```
// Copyright 2015 The GTFS Specifications Authors.
//
// Licensed under the Apache License, Version 2.0 (the "License");
// you may not use this file except in compliance with the License.
// You may obtain a copy of the License at
//
//     http://www.apache.org/licenses/LICENSE-2.0
//
// Unless required by applicable law or agreed to in writing, software
// distributed under the License is distributed on an "AS IS" BASIS,
// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
// See the License for the specific language governing permissions and
// limitations under the License.

// Protocol definition file for GTFS Realtime.
//
// GTFS Realtime lets transit agencies provide consumers with realtime
// information about disruptions to their service (stations closed, lines not
// operating, important delays etc), location of their vehicles and expected
// arrival times.
//
// This protocol is published at:
// https://github.com/google/transit/tree/master/gtfs-realtime

syntax = "proto2";
option java_package = "com.google.transit.realtime";
package GTFSv2.Realtime;

// The contents of a feed message.
// A feed is a continuous stream of feed messages. Each message in the stream is
// obtained as a response to an appropriate HTTP GET request.
// A realtime feed is always defined with relation to an existing GTFS feed.
// All the entity ids are resolved with respect to the GTFS feed.
// Note that "required" and "optional" as stated in this file refer to Protocol
// Buffer cardinality, not semantic cardinality.  See reference.md at
// https://github.com/google/transit/tree/master/gtfs-realtime for field
// semantic cardinality.
message FeedMessage {
  // Metadata about this feed and feed message.
  required FeedHeader header = 1;

  // Contents of the feed.
  repeated FeedEntity entity = 2;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// Metadata about a feed, included in feed messages.
message FeedHeader {
  // Version of the feed specification.
  // The current version is 2.0.  Valid versions are "2.0", "1.0".
  required string gtfs_realtime_version = 1;

  // Determines whether the current fetch is incremental.  Currently,
  // DIFFERENTIAL mode is unsupported and behavior is unspecified for feeds
  // that use this mode.  There are discussions on the GTFS Realtime mailing
  // list around fully specifying the behavior of DIFFERENTIAL mode and the
  // documentation will be updated when those discussions are finalized.
  enum Incrementality {
    FULL_DATASET = 0;
    DIFFERENTIAL = 1;
  }
  optional Incrementality incrementality = 2 [default = FULL_DATASET];
```

```
  // This timestamp identifies the moment when the content of this feed has been
  // created (in server time). In POSIX time (i.e., number of seconds since
  // January 1st 1970 00:00:00 UTC).
  optional uint64 timestamp = 3;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// A definition (or update) of an entity in the transit feed.
message FeedEntity {
  // The ids are used only to provide incrementality support. The id should be
  // unique within a FeedMessage. Consequent FeedMessages may contain
  // FeedEntities with the same id. In case of a DIFFERENTIAL update the new
  // FeedEntity with some id will replace the old FeedEntity with the same id
  // (or delete it - see is_deleted below).
  // The actual GTFS entities (e.g. stations, routes, trips) referenced by the
  // feed must be specified by explicit selectors (see EntitySelector below for
  // more info).
  required string id = 1;

  // Whether this entity is to be deleted. Relevant only for incremental
  // fetches.
  optional bool is_deleted = 2 [default = false];

  // Data about the entity itself. Exactly one of the following fields must be
  // present (unless the entity is being deleted).
  optional TripUpdate trip_update = 3;
  optional VehiclePosition vehicle = 4;
  optional Alert alert = 5;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

//
// Entities used in the feed.
//

// Realtime update of the progress of a vehicle along a trip.
// Depending on the value of ScheduleRelationship, a TripUpdate can specify:
// - A trip that proceeds along the schedule.
// - A trip that proceeds along a route but has no fixed schedule.
// - A trip that have been added or removed with regard to schedule.
//
// The updates can be for future, predicted arrival/departure events, or for
// past events that already occurred.
// Normally, updates should get more precise and more certain (see
// uncertainty below) as the events gets closer to current time.
// Even if that is not possible, the information for past events should be
// precise and certain. In particular, if an update points to time in the past
// but its update's uncertainty is not 0, the client should conclude that the
// update is a (wrong) prediction and that the trip has not completed yet.
