Protocol Buffers demo project

2015

Generated by Doxygen 1.8.11

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Alert Struct Reference

An alert, indicating some sort of incident in the public transit network.

Public Types

- enum Cause {
  UNKNOWN_CAUSE = 1, OTHER_CAUSE = 2, **TECHNICAL_PROBLEM** = 3, STRIKE = 4,
  DEMONSTRATION = 5, **ACCIDENT** = 6, **HOLIDAY** = 7, **WEATHER** = 8,
  MAINTENANCE = 9, **CONSTRUCTION** = 10, **POLICE_ACTIVITY** = 11, **MEDICAL_EMERGENCY** = 12 }

  Cause of this alert.
- enum Effect {
  NO_SERVICE = 1, **REDUCED_SERVICE** = 2, SIGNIFICANT_DELAYS = 3, **DETOUR** = 4,
  ADDITIONAL_SERVICE = 5, **MODIFIED_SERVICE** = 6, **OTHER_EFFECT** = 7, **UNKNOWN_EFFECT** = 8,
  STOP_MOVED = 9 }

  What is the effect of this problem on the affected entity.

Public Attributes

- repeated TimeRange active_period = 1

  Time when the alert should be shown to the user.
- repeated EntitySelector informed_entity = 5

  Entities whose users we should notify of this alert.
- optional Cause **cause** = 6 [default = UNKNOWN_CAUSE]
- optional Effect **effect** = 7 [default = UNKNOWN_EFFECT]
- optional TranslatedString url = 8

  The URL which provides additional information about the alert.
- optional TranslatedString header_text = 10

  Alert header. Contains a short summary of the alert text as plain-text.
- optional TranslatedString description_text = 11

  Full description for the alert as plain-text.
- extensions to

  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.

2.1.1 Detailed Description

An alert, indicating some sort of incident in the public transit network.

2.1.2 Member Enumeration Documentation

2.1.2.1 enum Alert::Cause

Cause of this alert.

Enumerator

OTHER_CAUSE Not machine-representable.

STRIKE Public transit agency employees stopped working.

DEMONSTRATION People are blocking the streets.

2.1.2.2 enum Alert::Effect

What is the effect of this problem on the affected entity.

Enumerator

SIGNIFICANT_DELAYS 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.

2.1.3 Member Data Documentation

2.1.3.1 repeated TimeRange Alert::active_period = 1

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.

2.1.3.2 optional TranslatedString Alert::description_text = 11

Full description for the alert as plain-text.

The information in the description should add to the information of the header.

2.1.3.3 extensions Alert::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.2 EntitySelector Struct Reference

A selector for an entity in a GTFS feed.

Public Attributes

- optional string agency_id = 1

  The values of the fields should correspond to the appropriate fields in the GTFS feed.
- optional string **route_id** = 2
- optional int32 route_type = 3

  corresponds to route_type in GTFS.
- optional TripDescriptor **trip** = 4
- optional string **stop_id** = 5
- extensions to

  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.

2.2.1 Detailed Description

A selector for an entity in a GTFS feed.

2.2.2 Member Data Documentation

2.2.2.1 optional string EntitySelector::agency_id = 1

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.

2.2.2.2 extensions EntitySelector::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.3 FeedEntity Struct Reference

A definition (or update) of an entity in the transit feed.

Public Attributes

- required string id = 1

  The ids are used only to provide incrementality support.
- optional bool is_deleted = 2 [default = false]

  Whether this entity is to be deleted.
- optional TripUpdate trip_update = 3

  Data about the entity itself.
- optional VehiclePosition **vehicle** = 4
- optional Alert **alert** = 5
- extensions to

  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.

2.3.1 Detailed Description

A definition (or update) of an entity in the transit feed.

2.3.2 Member Data Documentation

2.3.2.1 required string FeedEntity::id = 1

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).

2.3.2.2 optional bool FeedEntity::is_deleted = 2 [default = false]

Whether this entity is to be deleted.

Relevant only for incremental fetches.

2.3.2.3 extensions FeedEntity::to

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.

2.3.2.4 optional TripUpdate FeedEntity::trip_update = 3

Data about the entity itself.

Exactly one of the following fields must be present (unless the entity is being deleted).

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.4 FeedHeader Struct Reference

Metadata about a feed, included in feed messages.

