THE DEFINITIVE GUIDE TO GTFS-REALTIME

Quentin Zervaas

The Definitive Guide to GTFS-realtime

How to consume and produce real-time public transportation data with the GTFS-rt specification.

Quentin Zervaas

About This Book

This book is a comprehensive guide to GTFS-realtime, a specification for publishing of real-time public transportation data. GTFS-realtime is designed to complement the scheduled data that hundreds of transit agencies around the world publish using GTFS (General Transit Feed Specification).

This book begins with a description of the specification, with discussion about the three types of data contained in GTFS-realtime feeds: service alerts; vehicle positions; and trip updates.

Next the reader is introduced to *Protocol Buffers*, the data format that GTFS-realtime uses when it is being transmitted. This section then instructs the reader how to consume the three types of data from GTFS-realtime feeds (both from standard feeds and feeds with *extensions*).

Finally, the reader is shown how to produce a GTFS-realtime feed. A number of examples in this book use Java, but the lessons can be applied to a number of different languages.

This book complements *The Definitive Guide to GTFS*, available from http://gtfsbook.com.

About The Author

Quentin Zervaas (@HendX on Twitter) is a software developer from Adelaide, Australia.

He is the creator of the iOS & Android app TransitTimes (http://transittimesapp.com), which provides public transportation information in Australia, New Zealand, Canada and the United States. It uses many sources of data, including many GTFS and GTFS-realtime feeds.

Quentin has also created TransitFeeds.com, a web site that provides a comprehensive listing and archive of public transportation data available around the world. TransitFeeds.com is referenced various times throughout this book and is used for a number of examples.

Credits

Technical Reviewer

Nick Maher

Copy Editors

Miranda Little
Anne Delvizis

The Definitive Guide to GTFS-realtime

http://gtfsrealtime.com

First Edition. Published in August 2015.

Table of Contents

1. Introduction to GTFS-realtime

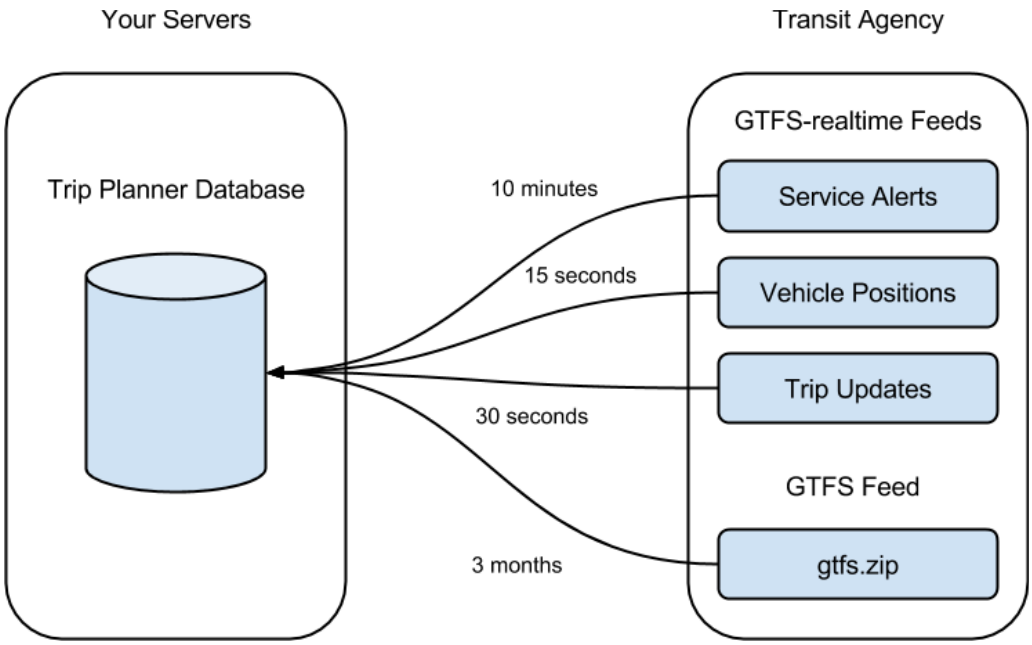
GTFS-realtime is a standard developed by Google in order to allow transit agencies to provide real-time information about their service.

