Contents

Payload Schema Language Reference

Complete reference for the LoRa Alliance Payload Schema specification (v0.3.2).

Document Structure

```
name: string                # REQUIRED: unique identifier
version: integer            # REQUIRED: schema version
endian: big|little          # Default: big
description: string         # Optional
direction: uplink|downlink|bidirectional  # Default: uplink
fields: [...]               # Field definitions (or use ports)
ports:                      # Port-based routing (or use fields)
  1: { fields: [...] }
  2: { fields: [...] }
definitions:                # Reusable field groups
  common_header: [...]
metadata:                   # Network metadata enrichment
  include: [...]
  timestamps: [...]
test_vectors: [...]         # Test cases
downlink_commands: [...]    # Command definitions (for downlink)
```

Field Types

Integer Types

Type	Bytes	Description
`u8`, `u16`, `u24`, `u32`, `u64`	1,2,3,4,8	Unsigned integer
`s8`, `s16`, `s24`, `s32`, `s64`	1,2,3,4,8	Signed integer (two's complement)

Note: 24-bit types (`u24`, `s24`) are commonly used for GPS coordinates in compact formats.

Floating Point Types

Type	Bytes	Description
`f16`, `f32`, `f64`	2,4,8	IEEE 754 float

Decimal Types

Type	Description
`udec`	Unsigned nibble-decimal (BCD-like)
`sdec`	Signed nibble-decimal

String/Byte Types

Type	Description
`ascii`	ASCII string (requires `length:`)
`hex`	Hex string output (requires `length:`)
`bytes`	Raw bytes (requires `length:`)
`base64`	Base64 encoded output (requires `length:`)

Bytes format options:

```
- name: device_eui
  type: bytes
  length: 8
  format: hex              # "0011223344556677"
  format: hex:upper        # "0011223344556677" uppercase
  format: base64           # Base64 encoded
  format: array            # [0, 17, 34, 51, ...]
  separator: ":"           # "00:11:22:33:44:55"
```

Special Types

Type	Description
`bool`	Boolean (0=false, nonzero=true)
`number`	Computed field (no wire bytes)
`string`	Literal string constant
`skip`	Skip bytes (padding)
`enum`	Enumerated values
`bitfield_string`	Bit flags as string

Bool Type

Boolean fields extract a single bit and convert to true/false:

```
- name: motion_detected
  type: bool
  bit: 0            # Bit position (0-7)
  consume: 1        # Advance past byte (optional)
```

By default, bool fields do not advance the position, allowing multiple bits from the same byte. Add `consume: 1` to advance after reading.

Bitfields

```
type: u8[0:3]       # Bits 0-3 of byte (4 bits)
type: u16[8:15]     # High byte of u16
```

Endian Prefix

```
type: le_u16        # Little-endian
type: be_u32        # Big-endian (explicit)
```

Byte Group (multiple values from shared bytes)

```
- byte_group:
    size: 3
    fields:
      - name: value_a
        type: u8[0:3]
      - name: value_b
        type: u8[4:7]
```

Shorthand (size inferred from field types):

```
- byte_group:
    - name: value_a
      type: u8[0:3]
    - name: value_b
      type: u8[4:7]
```

Arithmetic Modifiers

Applied in YAML key order:

```
- name: temperature
  type: s16
  div: 10           # Divide by 10
  add: -40          # Then subtract 40
  # Result: (raw / 10) - 40
```

Modifier	Effect
`add: n`	Add offset
`mult: n`	Multiply
`div: n`	Divide

Lookup Tables

```
- name: status
  type: u8
  lookup: ["off", "on", "error", "unknown"]
```

Computed Fields

Polynomial (calibration curves)

```yaml
- name: raw_value
  type: u16
  div: 50

- name: calibrated
  type: number
  ref: $raw_value
  polynomial: [0.0000043, -0.00055, 0.0292, -0.053]  # Descending powers
  # Result: ax³ + bx² + cx + d
```

Cross-Field Computation

```yaml
- name: ratio
  type: number
  compute:
    op: div           # add, sub, mul, div, mod, idiv
    a: $field1        # Field reference or literal
    b: $field2
```

Available Operations:

Op	Description	Example
`add`	Addition	`a + b`
`sub`	Subtraction	`a - b`
`mul`	Multiplication	`a * b`
`div`	Division	`a / b`
`mod`	Modulo (remainder)	`int(a) % int(b)`
`idiv`	Integer division	`int(a) // int(b)`

