Protobuffers

What are Protobuffers?

Protocol Buffers (Protobuffers or protobufs) are Google's binary serialization format for structured data. Compared to alternatives like JSON or XML, they are significantly more compact and faster to serialize/deserialize, making them well-suited for real-time communication between the rover's subsystems. They are also strongly typed and language-agnostic, the same `.proto` definition can generate code for C++, Python, Rust, and others. For a general introduction, refer to the official documentation:

https://protobuf.dev/overview/

The ERC-Protobufs Repository

All protobuf definitions used by the rover live in a shared repository called **ERC-Protobufs**, which is used as a dependency across the rover's software stack. You can find it linked from the Starting with Protobufs page.

The repository is organized under a `components/` folder, where each subfolder groups messages by the hardware board or subsystem they belong to:

```
components/
  arm_board/
  driving_board/
  sensor_board/
  container_board/
  basestation/
  common/
```

For example, `components/basestation/detected_object.proto` contains the message definition for a single YOLO-detected object sent from Jonny Boi to the Basestation. Placing it under `basestation/` makes its intended direction and purpose immediately clear.

Naming Conventions

Every message name in the repository must be **globally unique**, regardless of which folder it lives in. This is because proto3 imports reference files by path, but message names exist in a flat global namespace, two messages with the same name will cause build failures.

The convention used across the rover is to prefix every message name with its component:

- `ArmBoardControlSignals`
- `BasestationControlMode`
- `SensorBoardIMUInfo`
- `DrivingBoardMotorMessage`

Follow this convention strictly when adding new messages.

How Protobufs are Compiled

The `comms` package does not manually clone ERC-Protobufs. Instead, CMake fetches it automatically at build time using `FetchContent`, pinned to a specific commit hash in `CMakeLists.txt`:

```
FetchContent_Declare(
  erc_protobufs
  GIT_REPOSITORY https://github.com/RoboTeamTwente/ERC-Protobufs.git
  GIT_TAG <commit_hash>
)
```

Once fetched, `protoc` generates `.pb.h` and `.pb.cc` files into the build directory, which are then compiled into the `comms` node. This means that whenever new proto files are added to ERC-Protobufs and committed, the `GIT_TAG` in `CMakeLists.txt` must be updated to the new commit hash before the changes will be picked up by the build. The full procedure is covered in the **Adding a New Message** page.

The PBEnvelope

All protobuf messages on this rover are wrapped inside a `PBEnvelope` before being transmitted over UDP. The PBEnvelope acts as a typed container that allows the receiver to identify which type of message is inside before deserializing the payload. See the dedicated PBEnvelope page for a full explanation.

Revision #10
Created 2026-04-16 09:53:10 UTC by Andrei Badea

Updated 2026-04-17 13:28:15 UTC by Andrei Badea