Technical Memo

To:	SHRP2 - C10 Tri-Agency Project Implementation Files
From:	Stefan Coe
Date:	July 9th, 2015
Subject:	Transit Network Design Specification - Working Draft

Introduction:

The purpose of this document is to lay out a transit network design specification for the SHRP2 Fast-Trips implementation research project. This document examines and evaluates existing transit network specifications including previous versions of Fast-Trips and the General Transit Feed Specification (GTFS), identifies gaps and improvements, and proposes a final specification.

Network Design Considerations

The following considerations were taken into account when evaluating existing specifications and proposing a new one. They were developed based on feedback from various members of our team.

1. In order to make use of existing libraries and visualization capabilities, the format should use existing and mainstream conventions and standards when possible, which in this case means GTFS. Where GTFS has not yet incorporated a standard, the GTFS proposals should be evaluated to see if they could be used rather than developing something new.
2. In order to remain flexible to a variety of applications, the format should accommodate a variety of variables that may be used in route finding utility functions and rules, including on access links.
3. In order to increase the likelihood that errors are caught, the format should be legible and human readable without special tools.

Existing Fast-Trips and GTFS Network Specification

Fast-Trips currently uses a GTFS-derivative, schedule-based transit network. Transit vehicle trips are represented by a sequence of bus stops. Arrival and departure times for each stop for each trip are explicitly specified, with the travel time and dwell time being endogenous. The arrival and departure time between stops and can either be taken directly from the published GTFS schedule, or derived from a vehicle assignment model (i.e. DTA or SUE).

Transit riders are assigned a feasible itinerary from the schedule that takes them from a stop that is accessible to their origin to one that that is accessible to their destination. Transit demand is supplied to the network via TAZ centroids, which are connected to the transit network using access links. Some versions of Fast-Trips allow for park and ride (PNR) lots to be chosen within Fast-Trips, but the input to Fast-Trips are the origin and destination locations (i.e. traffic analysis zones).

The primary differences between Fast-Trips and GTFS specifications are the file format, the required files, and the variable definitions within the files. The file format and required files are outlined in sub-sections below. The detailed differences in variables within each file can be found in the sub-section discussing Fast-Trips Current Specification. The current Fast-Trips specification for the transit network is discussed in three parts: access links, transfer links, and transit service. Next, there is a section describing GTFS files related to fares, which are not included in the current Fast-Trips specification but will be needed for this project. The final sub-section discusses issues and drawbacks of the current Fast-Trips specification.

In addition to files that define the transit network, Fast-Trips has input files to describe zonal centroid locations, demand, path building parameters, and run-time parameters. These are outlined in detail in the Fast-Trips documentation and are not discussed here because they will be addressed in future design decisions as a part of Task 3 - Demand and Task 4 - Transit Rider behavior.

File Format

GTFS files are comma-delimited, while Fast-Trips reads in tab-delimited. In both cases, the first line of each file must contain the field names, which are case-sensitive. Field values may not contain tabs, carriage returns or new lines.

Required FIles

File	Filename (Fast-Trips / GTFS)	GTFS	Fast-Trips
Access Links	GTFS: NA FT: `ft_input_accessLinks.dat`	NA	Required
Transfer Links	GTFS: `transfers.txt` FT: `ft_input_transfers.dat`	Optional	Required
Trips	GTFS: `trips.txt` FT: `ft_input_trips.dat`	Required	Required
Routes	GTFS: `routes.txt` FT: `ft_input_routes.dat`	Required	Required
Stops	GTFS: `stops.txt` FT: `ft_input_stops.dat`	Required	Required

Stop Times	GTFS: `stop_times.txt` FT: `ft_input_stopTimes.dat`	Required	Required
Shapes	GTFS: `shapes.txt` FT: `ft_input_shapes.dat`	Optional	Optional
Agency	GTFS: `agency.txt` FT: NA	Required	NA
Calendar	GTFS: `calendar.txt` FT: NA	Required	NA
Calendar Dates (exceptions)	GTFS: `calendar_dates.txt` FT: NA	Optional	NA
Fare Attributes	GTFS: `fare_attributes.txt` FT: NA	Optional	NA
Fare Rules	GTFS: `fare_rules.txt` FT: NA	Optional	NA
Frequencies	GTFS: `frequencies.txt` FT: NA	Optional	NA
Feed publication info	GTFS: `feed_info.txt` FT: NA	Optional	NA

Current Fast-Trips Specification

The core Fast-Trips model uses the following input components: access links, transfer links, and transit service but does not include information about fares. Many of the variables names are similar, but not exact to the GTFS specification.

Note that in all tables below, the "requirements" column is encoded as follows:

- + : Required
- O: Optional
- NA: Not Applicable

Access links represent pedestrian links from a centroid to each accessible transit stop. They can be created using an existing Fast-Trips pre-processing tool, or a user can use more sophisticated techniques to create access links and supply this file themselves.

GTFS: *Not applicable*
Fast-Trips: **`ft_input_accessLinks.dat`**
Contains a record for each feasible stop <--> zone pair

Field Name	Required	Details
TAZ	**GTFS: NA** **FT: +**	Zone ID
stop	**GTFS: NA** **FT: +**	Stop ID
dist	**GTFS: NA** **FT: +**	Walking distance in miles between TAZ and stop
time	**GTFS: NA** **FT: +**	Walking time in minutes between TAZ and stop

Transfer links are links between stops that are traversed on foot. They are created for each stop to all other stops that are within a certain distance of said stop. These are the potential stops that are considered accessible during a transfer from an individual stop.

GTFS: **`transfers.txt`**
Fast-Trips: **`ft_input_transfers.dat`**
Contains a record for each feasible stop <--> stop pair

Field Name	Required	Details
GTFS: from_stop_id **FT:** fromStop	**GTFS:** + **FT:** +	From stop ID
GTFS: to_stop_id **FT:** toStop	**GTFS:** + **FT:** +	To stop ID
GTFS: NA **FT:** dist	**GTFS:** NA **FT:** +	Walking distance in miles between stops
GTFS: NA **FT:** time	**GTFS:** NA **FT:** +	Walking time in minutes between stops
GTFS: transfer_type **FT:** NA	**GTFS:** + **FT:** NA	Specifies the type of connection: 0 / Empty - a recommended transfer point 1 - timed transfer between two routes 2 - requires a minimum amount of time, specified by min_transfer_time 3 - transfers not possible between routes
GTFS: min_transfer_time **FT:** **NA**	**GTFS:** + **FT:** NA	When a connection between routes requires an amount of time between arrival and departure (transfer_type=2), this field defines the amount of

		time that must be available for a typical rider - in seconds.

Transit Service is specified by modifying five GTFS files to a format readable by Fast-Trips. There are files that represent transit vehicle trips, routes, stops, stop times, and route shapes.

GTFS: `trips.txt`
Fast-Trips: `ft_input_trips.dat`
Contains a record for every transit vehicle trip (i.e. the Muni 14 Local Outbound that leaves at 8:02 AM)

Field Name	Required	Details
GTFS: trip_id **FT:** tripID	**GTFS:** + **FT:** +	ID that uniquely identifies a vehicle trip
GTFS: route_id **FT:** routeId	**GTFS:** + **FT:** +	ID that uniquely identifies a route
GTFS: service_id **FT:** **NA**	**GTFS:** + **FT:** NA	ID that uniquely identifies set of dates when service is available from calendar or calendar dates files.
GTFS: trip_headsign **FT:** **NA**	**GTFS:** O **FT:** NA	Text that appears on the vehicle headsign to identify destination to passengers.
GTFS: trip_short_name **FT:** **NA**	**GTFS:** O **FT:** NA	Text that appears in schedules and sign boards.
GTFS: NA **FT:** type	**GTFS:** NA **FT:** +	Service type: 0 - Tram, streetcar, light rail 1 - Subway, metro 2 - Rail 3 - Bus 4 - Ferry 5 - Cable car 6 - Gondola
GTFS: block_id **FT:** NA	**GTFS:** O **FT:** NA	Two or more sequential trips made using same vehicle where passenger can transfer by staying on same vehicle. block_id must be referenced by two or more trips in trips.txt.
GTFS: shape_id **FT:** shapeId	**GTFS:** O **FT:** O	Defines shape for the trip from shapes.txt file.