Nibble Extraction Example:

```yaml
- name: rawByte
  type: u8

- name: upperNibble
  type: number
  compute:
    op: idiv
    a: $rawByte
    b: 16

- name: lowerNibble
  type: number
  compute:
    op: mod
    a: $rawByte
    b: 16
```

Guard Conditions

```yaml
- name: safe_ratio
  type: number
  compute:
    op: div
    a: $numerator
    b: $denominator
  guard:
    when:
      - field: $denominator
        gt: 0           # gt, gte, lt, lte, eq, ne
    else: 0             # Fallback if condition fails
```

Formula (Deprecated)

Legacy formula syntax for simple expressions. **Use `compute` instead.**

```
- name: temp_c
  type: number
  formula: "($raw_temp - 4000) / 100"  # String expression
```

Note: `formula` is deprecated in favor of the more explicit `compute` syntax which provides better validation and error handling.

Transform Operations

```
transform:
  - sqrt: true          # √x
  - abs: true           # |x|
  - pow: 2              # x²
  - floor: 0            # Clamp lower bound
  - ceiling: 100        # Clamp upper bound
  - clamp: [0, 100]     # Both bounds
  - log10: true         # Base-10 logarithm
  - log: true           # Natural logarithm
```

Conditional Parsing

Match (by field value)

```
- name: msg_type
  type: u8

- match:
    field: $msg_type
    cases:
      1:
        - name: temperature
          type: s16
      2:
        - name: humidity
          type: u8
```

Flagged (bitmask presence)

```
- name: flags
  type: u8

- flagged:
    field: flags
    groups:
      - bit: 0
        fields:
          - name: temperature
            type: s16
      - bit: 1
        fields:
          - name: humidity
            type: u8
```

Named Encodings

```
- name: signed_value
  type: u16
  encoding: sign_magnitude   # Also: bcd, gray
```

Encoding	Description
`sign_magnitude`	Bit 15 is sign, bits 0-14 are magnitude
`bcd`	Binary-coded decimal
`gray`	Gray code

Value-Range Matching

For value-dependent transformations:

```
- name: signed_value
  type: u16
  match_value:
    - when: "< 32768"
      # No transform (value as-is)
    - when: ">= 32768"
      add: -65536
```

Prefer `encoding:` for standard patterns; use `match_value` for custom ranges.

Bitfield String

Parse bits into formatted string (e.g., version numbers):

```
- name: firmware_version
  type: bitfield_string
  length: 2               # Bytes to read
  delimiter: "."          # Separator between parts
  prefix: "v"             # Optional prefix
  parts:
    - [8, 8]              # [start_bit, width] → major
    - [0, 8]              # [start_bit, width] → minor
# Input: 0x0102 → Output: "v1.2"
```

Test Vectors

```
test_vectors:
  - name: basic_reading
    description: "Normal temperature reading"
```

```yaml
    payload: "00 E7 32"        # Hex, spaces ignored
    expected:
      temperature: 23.1
      humidity: 50

  - name: encoding_test
    direction: encode          # Test encoding (JSON → binary)
    input:
      temperature: 23.1
      humidity: 50
    expected_payload: "00E732"
```

Enum Type

```yaml
- name: status
  type: enum
  base: u8
  values:
    0: "off"
    1: "on"
    2: "error"
```

Repeat (Arrays)

```yaml
# Count-based
- name: readings
  type: repeat
  count: 4
  fields:
    - name: value
      type: u16

# Field-based count
- name: readings
  type: repeat
  count_field: num_readings
  fields:
    - name: value
      type: u16

# Until end of payload
- name: entries
  type: repeat
  until: end
  fields:
    - name: value
      type: u16
```

Nested Objects

```yaml
- name: gps
  type: object
  fields:
    - name: latitude
```

```
      type: s32
      div: 10000000
    - name: longitude
      type: s32
      div: 10000000
```

Variables

Store values for later reference:

```
- name: device_type
  type: u8
  var: dev_type      # Store as variable

- match:
    field: $dev_type  # Reference variable
    cases:
      1: [...]
      2: [...]
```

TLV (Type-Length-Value)