There are three types of data a GTFS-realtime feed provides:

1. Vehicle positions
2. Trip updates
3. Service alerts

Vehicle positions contain data about events that have already occurred (e.g. “the vehicle was at this location one minute ago”), whereas trip updates contain data about events that are yet to occur (e.g. “the bus will arrive in three minutes”).

Typically, a single GTFS-realtime feed contains only one of these three types of data. Many agencies therefore have multiple GTFS-realtime feeds (that is, one for vehicle positions, one for trip updates and one for service alerts).



The above diagram shows how a GTFS-realtime feed is designed to complement a GTFS (General Transit Feed Specification) feed. It does this in two ways:

1. All identifiers for routes, trips and stops match those that appear in the corresponding GTFS feed.

2. A GTFS feed shows the projected schedule for a given period (such as the next six months), while the GTFS-realtime is used to make last-minute adjustments based on real-world conditions (such as traffic, roadworks, or weather).

Consuming GTFS-realtime Feeds

The format of GTFS-realtime feeds is based on Protocol Buffers, a language and platform-neutral mechanism for serializing structured data.

This is similar conceptually to JSON (JavaScript Object Notation), but the data transferred across the wire is binary data and not human-readable in its raw format.

Chapter 5. Protocol Buffers (page 33) shows you how to use Protocol Buffers and the associated `gtfs-realtime.proto` file (used to instruct Protocol Buffers how GTFS-realtime is structured).

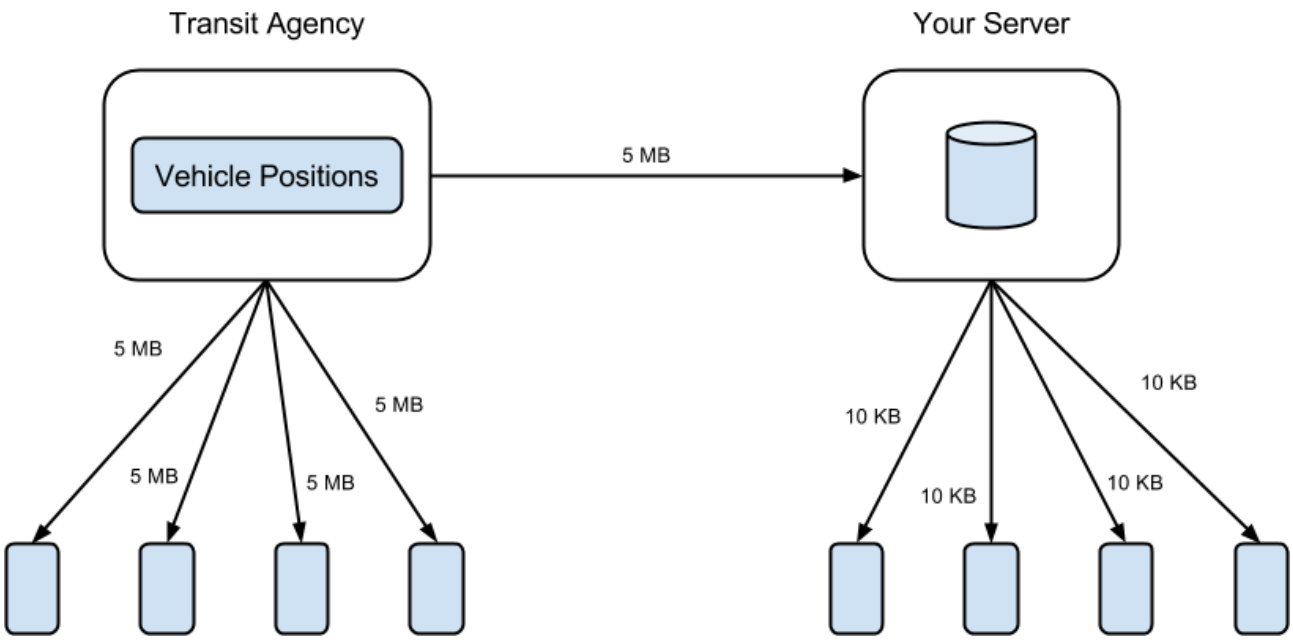
Consuming GTFS-realtime Feeds on Mobile Devices

A common use-case for GTFS and GTFS-realtime feeds is to build transit-related mobile apps that show scheduling data. However, it is important to note that GTFS-realtime feeds are not intended to be consumed directly by a mobile device.