GTFS: wheelchair_accessible **FT:** NA	**GTFS:** O **FT:** NA	0 - no accessibility info 1 - vehicle on this trip can accommodate at least one rider in a wheelchair 2 - no riders in wheelchairs can be accommodated on this trip
GTFS: bikes_allowed **FT:** NA	**GTFS:** O **FT:** NA	0 - no bike accessibility info 1 - vehicle on this trip can accommodate at least one bicycle 2 - no bicycles can be accommodated on this trip
GTFS: NA **FT:** startTime	**GTFS:** NA **FT:** +	Start time of the trip
GTFS: NA **FT:** capacity	**GTFS:** NA **FT:** +	Vehicle capacity of the trip (default is 60)
GTFS: direction_id **FT:** directionId	**GTFS:** O **FT:** +	ID that contains a binary value that indicates the direction of the trip: 0 - travel in one direction 1 - travel in opposite direction

GTFS: **`routes.txt`**
Fast-Trips: **`ft_input_routes.dat`**
Contains a record for every transit route (i.e. the Muni - 14 Local)

Field Name	Required	Details
GTFS: route_id **FT:** routeId	**GTFS:** + **FT:** +	ID that uniquely identifies a route
GTFS: agency_id **FT:** NA	**GTFS:** O **FT:** NA	ID that identifies the agency as specified in agency.txt
GTFS: route_short_name **FT:** routeShortName	**GTFS:** + **FT:** +	Short name of the route
GTFS: route_long_name **FT:** routeLongName	**GTFS:** + **FT:** +	Full name of the route
GTFS: route_desc **FT:** NA	**GTFS:** O **FT:** NA	Description of route.

GTFS: route_type **FT:** routeType	**GTFS:** + **FT:** +	Service type: 0 - Tram, streetcar, light rail 1 - Subway, metro 2 - Rail 3 - Bus 4 - Ferry 5 - Cable car 6 - Gondola 7 - Funicular
GTFS: route_url **FT:** NA	**GTFS:** O **FT:** NA	Webpage for that route
GTFS: route_color **FT:** NA	**GTFS:** O **FT:** NA	Display color for route in six-digit hexadecimal
GTFS: route_text_color **FT:** NA	**GTFS:** O **FT:** NA	Color of text to be drawn on top of route color, specified in six-digit hexadecimal

GTFS: `stops.txt`
Fast-Trips: `ft_input_stops.dat`
Contains a record for every transit stop or station (i.e. Embarcadero Station)

Field Name	Required	Details
GTFS: stop_id **FT:** stopID	**GTFS:** + **FT:** +	ID that uniquely identifies a stop or station
GTFS: stop_code **FT:** NA	**GTFS:** O **FT:** NA	Short text or a number that identifies the stop for passengers (i.e. EMB)
GTFS: stop_name **FT:** stopName	**GTFS:** + **FT:** +	Name of a stop or station
GTFS: stop_desc **FT:** stopDescription	**GTFS:** O **FT:** +	Description of a stop or station
GTFS: stop_lat **FT:** Latitude	**GTFS:** + **FT:** +	Latitude of stop or station in WGS 84
GTFS: stop_lon **FT:** Longitude	**GTFS:** + **FT:** +	Longitude of stop or station in WGS 84
GTFS: NA **FT:** capacity	**GTFS:** NA **FT:** +	Capacity of stop or station (use a large number if unknown)

GTFS: zone_id **FT:** NA	**GTFS:** O **FT:** NA	Defines a fare zone for the stop ID. Required if you want to provide fare information in fare_rules.txt that uses zones.
GTFS: location_type **FT:** NA	**GTFS:** O **FT:** NA	Identifies whether this stop is a stop or station. If nothing is specified or is blank, it is assumed it is a stop. Stations can have different properties from stops. 0 or blank - stop 1 - station (contains one or more stops)
GTFS: parent_station **FT:** NA	**GTFS:** O **FT:** NA	For stops inside stations, identifies station associated with the stop. Stops.txt must also contain a row where this stop id is assigned a location type 1.
GTFS: stop_timezone **FT:** NA	**GTFS:** O **FT:** NA	Contains timezone where stop or station is located. If omitted, stop assumed to be located in timezone in agency.txt.
GTFS: wheelchair_boarding **FT:** NA	**GTFS:** O **FT:** NA	Identifies whether wheelchair boardings are possible from the specified stop or station. 0 - no information 1 - some vehicles can be boarded by wheelchair 2 - wheelchair boarding not possible If a stop is part of a station: 0 - will inherit from parent station, if specified. 1 - there is an accessible path from outside station to stop 2 - no accessible path to specific stop

GTFS: `stop_times.txt`
Fast-Trips: `ft_input_stopTimes.dat`
Contains a record for every scheduled stop within a trip and route (i.e. the time when the Muni 14 Local Outbound that left at 8:02 gets to 24th St. and Mission St)

Field Name	Required	Details
GTFS: trip_id **FT:** tripId	**GTFS:** + **FT:** +	ID that uniquely identifies trip
GTFS: arrival_time **FT:** arrivalTime	**GTFS:** + **FT:** +	Arrival time at a specific stop for a specific trip on a route in HHMMSS format measured from midnight. For trips that span multiple dates, the time should be entered as a value greater than 2400000
GTFS: departure_time	**GTFS:** O **FT:** +	Departure time at a specific stop for a specific trip on a route in HHMMSS format measured from midnight. For

FT: departureTime		trips that span multiple dates, the time should be entered as a value greater than 2400000
GTFS: stop_id **FT:** stopId	**GTFS:** + **FT:** +	ID that uniquely identifies a stop
GTFS: stop_sequence **FT:** sequence	**GTFS:** + **FT:** +	Sequence number on a specific stop within a trip. The first stop sequence is 1 and subsequent stops in the trip are sequentially numbered.
GTFS: stop_headsign **FT:** NA	**GTFS:** O **FT:** NA	Text that appears on sign that identifies the trips destination to passengers. use this field to override default headsign when it changes at stops.
GTFS: pickup_type **FT:** NA	**GTFS:** O **FT:** NA	0/default - regular pickup 1 - no pickup available 2 - must phone agency 3 - must coordinate with driver
GTFS: drop_off_type **FT:** NA	**GTFS:** O **FT:** NA	0/default - regular drop off 1 - no drop off available 2 - must phone agency 3 - must coordinate with driver
GTFS: shape_dist_traveled **FT:** NA	**GTFS:** O **FT:** NA	Positions a stop as a distance from the first shape point in units that are used in this field in shapes.txt
GTFS: timepoint **FT:** NA	**GTFS:** O **FT:** NA	Indicates if specified arrival and departure times for a stop are strictly adhered to by the transit vehicle or if they are approximate and/or interpolated. empty - times considered exact 0 - times considered approximate 1 - times considered exact

GTFS: `shapes.txt`
Fast-Trips: `ft_input_shapes.dat`
Contains a record for shape points in a single shape that collectively describes the path transit vehicles take on their trips.

