to install the GUI versions requirements:

```
pip3 install -r requirements-gui.txt
```

## Usage

```
1 nsz.py --help
2 usage: nsz.py [-h] [-C] [-D] [-l LEVEL] [-L] [-B] [-S] [-s BS] [-V] [-Q]
3               [-F] [-p] [-P] [-t THREADS] [-m MULTI] [-o [OUTPUT]] [-w]
4               [-r]
5               [--rm-source] [-i] [--depth DEPTH] [-x]
6               [--extractregex EXTRACTREGEX] [--titlekeys] [--undupe]
7               [--undupe-dryrun] [--undupe-rename] [--undupe-hardlink]
8               [--undupe-prioritylist UNDUPE_PRIORITYLIST]
               [--undupe-whitelist UNDUPE_WHITELIST]
```

---

```

 9          [--undupe-blacklist UNDUPE_BLACKLIST] [--undupe-old-
10              versions]
11          [-c CREATE]
12          [file ...]
13 positional arguments:
14     file
15
16 options:
17     -h, --help            show this help message and exit
18     -C                    Compress NSP/XCI
19     -D                    Decompress NSZ/XCZ/NCZ
20     -l LEVEL, --level LEVEL
21                          Compression Level: Trade-off between
22                          compression speed
23                          and compression ratio. Default: 18, Max: 22
24     -L, --long            Enables zStandard long distance mode for even
25                          better
26                          compression
27     -B, --block           Use block compression option. This mode allows
28                          multi-threaded compression/decompression with
29                          random
30                          read access allowing compressed games to be
31                          played
32                          without decompression in the future however
33                          this comes
34                          with a slightly lower compression ratio cost.
35                          This is
36                          the default option for XCZ.
37     -S, --solid           Use solid compression option. Slightly higher
38                          compression ratio but won't allow for random
39                          read
40                          access. File compressed this way will never be
41                          mountable (have to be installed or decompressed
42                          first
43                          to run). This is the default option for NSZ.
44     -s BS, --bs BS       Block Size for random read access 2^x while x
45                          between
46                          14 and 32. Default: 20 => 1 MB
47     -V, --verify         Verifies files after compression raising an
48                          exception on hash mismatch and verify existing
49                          NSP and
50                          NSZ files when given as parameter. Requires --
51                          keep
52                          when used during compression.
53     -Q, --quick-verify   Same as --verify but skips the NSP SHA256 hash
54                          verification and only verifies NCA hashes. Does
55                          not
56                          require --keep when used during compression.

```

---

---

45	-K, --keep	Keep all useless files and partitions during
46		compression to allow bit-identical recreation
47	-F, --fix-padding intro	Fixes PFS0 padding to match the nxdumptool/no-
48		standard. Incompatible with --verify so --quick
49		-verify
49		will be used instead.
50	-p, --parseCnmt information	Extract TitleId/Version from Cnmt if this
51		cannot be obtained from the filename. Required
52		for
52		skipping/overwriting existing files and --rm-
53		old-
53		version to work properly if some not every file
54		is
54		named properly. Supported filenames:
55		*TitleID*[vVersion]*
56	-P, --alwaysParseCnmt	Always extract TitleId/Version from Cnmt and
57		never
58		trust filenames
59	-t THREADS, --threads	THREADS
60		Number of threads to compress with. Numbers < 1
61		corresponds to the number of logical CPU cores
62		for
62		block compression and 3 for solid compression
63	-m MULTI, --multi	MULTI
64		Executes multiple compression tasks in parallel
65		. Take
65		a look at available RAM especially if
66		compression
66		level is over 18.
67	-o [OUTPUT], --output	[OUTPUT]
68		Directory to save the output NSZ files
69	-w, --overwrite the	Continues even if there already is a file with
70		same name or title id inside the output
71		directory
71	-r, --rm-old-version	Removes older versions if found
72	--rm-source	Deletes source file/s after compressing/
73	decompressing.	
73		It's recommended to only use <b>this</b> in
74		combination with
74		--verify
75	-i, --info	Show info about title or file
76	--depth DEPTH	Max depth <b>for</b> file info and extraction
77	-x, --extract	Extract a NSP/XCI/NSZ/XCZ/NSPZ
78	--extractregex	EXTRACTREGEX
79		Regex specifying which files inside the
		container