Many of the vehicle positions and trip update feeds provided by transit agencies include a snapshot of their entire network at a single moment. For a large network, this could be multiple megabytes of data being updated every 10-15 seconds.

A mobile app downloading a full GTFS-realtime feed every 10-15 seconds would quickly download a large amount of data over their cellular connection, which could be very expensive. Additionally, their device would need to process a large amount of data – most of which would not be relevant – which would run down their battery unnecessarily. This would also put a huge amount of strain on the provider's servers.

Rather, GTFS-realtime is intended to be consumed by an intermediate server. In the case of a mobile app, this intermediate server would likely belong to the creator of the app. The mobile app can then query this intermediate server for the relevant data it needs at that time.



The above diagram demonstrates the different models. On the left, mobile devices download entire GTFS-realtime feeds from the provider. Each device is downloading 5 megabytes every 10-15 seconds.

On the right, an intermediate server records all vehicle positions, then mobile devices request only the data they need. This significantly reduces the amount of data transferred.

2. Introduction to Service Alerts

Service alerts are generally used by transit agencies to convey information that can not be conveyed using a trip update message (the GTFS-realtime trip update message is covered in *Chapter 4. Introduction to Trip Updates* on page 27).

For example, consider a bus stop that was to be closed for a period of time due to construction in the area. If the stop was to be closed for a long period of time, the transit agency could modify the long-term schedule (the GTFS feed). If the closure was unexpected and the stop will reopen later that day, the agency can reflect this temporary closure using trip updates in the GTFS-realtime feed (instructing each relevant trip to skip that stop).

However, in both of these cases the reason *why* there were no trips visiting the stop has not been conveyed. Using a service alert, you can explain why the stop is closed and when it will reopen.

You can attach one or more entities to a service alert (such as routes, trips or stops). In the above example of a stop being closed, you could include its stop ID as it appears in the corresponding GTFS feed, thereby allowing apps consuming the feed to display a message to their users.

Examples of Service Alerts

Some examples of common service alerts used by agencies include:

- **Holiday schedules.** If there is an upcoming holiday, agencies may use service alerts to remind travelers that a holiday schedule will be applied that day.
- **Stop closing.** If a stop is closed (temporarily or permanently) customers may be notified using a service alert. In this instance, the alert can be linked to the stop that is closing. If it is being replaced by a new stop, the new stop may also be linked.
- **Route detour.** Service alerts can be used to indicate that for a period of time in the future a route will be redirected, perhaps due to a road closure. In this instance, the alert would link to stops that lie on the closed part of the road, as well as to routes that will detour as a result of the closure.
- **Change to schedule.** If an upcoming change to a schedule results in far less (or far more) services operating a stop, service alerts might be used to notify customers.
- **Vehicle broken down.** If a bus has broken down, or if electric trains are not moving due to a power outage, passengers can be notified using service alerts. In this instance, the alert could link to a specific trip, or it could be more general and instead link to its route or to the stops affected.

The `EntitySelector` type described on page 14 describes how service alerts can be linked to routes, trips and stops accordingly.

Sample Feed

The following extract is from the service alerts feed of TriMet in Portland (http://developer.trimet.org/GTFS.shtml). It contains a single GTFS-realtime service alert *entity*. An entity contains either a service alert, a vehicle position or a trip update. This service alert indicates that a tree has fallen, causing potential delays to two routes.

Note: This extract has been converted from its binary format into a human-readable version. *Outputting Human-Readable GTFS-realtime Feeds* (page 39) illustrates how this is achieved.

```
entity {
  id: "35122"
  alert {
    active_period {
      start: 1415739180
      end: 1415786400
    }
    informed_entity {
      route_id: "19"
      route_type: 3
    }
    informed_entity {
      route_id: "71"
      route_type: 3
    }
    url {
      translation {
        text: "http://trimet.org/alerts/"
      }
    }
    description_text {
      translation {
        text: "Expect delays due to a tree down blocking northbound 52nd at Tolman. Police are
flagging traffic thru using the southbound lane."
      }
    }
  }
}
```

The elements of this service alert entity are as follows.