Field Name	Required	Details
GTFS: shape_id **FT:** shapeId	**GTFS:** + **FT:** +	ID that uniquely identifies a shape

GTFS: shape_pt_lat **FT:** latitude	**GTFS:** + **FT:** +	Latitude of a shape point (WGS 84)
GTFS: shape_pt_long **FT:** longitude	**GTFS:** + **FT:** +	longitude of a shape point (WGS 84)
GTFS: shape_pt_sequence **FT:** sequence	**GTFS:** + **FT:** +	Associates the latitude and longitude of a shape point sequence order along a shape
GTFS: shape_dist_traveled **FT:** distTraveled	**GTFS:** O **FT:** +	distance from the first shape point as a real distance in feet

GTFS Additional Relevant Files

Several files within the GTFS specification contain information that can be used to augment Fast-Trips, the most notable of which are the files used to describe fares.

Currently there are two GTFS files that define fares: Fare Attributes and Fare Rules, each described below.

GTFS: **`fare_attributes.txt`**
Fast-Trips: *Not Applicable*

Field Name	Required	Details
`fare_id`	Required	Contains an ID that uniquely identifies the fare class. The fare_id is dataset unique.
`price`	Required	Fare price in the unit specified by currency_type
`currency_type`	Required	Defines the currency used to pay the fare in ISO 4217 alphabetical currency codes
`payment_method`	Required	When the fare must be paid: 0 - on board 1 - before boarding

`transfers`	Required	Number of transfers permitted on this fare: 0 - none 1 - one 2 - two (empty) - unlimited
`transfer_duration`	Optional	Length of time in seconds before transfer expires. Omit or leave empty if they do not.

GTFS: **`fare_rules.txt`**
Fast-Trips: *Not Applicable*
Specifies how fares in the fare attributes file apply to an itinerary by O/D station, zones, or route.

Field Name	Required	Details
`fare_id`	Required	Unique identifier to fare class in fare attributes file
`route_id`	Optional	Associates a fare ID with a route ID from the routes file. If multiple route have the same attributes, create a row for each route.
`origin_id`	Optional	Origin fare zone ID, referenced from the stops file. If several origin IDs have the same fare attributes, create a row for each origin ID.
`destination_id`	Optional	Destination fare zone ID, referenced from the stops file. If several destination IDs have the same fare attributes, create a row for each destination ID.
`contains_id`	Optional	Associates a fare iD with a zone ID from the stops file and is associated with itineraries that pass through the contains_id zone.

Issues with current specification

There are a variety of issues with the current Fast-Trips specification that we would like to address with the revision:

(1) there is an unnecessary deviation from GTFS file and variables names and GTFS file formats. This can be remedied by adapting Fast-Trips to read in GTFS files directly and then create additional files with additional information and variables as needed.

(2) where possible and where it does not conflict with the GTFS specification, strings rather than integer codes should be used in order to facilitate legibility and increase the likelihood that errors are caught.

(3) there are additional variables that would be required in order to use them in route choice path-finding specifications. These include station attributes (i.e. bike and car parking), vehicle attributes (i.e. seated and standing capacity), fare rules, reliability, and additional network mode names.

(4) general functionality related to park and ride lots needs to be addressed. GTFS currently ignores issues of access and Fast-Trips' solution should have the flavor of GTFS.

(5) fare variables are currently ignored in fast trips, but need to be included.

Proposed Fast-Trips Transit Network Specification

This section considers the current Fast-Trips format and the overall objectives for a Fast-Trips transit network specification and makes specific recommendations for changes. Recommendations are discussed for the overall required files and format, transit service, fares, and access and non-transit links.

Required Files and Format

The proposed Fast-Trips network specification addresses one of the issues with the existing network specification by adopting plain GTFS as the primary format and supplementing the information available in GTFS with additional files. This will allow the existing multitude of GTFS file readers already available to read the network information without breaking. The two files (i.e. `routes.txt` and `routes_ft.txt`) can be joined together by a unique identifier within Fast-Trips. Accordingly, variables that were previously optional to Fast-Trips but mandatory in GTFS will become mandatory in Fast-Trips. Since there are both mandatory and optional variables, the Fast-Trips software should check for the presence of all the mandatory variables and read in the optional variables as kwargs or similar. All Unique IDs should be cast to string when appropriate (e.g. in a Pandas data frame) so that we can merge multiple GTFS datasets using a code concatenated to the ID. For example if you have transit agency a and b, unique IDs could be 1002_a and 1002_b.

For the purposes of consistency with GTFS, the following files are proposed to be required in Fast-Trips' specification:

agency.txt Shall now be a required file in Fast-Trips both because Fast-Trips should be able to accommodate multiple transit agencies, and because it is a required file in GTFS and we want to be able to reuse GTFS tools as much as possible.

calendar.txt Shall now be a required file in Fast-Trips because it is a required file in GTFS and we want to be able to reuse GTFS tools as much as possible.

The data standard in its entirety can be found in Appendix A.

Transit Service Provision

The following additional files are proposed to augment the GTFS representation transit service provision:

`routes_ft.txt`: A new table that is an extension of `routes.txt`. This table will include the following fields:

- **`route_id`**: Unique ID that links to `route_id` in `routes.txt`.
- **`proof_of_payment`**: Payment is/is not enforced via fare inspectors.
- **`fare_class`**: (Optional) It's possible that an agency has different fares for different service, e.g. Sound Transit bus and light rail, so this field will be used determine unique fare amounts/rules for fare specific service. Examples include: metro, community_transit, st_bus, st_light_rail, st_commuter_rail etc.
- **`mode`**: The purpose of this field is to enable both transit sub-mode skimming and the assignment of various parameters on in vehicle time (i.e. making the perceived in vehicle time for commuter rail less than the perceived in vehicle time for a bus). While the network mode values are flexible and adaptable to various agencies and situations, we have listed possible values to encourage inter-agency consistency. The mode choice model specifies a set of network modes that can be used for each mode choice mode based on a modal hierarchy, defined in the Fast-Trips parameters files. Mode choice modes can either be general (i.e. walk-transit, which allows the use of all transit so long as it is accessed by walking) or specific (i.e. walk-heavy_rail, which might allow the use of local bus so long as it is used to access heavy rail). Network mode definitions should have sufficient detail to be able to encapsulate the mode-choice mode definitions.
    - local_bus
    - premium_bus (e.g., Community Transit, Sound Transit, Golden Gate Transit)
    - rapid_bus (e.g., Van Ness BRT)
    - light_rail (e.g., VTA Rail, Muni Metro, Link)
    - heavy_rail (e.g., BART)
    - commuter_rail (e.g., Sounder, Caltrain)
    - regional_rail (e.g., SMART, eBART)
    - inter_regional_rail (e.g., Amtrak, ACE, Capital Corridor)
    - high_speed_rail
    - street_car (i.e. F-line, SLU)
    - ferry
    - cable_car
    - open_shuttle (i.e. Caltrain Shuttles, CPMC Shuttles)
    - employer_shuttle (i.e. Microsoft, Google, and Facebook shuttles)

vehicles_ft.txt: A new table describes transit vehicles. This table will include the following fields:

- **`vehicle_name`**: Unique identifier for vehicle name specified in `trips_ft.txt`
- **`vehicle_description`**: (Optional) A description of the vehicle type.
- **`seated_capacity`**: (Optional) If specified, will override capacity stated in `trips.txt`
- **`standing_capacity`**: (Optional) Of specified, will override capacity stated in `trips.txt`
- **`number_of_doors`**: (Optional) Required to be able to estimate dwell time by number of doors.
- **`max_speed`**: (Optional) Placeholder for future use in conjunction with DTA.
- **`vehicle_length`**: (Optional) Placeholder for future use in conjunction with DTA.
- **`vehicle_height`**: (Optional) Used in conjunction with platform height to determine level boarding.
- **`propulsion_type`**: (Optional) A potential summary variable for analyzing climate impacts.
- **`wheelchair_capacity`**: (Optional) Blank indicates that it is unknown and treated as unlimited, zero indicates that wheelchairs cannot access this vehicle. This value overrides the value in `trips.txt`.
- **`bicycle_capacity`**: (Optional) Blank indicates that it is unknown and treated as unlimited unless `trips.txt` says that it is not bicycle accessible.

trips_ft.txt: A new table that indexes `vehicles.txt` to `trips.txt` on `trip_id`. This table will include the following fields:

- **`trip_id`**: The unique trip ID
- **`vehicle_name`**: The unique vehicle name which corresponds to a valid vehicle name in `vehicles_ft.txt`.

stops_ft.txt: A new table that is indexed to `stops.txt` on `stop_id`. This table will include the following fields:

- **`stop_id`**: Unique ID that links to `stop_id` in `stops.txt`.
- **`shelter`** : (Optional) Valid entries include:
  - (blank) unknown
  - inside (e.g., bus tunnel, underground BART)
  - sheltered
  - none
- **`lighting`**: (Optional) A boolean field to indicate whether the stop has lighting.

