Meshtastic Message Protocol V1

Exported on 2026-03-14

```
# EmergencyHam Message Protocol (EMP) V1 Draft

## 1. Purpose

This document defines a compact, text-friendly message protocol for application traffic over
Meshtastic.

Primary goals:

- Keep frames small for LoRa payload limits.
- Support reliable command/response workflows.
- Borrow practical design patterns from APRS (addressing, ACK) and Winlink (structured, reliable
transfer).
- Be easy to parse on Raspberry Pi and Android.
- Provide a stable canonical model that can be bridged to multiple transport networks.

Status: Draft for discussion.

## 2. Design Principles

- Deterministic field order.
- Pipe-delimited top-level frame.
- Payload encoded to avoid delimiter collisions.
- Explicit ACK/NACK for control traffic.
- Duplicate suppression by `(SRC, MSGID)`.
- Simple, auditable behavior first; optional extensions later.

## 3. Transport Assumptions

- Underlying carrier is Meshtastic text messages.
- One EMP frame should fit in one mesh text payload when possible.
- Long messages can be fragmented using `FLAGS` and metadata.

## 4. Frame Format

Canonical frame:

`V1|TYPE|MSGID|SRC|DST|TS|TTL|FLAGS|PAYLOAD|CRC16`

Field delimiter: `|`

Character set for all non-payload fields: uppercase/lowercase letters, digits, `-`, `_`, `.`, `,`,
`:`.

Payload encoding: Base64url without padding.

CRC scope: everything from `V1` through the trailing `PAYLOAD` delimiter and payload bytes,
excluding final `|CRC16`.
```

CRC algorithm: CRC-16/CCITT-FALSE, represented as 4 hex chars uppercase.

5. Field Specification

Field	Required	Max Length	Description
`V1`	Yes	2	Protocol version literal.
`TYPE`	Yes	8	Message class: `TXT`, `CMD`, `ACK`, `NACK`, `TLM`, `FORM`, `PING`, `PONG`.
`MSGID`	Yes	16	Unique per source; recommended base36 timestamp+counter.
`SRC`	Yes	16	Source logical ID (callsign or node alias).
`DST`	Yes	16	Destination logical ID or `ALL`.
`TS`	Yes	20	UTC timestamp, ISO-like: `YYYYMMDDTHHMMSSZ`.
`TTL`	Yes	2	Hop/forward budget as integer string `00-15`.
`FLAGS`	Yes	64	Comma-separated key/value or bare tokens.
`PAYLOAD`	Yes	240	Base64url payload; may be empty string for ACK/PING/PONG.
`CRC16`	Yes	4	Frame integrity checksum.

6. TYPE Behavior

6.1 TXT

- Human-readable message content in payload.
- ACK optional.

6.2 CMD

- Structured command request.
- Requires `ACK` or `NACK` referencing same `MSGID`.

6.3 ACK

- Positive acknowledgement.
- `PAYLOAD` may include short status string.

6.4 NACK

- Negative acknowledgement.
- `PAYLOAD` should include concise reason code and optional detail.

6.5 TLM

- Telemetry snapshot (weather, voltage, service state).
- ACK optional unless requested by flag.

6.6 FORM

- Structured larger content.
- Fragmenting is expected for larger forms.

6.7 PING/PONG

- Link test and latency measurement.

7. FLAGS Definition

FLAGS is a comma-separated list.

Examples:

- `ACKREQ`
- `ENC=0`
- `FRAG=1/3`
- `PRIO=2`
- `CHAN=private`
- `RETRY=1`

Initial reserved keys:

- `ACKREQ` (bare token): sender requires explicit ACK/NACK.
- `FRAG=n/t`: fragment index and total.
- `CHAN=public|private|chN`: logical channel hint.
- `PRIO=0..3`: relative priority.
- `RETRY=n`: current retry count.
- `AUTH=tokenid`: token reference, not secret content.

8. Payload Encoding and Content

Payload is Base64url. Decoded content is UTF-8 text.

For structured content, decoded payload should be compact JSON.

Examples of decoded payloads:

- TXT: `"Team check-in at site A"`
- CMD: `{"cmd":"SET_INTERVAL","value":3600}`
- NACK: `{"code":"NOT_ALLOWED","detail":"sender not allowlisted"}`

9. Reliability Rules

9.1 Duplicate Suppression

- Receiver caches `(SRC, MSGID)` for at least 15 minutes.
- Duplicate frame with valid CRC is ignored but can be re-ACKed.