//
// Note that the update can describe a trip that is already completed.
// To this end, it is enough to provide an update for the last stop of the trip.
// If the time of that is in the past, the client will conclude from that that
// the whole trip is in the past (it is possible, although inconsequential, to
// also provide updates for preceding stops).
```

```
// This option is most relevant for a trip that has completed ahead of schedule,
// but according to the schedule, the trip is still proceeding at the current
// time. Removing the updates for this trip could make the client assume
// that the trip is still proceeding.
// Note that the feed provider is allowed, but not required, to purge past
// updates - this is one case where this would be practically useful.
message TripUpdate {
  // The Trip that this message applies to. There can be at most one
  // TripUpdate entity for each actual trip instance.
  // If there is none, that means there is no prediction information available.
  // It does *not* mean that the trip is progressing according to schedule.
  required TripDescriptor trip = 1;

  // Additional information on the vehicle that is serving this trip.
  optional VehicleDescriptor vehicle = 3;

  // Timing information for a single predicted event (either arrival or
  // departure).
  // Timing consists of delay and/or estimated time, and uncertainty.
  // - delay should be used when the prediction is given relative to some
  //   existing schedule in GTFS.
  // - time should be given whether there is a predicted schedule or not. If
  //   both time and delay are specified, time will take precedence
  //   (although normally, time, if given for a scheduled trip, should be
  //   equal to scheduled time in GTFS + delay).
  //
  // Uncertainty applies equally to both time and delay.
  // The uncertainty roughly specifies the expected error in true delay (but
  // note, we don't yet define its precise statistical meaning). It's possible
  // for the uncertainty to be 0, for example for trains that are driven under
  // computer timing control.
  message StopTimeEvent {
    // Delay (in seconds) can be positive (meaning that the vehicle is late) or
    // negative (meaning that the vehicle is ahead of schedule). Delay of 0
    // means that the vehicle is exactly on time.
    optional int32 delay = 1;

    // Event as absolute time.
    // In Unix time (i.e., number of seconds since January 1st 1970 00:00:00
    // UTC).
    optional int64 time = 2;

    // If uncertainty is omitted, it is interpreted as unknown.
    // If the prediction is unknown or too uncertain, the delay (or time) field
    // should be empty. In such case, the uncertainty field is ignored.
    // To specify a completely certain prediction, set its uncertainty to 0.
    optional int32 uncertainty = 3;

    // The extensions namespace allows 3rd-party developers to extend the
    // GTFS Realtime Specification in order to add and evaluate new features
    // and modifications to the spec.
    extensions 1000 to 1999;

    // The following extension IDs are reserved for private use by any organization.
    extensions 9000 to 9999;
  }

  // Realtime update for arrival and/or departure events for a given stop on a
  // trip. Updates can be supplied for both past and future events.
  // The producer is allowed, although not required, to drop past events.
  message StopTimeUpdate {
    // The update is linked to a specific stop either through stop_sequence or
    // stop_id, so one of the fields below must necessarily be set.
    // See the documentation in TripDescriptor for more information.

    // Must be the same as in stop_times.txt in the corresponding GTFS feed.
    optional uint32 stop_sequence = 1;
    // Must be the same as in stops.txt in the corresponding GTFS feed.
    optional string stop_id = 4;
```

```
  optional StopTimeEvent arrival = 2;
  optional StopTimeEvent departure = 3;

  // The relation between this StopTime and the static schedule.
  enum ScheduleRelationship {
    // The vehicle is proceeding in accordance with its static schedule of
    // stops, although not necessarily according to the times of the schedule.
    // At least one of arrival and departure must be provided. If the schedule
    // for this stop contains both arrival and departure times then so must
    // this update. Frequency-based trips (GTFS frequencies.txt with exact_times = 0)
    // should not have a SCHEDULED value and should use UNSCHEDULED instead.
    SCHEDULED = 0;

    // The stop is skipped, i.e., the vehicle will not stop at this stop.
    // Arrival and departure are optional.
    SKIPPED = 1;

    // No data is given for this stop. The main intention for this value is to
    // give the predictions only for part of a trip, i.e., if the last update
    // for a trip has a NO_DATA specifier, then StopTimes for the rest of the
    // stops in the trip are considered to be unspecified as well.