- **`bike_parking`**: (Optional) Indicates the availability of various types of bike parking. Valid entries include:
  - none
  - standard_outside

    - standard_inside
    - lockers
    - valet (e.g., 4th Street Caltrain Station)
- **`bike_share_station`:**(Optional) A boolean to indicate the presence of a bike share station
- **`Seating`:**(Optional) A boolean to indicate the presence of seating at the station or stop. Stop-level variables will overwrite station-level.
- **`platform_height`:** (Optional) Used with vehicle height to determine level boarding
- **`level`:** (Optional) Indicates number of floors up or below street level the stop is relative to the station, and the station relative to street level.
- **`off_board_payment`:**(Optional) A boolean to indicate if there are fare gates or tagging stations before the platform.

stop_times_ft.txt: A new table that is an extension of `stop_times.txt`. These variables are all dependent on both the route and stop, which is why they are here. To use **`pay_at_station variable`** as an example: in Seattle, you can pay/tap your transit pass for some routes at some bus stops (rapid ride), but not at all stops.

- **`trip_id`:** Contains an ID that identifies a trip. This field is used to index this table to `stop_times.txt` using both `trip_id` and `stop_id`.
- **`stop_id`:** Contains an ID that identifies a stop. This field is used to index this table to `stop_times.txt` using both `trip_id` and `stop_id`.
- **`front_board_only`:** (Optional) A boolean to indicate if all doors can be used for boarding or not in order to calculate dwell times.
- **`real_time_data`**: (Optional) A boolean to indicate presence of real time data displayed, where stop level overwrites station level.
- **`reliability`:** (Optional) Not yet defined.
- **`level_boarding`:** (Optional) A boolean to indicate if this trip/stop combo have a level boarding or not. Overrides logic from platform heights.

Fare Specification

While current versions of Fast-Trips do not take into account fares, the monetary cost of transit does influence route choice and should be incorporated. Fares as specified in GTFS have most, but not all, of the flexibility needed to be able to represent the fare systems in the Puget Sound and Bay area.

fare_rules_ft.txt: A new table that is an extension of `fare_rules.txt` and will allow variation in fares across time periods in order to account for peak pricing.

- **`fare_id`**: Unique ID that links to `fare_id` in `fare_rules.txt`.
- **`fare_class`:** The name of the `fare_class` which links to `fare_class` in `routes_ft.txt`.

- **start_time**: Enables fares that fluctuate by time of day. If no time of day is specified, it is assumed that this is the base fare and that other time of days will override it.
- **end_time**: Enables fares that fluctuate by time of day. If no time of day is specified, it is assumed that this is the base fare and that other time of days will override it.

fare_attributes_ft.txt: A new table that is a substitute for `fare_attribute.txt`. In the existing GTFS specification, the one-to-one relationship between `route_id` and `fare_id` in `fare_rules.txt` precludes the ability to represent fares that vary by time of day for the same route, e.g. peak/off-peak. Our work around is to use `fare_id`, `start_time` and `end_time` in `fare_rules_ft.txt` to return fare_class, which is then used in `fare_attributes_ft.txt` to return the correct fare. The only difference between `fare_attributes_ft.txt` and `fare_attributes.txt` is that `fare_class` is used instead of `fare_id`.

- **fare_class**: Unique ID that links `fare_class` in `fare_rules_ft.txt`.
- All other fields are the same as `fare_attributes.txt`

fare_tranfer_rules.txt: A new table that describes the amount a passenger will pay when transferring from one `fare_class` to another `fare_class`.

- **from_fare_class**: The name of the `fare_class` that is associated with the *from* leg of the transfer.
- **to_fare_class**: The name of the `fare_class` that is associated with the *to* leg of the transfer.
- **reduced_rate**: True Indicates that the full fare of the *to leg* of the transfer is not charged and the transfer_cost should be used instead. False indicates that the full fare of the *to leg* is paid.
- **transfer_cost**: The cost of the transfer.

The following examples will illustrate how we will model fares using the extended network structure.

Single leg, no transfer, flat-fare:
The example below illustrates how the fare for a single leg transit trip using a service that has flat fare system. First `fare_rules.txt` is queried on `route_id`, `origin_zone` & `destination_zone` to return it's `fare_id`. In this case, Origin and destination zones have values of None, which represent cases where stops are never used in a zonal fee structure. `Fare_rules_ft` is then queried on `fare_id` and the time of departure (>= to `start_time`, <= `end_time`) to return `fare_class`. `Fare_attributes_ft.txt` is then queried on `fare_class`, and the cost of the fare is returned by the `price` field.

stops.txt

stop_id	stop_name	zone_id	...
1	14th/Mission	--	...
2	30th/Mission	--	...

routes_ft.txt

route_id	mode	fare_class	proof_of_payment
MUN14	local_bus	muni_local	1
MUN14R	local_bus	muni_local	1

fare_rules.txt

fare_id	route_id	contains_id
muni-allday	muni_local	

fare_rules_ft.txt

fare_id	fare_class	start_time	end_time
muni-allday	muni_local	000001	240000

fare_attributes_ft.txt

fare_class	price	currency_type	transfers	transfer_duration
muni_local	2.50	USD	-	5400

Two or more legs, transfer

To capture the cost of this scenario, the cost of each leg is calculated using the same method proposed for a single leg trip. We then use the *from* fare_class and the *to* fare class to get the `reduced_fare` and `transfer_cost` attributes from `transfer_rules.txt.transfer_cost` is the amount of the second fare given that `reduced_fare`, which indicates a reduced rate, is True. For example, if `reduced_rate` is True, the fare for the second leg of the trip is the amount in `tranfer_cost`. If `reduced_rate` is False, then the full fare of the second leg is paid. To indicate a free transfer, `reduced_rate` is set to True and `transfer_cost` is set to 0.

Multiple Transfers

A second transfer would work in a similar fashion, however, it is possible (unlikely ?) that the fare would have to be calculated using the *from* `fare_class` for both the first and second leg to determine which `transfer_rule` to use. For example- a rider uses the same fare_class in the first and third leg of a three leg trip. This fare_class is entitled to a free transfer (tranfer_rule = 0) when staying with the same fare_class during a transfer (e.g. a metro bus to metro bus transfer). Assuming the transfer has not expired (and this scenario is permitted), the rider is eligible for a free transfer based on the fare_class associated with the first leg of the trip, even if there is a transfer cost (transfer_rule <> 0) associated with the second and third leg.