Public Types

- enum Incrementality { **FULL_DATASET** = 0, **DIFFERENTIAL** = 1 }

  Determines whether the current fetch is incremental.

Public Attributes

- required string gtfs_realtime_version = 1

  Version of the feed specification.
- optional Incrementality **incrementality** = 2 [default = FULL_DATASET]
- optional uint64 timestamp = 3

  This timestamp identifies the moment when the content of this feed has been created (in server time).
- extensions to

  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.

2.4.1 Detailed Description

Metadata about a feed, included in feed messages.

2.4.2 Member Enumeration Documentation

2.4.2.1 enum FeedHeader::Incrementality

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.

2.4.3 Member Data Documentation

2.4.3.1 required string FeedHeader::gtfs_realtime_version = 1

Version of the feed specification.

The current version is 1.0.

2.4.3.2 optional uint64 FeedHeader::timestamp = 3

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).

2.4.3.3 extensions FeedHeader::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.5 FeedMessage Struct Reference

The contents of a feed message.

Public Attributes

- required FeedHeader header = 1

  Metadata about this feed and feed message.
- repeated FeedEntity entity = 2

  Contents of the feed.
- extensions to

  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.

2.5.1 Detailed Description

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.

A feed depends on some external configuration:

- The corresponding GTFS feed.
- Feed application (updates, positions or alerts). A feed should contain only items of one specified application; all the other entities will be ignored.
- Polling frequency

2.5.2 Member Data Documentation

2.5.2.1 extensions FeedMessage::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.6 NyctFeedHeader Struct Reference

NYCT Subway extensions for the feed header.

Public Attributes

- required string nyct_subway_version = 1

  Version of the NYCT Subway extensions The current version is 1.0.
- repeated TripReplacementPeriod trip_replacement_period = 2

  For the NYCT Subway, the GTFS-realtime feed replaces any scheduled trip within the trip_replacement_period.

2.6.1 Detailed Description

NYCT Subway extensions for the feed header.

2.6.2 Member Data Documentation

2.6.2.1 repeated TripReplacementPeriod NyctFeedHeader::trip_replacement_period = 2

For the NYCT Subway, the GTFS-realtime feed replaces any scheduled trip within the trip_replacement_period.

This feed is a full dataset, it contains all trips starting in the trip_replacement_period. If a trip from the static GT↩FS is not found in the GTFS-realtime feed, it should be considered as cancelled. The replacement period can be different for each route, so here is a list of the routes where the trips in the feed replace all scheduled trips within the replacement period.

The documentation for this struct was generated from the following file:

- nyct-subway.proto

2.7 NyctStopTimeUpdate Struct Reference

NYCT Subway extensions for the stop time update.

Public Attributes

- optional string scheduled_track = 1

  Provides the planned station arrival track.
- optional string actual_track = 2

  This is the actual track that the train is operating on and can be used to determine if a train is operating according to its current schedule (plan).

2.7.1 Detailed Description

NYCT Subway extensions for the stop time update.

2.7.2 Member Data Documentation

2.7.2.1 optional string NyctStopTimeUpdate::actual_track = 2

This is the actual track that the train is operating on and can be used to determine if a train is operating according to its current schedule (plan).

The actual track is known only shortly before the train reaches a station, typically not before it leaves the previous station. Therefore, the NYCT feed sets this field only for the first station of the remaining trip.

Different actual and scheduled track is the result of manually rerouting a train off it scheduled path. When this occurs, prediction data may become unreliable since the train is no longer operating in accordance to its schedule. The rules engine for the 'countdown' clocks will remove this train from all schedule stations.

2.7.2.2 optional string NyctStopTimeUpdate::scheduled_track = 1

Provides the planned station arrival track.

The following is the Manhattan track configurations: 1: southbound local 2: southbound express 3: northbound express 4: northbound local

In the Bronx (except Dyre Ave line) M: bi-directional express (in the AM express to Manhattan, in the PM express away).

The Dyre Ave line is configured: 1: southbound 2: northbound 3: bi-directional

The documentation for this struct was generated from the following file:

- nyct-subway.proto

2.8 NyctTripDescriptor Struct Reference

NYCT Subway extensions for the trip descriptor.

Public Types

- enum Direction { **NORTH** = 1, **EAST** = 2, **SOUTH** = 3, **WEST** = 4 }

  The direction the train is moving.