9.2 ACK Policy

- `CMD` and `FORM`: mandatory ACK/NACK.
- `TXT` and `TLM`: optional unless `ACKREQ` flag present.

9.3 Retries

- Default retries for ACK-required messages: 3 attempts.
- Backoff suggestion: 8 s, 20 s, 45 s plus random jitter.
- Set/update `RETRY=n` in FLAGS on retransmit.

9.4 Timeout

- Mark as failed if no ACK/NACK after final retry window.

10. Fragmentation Rules

- Use `FRAG=n/t` in FLAGS.
- All fragments use same `MSGID`, `SRC`, `DST`, `TS`.
- Each fragment carries a payload chunk.
- Reassembly timeout: 120 seconds from first fragment.
- Reassembled message processed only after all fragments pass CRC and sequence checks.

11. Security Profile (Baseline)

- Maintain allowlist of trusted `SRC` for `CMD`.
- Require `AUTH` token reference for `CMD` and `FORM` writes.
- Validate timestamp freshness (recommended +/- 5 minutes).
- Reject commands if replayed `MSGID` appears in cache.

12. Error Codes for NACK

Recommended short codes:

- `BAD_CRC`
- `BAD_FORMAT`
- `UNSUPPORTED_TYPE`
- `NOT_ALLOWED`
- `AUTH_FAIL`
- `STALE_TS`
- `DUPLICATE`
- `TIMEOUT`
- `INVALID_ARG`
- `BUSY`

13. Worked Examples

13.1 Command Request (decoded payload shown for readability)

Decoded payload JSON:

`{"cmd":"GET_WEATHER"}`

Conceptual frame fields:

`V1|CMD|A1B2C3D4|PHONE01|GATEWAY|20260314T181500Z|03|ACKREQ,CHAN=private|eyJjbWQiOiJHRVRfV0VBVEhFU
iJ9|9F2A`

13.2 ACK Response

Decoded payload JSON:

`{"ok":true}`

Conceptual frame fields:

`V1|ACK|A1B2C3D4|GATEWAY|PHONE01|20260314T181503Z|03|CHAN=private|eyJvayI6dHJ1ZX0|A44C`

13.3 NACK Response

Decoded payload JSON:

`{"code":"NOT_ALLOWED"}`

Conceptual frame fields:

`V1|NACK|A1B2C3D4|GATEWAY|PHONE01|20260314T181503Z|03|CHAN=private|eyJjb2RlIjoiTk9UX0FMTE9XRUQifQ|4B71`

14. Open Discussion Items

- Confirm max frame length target after real-world Meshtastic path testing.
- Confirm whether CRC is redundant with mesh layer checks or still desired for app-level integrity.
- Decide whether `SRC`/`DST` should be callsign-based, node-id-based, or dual format.
- Decide whether command auth should move from token reference to HMAC signature in V2.
- Decide default `TTL` for normal operations.

15. Suggested Next Steps

1. Freeze field names and `TYPE` set for V1.
2. Implement parser/serializer library in gateway.
3. Add Android app prototype using same frame model.
4. Run airtime and reliability tests with ACK-required command traffic.

16. Multi-Network Gateway Architecture

EMP V1 is intended to be the internal canonical envelope for a multi-network message gateway.

Target external networks (initial discussion set):

- Meshtastic
- MeshCore
- Winlink
- APRS
- SMS

Core principle:

- Normalize all inbound traffic into EMP.
- Route and process internally in EMP.
- Render outbound traffic from EMP into each target network format.

17. Adapter Contract (Per Network)

Each network adapter should implement the same interface:

- `ingest(raw_network_message) -> EMP frame`
- `render(emp_frame) -> raw_network_message`
- `capabilities() -> capability descriptor`

Minimum adapter capability descriptor fields:

- `max_payload_bytes`
- `supports_ack` (bool)
- `supports_fragmentation` (bool)
- `supports_store_forward` (bool)
- `latency_class` (`realtime`, `delayed`, `opportunistic`)
- `cost_class` (`free_local`, `metered`, `high_cost`)

18. Routing and Policy Layer

Routing decisions should be policy-driven, not hardcoded.