System-wide Fare, one transfer:
The following two-leg (one transfer) trip demonstrates how a system-wide fare would be calculated using Pierce Transit as an example. First, fare_rules.txt is queried on the route_id, origin_zone and destination_zone of the first leg to return it's fare_id. In this case, Origin and destination have zones but are the same because these stops need zones for Sound Transit, our regional express bus service. Fare_rules_ft.txt is then queried on fare_id and the time of departure (>= to start_time, <= end_time) to return fare_class. Fare_attributes_ft is then queried on fare_class, and the cost of the fare is returned by the price field.

The next step is to determine the transfer rule for this particular transfer. We use the route_id of the second leg to get the fare_id which, along with departure time, is used to get fare_class from fare_rules_ft. The from_fare_class and the to_fare_class are used to get reduced_rate and transfer_cost from fare_transfer_rules.txt. In this case reduced_rate is True the and fare_cost is 0 indicating that there is no fee for the second leg of this trip.

1st leg:

routes_ft.txt

route_id	mode	fare_id	proof_of_payment
PT01	local_bus	Pierce	1

stops.txt

stop_id	stop_name	zone_id	...
1	Pacific Ave/166th St.	Tacoma	...
2	Pacific Ave & 112th St.	Seattle	...

fare_rules.txt

fare_id	route_id	origin_id	destination_id
PierceLocal	PT01	Pierce	Pierce

fare_rules_ft.txt

fare_id	fare_class	start_time	end_time
PierceLocal	PierceAllDay	000001	235959

fare_attributes.txt

fare_class	price	currency_type	payment_method	transfers
PierceAllDay	2.00	USD	1	-

2nd leg:

routes_ft.txt

route_id	mode	proof_of_payment
PT53	local_bus	1

stops.txt

stop_id	stop_name	zone_id	...
3	Pacific Ave & 112th St.	Pierce	...
4	SR 512 P&R	Pierce	...

fare_rules.txt

fare_id	route_id	contains_id
PierceLocal	PT04	

fare_rules_ft.txt

fare_id	fare_class	start_time	end_time
PierceLocal	PierceAllDay	000001	235959

fare_attributes.txt

fare_class	price	currency_type	payment_method	transfers
PierceAllDay	2.00	USD	1	-

fare_transfer_rules.txt

from_fare_class	to_fare_class	is_flat_fee	transfer_rule
PierceAllDay	PierceAllDay	False	0

Inter-Agency Fare, zonal fee structure:
The following illustrates how an inter-agency fare (one transfer, two different fare classes) would be calculated. First, `fare_rules.txt` is queried on the route_id, `origin_zone` and `destination zone` of the first leg to return it's `fare_id`. `Fare_rules_ft.txt` is then queried on `fare_id` and the time of departure (>= to `start_time`, <= end_time) to return `fare_class`. `Fare_attributes_ft.txt` is then queried on `fare_class`, and the cost of the fare is returned by the `price` field.

The next step is determine the transfer rule for this particular transfer. We use the route_id of the second leg to get the `fare_id` which can then be used to get `fare_class`. We then get the rule that applies to this transfer, which is returned by querying `fare_transfer_rules.txt` on `from_fare_class` and `to_fare_class`. In this case, the field `reduced_rate` in the returned record is False, indicating the full fare of the second leg applies. The full fare of the second leg (which is a peak fare) is returned using the same method as the first leg. If, however, `reduced_rate` was True, there would be no need to determine the standard fare of the second leg; instead the value in the `tranfer_cost` field would be used.

1st leg:
stops.txt

stop_id	stop_name	zone_id	...
1	TACOMA_DOME	Tacoma	...
2	4th Ave & Cherry	Seattle	...

fare_rules.txt

fare_id	origin_id	contains_id
ST_EXPRESS_2Z	Tacoma	0

fare_rules_ft.txt

fare_id	fare_class	start_time	end_time
ST_EXPRESS	ST_EXPRESS_2Z	000001	235959

fare_attributes_ft.txt

fare_class	price	currency_type	payment_method	transfers
ST_EXPRESS_2Z	3.40	USD	1	2

2nd leg:

stop_id	stop_name	zone_id	...
3	James St. & 3rd Ave.	Seattle	...
4	E Jefferson St & 17th Ave	Seattle	...

fare_rules.txt

fare_id	origin_id	destination_id
Metro_1Z	Seattle	Seattle

fare_rules_ft.txt

fare_id	fare_class	start_time	end_time
Metro_1Z	METRO_1Z_P	060000	085959

fare_attributes_ft.txt

fare_class	price	currency_type	payment_method	transfers
Metro_1Z_P	2.75	USD	1	-

fare_transfer_rules.txt

from_fare_class	to_fare_class	is_flat_fee	transfer_rule
ST_EXPRESS_2Z	Metro_1Z_P	False	1

Zone-Based Fares

Commuter rail frequently calculates fares based on the number of zones you traverse. This can be specified as follows.

`stops.txt`

stop_id	stop_name	zone_id	...
1	PIONEER_SQ	SEATTLE	...
2	EVERETT	EVERETT	...

`fare_rules.txt`

fare_id	origin_id	destination_id
SOUNDER_2Z	SEATTLE	EVERETT

`fare_rules_ft.txt`

fare_id	fare_class	start_time	end_time
SOUNDER_2Z	SOUNDER_2Z_ALL_DAY	000000	235959

`fare_attributes_ft.txt`

fare_class	price	currency_type	payment_method	transfers
SOUNDER_2Z_ALL_DAY	2.75	USD	1	-

OD-Based Fares

Origin-destination based fares are common on heavy rail systems, such as BART. They are a special case of zone-based fares, where every station has its own zone, and could be specified as follows.

`stops.txt`

stop_id	stop_name	zone_id	...
1	EMBARCADERO	B_EMB	...
2	FREMONT	B_FRE	...

`fare_rules.txt`

fare_id	origin_id	destination_id
B_EMB_FRE	B_EMB	B_FRE

`fare_rules_ft.txt`

fare_id	fare_class	start_time	end_time
B_EMB_FRE	B_EMB_FRE_ALL_DAY	000000	235959

`fare_attributes_ft.txt`

fare_class	price	currency_type	payment_method	transfers
B_EMB_FRE_ALL_DAY	2.75	USD	1	-

Transit Access and Transfer Specification

Per the scope of this project, each agency will keep its current demand resolution and access/egress link generation processes. It is expected that each agency will be able to generate appropriate access and egress links from TAZ centroids to transit stops.

walk_access.txt is proposed as a slight reformulation of the current Fast-Trips specification of `ft_input_access_links.dat` with additional optional fields in addition to distance to take advantage of potential variables in utility equations. We also propose the elimination of the time variable to allow for walking speed to vary based on the market segment. Additional optional variables include:

- **`elevation_gain`**: (Optional) the elevation in feet that one has to walk uphill to traverse this link.
- **`population_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`employment_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`retail_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`auto_capacity`**:(Optional) could be measured for the actual roadway, an area within ¼ mile, or other.
- **`indirectness`**: (Optional) the ratio of the manhattan distance to crow-fly distance.

transfers.txt will also include entries for links between PNR and KNR lots and stations/stops.

transfers_ft.txt contains necessary and optional information about transfer links that does not fit within the GTFS specification. We propose the elimination of the time variable (compared to the original Fast-Trips specification) to allow for walking speed to vary based on the market segment. Additional optional variables include:

- **`distance`**: represented here because it is not in GTFS format
- **`elevation_gain`**: (Optional) the elevation in feet that one has to walk uphill to traverse this link.
- **`population_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`employment_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`retail_density`**: (Optional) could be measured for the area within ¼ mile, or other.
- **`auto_capacity`**:(Optional) could be measured for the actual roadway, an area within ¼ mile, or other.
- **`indirectness`**: (Optional) the ratio of the manhattan distance to crow-fly distance.