Public Attributes

- optional string train_id = 1

  The nyct_train_id is meant for internal use only.
- optional bool is_assigned = 2

  This trip has been assigned to a physical train.
- optional Direction direction = 3

  Uptown and Bronx-bound trains are moving NORTH.

2.8.1 Detailed Description

NYCT Subway extensions for the trip descriptor.

2.8.2 Member Data Documentation

2.8.2.1 optional Direction NyctTripDescriptor::direction = 3

Uptown and Bronx-bound trains are moving NORTH.

Times Square Shuttle to Grand Central is also northbound.

Downtown and Brooklyn-bound trains are moving SOUTH. Times Square Shuttle to Times Square is also southbound.

EAST and WEST are not used currently.

2.8.2.2 optional bool NyctTripDescriptor::is_assigned = 2

This trip has been assigned to a physical train.

If true, this trip is already underway or most likely will depart shortly.

Train Assignment is a function of the Automatic Train Supervision (ATS) office system used by NYCT Rail Operations to monitor and track train movements. ATS provides the ability to "assign" the nyct_train_id attribute when a physical train is at its origin terminal. These assigned trips have the is_assigned field set in the TripDescriptor.

When a train is at a terminal but has not been given a work program it is declared unassigned and is tagged as such. Unassigned trains can be moved to a storage location or assigned a nyct_train_id when a determination for service is made.

2.8.2.3 optional string NyctTripDescriptor::train_id = 1

The nyct_train_id is meant for internal use only.

It provides an easy way to associated GTFS-realtime trip identifiers with NYCT rail operations identifier

The ATS office system assigns unique train identification (Train ID) to each train operating within or ready to enter the mainline of the monitored territory. An example of this is 06 0123+ PEL/BBR and is decoded as follows:

The first character represents the trip type designator. 0 identifies a scheduled revenue trip. Other revenue trip values that are a result of a change to the base schedule include; [= reroute], [/ skip stop], [$ turn train] also known as shortly lined service.

The second character 6 represents the trip line i.e. number 6 train The third set of characters identify the decoded origin time. The last character may be blank "on the whole minute" or + "30 seconds"

Note: Origin times will not change when there is a trip type change. This is followed by a three character "Origin Location" / "Destination Location"

The documentation for this struct was generated from the following file:

- nyct-subway.proto

2.9 OVapiStopTimeUpdate Struct Reference

Public Attributes

- optional string stop_headsign = 1

  The trip_headsign field contains the text that appears on a sign that identifies the trip's destination to passengers Overrides possible trip_headsign's and stop_headsign's in static GTFS feeds.
- optional string scheduled_track = 2

  The planned platform-code where the trip is scheduled to depart from.
- optional string actual_track = 3

  (optional) The actual platform-code where the deviates from the scheduled platform.
- optional string station_id = 4

  (optional) The (internal)identifier of a trainstation where the stoptimeupdate applies to

2.9.1 Member Data Documentation

2.9.1.1 optional string OVapiStopTimeUpdate::stop_headsign = 1

The trip_headsign field contains the text that appears on a sign that identifies the trip's destination to passengers Overrides possible trip_headsign's and stop_headsign's in static GTFS feeds.

The documentation for this struct was generated from the following file:

- gtfs-realtime-OVapi.proto

2.10 OVapiTripDescriptor Struct Reference

Public Attributes

- optional string realtime_trip_id = 1

  Matches with realtime_trip_id field in trips.txt.
- optional string trip_short_name = 2

  The trip_short_name of the (added) trip.
- optional string commercial_mode_id = 3

  The commercial_mode id of the.

The documentation for this struct was generated from the following file:

- gtfs-realtime-OVapi.proto

2.11 OVapiTripUpdate Struct Reference

Public Attributes

- optional string trip_headsign = 1

  The trip_headsign field contains the text that appears on a sign that identifies the trip's destination to passengers Overrides possible trip_headsign's in static GTFS feeds, overrides stop_headsigns in the static feed unless there stop_headsign in a StopTimeUpdate.

The documentation for this struct was generated from the following file:

- gtfs-realtime-OVapi.proto

2.12 OVapiVehicleDescriptor Struct Reference

Public Attributes

- optional bool **wheelchair_accessible** = 1
- optional string **vehicle_type** = 2
- optional string **vehicle_headsign** = 3

The documentation for this struct was generated from the following file:

- gtfs-realtime-OVapi.proto

2.13 OVapiVehiclePosition Struct Reference

Public Attributes

- optional int32 delay = 1

  Estimated arrival-delay at the next stop (in seconds) can be positive (meaning that the vehicle is late) or negative (meaning that the vehicle is ahead of schedule).

