Outbound TLS for the Console Through rustls

STATE

committed

STATUS

Committed

TYPE

Architectural Specification

AUTHORS

Vikrant Rathore

CREATED

2026-10-15

UPDATED

2026-10-16

SUMMARY

Admits rustls with the ring provider and the system trust store for the console's outbound HTTPS and WSS connections only, keeping the data plane free of external crates.

DISCUSSION

No external discussion URL assigned

LABELS

console, security, rust

Record status

The catalog defines the review process and the meaning of each state. This numbered record forms part of the catalog.

Contents

1. Abstract 1
2. Motivation and scope 2
3. Specification 2
4. Rationale and alternatives 2
5. Security and operational considerations 2
6. Backwards compatibility 3
7. Acceptance criteria and implementation status 3
8. Open questions 3
9. Review history 3
10. References 4

1. Abstract

The console makes outbound TLS connections: management peer streams to other consoles (WSS, SID 0007), webhook deliveries over HTTPS, GeoIP dataset downloads from the publishers' HTTPS hosts, and the `sibuna console` command-line client talking to a remote console. The Zig daemon used the TLS client of the Zig standard library. Rust's standard library has none. We admit `rustls` 0.23 with the `ring` cryptography provider and `rustls-native-certs` for the system trust store, for the console only. The data plane, which terminates no TLS and opens no TLS connection, stays free of external crates. The same choice gives PostgreSQL connections TLS (SID 0011's open question).

2. Motivation and scope

The console's outbound connections carry credentials (the peer key, webhook secrets) and data whose integrity matters (GeoIP datasets, policy exports). They must verify certificates against the system roots or an operator-supplied CA file, check host names, and refuse plaintext except to literal loopback addresses, exactly as SID 0007 specifies. Writing a TLS 1.3 client in-house means X25519, AES-GCM and ChaCha20-Poly1305, HKDF, ECDSA and RSA signature verification, X.509 parsing and path validation: a security-critical project larger than Sibuna's data plane, with a long tail of interoperability cases. Calling an installed `curl` covers one-shot downloads and webhooks but not long-lived bidirectional peer streams.

This record covers outbound client TLS. Sibuna still terminates no inbound TLS: ingress TLS remains the job of Caddy or Nginx (SID 0002), and the console listener stays behind a trusted HTTPS proxy for remote access (SID 0007).

3. Specification

- **Crates.** `rustls` 0.23 (features `std`, `tls12`, `ring`), `ring` 0.17, `rustls-pki-types`, `rustls-webpki`, `rustls-native-certs` 0.8, and their transitive dependencies (`untrusted`, `subtle`, `zeroize`, `getrandom`, `cfg-if`, `libc`, `once_cell`, `openssl-probe`): thirteen crates in all at run time, declared once in the root manifest with versions pinned by the lock file. The `logging` feature of `rustls` stays off, so `log` is not among them. `ring`'s build script adds three build-time crates (`cc`, `shlex`, `find-msvc-tools`) and needs a C compiler; they are not linked into the daemon.
- **Confinement.** One crate, `sibuna-tls`, wraps them behind a small blocking API: `connect(host, port, trust, deadline) -> TlsStream` (and a `Connector` built once from a `Trust` for repeated connections) implementing `Read` and `Write` over a `TcpStream`, and `Trust { system: bool, ca_file: Option<PathBuf> }`. Only the console crates and the PostgreSQL TLS feature of `sibuna-storage` depend on it, behind the daemon's `console` and `postgres-tls` features. A build without those features contains none of these crates.
- **Policy.** TLS 1.3 and TLS 1.2 with the `ring` provider's default cipher suites. Certificates are verified against the system roots, or the operator's PEM file when `--console-peer-ca-file` names one (which replaces the system roots for peers, as in the Zig daemon). Host-name verification is always on and cannot be disabled. Plain HTTP is allowed only to literal loopback addresses, decided before any connection is made. Every connection has a deadline covering connect, handshake and the exchange.
- **Bounds.** Response heads and bodies read over TLS keep the limits of SID 0007 (for example the GeoIP compressed-input bound). One redirect is followed for GeoIP downloads, only to the publisher hosts SID 0007 lists.
- **Plain HTTP to public addresses.** The Zig webhook validator accepted `http://` to public IP literals, although SID 0007 and the operator documentation allow plain HTTP only to loopback. The Rust console enforces the documented rule.
- **Replacing the interim path.** `sibuna-net`'s outbound module currently runs `curl` for HTTPS; it moves to `sibuna-tls`, and the `curl` path is removed.

4. Rationale and alternatives

`rustls` is the mature TLS implementation of the Rust ecosystem: actively maintained, used widely, independently audited, and memory-safe apart from the assembly in `ring`. An in-house TLS client was considered and rejected for the reasons above. `native-tls` (OpenSSL) adds a C library at run time. An installed `curl` cannot hold the peer streams. The maintainer chose `rustls` for the console on 2026-10-15.

5. Security and operational considerations

The admitted crates are reviewed when their versions change. The trust store is read at start-up; certificate rotation on peers needs no restart, but a change of the CA file does, as in the Zig daemon. A connection failure, a certificate error or a host-name mismatch is reported in the console's peer status and audit records and never falls back to plaintext.