    // Neither arrival nor departure should be supplied.
    NO_DATA = 2;

    // The vehicle is operating a trip defined in GTFS frequencies.txt with exact_times = 0.
    // This value should not be used for trips that are not defined in GTFS frequencies.txt,
    // or trips in GTFS frequencies.txt with exact_times = 1. Trips containing StopTimeUpdates
    // with ScheduleRelationship=UNSCHEDULED must also set TripDescriptor.ScheduleRelationship=UNSCHEDULED.
    // NOTE: This field is still experimental, and subject to change. It may be
    // formally adopted in the future.
    UNSCHEDULED = 3;
  }
  optional ScheduleRelationship schedule_relationship = 5
      [default = SCHEDULED];

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features
  // and modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// Updates to StopTimes for the trip (both future, i.e., predictions, and in
// some cases, past ones, i.e., those that already happened).
// The updates must be sorted by stop_sequence, and apply for all the
// following stops of the trip up to the next specified one.
//
// Example 1:
// For a trip with 20 stops, a StopTimeUpdate with arrival delay and departure
// delay of 0 for stop_sequence of the current stop means that the trip is
// exactly on time.
//
// Example 2:
// For the same trip instance, 3 StopTimeUpdates are provided:
// - delay of 5 min for stop_sequence 3
// - delay of 1 min for stop_sequence 8
// - delay of unspecified duration for stop_sequence 10
// This will be interpreted as:
// - stop_sequences 3,4,5,6,7 have delay of 5 min.
// - stop_sequences 8,9 have delay of 1 min.
// - stop_sequences 10,... have unknown delay.
repeated StopTimeUpdate stop_time_update = 2;

// Moment at which the vehicle's real-time progress was measured. In POSIX
// time (i.e., the number of seconds since January 1st 1970 00:00:00 UTC).
optional uint64 timestamp = 4;

// The current schedule deviation for the trip.  Delay should only be
```

```
  // specified when the prediction is given relative to some existing schedule
  // in GTFS.
  //
  // Delay (in seconds) can be positive (meaning that the vehicle is late) or
  // negative (meaning that the vehicle is ahead of schedule). Delay of 0
  // means that the vehicle is exactly on time.
  //
  // Delay information in StopTimeUpdates take precedent of trip-level delay
  // information, such that trip-level delay is only propagated until the next
  // stop along the trip with a StopTimeUpdate delay value specified.
  //
  // Feed providers are strongly encouraged to provide a TripUpdate.timestamp
  // value indicating when the delay value was last updated, in order to
  // evaluate the freshness of the data.
  //
  // NOTE: This field is still experimental, and subject to change. It may be
  // formally adopted in the future.
  optional int32 delay = 5;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// Realtime positioning information for a given vehicle.
message VehiclePosition {
  // The Trip that this vehicle is serving.
  // Can be empty or partial if the vehicle can not be identified with a given
  // trip instance.
  optional TripDescriptor trip = 1;

  // Additional information on the vehicle that is serving this trip.
  optional VehicleDescriptor vehicle = 8;

  // Current position of this vehicle.
  optional Position position = 2;

  // The stop sequence index of the current stop. The meaning of
  // current_stop_sequence (i.e., the stop that it refers to) is determined by
  // current_status.
  // If current_status is missing IN_TRANSIT_TO is assumed.
  optional uint32 current_stop_sequence = 3;
  // Identifies the current stop. The value must be the same as in stops.txt in
  // the corresponding GTFS feed.
  optional string stop_id = 7;

  enum VehicleStopStatus {
    // The vehicle is just about to arrive at the stop (on a stop
    // display, the vehicle symbol typically flashes).
    INCOMING_AT = 0;

    // The vehicle is standing at the stop.
    STOPPED_AT = 1;

    // The vehicle has departed and is in transit to the next stop.
    IN_TRANSIT_TO = 2;
  }
  // The exact status of the vehicle with respect to the current stop.
