Protocol buffers have multiple encodings for the same data, and allow surplus information to be added to the data blob. In fact, it is very general. The generality, though good (and sometimes required) for many uses cases works against security, in that it makes readers more complex, and error-prone.

For usage in Hedera precompiled contract ABIs, protobufs are a useful mechanism to encode rich blobs of data. But we don't want the security exposure of using the unrestricted protobuf serialization semantics. So we limit allowable protobuf schemas/encoding as follows:

- (These are *wire-format* restrictions, but in the following may be expressed in terms of the protobuf language.)

1. All protobufs MUST BE encoded using the Protocol Buffers Version 3 Language Specification as further described in the Language Guide (proto3) and Encoding
2. Statement restrictions
    - All imported proto definitions must obey these restrictions.
    - `service` statement not supported.
3. Naming restrictions
    - The package name MUST match the grammar rule `"hedera." ident`
        - *This might be unnecessary* as I think the package name is strictly compile-time, doesn't go on the wire (except in grpc).
4. Type restrictions
    - Field type MUST NOT be `float` (`double` allowed).
    - Field type MUST NOT be `sint32` or `sint64` (the "more inefficient" encoding for negative numbers `int32` and `int64` types are available).
    - Field type MUST NOT be any of the "fixed" types (`fixed32`, `fixed64`, `sfixed32`, `sfixed64`)
    - **TBD:** *alternative to prior two rules:* **Only** allow the 4 fixed types and *disallow* all the variable length integer encodings. **But this doesn't work** because enum values must use varint encoding, per the spec.
    - `string` MUST NOT be longer than **TBD** bytes.
        - (Some fixed number <2e6 bytes, or some lower limit?)
        - *n.b.:* Maximum string length given in UTF-8 encoded bytes, not code points.
    - `bytes` MUST NOT be longer than **TBD** bytes.
        - (Some fixed number <2e6 bytes, or some lower limit?)
    - `double` value MUST NOT be a NaN, -0.0, or ±∞.
5. Message restrictions
    - Field numbers MUST be <2000000 (2e6) (will fit in 3byte varint).
        - **TBD:** Or perhaps some lower limit, but probably needs to exceed 16K and therefore need more than two bytes.
6. Encoding restrictions
    - Unknown fields (fields not defined in the relevant HIP) MUST NOT be present.

        - Unknown fields are not ignored or retained: They cause the protobuf to be invalid and processing to fail.
    - Duplicate entries MUST NOT be present: the "last one wins" rule is NOT allowed.
        - Fields MUST NOT be repeated (unless it is defined as a repeated field in the relevant HIP),
        - Map keys MUST NOT be repeated in a map.
    - Fields MUST be encoded in order of their field number, increasing.
    - Maps MUST be encoded in order of their keys; if numeric, increasing, otherwise (string) lexicographically increasing
    - Map keys, if string, MUST match the grammar rule `fullIdent`
        - `fullIdent = ident { "." ident }; ident = letter { letter | decimalDigit | "_" }`
    - `MapFieldEntry` messages (the kv part of a map) MUST have a `key_type` field.
        - Thus, do *not* use the default value of a type for a map key.
            - **TBD:** Disallowing an empty string is ok, but is disallowing `0` for a numeric key ok?
    - Maps MUST have a value for each key.
        - This may just be an issue for serialization, not for the wire format.
    - Repeated primitive numeric fields MUST use packed encoding.
        - *And* there MUST only be *one* key-value pair for a packed repeated field
            - The wire encoding allows more than one, which are concatenated, but we disallow it.
        - They claim: "Protocol buffer parsers must be able to parse repeated fields that were compiled as packed as if they were not packed, and vice versa. This permits adding [packed=true] to existing fields in a forward- and backward-compatible way." Regardless, we disallow it as we *always* pack repeated primitive numeric fields.
    - Enum values MUST be restricted to those defined in the relevant HIP.