2.13.1 Member Data Documentation

2.13.1.1 optional int32 OVapiVehiclePosition::delay = 1

Estimated arrival-delay at the next stop (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.

The documentation for this struct was generated from the following file:

- gtfs-realtime-OVapi.proto

2.14 Position Struct Reference

A position.

Public Attributes

- required float latitude = 1

  Degrees North, in the WGS-84 coordinate system.
- required float longitude = 2

  Degrees East, in the WGS-84 coordinate system.
- optional float bearing = 3

  Bearing, in degrees, clockwise from North, i.e., 0 is North and 90 is East.
- optional double odometer = 4

  Odometer value, in meters.
- optional float speed = 5

  Momentary speed measured by the vehicle, in meters per second.
- extensions to

  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.

2.14.1 Detailed Description

A position.

2.14.2 Member Data Documentation

2.14.2.1 optional float Position::bearing = 3

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.

2.14.2.2 extensions Position::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.15 TripUpdate::StopTimeEvent Struct Reference

Timing information for a single predicted event (either arrival or departure).

Public Attributes

- optional int32 delay = 1

  Delay (in seconds) can be positive (meaning that the vehicle is late) or negative (meaning that the vehicle is ahead of schedule).
- optional int64 time = 2

  Event as absolute time.
- optional int32 uncertainty = 3

  If uncertainty is omitted, it is interpreted as unknown.
- extensions to

  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.

2.15.1 Detailed Description

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.

2.15.2 Member Data Documentation

2.15.2.1 optional int32 TripUpdate::StopTimeEvent::delay = 1

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.

2.15.2.2 optional int64 TripUpdate::StopTimeEvent::time = 2

Event as absolute time.

In Unix time (i.e., number of seconds since January 1st 1970 00:00:00 UTC).

2.15.2.3 extensions TripUpdate::StopTimeEvent::to

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.

2.15.2.4 optional int32 TripUpdate::StopTimeEvent::uncertainty = 3

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.16 TripUpdate::StopTimeUpdate Struct Reference

Realtime update for arrival and/or departure events for a given stop on a trip.

Public Types

- enum ScheduleRelationship { SCHEDULED = 0, SKIPPED = 1, NO_DATA = 2 }

  The relation between this StopTime and the static schedule.

Public Attributes

- optional uint32 stop_sequence = 1

  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.
- optional string stop_id = 4

  Must be the same as in stops.txt in the corresponding GTFS feed.
- optional StopTimeEvent **arrival** = 2
- optional StopTimeEvent **departure** = 3
- optional ScheduleRelationship **schedule_relationship**
- extensions to

  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.

2.16.1 Detailed Description

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.

2.16.2 Member Enumeration Documentation

2.16.2.1 enum TripUpdate::StopTimeUpdate::ScheduleRelationship

The relation between this StopTime and the static schedule.

Enumerator

SCHEDULED 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.

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

NO_DATA 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.

2.16.3 Member Data Documentation

2.16.3.1 optional ScheduleRelationship TripUpdate::StopTimeUpdate::schedule_relationship

Initial value:

```
= 5
        [default = SCHEDULED]
```

2.16.3.2 optional uint32 TripUpdate::StopTimeUpdate::stop_sequence = 1

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.

2.16.3.3 extensions TripUpdate::StopTimeUpdate::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.17 TimeRange Struct Reference

Low level data structures used above.

Public Attributes

- optional uint64 start = 1

  Start time, in POSIX time (i.e., number of seconds since January 1st 1970 00:00:00 UTC).
- optional uint64 end = 2

  End time, in POSIX time (i.e., number of seconds since January 1st 1970 00:00:00 UTC).
- extensions to

  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.

2.17.1 Detailed Description

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.

2.17.2 Member Data Documentation

2.17.2.1 optional uint64 TimeRange::end = 2

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.

2.17.2.2 optional uint64 TimeRange::start = 1

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.

2.17.2.3 extensions TimeRange::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.18 TranslatedString Struct Reference

An internationalized message containing per-language versions of a snippet of text or a URL.

