SmolTLV: A Small TLV Binary Encoding

Contents

1. Status of This Document

This document is a draft specification for **SmolTLV**, a compact TLV (Type–Length–Value) binary encoding designed for simple incremental parsing on resource-constrained systems.

Implementations **MUST** follow the normative requirements described using capitalized keywords (**MUST**, **SHOULD**, **MAY**), interpreted as in RFC 2119.

2. Overview

SmolTLV encodes a sequence of **items**. Each item is self-delimiting and consists of:

- a 1-byte **Type**,
- a 24-bit big-endian **Length** (payload length in bytes),
- a **Payload** of exactly **Length** bytes.

The format supports both primitive values (e.g. integers, byte strings) and containers (lists and dictionaries) whose payload is a concatenation of nested SmolTLV items.

3. Goals

- **Incremental parsing**: Decoders can parse items from a streaming source and skip unknown types.
- **Memory mapping friendly**: Length-delimited items allow walking a buffer without copying.
- **Small and obviously-correct implementations**: Fixed-size header (4 bytes) and no indefinite-length constructs.

4. Data Model

SmolTLV defines the following built-in types:

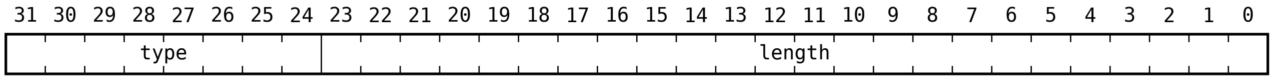
Type name	**Value**	**Meaning**
Null	`0x00`	An explicit null value. Payload length **MUST** be 0.
True	`0x01`	Boolean true. Payload length **MUST** be 0.
False	`0x02`	Boolean false. Payload length **MUST** be 0.
Int	`0x03`	Signed integer. Payload length **MUST** be 8.
Bytes	`0x04`	Byte string (opaque blob).
String	`0x05`	UTF-8 text string.
List	`0x06`	Ordered sequence of items (container).
Dict	`0x07`	Key/value mapping (container).

Type values outside this set are **reserved for extensions**. Decoders **MUST** be able to skip unknown types based solely on the length field.

5. Wire Format

5.1. Item Encoding

Each item has header with following layout (in big-endian byte order):



Where:

- **Type** is an unsigned 8-bit value.
- **Length** is an unsigned 24-bit integer, encoded big-endian, giving the number of payload bytes.
- Maximum payload length is 0xFFFFFF (16,777,215) bytes.

Followed by exactly **Length** bytes of payload data.

5.2. Length Encoding

Length is encoded as three bytes:

```
len = (b1 << 16) | (b2 << 8) | b3
```

where `b1`, `b2`, `b3` are the three length bytes in network (big-endian) order.

5.3. Endianness

All multi-byte integers in SmolTLV payloads **MUST** be encoded big-endian unless otherwise specified.

6. Integer Encoding

The `SMOLTLV_TYPE_INT` payload encodes a signed 64 bit integer using **two's complement big-endian**.

Payload length must be 8 bytes.

7. Bytes and Strings

7.1. Bytes

`SMOLTLV_TYPE_BYTES` payload is an opaque byte string of length `Length`. No interpretation is implied.

7.2. String

`SMOLTLV_TYPE_STRING` payload **MUST** be valid UTF-8. Implementations **MAY** reject invalid UTF-8.

Canonical form (if enforced by a profile) **SHOULD** use Unicode NFC, but SmolTLV does not require normalization by default.

8. Containers

8.1. List

`SMOLTLV_TYPE_LIST` payload is a concatenation of zero or more complete SmolTLV items.

A decoder processes a list by creating a bounded cursor over the list payload and repeatedly parsing items until the cursor reaches the end of the payload.

The list payload **MUST NOT** contain partial items or padding; i.e., the nested items **MUST** exactly fill the list payload length.

8.2. Dict

`SMOLTLV_TYPE_DICT` payload is a concatenation of SmolTLV items representing key/value pairs:

- The payload **MUST** contain an even number of nested items.
- Items are interpreted as `(key1, value1, key2, value2, ...)`.

8.2.1. Key Type

By default, SmolTLV permits any item type as a dictionary key. However, many deployments **SHOULD** restrict keys to `SMOLTLV_TYPE_STRING` for interoperability.

If a deployment profile restricts dict keys, encoders **MUST** follow that profile and decoders **SHOULD** enforce it.

8.2.2. Duplicate Keys

SmolTLV does not define duplicate-key semantics. A profile **MUST** specify one of:

- reject duplicates,
- last-wins,
- first-wins,
- collect-all.

9. Canonical / Deterministic Encoding (Optional Profile)

SmolTLV itself allows multiple equivalent encodings (e.g., dictionary order, integer minimality is required but key ordering is not). For deterministic encoding, a profile **MAY** require:

- Dict keys **MUST** be strings.
- Dict entries **MUST** be sorted lexicographically by the UTF-8 byte sequence of the key.
- Duplicate keys **MUST** be rejected.
- Strings **MUST** be valid UTF-8.

If deterministic encoding is required, all producers and consumers **MUST** implement the same profile.

10. Error Handling

Decoders **SHOULD** distinguish:

- **Need more data**: insufficient bytes to complete the header or payload.
- **Format error**: invalid structure (e.g., container payload not fully consumable into items, invalid minimal integer encoding, invalid UTF-8 if enforced).

Decoders **MUST** be able to skip unknown types by advancing the cursor by `4 + Length` bytes.

11. Security Considerations

Implementations **MUST** defend against:

- Length fields that exceed available buffer space (bounds checks).
- Excessive nesting depth (use a maximum depth).
- Excessive total allocation (cap sizes when materializing structures).
- Quadratic behavior when processing dictionaries with adversarial key patterns.

12. Example Encodings

12.1. A null value

The null value:

- Type = Null (0x00)
- Length = 0
- Payload = (none)

```
00 00 00 00
```

12.2. Boolean values

The boolean true value:

- Type = Bool True (0x01)
- Length = 0
- Payload = (none)

```
01 00 00 00
```

12.3. Integers

The integer value `42`:

- Type = Int (0x03)
- Length = 8
- Payload = 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x2A

```
03 00 00 08  00 00 00 00  00 00 00 2A
```

The integer value `-1`:

- Type = Int (0x03)
- Length = 8
- Payload = 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF

```
03 00 00 08  FF FF FF FF  FF FF FF FF
```

12.4. Simple string

The value `"hi"`:

- Type = String (0x05)
- Length = 2
- Payload = 0x68 0x69

```
05 00 00 02  68 69
```

12.5. List

The list: `["a", "b"]`

```
06 00 00 0A
   05 00 00 01  61
   05 00 00 01  62
```

12.6. Dictionary

The dictionary `{"k": 123}`

```
07 00 00 0B
   05 00 00 01  6B
   03 00 00 01  7B
```

13. Appendix: Type Registry

The following type assignments are defined by this specification:

```
SMOLTLV_TYPE_NULL       = 0x00,
SMOLTLV_TYPE_BOOL_TRUE  = 0x01,
SMOLTLV_TYPE_BOOL_FALSE = 0x02,
SMOLTLV_TYPE_INT        = 0x03,
SMOLTLV_TYPE_BYTES      = 0x04,
SMOLTLV_TYPE_STRING     = 0x05,
SMOLTLV_TYPE_LIST       = 0x06,
SMOLTLV_TYPE_DICT       = 0x07,
```

Values `0x08..0xFF` are reserved for future assignment or application-specific extensions.