Survale API & Backend Specification — v1.0

Scope: This document defines backend interfaces and contracts for the Survale MVP. Backend stack: self-hosted Supabase (Auth, Postgres, Realtime, Storage) plus a lightweight Node.js service for PDF exports, proximity alerts, invite expiry, auto-end, and purges.

1) Principles

• Least-privilege via Postgres Row-Level Security (RLS). • Mobile clients use Supabase JWT. • All writes run through RPCs or policy-guarded inserts. • Realtime is used for op chat and live locations. • All endpoints use TLS.

2) Authentication & Identity

Auth Provider: Supabase email+password. Users belong to exactly one primary team and agency. Users can be invited to other teams' operations. Token includes user id (auth.uid), team_id, agency_id claims.

3) Supabase Data Model (Summary)

Core tables: agencies, teams, users, operations, operation_members, operation_invites, operation_join_requests, targets (+person/vehicle/location), target_photos, staging_areas, locations_stream, locations_archive, op_messages, media_assets, proximity_alerts, audit_log, exports.

4) Realtime Channels

• op_{operation_id}_locations — broadcast latest point per user. • op_{operation_id}_chat — broadcast operation-wide chat messages. • dm_{sorted_userA_userB} — ephemeral (Node relay), not persisted.

5) Supabase RPC Functions (secure writes)

- rpc_create_operation(name, incident_number): returns operation_id; inserts operations row as Draft with CA=auth.uid.
- rpc_start_operation(operation_id): sets status=active, started_at=now(); allowed for CA.
- rpc_end_operation(operation_id): sets status=ended, ended_at=now(); allowed for CA.
- rpc_invite_user(operation_id, invitee_user_id, expires_at): inserts operation_invites; sends push via Node job.
- rpc_accept_invite(invite_id): adds row in operation_members and marks invite accepted.
- rpc_request_join(operation_id): inserts join_request (expires in 1h); auto-approve if same team.
- rpc_approve_join(request_id, approve_bool): CA approves/denies; on approve, add member.
- rpc_post_message(operation_id, body_text, media_path, media_type): inserts op_messages.
- rpc_publish_location(operation_id, lat, lon, accuracy, speed, heading): inserts into locations_stream and upserts latest table; snapshots to archive per policy.

- rpc_tag_export_segments(operation_id, segments_json): store user-selected ranges for map/coordinate export.
- rpc_request_export_pdf(operation_id, include_maps_bool): enqueue export job; Node renders and stores file.

6) Node 'Ops Service' — REST Endpoints

- POST /v1/exports/:operationId/pdf — Auth: CA. Body: {include_maps: bool, tagged_segments:[{start_ts,end_ts}]}. Returns {export_id}.
- GET /v1/exports/:exportId — Auth: op member or CA. Returns {status, signed_url?}.
- POST /v1/push/invite — internal (from DB trigger/job).
- POST /v1/alerts/proximity-scan — cron-invoked; computes cross-tenant 1km alerts and pushes.
- POST /v1/ops/auto-end-scan — cron; notifies CA when all members disconnected, ends op after 10 min if no action.
- POST /v1/ops/purge — cron; deletes ended ops >7 days (DB + Storage).

7) Storage & Buckets

Buckets: media/ (signed URLs for op members), exports/ (private, CA-only). Media max sizes follow iOS best-practice compression (photos ~2–3MP; videos default 720p short clips).

8) Security & RLS Highlights

- Enable RLS on all tables.
- Helper fn is_op_member(op_id) to gate selects/inserts for most operation-scoped tables.
- operations: insert by team members; update (status) by CA; select by same-agency; ended operations visible only to members until purge.
- op_messages, targets, staging_areas, locations_*: select/insert allowed only for op members.
- View other_ops_nearby exposes limited metadata for cross-tenant proximity (op id, centroid, CA contact).

9) Data Contracts (JSON)

Location publish

```
{ "operation_id": "uuid", "user_id": "uuid", "ts": "2025-10-14T15:03:12Z",
"lat": 37.7749, "lon": -122.4194, "accuracy_m": 4.2, "speed_mps": 6.1,
"heading_deg": 182.0 }
```

Chat message

```
{ "operation_id": "uuid", "sender_user_id": "uuid", "body_text": "Target
leaving northbound.", "media_path": null, "media_type": "text",
"created_at": "2025-10-14T15:03:22Z" }
```

Export request

```
{ "include_maps": false, "tagged_segments":
[{"start_ts":"2025-10-14T15:15:00Z","end_ts":"2025-10-14T15:45:00Z"}] }
```

10) Error Handling

Standard format: { error_code, message, details? }. Common codes: AUTH_REQUIRED, FORBIDDEN, NOT_FOUND, VALIDATION_FAILED, RATE_LIMIT, STORAGE_LIMIT, EXPORT_FAILED, OP_ENDED, INVITE_EXPIRED.

11) Scheduled Jobs

- invite-expiry: marks pending invites/join-requests expired at +1h; sends notifications.
- auto-end: detects ops with all members disconnected; notifies CA; ends after 10 min if no action.
- purge: deletes data for ops ended >7 days; clears media and exports; logs audit.
- proximity: computes nearest active operations across tenants; pushes banner with CA contact when <1km.

12) Performance & Limits

• Locations: client sends every 3–5s; archive snapshot each ~5s per user; stream retention ~2h. • Indexing: (operation_id, ts) + GiST geography on points. • Throttle: per-user rate limits on location publishes and media uploads. • Exports: queue jobs; cap chat export to 7 days; PDF timeout safeguards.

13) Security Notes (MVP)

TLS for all traffic; rotate keys quarterly; private buckets for exports; signed URLs with short TTL; audit log for op create/end, invite, join/leave, edit target, export creation, replay views. E2E for messages is a post-MVP upgrade.

14) Next Steps

Deliver SQL migration (survale_schema_v1.sql), Node service scaffold, and iOS client API layer. Then deploy Supabase + Node via Docker Compose and run pilot.