Oboron CLI Specification

oboron.org

version 1.0

1 Overview

This document specifies the command-line interface for conforming Oboron implementations. Its sole purpose is cross-implementation conformance testing through a shared command contract: a test harness passes a key and a format and checks the resulting obtext, so the CLI needs only to expose key input, format selection, and the `enc` / `dec` operations.

Conforming implementations provide the `ob` binary — the CLI for the authenticated Oboron core schemes (`dsiv`, `dgcmsiv`, `psiv`, `pgcmsiv`). A conforming core CLI MUST provide the `enc`, `dec`, and `keygen` commands and the global `--help` and `--version` options.

Implementations MAY offer additional convenience features — key profiles, persistent configuration files, and the like. Such features are outside this specification and MUST NOT be required for conformance; this document specifies only a key and a format passed as parameters.

The unauthenticated schemes have their own optional CLI, specified in the Oboron Unauthenticated Layer (obu) Specification. For cryptographic details, format definitions, key handling, and UTF-8 behavior, see the Oboron Protocol Specification.
The reference CLI implementation (the `ob` binary) is available at `https://gitlab.com/oboron/oboron-tools-rs`; the reference protocol library is at `https://gitlab.com/oboron/oboron-rs`.

2 Conventions

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in BCP 14 (RFC 2119, RFC 8174) when, and only when, they appear in all capitals, as shown here.

Command Syntax Notation

- `<REQUIRED>` — a required argument or value.
- `[OPTIONAL]` — an optional argument or value.
- `--flag` — a long-form flag.
- `-f` — a short-form flag.
- `--option <VALUE>` — an option that takes a value.

3 The ob Binary

The `ob` binary is REQUIRED for all conforming implementations. It handles the authenticated Oboron core schemes. Its built-in default format is `dsiv.c32` (§5).

`--help` and `--version` are global options: an implementation MUST accept them before or after any command name, and MUST NOT require a key, configuration, profile, or readable stdin to service them. Both exit with status `0`.

Version Output

`ob --version` MUST write a single UTF-8 line to stdout and exit `0`. The line MUST have the form:

```
ob <implementation> <version> protocol=<v> cli=<v>
```

for example `ob oboron-tools-rs 1.0.0 protocol=1.0 cli=1.0`, so that conformance tooling can parse the implementation name, the implementation version, the supported protocol version, and the CLI specification version. Each field is a single whitespace-free token, so the line can be split on spaces.

4 Core Commands (REQUIRED)

The `enc`, `dec`, and `keygen` commands MUST be implemented by a conforming core CLI. The `enc` (`e`) and `dec` (`d`) aliases are OPTIONAL.

4.1 enc

Encrypt and encode a plaintext string.

```
ob enc [OPTIONS] [TEXT]
```

Flags

Flag / Option	Short	Description
`--key <KEY>`	`-k`	Encryption key (128 hex chars); conflicts with `--keyless`.
`--keyless`	`-K`	Use the fixed public test key (INSECURE — testing only); conflicts with `--key`.
`--format <FORMAT>`	`-f`	Format string (e.g. `dsiv.b64`); MUST NOT be combined with scheme/encoding flags.
`--dsiv`	`-s`	Deterministic AES-SIV.
`--psiv`	`-S`	Probabilistic AES-SIV.
`--dgcmsiv`	`-g`	Deterministic AES-GCM-SIV.
`--pgcmsiv`	`-G`	Probabilistic AES-GCM-SIV.
`--c32`	`-c`	Crockford base32 encoding.

Flag / Option	Short	Description
`--b32`	`-b`	RFC 4648 base32 encoding.
`--b64`	`-B`	URL-safe base64 encoding.
`--hex`	`-x`	Hex encoding.
`--raw`	`-0`	Disable CLI line framing (§7).
`--help`	`-h`	Print help.
`--version`	`-V`	Print version.