  // Ignored if current_stop_sequence is missing.
  optional VehicleStopStatus current_status = 4 [default = IN_TRANSIT_TO];

  // Moment at which the vehicle's position was measured. In POSIX time
  // (i.e., number of seconds since January 1st 1970 00:00:00 UTC).
  optional uint64 timestamp = 5;

  // Congestion level that is affecting this vehicle.
```

```
  enum CongestionLevel {
    UNKNOWN_CONGESTION_LEVEL = 0;
    RUNNING_SMOOTHLY = 1;
    STOP_AND_GO = 2;
    CONGESTION = 3;
    SEVERE_CONGESTION = 4;  // People leaving their cars.
  }
  optional CongestionLevel congestion_level = 6;

  // The degree of passenger occupancy of the vehicle. This field is still
  // experimental, and subject to change. It may be formally adopted in the
  // future.
  enum OccupancyStatus {
    // The vehicle is considered empty by most measures, and has few or no
    // passengers onboard, but is still accepting passengers.
    EMPTY = 0;

    // The vehicle has a relatively large percentage of seats available.
    // What percentage of free seats out of the total seats available is to be
    // considered large enough to fall into this category is determined at the
    // discretion of the producer.
    MANY_SEATS_AVAILABLE = 1;

    // The vehicle has a relatively small percentage of seats available.
    // What percentage of free seats out of the total seats available is to be
    // considered small enough to fall into this category is determined at the
    // discretion of the feed producer.
    FEW_SEATS_AVAILABLE = 2;

    // The vehicle can currently accommodate only standing passengers.
    STANDING_ROOM_ONLY = 3;

    // The vehicle can currently accommodate only standing passengers
    // and has limited space for them.
    CRUSHED_STANDING_ROOM_ONLY = 4;

    // The vehicle is considered full by most measures, but may still be
    // allowing passengers to board.
    FULL = 5;

    // The vehicle is not accepting additional passengers.
    NOT_ACCEPTING_PASSENGERS = 6;

  }
  optional OccupancyStatus occupancy_status = 9;

  // A percentage value representing the degree of passenger occupancy of the vehicle.
  // The values are represented as an integer without decimals. 0 means 0% and 100 means 100%.
  // The value 100 should represent the total maximum occupancy the vehicle was designed for,
  // including both seated and standing capacity, and current operating regulations allow.
  // It is possible that the value goes over 100 if there are currently more passengers than what the vehicle was designed for.
  // The precision of occupancy_percentage should be low enough that you can't track a single person boarding and alighting for privacy reasons.
  // This field is still experimental, and subject to change. It may be formally adopted in the future.
  optional uint32 occupancy_percentage = 10;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// An alert, indicating some sort of incident in the public transit network.
message Alert {
  // Time when the alert should be shown to the user. If missing, the
  // alert will be shown as long as it appears in the feed.
  // If multiple ranges are given, the alert will be shown during all of them.
  repeated TimeRange active_period = 1;
```

```
// Entities whose users we should notify of this alert.
repeated EntitySelector informed_entity = 5;

// Cause of this alert.
enum Cause {
  UNKNOWN_CAUSE = 1;
  OTHER_CAUSE = 2;        // Not machine-representable.
  TECHNICAL_PROBLEM = 3;
  STRIKE = 4;             // Public transit agency employees stopped working.
  DEMONSTRATION = 5;      // People are blocking the streets.
  ACCIDENT = 6;
  HOLIDAY = 7;
  WEATHER = 8;
  MAINTENANCE = 9;
  CONSTRUCTION = 10;
  POLICE_ACTIVITY = 11;
  MEDICAL_EMERGENCY = 12;
}
optional Cause cause = 6 [default = UNKNOWN_CAUSE];

// What is the effect of this problem on the affected entity.
enum Effect {
  NO_SERVICE = 1;
  REDUCED_SERVICE = 2;

  // We don't care about INsignificant delays: they are hard to detect, have
  // little impact on the user, and would clutter the results as they are too
  // frequent.