- Active Period

- Informed Entity
- URL
- Description text

Each of these fields are discussed below, as are some ways this particular feed could be improved.

Active Period

The active period element in this example states that the alert is active between 12:53 PM on one day and 2:00 AM the following day (Portland time). It is likely they included this finishing time to say "this might last all day, but it definitely won't be a problem tomorrow".

Often precise timing isn't known. If the active period is omitted, then the alert is assumed to be active for as long as it appears in the feed.

Informed Entity

In a GTFS-realtime service alerts feed, *informed entity* refers to a route, trip, stop, agency or route type, or any combination of these.

In this example, there are two informed entities, both of which are bus routes (as indicated by a route type of `3`). Referring to the TriMet GTFS feed (http://developer.trimet.org/schedule/gtfs.zip), the routes with an ID of `19` and `71` are as follows.

```
route_id,route_short_name,route_long_name,route_type
19,19,Woodstock/Glisan,3
71,71,60th Ave/122nd Ave,3
```

Technically in this case the `route_type` value need not be specified, as this can be derived using the GTFS feed. Sometimes, however, an alert may impact *all* routes for a given mode of transport, so the route type would be specified only.

For instance, if an electrical outage affects all subway trains, then an informed entity containing only a route type of `1` (the GTFS value for Subway routes) would be sufficient, rather than including a service alert for each subway route.

URL

This field contains a web address where additional information about the alert can be found.

In this particular example, TriMet has included a generic URL for their service alert. Other alerts in the same feed also use the same URL. It would be far more useful if instead each alert pointed to a URL specifically related to that alert. This would make it easier to provide the end user with additional information specific to that alert.

Description Text

This field contains a textual description of the alert that can be presented to the users of your web site or app. Just like the URL field, it has a type of `TranslatedString`, which is the mechanism by which GTFS-realtime can provide translations in multiple languages if required. The sample feed on page 10 contains an English version of the description.

Improvements

In addition to providing a more specific URL, this alert could be improved by including values for the `cause` and `effect` fields. For instance, `cause` could have a value of `WEATHER` or `ACCIDENT`, while `effect` could have a value of `DETOUR` or `SIGNIFICANT_DELAYS`.

Additionally, this alert could include the `header_text` field to complement the `description_text` field. This would allow you to display a summary of the alert (`header_text`), then supply more information if the user of your app or web site requests it (`description_text`).

Specification

This section contains the specification for the `Alert` entity type. Some of this information has been sourced from the GTFS-realtime reference page (https://developers.google.com/transit/GTFS-realtime/reference).

Alert

An `Alert` message makes it possible to provide extensive information about a given service alert, including the ability to match it to any number of routes, stops or trips.

The fields for any single service alert are as described in the following table.

`active_period`	`TimeRange`	Zero or more occurrences
The time or times when this alert should be displayed to the user. If no times are specified, then the alert should be considered active as long as it appears in the feed. Sometimes there may be multiple active periods specified. For example, if some construction was occurring daily between a certain time, there might be an active period record for each day it will occur.		
`informed_entity`	`EntitySelector`	Zero or more occurrences
These are the entities this service alert relates to (such as route, trips or stops).		
`cause`	`Cause`	Optional
This is the event that occurred to trigger the alert. Possible values for the `Cause` enumerator are listed below this table.		
`effect`	`Effect`	Optional
This indicates what action was taken as a result of the incident. Possible values for the `Effect` enumerator are listed below this table.		
`url`	`TranslatedString`	Optional
A URL which provides additional information that can be shown to users.		
`header_text`	`TranslatedString`	Optional
A brief summary of the alert that can be used as a heading. This is to be in plain-text (no HTML markup).		
`description_text`	`TranslatedString`	Optional
A description of the alert that complements the header text. Similarly, it is to be in plain-text (no HTML markup).		