The scheme short flags are mnemonic: the letter selects the algorithm (`-s` = SIV, `-g` = GCM-SIV) and the case selects the property — lowercase is deterministic, uppercase is probabilistic. Thus `-s`/`-S` are `dsiv`/`psiv` and `-g`/`-G` are `dgcmsiv`/`pgcmsiv`.

Behavioral Rules

- `[TEXT]` is an optional single positional argument; more than one positional argument is a usage error. If `[TEXT]` is supplied, it is the input string exactly as given — the implementation MUST NOT read stdin and MUST NOT strip a trailing newline. If `[TEXT]` is omitted, the command reads stdin per §7.
- `--key` and `--keyless` are mutually exclusive (§6).
- `--format` MUST NOT be combined with individual scheme or encoding flags. More than one scheme flag, or more than one encoding flag, MUST be rejected.
- `enc` MUST fail if the key is missing or invalid, the plaintext is empty after any newline processing (§7), the plaintext is not valid UTF-8, or a probabilistic scheme requires randomness that is unavailable. (A malformed or unknown effective format is a usage error, resolved before encryption; §8.)
- On success: write the obtext to stdout (newline per §7) and exit `0`.

4.2 dec

Decode and decrypt an obtext string.

```
ob dec [OPTIONS] [TEXT]
```

Flags

The same flag set as `enc` (`--key`/`--keyless`, `--format`, the scheme flags, the encoding flags, `--raw`, `--help`, `--version`).

Behavioral Rules

- `[TEXT]`, stdin, mutual-exclusion, and format-flag handling are as for `enc`.
- The scheme is supplied by the caller; when no scheme or `--format` flag is given, the built-in default scheme (§5) is used. `dec` MUST NOT auto-detect the scheme from the obtext and MUST NOT trial-decrypt across schemes or encodings.
- Invalid obtext MUST be rejected; §8 defines what counts as invalid.
- On success: write the plaintext to stdout (newline per §7) and exit `0`.

4.3 keygen

Print a freshly generated random key to stdout and exit. `keygen` needs no key, configuration, or stdin.

```
ob keygen
```

- Output is the canonical hex form of a fresh 512-bit key — exactly 128 lowercase hexadecimal characters (`0`–`9`, `a`–`f`) — followed by a single newline; exit `0`.
- If a cryptographically secure random number generator is unavailable, `keygen` MUST fail with status `1` (§8) and MUST NOT write a key.

5 Format Resolution

The CLI format identifiers are exactly those defined by the Oboron Protocol Specification; they are restated here for reference:

- Schemes: `dsiv`, `dgcmsiv`, `psiv`, `pgcmsiv`.
- Encodings: `c32` (Crockford base32), `b32` (RFC 4648 base32), `b64` (URL-safe base64), `hex` (hexadecimal).

Format identifiers are lowercase ASCII and case-sensitive; implementations MUST reject a malformed format identifier — including one whose scheme or encoding is not in the sets above — as a usage error (§8). The `ob:` prefix is *not* accepted by `--format`.

For `enc` and `dec`, the effective format MUST be resolved as follows:

1. If `--format <FORMAT>` is supplied, it specifies both the scheme and the encoding, and no scheme or encoding flag may also be supplied.
2. Otherwise the scheme and encoding resolve independently: the scheme is the single scheme flag supplied, or the built-in default `dsiv`; the encoding is the single encoding flag supplied, or the built-in default `c32`.
3. Supplying more than one scheme flag, or more than one encoding flag, is a usage error.

Scheme and encoding resolve independently: the built-in default format is `dsiv.c32`, but, for example, `ob enc --b64` uses scheme `dsiv` with encoding `b64`.

6 Key Resolution

A key is supplied to `enc` and `dec` by `--keyless`, `--key <KEY>`, or the `$OBORON_KEY` environment variable. `--key` and `--keyless` are mutually exclusive; their co-presence is a usage error, rejected before any key lookup, and no precedence rule applies between mutually exclusive options. After that check, the key source is resolved in the following order (highest to lowest):

Priority	Source
1	`--keyless` — the fixed public test key (§9; INSECURE, testing only).
2	`--key` CLI flag.
3	`$OBORON_KEY` environment variable.
4	Error (no key source).

