Ceedoku Save Format (CSF) Specification 1.0

1. Overview

The **Ceedoku Save Format (CSF)** is a binary file format used to store Ceedoku save data.

A CSF file consists of:

1. An optional file header
2. A mandatory NUL byte
3. A mandatory CRC-32C checksum
4. Save key/value data
5. End of file (EOF)

General structure:

[OPTIONAL HEADER]
[0x00]
[CRC-32C CHECKSUM]
[SAVE DATA]
[EOF]

---

2. File Header

The standard CSF 1.0 header is:

CEEDOKU-CSF/1

The header is optional.

A CSF parser must accept a valid CSF file with or without the header.

The header is **not included in the checksum**.

---

3. Mandatory NUL Separator

A single NUL byte (`0x00`) is mandatory after the optional header.

The NUL byte marks the beginning of the CSF binary data.

The separator NUL is **not included in the checksum**.

If the optional header is omitted, the file begins with the mandatory NUL byte.

4. Checksum

A checksum is mandatory.

CSF 1.0 uses **CRC-32C**.

The checksum is exactly **4 bytes (32 bits)**.

It is stored immediately after the mandatory NUL separator and before the save data.

Checksum coverage

The checksum covers **all bytes after the checksum**, including:

- All normal key/value data
- The `undoRedoStack` shortcode
- The undo stack JSON
- The NUL separator between the undo and redo stacks
- The redo stack JSON

The checksum does **not** cover:

- The optional header
- The mandatory NUL separator before the checksum
- The checksum itself

Structure:

[OPTIONAL HEADER]
[0x00]
[CRC-32C]
[SAVE DATA]

A CRC mismatch indicates that the save data has been **corrupted or modified**.

CRC-32C is intended for corruption detection and is not a cryptographic security mechanism.

---

5. Key Shortcodes

Every normal CSF key begins with exactly **one byte**.

The byte is the key's shortcode.

The shortcode determines:

- Which save property is being stored
- The property's type
- The property's width

The parser uses the CSF key table to interpret the value.

6. CSF 1.0 Key Table

Byte	JSON Key	Type	Width
`0x01`	`solution`	Sudoku board	324 bits
`0x02`	`values`	Sudoku board	324 bits
`0x03`	`givens`	Given-state board	81 bits
`0x04`	`notes`	Notes	729 bits
`0x05`	`selected`	Cell index	7 bits
`0x06`	`difficulty`	Enum	3 bits
`0x07`	`pencilMode`	Boolean	1 bit
`0x08`	`eraseMode`	Boolean	1 bit
`0x09`	`mistakes`	Unsigned integer	64 bits
`0x0A`	`elapsedMs`	Time	128 bits
`0x0B`	`timerPaused`	Boolean	1 bit
`0x0C`	`undoRedoStack`	Special JSON	Variable
`0x0D`	`finished`	Boolean	1 bit
`0x0E`	`hintcount`	Unsigned integer	64 bits
`0x0F`	`hintcounter`	Unsigned integer	64 bits
`0x10`	`cooldownmoves`	Unsigned integer	64 bits
`0x11`	`cooldowntime`	Unsigned integer	64 bits
`0x12`	`cooldowntypetouse`	Enum	1 bit
`0x13`	`canusehelp`	Boolean	1 bit

7. Shortcode Assignment

Shortcodes are initially assigned according to the order of the keys in the Ceedoku JSON save structure.

Once a shortcode has been assigned, its meaning must never change.

Existing shortcodes must never be shifted.

When a new save key is added:

1. It is added to the bottom of the JSON key order.
2. It receives the next available shortcode.
3. Existing shortcode assignments remain unchanged.

For example, if `0x12` is currently the last assigned shortcode, the next new key receives `0x13`.

---

8. Key Ordering

Normal keys may appear in **any order** in a CSF file.

For example, this is valid:

[difficulty]
[values]
[solution]
[finished]

The parser identifies each key by its shortcode.
The physical order of normal keys does not determine their meaning.

Exception

`undoRedoStack` is special.

`undoRedoStack`:

- Must be present at the exact end if stored
- Must be the final key
- Must have no key or save data after it
- Is terminated only by EOF

---

9. Fixed-Width Values

Normal CSF values use the exact width specified by the key table.

The general representation is:

[1-byte SHORTCODE]
[VALUE]

No separator is required between normal key/value pairs.

10. Boolean Values

Boolean values occupy exactly **1 bit**.

0 = false
1 = true

No other boolean value is valid.

---

11. Difficulty

`difficulty` occupies exactly **3 bits**.

Binary	Decimal	Difficulty
`000`	0	Easy
`001`	1	Medium
`010`	2	Hard
`011`	3	Expert
`100`	4	Master
`101`	5	Extreme
`110`	6	Impossible
`111`	7	Godlike

---

12. Sudoku Board Encoding

The `solution` and `values` fields each contain exactly 81 Sudoku cells.

Each cell occupies **4 bits**.

Binary	Meaning
`0000`	Empty
`0001`	1
`0010`	2
`0011`	3
`0100`	4

`0101`	5
`0110`	6
`0111`	7
`1000`	8
`1001`	9

The values `1010` through `1111` are invalid.

Each board occupies:

81 × 4 = 324 bits

---

13. Givens Encoding

The `givens` field contains one bit for every Sudoku cell.

Total size:

81 bits

Encoding:

0 = not given
1 = given

The bits use the same cell ordering as the `solution` and `values` fields.

---

14. Notes Encoding

The `notes` field contains nine bits for every Sudoku cell.