The following values are valid for the `Cause` enumerator:

`ACCIDENT`	`MAINTENANCE`	`TECHNICAL_PROBLEM`
`CONSTRUCTION`	`MEDICAL_EMERGENCY`	`WEATHER`
`DEMONSTRATION`	`POLICE_ACTIVITY`	`UNKNOWN_CAUSE`
`HOLIDAY`	`STRIKE`	`OTHER_CAUSE`

The following values are valid for the `Effect` enumerator:

`ADDITIONAL_SERVICE`	`DETOUR`	`REDUCED_SERVICE`
`NO_SERVICE`	`STOP_MOVED`	`UNKNOWN_EFFECT`
`SIGNIFICANT_DELAYS`	`MODIFIED_SERVICE`	`OTHER_EFFECT`

TimeRange

A `TimeRange` message specifies a time interval. It is not mandatory to include both the start and finish times, but at least one of those is required if a `TimeRange` is included.

`start`	`uint64` (64-bit unsigned integer)	Optional
The start time specified in number of seconds since 1-Jan-1970 00:00:00 UTC.		
`end`	`uint64` (64-bit unsigned integer)	Optional
The end time specified in number of seconds since 1-Jan-1970 00:00:00 UTC.		

If only the start time is specified, the time range is considered active after the starting time.

If only the end time is specified, the time range is considered active before the end time.

If both the start and finish times are specified, the time range is considered active between these times.

EntitySelector

An `EntitySelector` message is used to specify an entity from within a GTFS feed. Doing so allows you to match up a service alert with a route (or all routes of a given type), trip, stop or agency from the GTFS feed that corresponds to the GTFS-realtime feed.

`agency_id`	`string`	Optional
This is the ID of an agency as it appears in the `agency.txt` file of the corresponding GTFS feed.		
`route_id`	`string`	Optional
This is the ID of a route as it appears in the `routes.txt` file of the corresponding GTFS feed.		
`route_type`	`int32` (32-bit signed integer)	Optional
This is a GTFS route type, such as `3` for bus routes or `4` for ferry routes. Extended GTFS route types can also be used for this value.		
`trip`	`TripDescriptor`	Optional
This is used to match a specific trip from the corresponding GTFS feed's `trips.txt` file. Trip matching can be potentially more complex than just matching the `trip_id`, which is why this field differs to the other ID-related fields in `EntitySelector`. You can find more discussion of this below in the `TripDescriptor` section.		
`stop_id`	`string`	Optional
This is the ID of a stop as it appears in the `stops.txt` file of the corresponding GTFS feed. If the corresponding stop is of type “station” (a `location_type` value of `1`), then you may consider matching this entity to its child stops also.		

All of these elements are optional, but at least one of them must occur. If multiple elements are specified, then all must be matched.

Note: Conversely, if you want multiple matches, then you should instead include multiple `EntitySelector` values. For instance, if you want a service alert that covers all buses and ferries, then the `informed_entity` field would contain one `EntitySelector` for buses, and another for ferries.

The following table shows some different combinations that can occur, and what each of them mean.

Fields Specified	Meaning
`agency_id`	The alert applies to anything relating to the given agency. This may include any routes or trips that match back to the agency, or even stops that the trips stop at.
`route_id`	The alert applies to the given route. For instance, if a user is viewing upcoming departures for the matched route, then it would be appropriate to display the alert.
`route_type`	The alert is relevant when showing the user any data related to the given route type. For example, if a route type of `3` (buses) is specified, then it would be appropriate to display the alert when a user is viewing upcoming departures for any bus route in the corresponding GTFS feed.
`trip`	If a trip is matched, then it would be appropriate to display the alert when the user is viewing anything related to that trip. For instance, if you are showing a list of stop times for the trip then it would be relevant. If you have received a real-time vehicle position for the trip and are showing it to the user on a map, you might show the service alert if the user taps on the vehicle.
`stop_id`	If a stop is matched here then it would be appropriate to show an alert in a number of situations, such as when viewing upcoming departures for the stop, or if the user is taking a trip that embarks or disembarks at the matched stop.
`agency_id` + `route_id`	This kind of match is redundant, because there should only ever be a maximum of one route that matches a given `route_id` value in a GTFS feed.
`route_id` + `trip`	Similar to the previous case, any matched trip will only belong to a single route, so specifying the `route_id` has no real meaning.
`route_id` + `stop_id`	Matching both a route and a stop can be useful if an alert relating to a stop only applies to certain routes. For instance, if a stop is serviced by two different routes and you want to notify users that one of the routes will no longer stop here, the alert does not apply to the route that will continue to service the stop.
`trip` + `stop_id`	In this case, a service alert is matched to a combination of a trip and a stop. The alert would be relevant to a user waiting at a stop for a particular vehicle. It would not apply to other people at the same stop waiting for a different route.
`route_type` + `stop_id`	Sometimes a stop is shared by multiple travel modes. For instance, some light rail services share stops with buses. This combination can be useful if a stop-related alert only applies to one of those modes.