Recommended policy inputs:

- `TYPE`
- source/destination classes
- priority (`PRIO`)
- message size
- required reliability
- network availability and health

Example policy outcomes:

- `TXT`: route to Meshtastic and APRS.
- `CMD`: route to Meshtastic or MeshCore with mandatory ACK.
- `FORM`: route to Winlink primary, Meshtastic fallback as fragments only when allowed.
- urgent alerts: fan-out to all available networks with per-network throttles.

19. Delivery State Model

For multi-network use, maintain delivery state per network and per message.

Suggested states:

- `queued`
- `sent`
- `acked`
- `nacked`
- `timeout`
- `failed`

Suggested key:

- `(MSGID, target_network, target_endpoint)`

This allows partial success handling (for example, ACK on APRS but timeout on Winlink).

20. EMP V1 Fields Reserved for Multi-Network Bridging

Current fields remain unchanged. The following `FLAGS` keys are reserved for gateway bridging behavior:

- `NET=...` target network hint (`meshtastic`, `meshcore`, `winlink`, `aprs`, `sms`)
- `MULTI=...` comma list for fan-out targets
- `EXP=YYYYMMDDTHHMMSSZ` message expiry timestamp
- `RCLASS=...` reliability class (`best_effort`, `ack_required`, `store_forward`)

Note: these are optional hints. The gateway policy engine is authoritative.

21. Network-Specific Guidance (Initial)

21.1 APRS

- Use for short tactical text/status.
- Keep payload compact.
- Prefer human-readable summaries and terse telemetry.

21.2 Winlink

- Use for larger structured form traffic.
- Prefer `FORM` mapping with stronger delivery assurances.
- Expect higher latency and store-and-forward behavior.

21.3 MeshCore and Meshtastic

- Use for field-local command, acknowledgement, and short text workflows.
- Reuse `ACKREQ`, duplicate suppression, and retry behaviors.

21.4 SMS

- Use as fallback or edge bridge where IP/radio integration is constrained.
- Keep strict size limits and concise payloads.

22. Interop Validation Plan

Before declaring interop stable:

1. Build deterministic parser/serializer tests for EMP.
2. Build adapter conformance tests for each network adapter.
3. Run end-to-end routing tests for each `TYPE` and reliability class.
4. Validate duplicate suppression and replay protections under fan-out conditions.
5. Measure airtime and latency impact for fragmented versus compact payloads.

23. Practical Scenarios

The examples below show how EMP can combine low-latency LoRa behavior with resilient store-and-forward delivery when normal infrastructure is unavailable.

23.1 Water Leak Alert (Fast Local + Durable Relay)

Operational intent:

- Immediate local notification over LoRa mesh.
- Reliable escalation path for detailed report when a forwarding path is available.

Suggested flow:

1. Sensor event at remote site triggers EMP `TLM` alert frame:
        - `TYPE=TLM`
        - `FLAGS=PRIO=3,ACKREQ,CHAN=private,MULTI=meshtastic,winlink,RCLASS=ack_required`
        - Decoded payload example:
          - `{"event":"LEAK","site":"tank-shed-2","severity":"high","vin_v":5.02,"vbat_v":3.78,"ts":"20260314T192212Z"}`
2. Meshtastic adapter sends compact alert immediately to nearby field operators.
3. Gateway expects quick ACK from local command node (LoRa advantage: low dependency, local coverage).
4. In parallel, Winlink adapter stages a structured incident form (`FORM`) with richer metadata and queued delivery.
5. If internet/backhaul is down, Winlink message remains queued for later transfer (store-and-

forward robustness).
6. When Winlink path becomes available, detailed message is delivered without needing live site infrastructure.

Why this plays to strengths:

- LoRa/Meshtastic gives immediate, infrastructure-light local alerting.
- Winlink provides durable, delayed-but-reliable long-form incident reporting.

23.2 Wellness Check (Low Airtime + Positive Accountability)

Operational intent:

- Periodic proof-of-life from field unit with minimal airtime.
- Escalate only if acknowledgements are missing.

Suggested flow:

1. Field unit sends EMP `TXT` or `TLM` wellness ping every interval:
    - `TYPE=TLM`
    - `FLAGS=ACKREQ,PRIO=1,CHAN=private,MULTI=meshtastic,RCLASS=best_effort`
    - Decoded payload example:
      - `{"event":"WELLNESS","unit":"team-3","state":"OK","battery":"3.71"}`
2. Command node returns EMP `ACK` with same `MSGID`.
3. If ACK missing after retry budget, gateway raises a local `NACK/TIMEOUT` event.
4. Escalation policy creates a Winlink `FORM` summary after N consecutive missed checks:
    - includes last heard time, retries attempted, and last telemetry snapshot.
5. Winlink carries the escalation report when possible, even if primary infrastructure remains degraded.