Parse tag-based variable content. Supports single and multi-byte tags.

```
- tlv:
    tag_size: 1           # Tag size in bytes (1, 2, or more)
    length_size: 1        # Length field size (0 = implicit/no length field)
    merge: true           # Merge results into parent (default)
    unknown: skip         # skip|error|raw for unknown tags
    cases:
      0x01:
        - name: temperature
          type: s16
      0x02:
        - name: humidity
          type: u8
```

Multi-Byte Tags (Tektelic-style)

```
- tlv:
    tag_size: 2           # 2-byte tags (big-endian)
    length_size: 0        # Implicit length from case definition
    cases:
      0x00BA:             # Battery status
        - name: battery_level
          type: u8
      0x0B67:             # Ambient temperature
        - name: temperature
          type: s16
          div: 10
```

Composite Tags

For protocols with multi-field tag structures:

```
- tlv:
    tag_fields:
      - name: channel
        type: u8
      - name: sensor_type
        type: u8
    tag_key: [channel, sensor_type]
    cases:
      [1, 0x67]:          # Channel 1, temperature type
        - name: ch1_temperature
          type: s16
```

Match Patterns

```
- match:
    field: $msg_type
    cases:
      1: [...]                    # Exact match
      2..5: [...]                 # Range (2,3,4,5)
      0x10..0x1F: [...]           # Hex range
      _: [...]                    # Default case
```

Skip (Padding)

```
- name: _reserved
  type: skip
  length: 2          # Skip 2 bytes
```

Definitions (Reusable Groups)

```
definitions:
  header:
    - name: version
      type: u8
    - name: flags
      type: u8

fields:
  - use: header      # Include definition
  - name: payload
    type: bytes
    length: 10
```

Schema Composition

Cross-File References

```
# Reference definitions from other files
fields:
  - use: common/headers.yaml#message_header
  - use: ./local-defs.yaml#sensor_block
  - name: data
    type: u16
```

Standard Library

```yaml
# Use standard sensor definitions
fields:
  - use: std/sensors/temperature
    rename: ambient_temp
  - use: std/sensors/humidity
  - use: std/sensors/battery_percent
```

Field Renaming

```yaml
- use: gps_position
  rename: device_location    # Rename single field

- use: sensor_block
  prefix: indoor_           # Prefix all fields: indoor_temp, indoor_humidity
```

Port-Based Routing

```yaml
ports:
  1:
    description: "Sensor data"
    fields:
      - name: temperature
        type: s16
  2:
    description: "Status"
    fields:
      - name: battery
        type: u8
```

Downlink Encoding

Direction Property

```yaml
name: device_config
direction: downlink        # or: bidirectional

fields:
  - name: interval
    type: u16
    mult: 60               # Minutes to seconds
  - name: threshold
    type: u8
```

Arithmetic Reversal

When encoding downlinks, arithmetic is reversed automatically:

Decode (uplink)	Encode (downlink)
`mult: n`	`div: n`
`div: n`	`mult: n`
`add: n`	`sub: n`

Command-Based Downlinks

```yaml
name: device_commands
direction: downlink

downlink_commands:
  set_interval:
    command_id: 0x01
    fields:
      - name: interval_minutes
        type: u16

  reboot:
    command_id: 0x02
    fields: []            # No payload

  set_threshold:
    command_id: 0x03
    fields:
      - name: low
        type: u8
      - name: high
        type: u8
```

Bidirectional Schema

```yaml
name: env_sensor
direction: bidirectional

fields:
  # Uplink: sensor readings
  - name: temperature
    type: s16
    div: 10
  - name: humidity
    type: u8

downlink_commands:
  # Downlink: configuration
  set_interval:
    command_id: 0x01
    fields:
      - name: interval
        type: u16
```

Network Metadata Enrichment

Include TS013 input fields in decoder output:

```yaml
metadata:
  include:
    - name: received_at
      source: $recvTime
    - name: rssi
      source: $rxMetadata[0].rssi
    - name: snr
      source: $rxMetadata[0].snr
    - name: port
      source: $fPort
```

Timestamp Modes

```
metadata:
  timestamps:
    # Mode 1: Use network receive time
    - name: timestamp
      mode: rx_time

    # Mode 2: Compute from offset field
    - name: measurement_time
      mode: subtract
      offset_field: seconds_ago

    # Mode 3: Convert Unix epoch from payload
    - name: device_time
      mode: unix_epoch
      field: unix_timestamp
```