The variety in approaches among agencies for park and rides (PNR) necessitates flexibility within Fast-Trips for how they are defined. One one end of the spectrum, park and ride choice is done completely within the network model (e.g., Travel Model One), which allows for a joint choice of transit path and park and ride lot, but must wait for an entire global iteration to determine if park and ride lots are full. On the other end, it is done completely within the ABM in order to have an accounting of the capacities and use person-based variables in the park and ride choice (e.g., DaySim), but artificially constrains the transit path-finder to a single park and ride lot when a more optimal route may exist. If enough iterations between the ABM and network model are completed, this would be fine, but it may provide more noise than is desired on an iteration-to-iteration basis in order to close the swings between iterations. A hybrid approach is to specify a handful of likely PNR lots in the demand model to feed into the network model (e.g., SF-CHAMP). We propose a solution that is flexible enough to accommodate the spectrum of approaches adopted by each of the three agencies

drive_access.txt has one entry for each park and ride or kiss and ride lot that can be accessed from each zone. For each trip, the demand file will be referenced to further constrain the park and ride entries that are available. Additional optional variables include:

- **`taz`** (integer)
- **`lot_id`** (integer) , which can be a special PNR TAZ if they exist, or a KNR node.

- **`direction`** (string) to determine if it is an access or egress link, with possible values of:
    - access
    - egress
- **`dist`** (float, miles)
- **`travel_time`** (float, minutes)
- **`cost`** (integer, cents) represents tolls and out of pocket costs for the access link (separate from parking cost, which is specified below)
- **`start_time`** (HHMMSS from midnight) This is so we can model attributes that fluctuate by time of day. If blank no time of day is specified, it is assumed that this is the base condition other time of days will override it.
- **`end_time`** (HHMMSS from midnight) This is so we can model attributes that fluctuate by time of day. If blank and no time of day is specified, it is assumed that this is the base condition other time of days will override it.

pnr.txt represents the characteristics of the park and ride lot itself and is connected to a stop or station by a transfer link in `transfers.txt`.

- **`lot_id`**: Unique identifier for park and ride lot
- **`lot_lat`**: Latitude of lot in WGS 84
- **`lot_long`**: Longitude of lot in WGS 84
- **`name`**: (Optional)
- **`capacity`**: (Optional) If not specified, assumed to be infinite
- **`overflow_capacity`**: (Optional) If not specified, assumed to be zero. This is to represent "hide and ride" or unofficial parking availability in surrounding area.
- **`hourly_cost`** (integer, cents), optional - hourly cost
- **`max_cost`** (integer, cents), optional - maximum daily cost
- **`type`** (string), with possible values of:
    - surface
    - underground
    - structure

knr.txt represents the characteristics of the kiss and ride lot itself and is connected to a stop or station by a transfer link in `transfers.txt`.

- **`lot_id`** (integer), required
- **`lot_lat`** (float), required, latitude of stop or station in WGS 84
- **`lot_long`** (float), required longitude of stop or station in WGS 84
- **`name`** (string), optional

Appendix A - Transit Network Data Standard

This Appendix describes the proposed network specification for Fast-Trips in detail, comparing it to the existing GTFS specification where applicable. Note that in all tables below, the "requirements" column is encoded as follows:

- + : Required
- O: Optional
- NA: Not Applicable

File Format

Comma-delimited text files where the first line of each file contains the field names, which are case-sensitive. Field values may not contain tabs, carriage returns or new lines.

Required Files

In the table below, green text indicates additions or changes from the existing specification.

File	Filename	GTFS	Fast-Trips
Walk access Links	`walk_access.txt`	NA	Required
Transfer Links	`transfers.txt`	Optional	Required
Transfer Links - Additional Info	`transfers_ft.txt`	NA	Implementation -dependent
Drive Access Links	`drive_access.txt`	NA	Implementation -dependent
Park and Ride Lots	`pnr.txt`	NA	Implementation -dependent
Kiss and Ride Drop-offs	`knr.txt`	NA	Implementation -dependent
Trips	`trips.txt`	Required	Required
Trips - Additional Info	`trips_ft.txt`	NA	Required
Routes	`routes.txt`	Required	Required
Routes - Additional Info	`routes_ft.txt`	NA	Required
Stops	`stops.txt`	Required	Required
Stops - Additional Info	`stops_ft.txt`	NA	Required
Stop Times	`stop_times.txt`	Required	Required

Stop Times - Additional Info	`stop_times_ft.txt`	NA	Required
Vehicles	`vehicles_ft.txt`	NA	Required
Shapes	`shapes.txt`	Optional	Optional
Agency	`agency.txt`	Required	Required
Calendar	`calendar.txt`	Required	Required
Calendar Dates (exceptions)	`calendar_dates.txt`	Optional	NA
Fare Attributes	`fare_attributes.txt`	Optional	Implementation -dependent
Fare Rules	`fare_rules.txt`	Optional	Implementation -dependent
Fare Rules - Additional Info	`fare_rules_ft.txt`	NA	Implementation -dependent
Fare Transfer Rules	`fare_transfer_rules.txt`	NA	Implementation -dependent
Frequencies	`frequencies.txt`	Optional	NA
Feed publication info	`feed_info.txt`	Optional	NA

Transit Service Provision

All files below are required unless specified otherwise.

trips.txt

Contains a record for every transit vehicle trip (i.e. the Muni 14 Local Outbound that leaves at 8:02 AM)

Field Name	Required	Details
`trip_id`	**GTFS:** + **FT:** +	ID that uniquely identifies a vehicle trip
`route_id`	**GTFS:** + **FT:** +	ID that uniquely identifies a route
`service_id`	**GTFS:** + **FT:** O	ID that uniquely identifies set of dates when service is available from calendar or calendar dates files.
`trip_headsign`	**GTFS:** O **FT:** O	Text that appears on the vehicle headsign to identify destination to passengers.
`trip_short_name`	**GTFS:** O **FT:** O	Text that appears in schedules and sign boards.
`block_id`	**GTFS:** O **FT:** O	Two or more sequential trips made using same vehicle where passenger can transfer by staying on same vehicle. block_id must be referenced by two or more trips in trips.txt.
`shape_id`	**GTFS:** O **FT:** O	Defines shape for the trip from shapes.txt file.
`wheelchair_accessible`	**GTFS:** O **FT:** O	0 - no accessibility info 1 - vehicle on this trip can accommodate at least one rider in a wheelchair 2 - no riders in wheelchairs can be accommodated on this trip
`bikes_allowed`	**GTFS:** O **FT:** O	0 - no bike accessibility info 1 - vehicle on this trip can accommodate at least one bicycle 2 - no bicycles can be accommodated on this trip
`direction_id`	**GTFS:** O **FT:** O	ID that contains a binary value that indicates the direction of the trip: 0 - travel in one direction 1 - travel in opposite direction

`trips_ft.txt`
Contains a record for every transit vehicle trip (i.e. the Muni 14 Local Outbound that leaves at 8:02 AM)

Field Name	Required	Details
`trip_id`	+	ID that uniquely identifies a vehicle trip
`vehicle_name`	+	Name of vehicle type, which is to match a description in vehicles.txt

routes.txt
Contains a record for every transit route (i.e. the Muni - 14 Local)

Field Name	Required	Details
route_id	**GTFS:** + **FT:** +	ID that uniquely identifies a route
agency_id	**GTFS:** O **FT:** O	ID that identifies the agency as specified in agency.txt
route_short_name	**GTFS:** + **FT:** +	Short name of the route
route_long_name	**GTFS:** + **FT:** +	Full name of the route
route_desc	**GTFS:** O **FT:** O	Description of route.
route_type	**GTFS:** + **FT:** +	Service type: 0 - Tram, streetcar, light rail 1 - Subway, metro 2 - Rail 3 - Bus 4 - Ferry 5 - Cable car 6 - Gondola 7 - Funicular
route_url	**GTFS:** O **FT:** O	Webpage for that route
route_color	**GTFS:** O **FT:** O	Display color for route in six-digit hexadecimal
route_text_color	**GTFS:** O **FT:** O	Color of text to be drawn on top of route color, specified in six-digit hexadecimal

routes_ft.txt

Field Name	Required	Details
route_id	+	The **route_id** field is an ID that uniquely identifies a route. This field is used to index this table to routes.txt.
mode	+	The **mode** field is used to specify the network mode of the route. Valid entries include :

