Funlet Behavioral Logging Specification (Draft v1)

1. Purpose & Principles

Funlet's long-term value comes from its behavioral dataset: structured metadata describing how real groups coordinate through Funlet. The purpose of behavioral logging is **not** analytics for charts — it is to capture high-quality signals from organizer and invitee actions that can later be used for agent reasoning, behavior reports, and API integration.

Logging must be:

- **Consistent** (same event logged the same way every time)
- **Metadata-only** (no message body, no PII beyond phone numbers already in the system)
- **Event-based** (each meaningful action creates a log row)
- **Complete** (all organizer and invitee actions recorded)

Organizer actions happen in app/web chat. Invitee actions happen via SMS. Every action from both sides must generate a log entry.

---

2. Event Model (What We Log)

Below is the canonical list of event types for organizers, invitees, and system processes.

These mirror the structure in the internal spec.

2.1 Organizer Events

Organizer-side logging captures all structured actions taken through the app/web chat.

- session_start — User opens chat or app session.
- session_end — User leaves chat or session expires.

- flow_started — Any major flow begins (crew creation, event creation, sync-up, etc.).
- flow_step — Each step within a flow (adding contacts, naming crew, selecting options).
- flow_completed — Flow completes successfully.
- user_action — Generic action that is not part of a defined flow.
- contact_picker_opened — iOS native picker launched.
- contact_picker_submitted — Count of contacts selected.
- crew_created — Crew created.
- crew_updated — Members added/removed.
- event_created — Event created.
- syncup_created — Sync-up created.
- finalize_triggered — Organizer manually finalizes an event or sync-up.
- reminder_sent — Organizer manually triggers reminders.
- drop_off — Flow started but not completed (timeout or exit).
- push_received — Device receives a push notification.
- push_opened — Organizer taps push and returns to chat.

2.2 Invitee Events

Invitee-side logging captures natural SMS behavior.

- invite_sent — SMS invite sent to an invitee.
- invitee_reply_yes
- invitee_reply_no
- invitee_reply_unknown — Message not recognized.

- invitee_vote — Vote on sync-up option (“1”, “2”, “3”).
- invitee_timeout — No reply after X hours/days (system-defined).
- invitee_confirmed — Invitee confirms after organizer follow-up.

2.3 System Events

System-level processes that help us debug and validate flows.

- sms_sent — Outbound SMS delivered via Twilio.
- sms_received — Inbound SMS from invitee handled.
- error — Any backend or flow error.
- latency — Round-trip processing time measurement.
- state_snapshot — Debug snapshot of any event state (optional during early beta).

---

3. Event Schema (How We Log It)

All behavioral logs go into a single table (or a small set of tables, developer’s choice) with the following fields:

Field	Description
id	Unique log entry ID
event_type	One of the defined event types

timestamp	UTC timestamp
organizer_id	Organizer performing action (nullable for invitee events)
invitee_phone	Phone number (only for invitee-related events)
event_id	Event associated with action (nullable if not applicable)
crew_id	Crew associated with action (nullable)
syncup_id	Sync-up ID (nullable)
session_id	Session identifier for grouping organizer actions
platform	web, ios_app, sms, or system
version	App or backend version
metadata (JSON)	Optional structured details (counts, selected options, etc.)

Notes:

- metadata must only contain structured fields — no message bodies or raw SMS text.
- Phone numbers are already part of the Funlet system and allowed to appear.
- session_id helps us see full session-level sequences.

4. Implementation Rules (Where and When We Log)

4.1 Organizer Action Logging (app/web)

Every meaningful user action inside the chat must log one of the organizer event types. This includes:

- beginning a flow (crew, event, sync-up)
- selecting contacts
- creating the event
- adding more people
- sending reminders
- finalizing
- completing a flow
- abandoning a flow

The WebView triggers the backend; the backend writes logs.

4.2 Invitee SMS Logging

Every message Funlet sends or receives must generate corresponding invitee or system logs:

- Outbound SMS → sms_sent
- Inbound SMS → sms_received
- Valid replies → invitee_reply_yes/no/vote
- Unknown replies → invitee_reply_unknown

- Non-response timeout → invitee_timeout

4.3 Avoid Double-Logging

Only the backend writes logs to the table.

Frontend actions should call backend endpoints, and backend endpoints should log.

4.4 Logging Reliability Requirements

- Logging must not slow down user-facing flows.
- Failures in logging must not break the main workflow.
- Errors should be captured as error events.

4.5 QA Expectations

During beta, we will manually verify:

- Each organizer flow produces the correct event sequence.
- Each invitee reply logs properly.
- Each SMS sent logs properly.
- Push notifications generate push_received and push_opened.
- No duplicate logs appear per action.

---

5. Summary for Developer

You must implement behavioral logging across all organizer and invitee flows.

Focus on:

- Using the event types defined above

- Writing every log entry in the standard schema
- Ensuring all logs pass through the backend
- Capturing structured behavior, not message text
- Keeping logging lightweight and reliable
- Producing clean data for future analysis and agent integration

This logging system is the backbone of Funlet’s dataset and the most important feature for validating the new product direction.