Available TS013 Input Fields

Field	Description
`$fPort`	LoRaWAN FPort
`$recvTime`	Server receive time (ISO 8601)
`$devEui`	Device EUI
`$rxMetadata[n].rssi`	RSSI from gateway n
`$rxMetadata[n].snr`	SNR from gateway n
`$rxMetadata[n].gatewayId`	Gateway identifier

Output Format Hints

```
- name: temperature
  type: s16
  div: 10
  unit: "°C"
  ipso: 3303          # IPSO Smart Object ID
  senml_unit: "Cel"   # SenML unit
```

Common IPSO Smart Objects:

ID	Name	Use Case
3200	Digital Input	Binary sensors
3301	Illuminance	Light (lux)
3303	Temperature	Temperature (°C)
3304	Humidity	Humidity (%)
3308	Set Point	Thermostat setpoints
3316	Voltage	Battery voltage
3323	Pressure	Pressure sensors
3325	Concentration	CO2/gas (ppm)

ID	Name	Use Case
3330	Distance	Range/level sensors
3337	Positioner	Valve position (%)

M-Bus / Utility Format Hints

```yaml
- name: volume
  type: u32
  div: 1000
  unit: "m³"
  mbus_dif: 0x04        # 32-bit integer
  mbus_vif: 0x14        # Volume in 0.001 m³
```

Semantic Fields

Fields for value quality tracking and IoT interoperability.

Valid Range

Declares expected output value bounds. Out-of-range values produce quality warnings but are not modified (unlike `clamp`).

```yaml
- name: temperature
  type: s16
  div: 100
  unit: "°C"
  valid_range: [-40, 85]    # Expected operating range
```

Interpreter Behavior:

```
# Normal reading
{
    "temperature": 23.45,
    "_quality": {"temperature": "good"}
}

# Out-of-range (e.g., sensor failure reads -999)
{
    "temperature": -999.0,
    "_quality": {"temperature": "out_of_range"},
    "_warnings": ["temperature: value -999.0 outside valid range [-40, 85]"]
}
```

Resolution

Documents minimum detectable change. Useful for fixed-point scaling and code generation.

```yaml
- name: temperature
  type: s16
  div: 100
  unit: "°C"
  resolution: 0.01    # 0.01°C steps
```

Interpreter Behavior: Included in metadata output. Optional rounding to resolution in strict mode.

UNECE Unit Codes

Standard unit identifiers per UNECE Recommendation 20.

```
- name: temperature
  type: s16
  div: 100
  unit: "°C"
  unece: "CEL"        # UNECE code for Celsius
```

Common UNECE Codes:

Measurement	Code	Display
Temperature (C)	CEL	°C
Temperature (F)	FAH	°F
Humidity (%)	P1	%
Pressure (Pa)	PAL	Pa
Pressure (bar)	BAR	bar
Voltage	VLT	V
Current	AMP	A
Power (W)	WTT	W
Distance (m)	MTR	m
Distance (mm)	MMT	mm
Mass (kg)	KGM	kg
Time (s)	SEC	s

Combined Example

```
- name: temperature
  type: s16
  div: 100
  unit: "°C"
  valid_range: [-40, 85]
  resolution: 0.01
  unece: "CEL"
  ipso: 3303
```

Compact Format (Alternative Syntax)

Basic Compact

```
# Verbose
fields:
  - name: temp
    type: s16
  - name: hum
    type: u8

# Compact equivalent
format: ">hB"         # struct-like format string
names: [temp, hum]
```

Inline Field Names

```
# Single-line format with names
fields: ">B:version H:length I:timestamp"
```

```
# With padding (2x = skip 2 bytes)
fields: ">B:type 2x H:value I:timestamp"
```

Format Characters

Char	Type	Bytes
`b/B`	s8/u8	1
`h/H`	s16/u16	2
`i/I`	s32/u32	4
`q/Q`	s64/u64	8
`f`	f32	4
`d`	f64	8
`x`	skip	1
`>`	big-endian	-
`<`	little-endian	-

OTA Schema Transfer

Schemas can be transmitted over-the-air from device to network.