Classes

- struct Translation

Public Attributes

- repeated Translation translation = 1

  At least one translation must be provided.
- extensions to

  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.

2.18.1 Detailed Description

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.

2.18.2 Member Data Documentation

2.18.2.1 extensions TranslatedString::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.19 TranslatedString::Translation Struct Reference

Public Attributes

- required string text = 1

  A UTF-8 string containing the message.
- optional string language = 2

  BCP-47 language code.
- extensions to

  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.

2.19.1 Member Data Documentation

2.19.1.1 optional string TranslatedString::Translation::language = 2

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.

2.19.1.2 extensions TranslatedString::Translation::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.20 TripDescriptor Struct Reference

A descriptor that identifies an instance of a GTFS trip, or all instances of a trip along a route.

Public Types

- enum ScheduleRelationship { SCHEDULED = 0, ADDED = 1, UNSCHEDULED = 2, CANCELED = 3 }

  The relation between this trip and the static schedule.

Public Attributes

- optional string trip_id = 1

  The trip_id from the GTFS feed that this selector refers to.
- optional string route_id = 5

  The route_id from the GTFS that this selector refers to.
- optional uint32 direction_id = 6

  The direction_id from the GTFS feed trips.txt file, indicating the direction of travel for trips this selector refers to.
- optional string start_time = 2

  The initially scheduled start time of this trip instance.
- optional string start_date = 3

  The scheduled start date of this trip instance.
- optional ScheduleRelationship **schedule_relationship** = 4
- extensions to

  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.

2.20.1 Detailed Description

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.

2.20.2 Member Enumeration Documentation

2.20.2.1 enum TripDescriptor::ScheduleRelationship

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.

Enumerator

SCHEDULED Trip that is running in accordance with its GTFS schedule, or is close enough to the scheduled trip to be associated with it.

ADDED 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.

UNSCHEDULED A trip that is running with no schedule associated to it, for example, if there is no schedule at all.

CANCELED A trip that existed in the schedule but was removed.

2.20.3 Member Data Documentation

2.20.3.1 optional uint32 TripDescriptor::direction_id = 6

The direction_id from the GTFS feed trips.txt file, indicating the direction of travel for trips this selector refers to.

This field is still experimental, and subject to change. It may be formally adopted in the future.

2.20.3.2 optional string TripDescriptor::start_date = 3

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.

2.20.3.3 optional string TripDescriptor::start_time = 2

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 StopTime↩ Update. 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.

2.20.3.4 extensions TripDescriptor::to

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.

2.20.3.5 optional string TripDescriptor::trip_id = 1

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.21 TripReplacementPeriod Struct Reference

Public Attributes

- optional string route_id = 1

  The replacement period is for this route.
- optional transit_realtime TimeRange replacement_period = 2

  The start time is omitted, the end time is currently now + 30 minutes for all routes of the A division.

The documentation for this struct was generated from the following file:

- nyct-subway.proto

2.22 TripUpdate Struct Reference

Entities used in the feed.

Classes

- struct StopTimeEvent

  Timing information for a single predicted event (either arrival or departure).
- struct StopTimeUpdate

  Realtime update for arrival and/or departure events for a given stop on a trip.

Public Attributes

- required TripDescriptor trip = 1

  The Trip that this message applies to.
- optional VehicleDescriptor vehicle = 3

  Additional information on the vehicle that is serving this trip.
- repeated StopTimeUpdate stop_time_update = 2

  Updates to StopTimes for the trip (both future, i.e., predictions, and in some cases, past ones, i.e., those that already happened).
- optional uint64 timestamp = 4

  Moment at which the vehicle's real-time progress was measured.
- optional int32 delay = 5

  The current schedule deviation for the trip.
- extensions to

  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.

2.22.1 Detailed Description

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.

2.22.2 Member Data Documentation

2.22.2.1 optional int32 TripUpdate::delay = 5

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.

2.22.2.2 repeated StopTimeUpdate TripUpdate::stop_time_update = 2

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.

2.22.2.3 optional uint64 TripUpdate::timestamp = 4

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).

2.22.2.4 extensions TripUpdate::to

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.

2.22.2.5 required TripDescriptor TripUpdate::trip = 1

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.23 VehicleDescriptor Struct Reference

Identification information for the vehicle performing the trip.