  SIGNIFICANT_DELAYS = 3;

  DETOUR = 4;
  ADDITIONAL_SERVICE = 5;
  MODIFIED_SERVICE = 6;
  OTHER_EFFECT = 7;
  UNKNOWN_EFFECT = 8;
  STOP_MOVED = 9;
  NO_EFFECT = 10;
  ACCESSIBILITY_ISSUE = 11;
}
optional Effect effect = 7 [default = UNKNOWN_EFFECT];

// The URL which provides additional information about the alert.
optional TranslatedString url = 8;

// Alert header. Contains a short summary of the alert text as plain-text.
optional TranslatedString header_text = 10;

// Full description for the alert as plain-text. The information in the
// description should add to the information of the header.
optional TranslatedString description_text = 11;

// Text for alert header to be used in text-to-speech implementations. This field is the text-to-speech version of header_text.
// This field is still experimental, and subject to change. It may be formally adopted in the future.
optional TranslatedString tts_header_text = 12;

// Text for full description for the alert to be used in text-to-speech implementations. This field is the text-to-speech version of description_text.
// This field is still experimental, and subject to change. It may be formally adopted in the future.
optional TranslatedString tts_description_text = 13;

// Severity of this alert.
enum SeverityLevel {
      UNKNOWN_SEVERITY = 1;
      INFO = 2;
      WARNING = 3;
      SEVERE = 4;
}

optional SeverityLevel severity_level = 14 [default = UNKNOWN_SEVERITY];
```

```
  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features
  // and modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

//
// Low level data structures used above.
//

// A time interval. The interval is considered active at time 't' if 't' is
// greater than or equal to the start time and less than the end time.
message TimeRange {
  // Start time, in POSIX time (i.e., number of seconds since January 1st 1970
  // 00:00:00 UTC).
  // If missing, the interval starts at minus infinity.
  optional uint64 start = 1;

  // End time, in POSIX time (i.e., number of seconds since January 1st 1970
  // 00:00:00 UTC).
  // If missing, the interval ends at plus infinity.
  optional uint64 end = 2;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// A position.
message Position {
  // Degrees North, in the WGS-84 coordinate system.
  required float latitude = 1;

  // Degrees East, in the WGS-84 coordinate system.
  required float longitude = 2;

  // Bearing, in degrees, clockwise from North, i.e., 0 is North and 90 is East.
  // This can be the compass bearing, or the direction towards the next stop
  // or intermediate location.
  // This should not be direction deduced from the sequence of previous
  // positions, which can be computed from previous data.
  optional float bearing = 3;

  // Odometer value, in meters.
  optional double odometer = 4;
  // Momentary speed measured by the vehicle, in meters per second.
  optional float speed = 5;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// A descriptor that identifies an instance of a GTFS trip, or all instances of
// a trip along a route.
// - To specify a single trip instance, the trip_id (and if necessary,
//   start_time) is set. If route_id is also set, then it should be same as one
//   that the given trip corresponds to.
// - To specify all the trips along a given route, only the route_id should be
```

```
//   set. Note that if the trip_id is not known, then stop sequence ids in
//   TripUpdate are not sufficient, and stop_ids must be provided as well. In
//   addition, absolute arrival/departure times must be provided.
message TripDescriptor {
  // The trip_id from the GTFS feed that this selector refers to.
  // For non frequency-based trips, this field is enough to uniquely identify
  // the trip. For frequency-based trip, start_time and start_date might also be
  // necessary.
  optional string trip_id = 1;

  // The route_id from the GTFS that this selector refers to.
  optional string route_id = 5;

  // The direction_id from the GTFS feed trips.txt file, indicating the
  // direction of travel for trips this selector refers to.
  optional uint32 direction_id = 6;

  // The initially scheduled start time of this trip instance.
  // When the trip_id corresponds to a non-frequency-based trip, this field
  // should either be omitted or be equal to the value in the GTFS feed. When
  // the trip_id correponds to a frequency-based trip, the start_time must be
  // specified for trip updates and vehicle positions. If the trip corresponds
  // to exact_times=1 GTFS record, then start_time must be some multiple
  // (including zero) of headway_secs later than frequencies.txt start_time for
  // the corresponding time period. If the trip corresponds to exact_times=0,
  // then its start_time may be arbitrary, and is initially expected to be the
  // first departure of the trip. Once established, the start_time of this
  // frequency-based trip should be considered immutable, even if the first
  // departure time changes -- that time change may instead be reflected in a
  // StopTimeUpdate.