6. Backwards compatibility

Command-line options and behaviour are those of SID 0007. Peers and webhook endpoints that worked with the Zig daemon's TLS client work with `rustls` when they offer TLS 1.2 or 1.3 with the `ring` cipher suites, which every current server does.

7. Acceptance criteria and implementation status

1. `cargo tree -p sibuna --no-default-features` lists no external crate; with `console` it lists exactly the crates above in addition to `sqlx` and `tokio`.
2. Tests connect to a local TLS server with a test CA (accept), a wrong host name (refuse), an untrusted certificate (refuse), and plain HTTP to a non-loopback address (refuse before connecting).
3. A two-console peer stream over WSS with a private CA passes the console peer tests of SID 0007.

Status: all three criteria are met (see the review history). Committed on 2026-10-16; released in 1.0.0-beta.1.

8. Open questions

- None.

9. Review history

- 2026-10-15: drafted after the maintainer chose `rustls` over an in-house TLS client and over running `curl`.
- 2026-10-15: corrected the crate list and the API signature after the `sibuna-tls` implementation; recorded the stricter webhook rule.
- 2026-10-15 (Rust port): two further differences from the Zig TLS client, recorded in `sibuna-tls`. Zig's certificate bundle skipped CA certificates that had already expired when the file was read; rustls (webpki) uses a trust anchor's name and key only and ignores its validity period, the convention of RFC 5280, while the leaf and the intermediates are still checked for expiry. The per-connection 6 MiB allocation ceiling of the Zig peer client is not carried over, because rustls bounds its own buffers (handshake messages, records, and 64 KiB of buffered plaintext by default).
- 2026-10-15 (Rust port): the console's outbound HTTP client (`sibuna-console::client`, used by the CLI, GeoIP downloads and webhooks) is one module: an `Origin` and `Endpoint` with an optional pinned address and a `Connector`, one deadline over connect and exchange, a body bound, and redirects returned to the caller.
- 2026-10-15 (Rust port): `sibuna-net`'s outbound module keeps only the address policy and bounded resolution. `Post`, `post`, `Pinned` and the curl HTTPS path are removed; webhooks and GeoIP downloads use `sibuna-console::client`.
- 2026-10-15 (Rust port): the test ingress lives in `sibuna-tls` behind the `testing` feature, so rustls stays confined to `sibuna-tls`. Criterion 3 is verified by `crates/console/tests/peers.rs`: two consoles on SQLite peer over WSS through the ingress with the private test CA; an untrusted certificate and a host-name mismatch are refused in the handshake and nothing reaches the console; a replayed, duplicated or forged proof gets 403 `CONSOLEPEER`; peer statistics reach the dashboard and the nodes topic. A certificate refusal shows as `rejected` in the peer status. Each peer worker reads the trust store once.
- 2026-10-15 (Rust port): a peer that becomes `rejected` (refused certificate, proof, upgrade reply or configuration) writes one `peer.rejected` audit row (subject: node, target: origin, reason in the after summary) on that transition, never per retry. Unreachable peers are reported in the peer status only, because they retry every few seconds. A peer whose worker cannot start is reported `stale`.
- 2026-10-15 (Rust port, PostgreSQL TLS built): the planned `postgres-tls` feature exists. It enables sqlx's `tls-rustls-ring-native-roots`, which reuses the rustls 0.23, ring 0.17 and rustls-native-certs 0.8 that `sibuna-tls` admits. It adds no crate to a build with the console, and the ten crates `sibuna-tls` brings to a build without it; there is one version of rustls. rustls is therefore reached in two places, `sibuna-tls` (the console) and sqlx (PostgreSQL), while our code calls rustls only in `sibuna-tls`. sqlx 0.9 cannot take a stream that `sibuna-tls` built, because `PgConnection` opens its own socket. `tls-rustls-ring-webpki` (webpki-roots), `tls-rustls-aws-lc-rs` (aws-lc-sys) and `tls-native-tls` (OpenSSL) were rejected because they add crates.

10. References

- SID 0002: deployment behind a TLS-terminating ingress.
- SID 0007: the console, its peers, webhooks and GeoIP downloads.
- SID 0010: the dependency policy of the Rust port.
- SID 0011: storage through sqlx, and PostgreSQL TLS.
- RFC 8446 (TLS 1.3); the rustls project and its audit reports.

Lifecycle event

```
2026-10-15: prediscussion -> discussion. Maintainer decision on console TLS,
recorded before implementation
```

Lifecycle event

```
2026-10-15: discussion -> accepted. The maintainer chose rustls for the
console on 2026-10-15
```

Lifecycle event

```
2026-10-16: accepted -> committed. Implemented and released in Sibuna
1.0.0-beta.1; reviewed by the maintainer on 2026-10-16. cargo tree -p sibuna
--no-default-features lists no external crate; crates/tls/tests/connect.rs
(accept, wrong host name, untrusted issuer, plain HTTP only to loopback);
crates/console/tests/peers.rs (two consoles over WSS with a private CA);
release v1.0.0-beta.1
```