		● local_bus ● premium_bus (e.g., Community Transit, Sound Transit, Golden Gate Transit) ● rapid_bus (e.g., Van Ness BRT) ● light_rail (e.g., VTA Rail, Muni Metro, Link) ● heavy_rail (e.g., BART) ● commuter_rail (e.g., Sounder, Caltrain) ● regional_rail (e.g., SMART, eBART) ● inter_regional_rail (e.g., Amtrak, ACE, Capital Corridor) ● high_speed_rail ● street_car (i.e. F-line, SLU) ● ferry ● cable_car ● open_shuttle (i.e. Caltrain Shuttles, CPMC Shuttles) ● employer_shuttle (i.e. Microsoft, Google, and Facebook shuttles)
`fare_class`	O	String. The **fare_entity** field contains a string that uniquely defines an agency and service type that has a uniform fare structure since multiple fare structures could exist within a single agency. Examples for Sound Transit could be st_light_rail, st_commuter_rail, st_bus.
`proof_of_payment`	+	Boolean. The **proof_of_payment** field contains a boolean value indicating if the route has fare enforcement through random inspection (true) or if the driver oversees payment (false).

`stops.txt`
Contains a record for every transit stop or station (i.e. Embarcadero Station)

Field Name	Required	Details
stop_id	**GTFS:** + **FT:** +	ID that uniquely identifies a stop or station
stop_code	**GTFS:** O **FT:** O	Short text or a number that identifies the stop for passengers (i.e. EMB)
stop_name	**GTFS:** + **FT:** +	Name of a stop or station
stop_desc	**GTFS:** O **FT:** O	Description of a stop or station
stop_lat	**GTFS:** + **FT:** +	Latitude of stop or station in WGS 84
stop_lon	**GTFS:** + **FT:** +	Longitude of stop or station in WGS 84
zone_id	**GTFS:** O **FT:** O	Defines a fare zone for the stop ID. Required if you want to provide fare information in fare_rules.txt that uses zones.
location_type	**GTFS:** O **FT:** O	Identifies whether this stop is a stop or station. If nothing is specified or is blank, it is assumed it is a stop. Stations can have different properties from stops. 0 or blank - stop 1 - station (contains one or more stops)
parent_station	**GTFS:** O **FT:** O	For stops inside stations, identifies station associated with the stop. Stops.txt must also contain a row where this stop id is assigned a location type 1.
stop_timezone	**GTFS:** O **FT:** O	Contains timezone where stop or station is located. If omitted, stop assumed to be located in timezone in agency.txt.
wheelchair_boarding	**GTFS:** O **FT:** O	Identifies whether wheelchair boardings are possible from the specified stop or station. 0 - no information 1 - some vehicles can be boarded by wheelchair 2 - wheelchair boarding not possible If a stop is part of a station: 0 - will inherit from parent station, if specified.

		1 - there is an accessible path from outisde station to stop 2 - no accessible path to specific stop

`stops_ft.txt`
Contains a record for every transit stop or station (i.e. Embarcadero Station)

Field Name	Required	Details
`stop_id`	+	ID that uniquely identifies a station. This field is used to index this table to stops.txt.
`shelter`	O	String. Contains a description of the the shelter facility at the station. Valid entries include: ● blank / unknown ● inside (i.e. underground) ● sheltered ● none
`lighting`	O	Boolean. Indicates the presence or absence of lighting.
`bike_ parking`	O	Describes the bike parking facilities at the station. Valid entries include: ● none ● standard_outside ● standard_inside ● lockers ● valet (i.e. bike station)
`bike_share_station`	O	Boolean. Indicates the presence of bike share location.
`seating`	O	Boolean. Indicates the presence of seating at the station. Stop-level overrides station-level.
`platform_height`	O	Float, inches. Used with vehicle height to determine level boarding.
`level`	O	Integer, floors from street level. Indicates how far up or below street level the stop is relative to the station and the station relative to the street level.
`off_board_payment`	O	Boolean. INdicates if there are fare gates or tagging stations before the platform. Can be overridden by stop_times _ft value for specific service.

`stop_times.txt`
Contains a record for every scheduled stop within a trip and route (i.e. the time when the Muni 14 Local Outbound that left at 8:02 gets to 24th St. and Mission St)

Field Name	Required	Details
`trip_id`	**GTFS:** + **FT:** +	ID that uniquely identifies trip
`arrival_time`	**GTFS:** + **FT:** +	Arrival time at a specific stop for a specific trip on a route in HHMMSS format measured from midnight. For trips that span multiple dates, the time should be entered as a value greater than 2400000
`departure_time`	**GTFS:** O **FT:** +	Departure time at a specific stop for a specific trip on a route in HHMMSS format measured from midnight. For trips that span multiple dates, the time should be entered as a value greater than 2400000
`stop_id`	**GTFS:** + **FT:** +	ID that uniquely identifies a stop
`stop_sequence`	**GTFS:** + **FT:** +	Sequence number on a specific stop within a trip. The first stop sequence is 1 and subsequent stops in the trip are sequentially numbered.
`stop_headsign`	**GTFS:** O **FT:** O	Text that appears on sign that identifies the trips destination to passengers. use this field to override default headsign when it changes at stops.
`pickup_type`	**GTFS:** O **FT:** O	0/default - regular pickup 1 - no pickup available 2 - must phone agency 3 - must coordinate with driver
`drop_off_type`	**GTFS:** O **FT:** O	0/default - regular drop off 1 - no drop off available 2 - must phone agency 3 - must coordinate with driver
`shape_dist_traveled`	**GTFS:** O **FT:** O	Positions a stop as a distance from the first shape point in units that are used in this field in shapes.txt
`timepoint`	**GTFS:** O **FT:** O	Indicates if specified arrival and departure times for a stop are strictly adhered to by the transit vehicle or if they are approximate and/or interpolated. empty - times considered exact 0 - times considered approximate

		1 - times considered exact

`stop_times_ft.txt`
Contains a record for every scheduled stop within a trip and route (i.e. the time when the Muni 14 Local Outbound that left at 8:02 gets to 24th St. and Mission St)

Field Name	Required	Details
`trip_id`	+	ID that identifies a trip. This field is used to index this table to stop_times.txt using both trip_id and stop_id.
`stop_id`	+	ID that identifies a stop. This field is used to index this table to stop_times.txt using both trip_id and stop_id.
`pay_at_station`	O	Boolean. Indicates if the passenger can pay at the stop. Boolean.
`real_time_data`	O	Boolean. Indicates presences of real time data displayed while waiting. Stop level overrides station level.
`front_board_only`	O	Boolean. Indicates the boarding can only be made through the front doors.
`reliability`	O	Not yet defined.
`level_boarding`	O	Boolean. The **level_boarding** field indicates if the platform and the bus are level. Overrides logic from platform height.

`shapes.txt` - *Optional*
Contains a record for shape points in a single shape that collectively describes the path transit vehicles take on their trips.