A key supplied via `--key` or `$OBORON_KEY` MUST be a canonical key string: exactly 128 lowercase ASCII hexadecimal characters, as defined by the protocol specification. Implementations MUST reject any other length or character set. An empty `$OBORON_KEY` is an invalid key, not an absent one. If no key source is available, the implementation MUST exit with status `1` (§8).

7 Input and Output

All CLI input and output defined by this specification is UTF-8. Implementations MUST NOT interpret stdin, stdout, stderr, or test-vector files through a locale-dependent character encoding. Where a platform supplies command-line arguments as Unicode strings, implementations MUST encode them as UTF-8 before use; where it supplies them as bytes, implementations MUST validate them as UTF-8. `enc` MUST reject input that is not valid UTF-8. Implementations MUST treat stdin, stdout, and stderr as binary byte streams with no platform newline translation: the only line endings written or stripped are those specified below, so output is byte-identical across platforms.

Input

- If `[TEXT]` is supplied as a positional argument, it is used exactly; stdin is not read and no trailing newline is stripped.
- Implementations MUST support the `--` end-of-options separator: every argument after `--` is data, so a `[TEXT]` that begins with `-` can be passed as `ob enc [OPTS] -- <TEXT>` (or delivered on stdin).
- If `[TEXT]` is omitted, the command reads all bytes from stdin until EOF. In default mode it then removes exactly one trailing line ending: if the input ends with `\r\n` that two-byte sequence is removed, otherwise if it ends with `\n` that one byte is removed; no other bytes are removed. (Thus `printf '\n' | ob enc` yields an empty plaintext and is rejected, whereas `printf '\n' | ob enc --raw` encrypts a one-byte `U+000A` plaintext.)

Output

On success, `enc` writes the obtext and `dec` writes the plaintext to stdout, followed by a single `\n` in default mode.

Raw framing

The `--raw` flag (alias `-0`) disables CLI line framing. In `--raw` mode, for both `enc` and `dec`, the implementation MUST NOT strip a trailing line ending from stdin and MUST NOT append a line ending to stdout. Consequently this pipeline MUST round-trip exactly for any valid non-empty UTF-8 plaintext:

```
printf '%s' "$plaintext" | ob enc --raw [OPTS] | ob dec --raw [OPTS]
```

`--raw` is a framing option only: it does not change the Oboron plaintext domain and does not permit invalid UTF-8 plaintext.

8 Error Handling and Exit Status

Status	Meaning
`0`	Success.
`1`	Operation failed: invalid input, invalid obtext, authentication failure, invalid UTF-8 after decryption, missing or invalid key, or unavailable randomness.
`2`	Usage error: invalid or conflicting flags, unknown command, malformed format identifier, or wrong argument count.

On any failure, implementations MUST NOT write obtext or plaintext to stdout; diagnostic output MUST go to stderr only. Except for explicitly specified warnings, implementations MUST NOT write to stderr on success.

For `dec`, *invalid obtext* is any input that is not a canonical encoding for the selected encoding, has an invalid scheme-output length for the selected scheme, fails authentication, decrypts to invalid UTF-8, or decrypts to the empty plaintext. All such `dec` failures MUST exit with status `1` and MUST share one uniform stderr message, so that `dec` does not become a distinguishing oracle: separating the causes (non-canonical encoding, invalid scheme-output length, authentication failure, invalid UTF-8, or the empty plaintext) could leak information about secret input. This is stricter than the protocol API, where a uniform error is RECOMMENDED but not required; the CLI requires it. Usage errors (status `2`) are exempt, as they do not depend on secret input. This requirement governs the stderr message and exit status only; timing and other side channels are outside the CLI contract and are addressed by the protocol specification.