Public Attributes

- optional string id = 1

  Internal system identification of the vehicle.
- optional string label = 2

  User visible label, i.e., something that must be shown to the passenger to help identify the correct vehicle.
- optional string license_plate = 3

  The license plate of the vehicle.
- extensions to

  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.

2.23.1 Detailed Description

Identification information for the vehicle performing the trip.

2.23.2 Member Data Documentation

2.23.2.1 optional string VehicleDescriptor::id = 1

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.

2.23.2.2 optional string VehicleDescriptor::label = 2

User visible label, i.e., something that must be shown to the passenger to help identify the correct vehicle.

2.23.2.3 extensions VehicleDescriptor::to

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.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

2.24 VehiclePosition Struct Reference

Realtime positioning information for a given vehicle.

Public Types

- enum VehicleStopStatus { INCOMING_AT = 0, STOPPED_AT = 1, IN_TRANSIT_TO = 2 }
- enum CongestionLevel {
  UNKNOWN_CONGESTION_LEVEL = 0, **RUNNING_SMOOTHLY** = 1, **STOP_AND_GO** = 2, **CONGESTION** = 3,
  SEVERE_CONGESTION = 4 }

  Congestion level that is affecting this vehicle.
- enum OccupancyStatus {
  EMPTY = 0, MANY_SEATS_AVAILABLE = 1, FEW_SEATS_AVAILABLE = 2, STANDING_ROOM_ONLY = 3,
  CRUSHED_STANDING_ROOM_ONLY = 4, FULL = 5, NOT_ACCEPTING_PASSENGERS = 6 }

  The degree of passenger occupancy of the vehicle.

Public Attributes

- optional TripDescriptor trip = 1

  The Trip that this vehicle is serving.
- optional VehicleDescriptor vehicle = 8

  Additional information on the vehicle that is serving this trip.
- optional Position position = 2

  Current position of this vehicle.
- optional uint32 current_stop_sequence = 3

  The stop sequence index of the current stop.
- optional string stop_id = 7

  Identifies the current stop.
- optional VehicleStopStatus current_status = 4 [default = IN_TRANSIT_TO]

  The exact status of the vehicle with respect to the current stop.
- optional uint64 timestamp = 5

  Moment at which the vehicle's position was measured.
- optional CongestionLevel **congestion_level** = 6
- optional OccupancyStatus **occupancy_status** = 9
- extensions to

  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.

2.24.1 Detailed Description

Realtime positioning information for a given vehicle.

2.24.2 Member Enumeration Documentation

2.24.2.1 enum VehiclePosition::CongestionLevel

Congestion level that is affecting this vehicle.

Enumerator

SEVERE_CONGESTION People leaving their cars.

2.24.2.2 enum VehiclePosition::OccupancyStatus

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.

Enumerator

EMPTY The vehicle is considered empty by most measures, and has few or no passengers onboard, but is still accepting passengers.

MANY_SEATS_AVAILABLE 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.

FEW_SEATS_AVAILABLE 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.

STANDING_ROOM_ONLY The vehicle can currently accommodate only standing passengers.

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

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

NOT_ACCEPTING_PASSENGERS The vehicle is not accepting additional passengers.

2.24.2.3 enum VehiclePosition::VehicleStopStatus

Enumerator

INCOMING_AT The vehicle is just about to arrive at the stop (on a stop display, the vehicle symbol typically flashes).

STOPPED_AT The vehicle is standing at the stop.

IN_TRANSIT_TO The vehicle has departed and is in transit to the next stop.

2.24.3 Member Data Documentation

2.24.3.1 optional VehicleStopStatus VehiclePosition::current_status = 4 [default = IN_TRANSIT_TO]

The exact status of the vehicle with respect to the current stop.

Ignored if current_stop_sequence is missing.

2.24.3.2 optional uint32 VehiclePosition::current_stop_sequence = 3

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.

2.24.3.3 optional string VehiclePosition::stop_id = 7

Identifies the current stop.

The value must be the same as in stops.txt in the corresponding GTFS feed.

2.24.3.4 optional uint64 VehiclePosition::timestamp = 5

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).

2.24.3.5 extensions VehiclePosition::to

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.

2.24.3.6 optional TripDescriptor VehiclePosition::trip = 1

The Trip that this vehicle is serving.

Can be empty or partial if the vehicle can not be identified with a given trip instance.

The documentation for this struct was generated from the following file:

- gtfs-realtime.proto

Index