Total size:

81 × 9 = 729 bits

For each cell:

Bit	Note
0	1

1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9

For every note bit:

0 = note absent
1 = note present

---

15. Selected Cell

The `selected` field occupies exactly **7 bits**.

The value represents the currently selected cell.

0 = no cell selected
1 = first cell
2 = second cell
...
81 = last cell

Values `82` through `127` are invalid.

---

16. Unsigned Integer Values

All normal integer fields use an **unsigned 64-bit integer**.

The valid range is:

0 to 18,446,744,073,709,551,615

The following fields use this representation:

- `mistakes`
- `hintcount`
- `hintcounter`

- `cooldownmoves`
- `cooldowntime`

---

17. Time Values

Time values use exactly **128 bits (16 bytes)**.

The following field uses this representation:

- `elapsedMs`

`startTime` is **not stored in CSF 1.0**.

---

18. Special Key: undoRedoStack

`undoRedoStack` uses shortcode:

0x0B

It is a special variable-length field.

It has no fixed maximum length.

The structure is:

0x0B
[UNDO STACK JSON]
0x00
[REDO STACK JSON]
EOF

The undo stack and redo stack are stored as their corresponding JSON values.

The `undoRedoStack` key must be the **final key in the entire file**.

Nothing may follow the redo stack JSON.

---

19. Undo/Redo JSON Encoding

The undo and redo stack data is encoded as **UTF-8 JSON**.

The JSON may contain arbitrary nested objects and arrays required by Ceedoku's save system.

Example:

```
[
  {
    "selected": 60,
    "changes": [
      {
        "index": 60,
        "before": {
          "value": 0,
          "notes": []
        },
        "after": {
          "value": 1,
          "notes": []
        }
      }
    ]
  }
]
```

Raw NUL bytes (`0x00`) must not occur inside either JSON value.

If a JSON value needs to represent a NUL character, JSON escaping must be used:

\u0000

The escaped sequence does not contain an actual `0x00` byte.

Therefore the first raw `0x00` after the undo JSON is always the separator between the undo and redo stacks.

---

20. cooldowntypetouse

`cooldowntypetouse` is a **1-bit enum**.

0 = time
1 = moves

No other value is valid.

---

21. Missing Keys

A key does not have to be present in every CSF file.

If a key is absent, no placeholder value is required.

An absent key represents an omitted save property.

---

22. Undefined Shortcodes

A shortcode that is not defined by the applicable CSF specification is invalid.

Future CSF versions may assign meanings to previously unused shortcode values.

Existing shortcode meanings must never change.

---

23. Invalid Values

A save is invalid if any value violates its field definition.

Examples include:

- Invalid boolean value
- Invalid difficulty value
- Invalid Sudoku cell value
- Invalid selected value
- Invalid `cooldowntypetouse` value
- Invalid JSON
- Missing undo/redo separator
- Data after the redo JSON
- Undefined shortcode
- Incorrect field width
- Incorrect key/value structure

---

24. Corruption vs Invalid Save

CSF distinguishes between corruption and an invalid save.

CRC mismatch

If the calculated CRC-32C does not match the stored checksum:

Save corrupted or modified

CRC valid but specification violation

If the checksum matches but the file violates the CSF specification:

Invalid save

CRC valid and specification compliant

If the checksum matches and the complete file follows the CSF specification:

Valid save

---

25. EOF

CSF does not use a special EOF byte.

EOF is the actual end of the file.

For a file containing `undoRedoStack`, the file must end immediately after the redo stack JSON.

Example:

0B
[undo JSON]
00
[redo JSON]
EOF

---

26. Complete File Layout

Complete layout of a .CSF file:

Optional CEEDOKU-CSF/1 header

00 (Mandatory NUL separator)

CRC-32C (4 bytes)

Normal key/value data

0B (undoRedoStack shortcode)

Undo stack JSON

00 (Undo/redo separator)

Redo stack JSON

EOF (End of File character)

---

27. CSF 1.0 Key Summary

Byte	JSON Key	Width
`00`	`solution`	324 bits
`01`	`values`	324 bits

02	givens	81 bits
03	notes	729 bits
04	selected	7 bits
05	difficulty	3 bits
06	pencilMode	1 bit
07	eraseMode	1 bit
08	mistakes	64 bits
09	elapsedMs	128 bits
0A	timerPaused	1 bit
0B	undoRedoStack	Variable
0C	finished	1 bit
0D	hintcount	64 bits
0E	hintcounter	64 bits
0F	cooldownmoves	64 bits
10	cooldowntime	64 bits
11	cooldowntypetouse	1 bit
12	canusehelp	1 bit

---

28. Design Principles

1. The header is optional.
2. The NUL separator after the header is mandatory.
3. The CRC-32C checksum is mandatory.
4. The checksum is stored before all save keys.
5. The checksum covers all save data after the checksum.
6. Normal keys use one-byte shortcodes.
7. Normal values have fixed widths.
8. Keys may appear in any order.
9. Existing shortcode assignments never change.
10. New keys are appended.
11. Boolean values use one bit.
12. Difficulty uses three bits.
13. Sudoku cells use four bits.
14. Given states use one bit per cell.
15. Notes use nine bits per cell.

16. `selected` uses seven bits.
17. Ordinary integers use unsigned 64-bit values.
18. Time values use 128 bits.
19. `undoRedoStack` is variable-length UTF-8 JSON.
20. `undoRedoStack` must always be the final key.
21. A NUL separates the undo and redo JSON.
22. EOF immediately follows the redo JSON.
23. `pageMode` is not stored.
24. `startTime` is not stored.
25. CRC mismatch indicates corruption or modification.
26. A specification violation produces an Invalid Save result.