9 Fixed Public Test Key

For cross-implementation testing, `ob` supports a **fixed public test key** via the `-K`/`--keyless` flag. Implementations MUST use this exact value for test-vector compatibility. The key is public and provides no security; implementations SHOULD warn when it is supplied through `--key` or `$OBORON_KEY` rather than `--keyless`. On a failed `dec`, that warning MUST be suppressed, so that only the uniform decryption-failure message (§8) appears.

Hex (128 chars) — canonical:

```
381284633d02ea5f35df8596b5cc4218310060468e8b465455a415174ea6e966a9f48eec4ba446ddfc
```

10 Test Vector Format

Test vectors are UTF-8 JSON Lines (JSONL) files, published at https://gitlab.com/oboron/oboron-test-vectors; an implementation MUST pass the vector set carried by the release tag matching this specification's version. Each non-empty line MUST be a single JSON object; blank lines and a leading byte-order mark MUST be rejected, while a single trailing newline at end of file is permitted. Core-scheme vectors bind to the fixed public test key (§9) and carry no per-vector key field. An `ob` conformance run processes only vectors whose scheme is a core scheme (§5); a shared file MAY also carry non-core (obu) vectors, which belong to the obu CLI and are skipped here.

A positive vector carries `"format"`, `"plaintext"`, and `"obtext"`; `"description"` is optional, and implementations MUST ignore unknown fields unless configured for strict schema validation:

```
{"format": "dgcmsiv.c32", "plaintext": "hello", "obtext": "..."}
```

A *negative vector* (in a separate file) asserts that an operation MUST fail. It carries an `op` (`enc` or `dec`), a `format`, the offending `input` string, and an informative `reason`; running `op` on `input` under the test key MUST fail as an operation failure with status `1`:

```
{"op": "dec", "format": "dsiv.c32", "input": "...", "reason": "..."}
```

Test Strategy by Scheme Type

Scheme type	Enc test	Dec test	Roundtrip
Deterministic (`dsiv`, `dgcmsiv`)	Exact match against vector obtext	Exact match against vector plaintext	N/A (exact enc match proves roundtrip)
Probabilistic (`psiv`, `pgcmsiv`)	N/A (output differs each run)	Exact match against vector plaintext	enc then dec; assert equals original plaintext

Any vector whose plaintext or obtext begins with `-`, or whose plaintext contains leading or trailing line breaks, MUST be delivered on stdin (with `--raw` when a line ending must be preserved) or after the `--` separator — not as a bare positional argument, where a leading `-` would be parsed as an option.

11 Completion (OPTIONAL)

Generating shell completion scripts is an OPTIONAL extension, not required for conformance. If provided:

```
ob completion <SHELL>
```

where <SHELL> is one of bash, zsh, fish, or powershell. The completion script is written to stdout and the command exits 0; a missing or unsupported shell identifier is a usage error (status 2).

12 Security Considerations

Keys in arguments. Supplying a key with --key can expose it through process listings, shell history, terminal scrollback, and logs. Prefer $OBORON_KEY or a dedicated secret manager where available.

Environment variables. $OBORON_KEY is a convenience interface, not a secret-management system; environment variables can be visible to child processes and, on some systems, to other same-user tools.

Diagnostics. Error messages MUST NOT include key material, plaintext, or decrypted output, and SHOULD NOT include full obtexts.

Test key. The fixed public test key provides no security; implementations SHOULD make this clear in help output and documentation. Because -K (keyless) and -k (key) differ only in case, a real secret passed to -K by mistake is encrypted under the public key with no protection.

Cryptographic considerations (authentication, deterministic leakage, randomness, key scope) are specified in the Oboron Protocol Specification and apply to the CLI unchanged.

References

- BCP 14 — RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels*; RFC 8174, *Ambiguity of Uppercase vs. Lowercase in RFC 2119 Key Words*.
- Oboron Protocol Specification — https://oboron.org/protocol-spec/oboron-v1.0.html (formats, key handling, UTF-8, and cryptographic considerations).
- Oboron test vectors — https://gitlab.com/oboron/oboron-test-vectors.