# NexusGraph Connectivity Protobuf

## Overview

The NexusGraph service provides the source of truth for all navigable topology. When other Google teams (such as the autonomous vehicle or logistics groups) need to overlay custom data onto the base map, they must conform to the EdgeWeight and NodeDescriptor schemas defined below. These schemas ensure that any data "side-loaded" into the map is semantically compatible with the **TerraFlow** rendering pipeline.

### Protobuf Structure: connectivity_metadata.proto

The core of our normalization logic resides in how we define a "Road Segment." Below is a simplified representation of the internal schema used to synchronize routing logic with visual rendering.

## Protocol Buffers

```
syntax = "proto3";

package maps.core.nexusgraph;

// Represents a single navigable segment in the global map graph.
message NavigableEdge {
  fixed64 edge_id = 1; // Unique identifier linked to TerraFlow geometry

  // The physical properties of the segment
  message PhysicalAttributes {
    float length_meters = 1;
    uint32 lane_count = 2;
    SurfaceType surface = 3;
    bool is_covered = 4; // Bridges/Tunnels
  }

  // The logic that dictates how the Routing Engine treats this edge
  message RoutingConstraints {
    bool allows_pedestrians = 1;
    bool allows_bicycles = 2;
    uint32 speed_limit_kph = 3;
    repeated TurnRestriction restrictions = 4;
  }

  PhysicalAttributes physical = 2;
  RoutingConstraints logic = 3;

  // Maps this logical edge to specific UI layers in GeoStencils
  repeated string stencil_layer_ids = 4;
}
```

## Post-Launch Implementation Details

### 1. Field Masking for Latency Control

When documenting this for the **GridStream** API, we emphasized the use of FieldMasks. For mobile clients (iOS/Android), requesting the full NavigableEdge proto is often too expensive. By using masks, the mobile SDK can request only the edge_id and speed_limit_kph for real-time display, while backend routing services can pull the full RoutingConstraints for pathfinding.

### 2. The "Z-Level" Normalization

One of the most complex aspects of the documentation involved the is_covered and stencil_layer_ids fields. In a 2D rendering world, an overpass and a surface road look like they intersect.

- Our post-processing logic assigns unique **Z-indices** to edges.
- The documentation guides developers on how to use these indices to ensure that emergency incident markers (e.g., a "Car Accident" icon) appear on the *top* deck of a bridge rather than on the road beneath it.

### 3. Handling Mobile Variability

Because the **GridStream-A** (Android) and **GridStream-i** (iOS) stacks use different underlying memory models, the Protobuf was designed to be "Stream-Friendly." The edges are sorted by a **Spatial S2 Cell ID** rather than a sequential ID. This allows mobile devices to deserialize only the map segments currently within the user's viewport, preventing memory overflows during cross-country navigation.

## Comparison: Design Doc vs. Implementation Guide

| Feature | Design Doc (Pre-Launch) | Technical Reference (Post-Launch) |
|---|---|---|
| Primary Goal | Argue for a specific architecture. | Enable third-party integration. |
| Error Handling | Theoretical failure modes. | Actionable GRPC error codes and recovery steps. |
| Payload Examples | Conceptual JSON/Protos. | Production-ready, wire-sniffed examples. |

## Troubleshooting & Integration Pitfalls

Below are the most common failure modes identified during the initial rollout.

### 1. The "Off-Road" Snap Failure

**Symptom:** The navigation blue line "jumps" to a parallel frontage road or a parking lot path instead of staying on the highway.

- **The Cause:** This usually occurs when the caller fails to provide the `RoadClass` filter in their `FieldMask`. If the client only pulls raw coordinates without the functional road class (FRC) metadata, the snap-to-road algorithm is not supported.
- **The Fix:** Ensure your `GridStreamRequest` includes `PhysicalAttributes.surface` and `RoutingConstraints.speed_limit_kph`. The client-side interpolator uses these to weight the probability of the user's GPS "ping" belonging to a specific `edge_id`.

### 2. Protobuf Version Mismatch (Shadow Deprecations)

**Symptom:** `NavigableEdge` fields return as default values (0 or null) despite data existing in the UI.

- **The Cause:** Because **TerraFlow** is a rolling pipeline, we occasionally "shadow-deprecate" fields. For example, moving from `uint32 speed_limit` to a more granular `SpeedLimit proto` message.
- **The Fix:** Check your `stubby` logs for `DEPRECATED_FIELD_ACCESS` warnings. Always use the latest generated `.proto` definitions from the `//geostore/base/public` depot rather than local copies.

## For Further Reading

To provide a deeper context for engineers who need to extend the **Core Maps Platform** functionality, please refer to the following internal resources.

- **go/terraflow-pipelines:** The canonical guide to how raw satellite and street-view imagery is vectorized into the `PhysicalAttributes` message. Use this if you need to add a new road surface type (e.g., "gravel" or "cobblestone").
- **go/nexusgraph-s2-sharding:** A deep dive into how we use **S2 Geometry** to shard the global map graph. Essential reading if you are experiencing high latency in high-density urban areas like Tokyo or Manhattan.
- **go/gridstream-api-best-practices:** A comprehensive list of `gRPC` deadlining strategies to ensure that map rendering doesn't block the main UI thread on lower-end Android hardware.