---

---

```

80      should be extracted. Example: "^.*\.(cert|tik)$"
81      --titlekeys      Extracts titlekeys from your NSP/NSZ files and
      adds
82      missing keys to ./titlekeys.txt and JSON files
      inside
83      ./titledb/ (obtainable from
84      https://github.com/blawar/titledb).
85      --undupe          Deleted all duplicates (games with same ID and
86      Version). The Files folder will get parsed in
      order so
87      the later in the argument list the more likely
      the
88      file is to be deleted
89      --undupe-dryrun   Shows what files would get deleted using --
      undupe
90      --undupe-rename   Renames files to minimal standard:
91      [TitleId][vVersion].nsz
92      --undupe-hardlink Hardlinks files to minimal standard:
93      [TitleId][vVersion].nsz
94      --undupe-prioritylist UNDUPED_PRIORITYLIST
95      Regex specifying which duplicate deletion
      should be
96      prioritized before following the normal
      deletion
97      order. Example: "^.*\.(nsp|xci)$"
98      --undupe-whitelist UNDUPED_WHITELIST
99      Regex specifying which duplicates should under
      no
100     circumstances be deleted. Example: "^.*\.(nsz|
      xcz)$"
101     --undupe-blacklist UNDUPED_BLACKLIST
102     Regex specifying which files should always be
      deleted
103     - even if they are not even a duplicate! Be
      careful!
104     Example: "^.*\.(nsp|xci)$"
105     --undupe-old-versions
106     Removes every old version as long there is a
      newer one
107     of the same titleID.
108     -c CREATE, --create CREATE
109     Inverse of --extract. Repacks files/folders to
      an NSP.
110     Example: --create out.nsp .\in

```

## Few Usage Examples

- To compress all files in a folder: `nsz -C /path/to/folder/with/dumps/`

- 
- To compress all files in a folder and verifying integrity of compressed files: `nsz --verify -C /path/to/folder/with/dumps/`
  - To compress all files in a folder with 8 threads and outputting resulting files to a new directory: `nsz --threads 8 --output /path/to/out/dir/ -C /path/to/folder/with/dumps/`
  - To compress all files in a folder with level 22 compression level: `nsz --level 22 -C /path/to/folder/with/dumps/`
  - To decompress all files in a folder: `nsz -D /path/to/folder/with/dumps/`

To view all the possible flags and a description on what each flag, check the Usage section.

## File Format Details

### NSZ

NSZ files are functionally identical to NSP files. Their sole purpose to alert the user that it contains compressed NCZ files. NCZ files can be mixed with NCA files in the same container.

As an alternative to this tool NSC\_Builder also supports compressing NSP to NSZ, and decompressing NSZ to NSP. NSC\_Builder can be downloaded at [https://github.com/julesontheroad/NSC\\_BUILDER](https://github.com/julesontheroad/NSC_BUILDER)

### XCZ

XCZ files are functionally identical to XCI files. Their sole purpose to alert the user that it contains compressed NCZ files. NCZ files can be mixed with NCA files in the same container.

### NCZ

These are compressed NCA files. The NCA's are decrypted, and then compressed using zStandard.

The first 0x4000 bytes of an NCZ file is exactly the same as the original NCA (and still encrypted). This applies even if the first section doesn't start at 0x4000.

At 0x4000, there is the variable sized NCZ Header. It contains a list of sections which tell the decompressor how to re-encrypt the NCA data after decompression. It can also contain an optional block compression header allowing random read access.

All of the information in the header can be derived from the original NCA + Ticket, however it is provided pre-parsed to make decompression as easy as possible for third parties.

---

Directly after the NCZ header, the zStandard stream begins and ends at EOF. The stream is decompressed to offset 0x4000. If block compression is used the stream is splitted into independent blocks and can be decompressed as shown in <https://github.com/nicoboss/nsz/blob/master/nsz/BlockDecompressorReader.py>. `CompressedBlockSizeList[blockID]` must not exceed `decompressedBlockSize`. If smaller the block must be decompressed. If equal the block is stored in plain text.

```
1 class Section:
2     def __init__(self, f):
3         self.magic = f.read(8) # b'NCZSECTN'
4         self.offset = f.readInt64()
5         self.size = f.readInt64()
6         self.cryptoType = f.readInt64()
7         f.readInt64() # padding
8         self.cryptoKey = f.read(16)
9         self.cryptoCounter = f.read(16)
10
11 class Block:
12     def __init__(self, f):
13         self.magic = f.read(8) # b'NCZBLOCK'
14         self.version = f.readInt8()
15         self.type = f.readInt8()
16         self.unused = f.readInt8()
17         self.blockSizeExponent = f.readInt8()
18         self.numberOfBlocks = f.readInt32()
19         self.decompressedSize = f.readInt64()
20         self.compressedBlockSizeList = []
21         for i in range(self.numberOfBlocks):
22             self.compressedBlockSizeList.append(f.readInt32())
23
24 nspf.seek(0x4000)
25 sectionCount = nspf.readInt64()
26 for i in range(sectionCount):
27     sections.append(Section(nspf))
28
29 if blockCompression:
30     BlockHeader = Block(nspf)
```

## References

NSZ pip package: <https://pypi.org/project/nsz/>

Forum thread: <https://gbatemp.net/threads/nsz-homebrew-compatible-nsp-xci-compressor-decompressor.550556/>

---

## Credits

SciresM for his hardware crypto functions; the blazing install speeds (50 MB/sec +) achieved here would not be possible without this.

Thanks to our contributors: nicoboss, blawar, plato79, eXhumer, Taorni, anthonyu, teknoraver, KWottrich, gabest11, siddhartha77, alucryd, seiya-git, drizzt, 16BitWonder, 2weak2live, thatch, maki-chan, pR0Ps