Binary Schema Encoding

Schemas compile to compact binary for transmission:

```
# ~5 bytes for simple field
- name: temperature
  type: s16
  div: 10

# Compiles to: 0x11 0x0A 0x00 [name...]
```

QR Code Embedding

```
LW:1:DevEUI:AppEUI:AppKey:SCHEMA:Base64EncodedSchema
```

Schema Validation

Validation Levels

Level	Description
ERROR	Must fix before use
WARNING	Should review
INFO	Best practice suggestion

Common Validations

```
# ERROR: Undefined variable reference
- match:
    field: $undefined_var    # Error: variable not defined

# WARNING: Missing IPSO for known sensor type
- name: temperature          # Warning: detected as temperature sensor
```

```
  type: s16                   # but missing ipso: annotation
  div: 10

# INFO: Consider adding unit
- name: voltage
  type: u16
  div: 1000                   # Info: consider adding unit: "V"
```

Quality Scoring

Schemas are scored for certification readiness:

Tier	Requirements
Bronze	Valid schema, parses without error
Silver	Test vectors present, all pass
Gold	Full metadata (units, IPSO), edge cases tested
Platinum	Bidirectional support, fuzz tested

Test Coverage Requirements

- Minimum 3 test vectors for basic coverage
- Edge cases: min/max values, error conditions
- All message types / ports exercised

TS013 Code Generation

Schemas generate TS013-compliant JavaScript decoders:

```
// Generated from schema
function decodeUplink(input) {
  // ... generated decoder logic
  return {
    data: { temperature: 23.1, humidity: 50 },
    warnings: [],
    errors: []
  };
}

function encodeDownlink(input) {
  // ... generated encoder logic
  return {
    fPort: 1,
    bytes: [0x00, 0xE7, 0x32]
  };
}
```

Complete Example

```
name: environment_sensor
version: 1
endian: big
direction: bidirectional
description: Temperature and humidity sensor with battery
```

```yaml
fields:
  - name: temperature
    type: s16
    div: 10
    unit: "°C"
    ipso: 3303
    valid_range: [-40, 85]

  - name: humidity
    type: u8
    unit: "%"
    ipso: 3304
    valid_range: [0, 100]

  - name: battery_mv
    type: u16
    unit: "mV"

  - name: battery_percent
    type: number
    ref: $battery_mv
    transform:
      - add: -2000        # 2000mV = 0%
      - div: 12           # 3200mV = 100%
      - clamp: [0, 100]
    unit: "%"
    ipso: 3316

downlink_commands:
  set_interval:
    command_id: 0x01
    fields:
      - name: interval_minutes
        type: u16

metadata:
  include:
    - name: rssi
      source: $rxMetadata[0].rssi

test_vectors:
  - name: normal
    payload: "00E7 32 0C80"
    expected:
      temperature: 23.1
      humidity: 50
      battery_mv: 3200
      battery_percent: 100

  - name: cold
    payload: "FF9C 5A 0BB8"
    expected:
      temperature: -10.0
      humidity: 90
      battery_mv: 3000
      battery_percent: 83.3
```

Quick Reference Card

```
TYPES:         u8 u16 u24 u32 u64 | s8 s16 s24 s32 s64 | f16 f32 f64 | bool
               ascii hex bytes base64 | number string | skip enum
               udec sdec | bitfield_string

STRUCTURES:    object | repeat | byte_group | tlv

MODIFIERS:     add mult div | lookup | polynomial | compute | guard | transform | match_value

CONDITIONALS: match (value dispatch) | flagged (bitmask) | tlv (tag dispatch)

TRANSFORMS:    sqrt abs pow floor ceiling clamp log10 log

COMPUTE OPS:   add sub mul div mod idiv

GUARD OPS:     gt gte lt lte eq ne

ENCODINGS:     sign_magnitude bcd gray

MATCH:         exact | range (n..m) | default (_)

REFERENCES:    $field_name | use: definition_name | file: path.yaml#def

SEMANTICS:     unit | ipso | senml_unit | valid_range | resolution | unece

DIRECTIONS:    uplink | downlink | bidirectional

METADATA:      include | timestamps (rx_time, subtract, unix_epoch)
```

See Also

- SCHEMA-DEVELOPMENT-GUIDE.md - Tutorial and best practices
- OUTPUT-FORMATS.md - Output format specifications (IPSO, SenML)
- C-CODE-GENERATION.md - Embedded firmware codec generation
- BIDIRECTIONAL-CODEC.md - Downlink encoding details