Why this plays to strengths:

- LoRa mesh supports low-bandwidth, regular accountability messages.
- ACK tracking gives clear operational status without heavy payloads.
- Winlink escalation preserves continuity when immediate paths are unavailable.

24. Example Policy Snippet (Discussion)

Illustrative policy behavior for the two scenarios:

- `event=LEAK`: send now via Meshtastic + queue detailed Winlink form.
- `event=WELLNESS`: send via Meshtastic only; escalate to Winlink after missed ACK threshold.
- `priority>=3`: force `ACKREQ` and at least one durable path target.

This policy pattern keeps routine traffic light while preserving high-reliability escalation under degraded conditions.

25. Network Routing Process (Gateway)

This section defines how the gateway selects transport paths when multiple routes exist.

Scope:

- Gateway-level transport selection (Meshtastic, MeshCore, Winlink, APRS, SMS).
- Not RF hop-by-hop control inside a mesh radio network.

25.1 Inputs to Route Selection

Per message:

- `TYPE`
- `PRIO`
- `RCLASS`
- `NET` and `MULTI` hints
- payload size and fragment count
- expiry (`EXP`) and timeout budget

Per network adapter (live state):

- `available` (bool)
- rolling success and ACK rate
- rolling timeout and NACK rate
- median end-to-end latency
- queue depth and send backlog
- `cost_class` and `latency_class`

25.2 Policy Precedence

Order of authority:

1. Safety/admin policy (hard constraints, allow/deny)
2. Message class defaults (`TYPE`, `RCLASS`)
3. User hints (`NET`, `MULTI`, `PRIO`, `EXP`)
4. Live health scoring and fallback logic

User hints are advisory unless explicitly allowed as strict overrides by admin policy.

25.3 Candidate Route Construction

Build candidate network list as follows:

1. Start from adapters marked `available=true`.
2. Remove adapters denied by policy for this `TYPE`.
3. Remove adapters that cannot carry payload size/fragment profile.
4. If `NET` provided, place it first if policy permits.
5. If `MULTI` provided, include those as fan-out targets if policy permits.

25.4 Route Scoring

For each candidate, compute a score from normalized components:

- reliability score (ACK and timeout history)
- latency score
- cost score
- queue pressure score
- policy affinity bonus (for preferred networks by message class)

Example linear model:

`score = 0.40*reliability + 0.25*latency + 0.15*queue + 0.10*cost + 0.10*affinity`

Weights are policy-configurable and can vary by message class.

25.5 Selection Modes

- `best_effort`: choose top-scoring single route.
- `ack_required`: choose top route, arm fallback on timeout threshold.
- `store_forward`: choose durable route primary (for example Winlink), optional realtime mirror.
- critical fan-out (`PRIO>=3` or policy): send to multiple routes with throttles.

25.6 Fallback and Escalation

If primary route does not reach `acked` before threshold:

1. transition primary to `timeout` or `failed`
2. activate next ranked candidate
3. preserve original `MSGID` correlation with per-network delivery state

Escalation policy may add a durable route when retries exceed threshold.

25.7 User Influence Model

User can influence route selection through EMP hints:

- `NET=...` preferred target
- `MULTI=...` fan-out hint
- `RCLASS=...` reliability intent
- `PRIO=...` urgency
- `EXP=...` expiry constraint

Admin policy controls what is allowed:

- allowlist message classes by role
- enforce mandatory durable path for selected traffic
- ignore or cap user fan-out when network load is high

25.8 Observability and Audit

Gateway should log routing decisions with reasons:

- selected candidates and scores
- policy accepts/rejects
- chosen primary and fallbacks
- final per-network outcomes (`acked`, `timeout`, `failed`)

This enables post-incident analysis and policy tuning.

25.9 Minimal Deterministic Algorithm (Reference)

1. Validate message and policy constraints.
2. Build candidate routes from live adapters.
3. Apply policy precedence and user hints.
4. Score candidates and select mode by `RCLASS` and `PRIO`.
5. Dispatch primary (and fan-out targets if required).
6. Track delivery state by `(MSGID, network, endpoint)`.
7. Trigger fallback/escalation on timeout policy.

```
8. Close message when success criteria are met or retries exhausted.
```