  // Format and semantics of the field is same as that of
  // GTFS/frequencies.txt/start_time, e.g., 11:15:35 or 25:15:35.
  optional string start_time = 2;
  // The scheduled start date of this trip instance.
  // Must be provided to disambiguate trips that are so late as to collide with
// a scheduled trip on a next day. For example, for a train that departs 8:00
// and 20:00 every day, and is 12 hours late, there would be two distinct
// trips on the same time.
// This field can be provided but is not mandatory for schedules in which such
// collisions are impossible - for example, a service running on hourly
// schedule where a vehicle that is one hour late is not considered to be
// related to schedule anymore.
// In YYYYMMDD format.
optional string start_date = 3;

// The relation between this trip and the static schedule. If a trip is done
// in accordance with temporary schedule, not reflected in GTFS, then it
// shouldn't be marked as SCHEDULED, but likely as ADDED.
enum ScheduleRelationship {
  // Trip that is running in accordance with its GTFS schedule, or is close
  // enough to the scheduled trip to be associated with it.
  SCHEDULED = 0;

  // An extra trip that was added in addition to a running schedule, for
  // example, to replace a broken vehicle or to respond to sudden passenger
  // load.
  ADDED = 1;

  // A trip that is running with no schedule associated to it (GTFS frequencies.txt exact_times=0).
  // Trips with ScheduleRelationship=UNSCHEDULED must also set all StopTimeUpdates.ScheduleRelationship=UNSCHEDULED.
  UNSCHEDULED = 2;

  // A trip that existed in the schedule but was removed.
  CANCELED = 3;

  // Should not be used - for backwards-compatibility only.
  REPLACEMENT = 5 [deprecated=true];
}
optional ScheduleRelationship schedule_relationship = 4;
```

```
  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// Identification information for the vehicle performing the trip.
message VehicleDescriptor {
  // Internal system identification of the vehicle. Should be unique per
  // vehicle, and can be used for tracking the vehicle as it proceeds through
  // the system.
  optional string id = 1;

  // User visible label, i.e., something that must be shown to the passenger to
  // help identify the correct vehicle.
  optional string label = 2;

  // The license plate of the vehicle.
  optional string license_plate = 3;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// A selector for an entity in a GTFS feed.
message EntitySelector {
  // The values of the fields should correspond to the appropriate fields in the
  // GTFS feed.
  // At least one specifier must be given. If several are given, then the
  // matching has to apply to all the given specifiers.
  optional string agency_id = 1;
  optional string route_id = 2;
  // corresponds to route_type in GTFS.
  optional int32 route_type = 3;
  optional TripDescriptor trip = 4;
  optional string stop_id = 5;
  // Corresponds to trip direction_id in GTFS trips.txt. If provided the
  // route_id must also be provided.
  optional uint32 direction_id = 6;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}

// An internationalized message containing per-language versions of a snippet of
// text or a URL.
// One of the strings from a message will be picked up. The resolution proceeds
// as follows:
// 1. If the UI language matches the language code of a translation,
//    the first matching translation is picked.
// 2. If a default UI language (e.g., English) matches the language code of a
//    translation, the first matching translation is picked.
// 3. If some translation has an unspecified language code, that translation is
//    picked.
message TranslatedString {
  message Translation {
    // A UTF-8 string containing the message.
```

```
    required string text = 1;
    // BCP-47 language code. Can be omitted if the language is unknown or if
    // no i18n is done at all for the feed. At most one translation is
    // allowed to have an unspecified language tag.
    optional string language = 2;

    // The extensions namespace allows 3rd-party developers to extend the
    // GTFS Realtime Specification in order to add and evaluate new features and
    // modifications to the spec.
    extensions 1000 to 1999;

    // The following extension IDs are reserved for private use by any organization.
    extensions 9000 to 9999;
  }
  // At least one translation must be provided.
  repeated Translation translation = 1;

  // The extensions namespace allows 3rd-party developers to extend the
  // GTFS Realtime Specification in order to add and evaluate new features and
  // modifications to the spec.
  extensions 1000 to 1999;

  // The following extension IDs are reserved for private use by any organization.
  extensions 9000 to 9999;
}
```