As this demonstrates, it is possible to match service alerts to real-world entities in any number of ways. This allows you to keep relevant users informed. The alternative to matching on this granular level would be to show all of your users all service alerts, meaning most alerts would be irrelevant to most people.

TripDescriptor

One of the files in a GTFS feed is `frequencies.txt`, which is used to specify trips that repeat every *x* minutes. This file is used when an agency does not have a specific schedule for trips, other than guaranteeing, for instance, that a new trip departs every five minutes.

For example, it is possible for a particular route to run every five minutes for an entire day, while only having one entry in `trips.txt` (and one set of corresponding stop times in `stop_times.txt`).

When using trip frequencies the `trip_id` value may not be enough to uniquely identify a single trip from the GTFS feed. This means that in order to match a trip, additional information may need to be supplied, which the `TripDescriptor` message allows for.

`trip_id`	`string`	Optional
This is the ID of a trip as it appears in the `trips.txt` of the corresponding GTFS feed. Alternatively, this value may refer to a trip that has been added via a `TripUpdate` message and does not exist in the GTFS feed.		
`route_id`	`string`	Optional
If this value is specified, it should match the route ID for the trip specified in `trip_id`. If the `route_id` is specified but no `trip_id` is specified, then this trip descriptor references all trips for the given route.		
`direction_id`	`uint32` (32-bit unsigned integer)	Optional
This value corresponds to the `direction_id` value as specified in the `trips.txt` file of the corresponding GTFS feed. At time of writing this is an experimental field in the GTFS-realtime specification.		
`start_time`	`string`	Optional
If the specified trip in `trip_id` is a frequency-expanded trip, this value must be specified in order to determine which instance of a trip this selector refers to. Its value is in the format `HH:MM:SS`, as in the `stop_times.txt` and `frequencies.txt` files.		
`start_date`	`string`	Optional
It is possible that knowing the `trip_id` may not be enough to determine a specific trip. For instance, if a train is scheduled to depart at 11:30 PM but is running 40 minutes late, then you would need to know its date in order to match up with the original trip (40 minutes late), and not the next day's instance of the trip (23 hours 20 minutes early). This field helps to avoid this ambiguity. The date is specified in `YYYYMMDD` format.		
`schedule_relationship`	`ScheduleRelationship`	Optional
This value indicates the relationship between the trip(s) specified in this selector and its regular schedule.		

The following values are valid for the `ScheduleRelationship` enumerator:

- `SCHEDULED`. Used when the trip being described is running in accordance with a trip in the GTFS feed.
- `ADDED`. A trip that was added in addition to the schedule. For instance, if an extra trip was added because there were more passengers than normal, it would be represented using this value.
- `UNSCHEDULED`. A trip that is running with no schedule associated with it. For instance, if this trip is expected to run but there is no static schedule associated with it, it would be marked with this value.
- `CANCELED`. A trip that existed in the schedule but was removed. For example, if a vehicle broke down and could not complete the trip, then it would be marked as canceled.

If a trip has been added, then the `route_id` should be populated, as without this it may not be possible to determine which route the added trip corresponds to (since the `trip_id` value would not appear in the GTFS `trips.txt` file).

With the newly-added `direction_id` field (still experimental at time of writing this book), an added trip can also have its direction specified, meaning you can present information to your users about which direction the vehicle is traveling, even if you do not know its specific stops.

TranslatedString

A `TranslatedString` message contains one or more `Translation` elements. This allows for alerts to be issued in multiple languages. A `Translation` element is structured as follows.

`text`	`string`	Optional
A UTF-8 string containing the message. This string will typically be read by the users of your web site or app.		
`language`	`string`	Optional
This is the language code for the given text (such as `en-US` for United States English). It can be omitted, but if there are multiple translations then at most only one translation can have this value omitted.		

Thanks for reading a sample of

The Definitive Guide to GTFS-realtime

Download the full book today:

http://gtfsbook.com

Index

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z