Field Name	Required	Details
`shape_id`	**GTFS:** + **FT:** +	ID that uniquely identifies a shape
`shape_pt_lat`	**GTFS:** + **FT:** +	Latitude of a shape point (WGS 84)
`shape_pt_long`	**GTFS:** + **FT:** +	Longitude of a shape point (WGS 84)
`shape_pt_sequence`	**GTFS:** + **FT:** +	Associates the latitude and longitude of a shape point sequence order along a shape
`shape_dist_traveled`	**GTFS:** O **FT:** O	Distance from the first shape point as a real distance in feet

vehicles.txt
Contains a record for each vehicle type

Field Name	Required	Details
vehicle_ name	+	String. Uniquely identifies a vehicle type.
vehicle_ description	O	String. Description of the vehicle. For example, 'metro_articulated'.
seated_ capacity	O	Integer. Total seated capacity per vehicle. If specified, this will override capacity from trip file.
standing_ capacity	O	Integer. Number of standing riders at capacity. If specified, this will override capacity from trip file.
number_ of_doors	O	Integer. Number of doors.
max_ speed	O	Float. Maximum speed of the vehicle in mph.
vehicle_ length	O	Float. Length of the vehicle in feet.
platform_ height	O	Float. Height of the platform in inches.
propulsion_ type	O	String. Name of the propulsion type. Possible values include: ● diesel, ● bio-diesel, ● CNG, ● diesel-hybrid, ● electric.
wheelchair_capacity	O	Integer, overrides value in trip file. Blank indicates that it is unknown and is treated as infinite. Zero indicates that wheelchairs cannot access this vehicle.
bicycle_capacity	O	Integer. Blank indicates that it is unknown and is treated as infinite unless the trip file says that it is not bicycle accessible.

Fare Definition

`fare_attributes.txt` - *Implementation Specific Requirements*

Field Name	Required	Details
`fare_id`	GTFS: + FT: +	Contains an ID that uniquely identifies the fare class. The fare_id is dataset unique.
`price`	GTFS: + FT: +	Fare price in the unit specified by currency_type
`currency_type`	GTFS: + FT: +	Defines the currency used to pay the fare in ISO 4217 alphabetical currency codes
`payment_method`	GTFS: + FT: +	When the fare must be paid: 0 - on board 1 - before boarding
`transfers`	GTFS: + FT: +	Number of transfers permitted on this fare: 0 - none 1 - one 2 - two (empty) - unlimited
`transfer_duration`	GTFS: O FT: O	Length of time in seconds before transfer expires. Omit or leave empty if they do not.

`fare_attributes_ft.txt` - *Implementation Specific Requirements*
The one-to-one relationship between route_id and fare_id in fare_rules.txt precludes the ability to represent fares that vary by time of day for the same route, e.g. peak/off-peak. Our work around is to use fare_id, start_time and end_time in fare_rules_ft.txt to return fare_class, which is then used in fare_attributes_ft.txt to return the correct fare.

Field Name	Required	Details
`fare_class`	GTFS: + FT: +	Contains an ID that uniquely identifies the fare class. The fare_class is dataset unique.
`price`	GTFS: + FT: +	Fare price in the unit specified by currency_type
`currency_type`	GTFS: + FT: +	Defines the currency used to pay the fare in ISO 4217 alphabetical currency codes

payment_method	GTFS: + FT: +	When the fare must be paid: 0 - on board 1 - before boarding
transfers	GTFS: + FT: +	Number of transfers permitted on this fare: 0 - none 1 - one 2 - two (empty) - unlimited
transfer_duration	GTFS: O FT: O	Length of time in seconds before transfer expires. Omit or leave empty if they do not.

fare_rules.txt - *Implementation Specific Requirements*
Specifies how fares in the fare attributes file apply to an itinerary by O/D station, zones, or route.

Field Name	Required	Details
fare_id	+	Unique identifier to fare class in fare attributes file
route_id	O	Associates a fare ID with a route ID from the routes file. If multiple route have the same attributes, create a row for each route.
origin_id	O	Origin fare zone ID, referenced from the stops file. If several origin IDs have the same fare attributes, create a row for each origin ID.
desitnation_id	O	Destination fare zone ID, referenced from the stops file. If several destination IDs have the same fare attributes, create a row for each destination ID.
contains_id	O	Associates a fare iD with a zone ID from the stops file and is associated with itineraries that pass through the contains_id zone.

fare_rules_ft.txt - *Implementation Specific Requirements*

Field Name	Required	Details
fare_id	+	An ID that links to fare_id in fare_rules.txt.
fare_class	+	Contains the name of the fare_class that links to the same attribute in routes_ext.txt.

`start_time`	+	(HHMMSS from midnight) This is so we can model fares that fluctuate by time of day. If no time of day is specified, it is assumed that this is the base fare and that other time of days will override it.
`end_time`	+	(HHMMSS from midnight) This is so we can model fares that fluctuate by time of day. If no time of day is specified, it is assumed that this is the base fare and that other time of days will override it.

`fare_transfer_rules.txt` - *Implementation Specific Requirements*

Field Name	Required	Details
`from_fare_class`	+	An ID that identifies the fare_class that the passenger is coming from.
`to_fare_class`	+	An ID that identifies the fare_class that the passenger is going to.
`is_flat_fee`	+	A flag that indicates if a flat fare is paid or the fare is a percentage of the full fare for that leg. If True, a flat fee is expected in the tranfer_rule field, e.g. 1.50. Otherwise the value in tranfer_rule should range from 0-1.
`transfer_ rule`	+	If is_flat_fee is true, value should be a monetary amount, e.g 1.50. Otherwise, this field contains the amount, from 0-1, that will be multiplied to the fare of the transfer leg to return the amount of the transfer.

Access Files

`walk_access.txt` - *required by Fast-Trips, not a GTFS format*
Contains a record for each feasible stop <--> zone pair

Field Name	Required	Details
`taz`	+	Zone ID
`stop_id`	+	Stop ID
`dist`	+	Walking distance in miles between TAZ and stop
`elevation_gain`	O	integer, feet. The elevation walked along this link.
`population_density`	O	float, employees per square mile per mile. Can be measured for the area within ¼ mile, or other.
`retail_density`	O	float, employees per square mile per mile. Can be measured for the area within ¼ mile, or other.
`auto_capacity`	O	float, vehicles per hour per mile. Can be measured for the actual roadway, an area within ¼ mile, or other.
`indirectness`	O	float, ratio. Measured as the ratio of the manhattan distance to crow-fly distance.

`transfers.txt` - *required by Fast-Trips, Optional for GTFS*
Transfers are links traversed on foot. They are created for each stop to all other stops that are considered accessible during a transfer from an individual stop as well as between stops accessible from PNR and KNR lots.

Contains a record for each feasible stop <--> stop pair in addition to PNR <-->stops and KNR ←> stops.

Field Name	Required	Details
`from_stop_id`	**GTFS:** + **FT:** +	From stop ID
`to_stop_id`	**GTFS:** + **FT:** +	To stop ID
`transfer_type`	**GTFS:** + **FT:** +	Specifies the type of connection: 0 / Empty - a recommended transfer point 1 - timed transfer between two routes 2 - requires a minimum amount of time, specified by min_transfer_time 3 - transfers not possible between routes
`min_transfer_time`	**GTFS:** + **FT:** +	When a connection between routes requires an amount of time between arrival and departure (transfer_type=2), this field defines the amount of time that must be available for a typical rider - in seconds.

`transfers_ft.txt` - *Implementation Specific Requirements*

Field Name	Required	Details
`from_stop_id`	+	From stop ID
`to_stop_id`	+	To stop ID
`dist`	+	float, miles
`elevation_gain`	O	integer, feet. The elevation walked along this link.
`population_density`	O	float, employees per square mile per mile. Can be measured for the area within ¼ mile, or other.

`retail_density`	O	float, employees per square mile per mile. Can be measured for the area within ¼ mile, or other.
`auto_capacity`	O	float, vehicles per hour per mile. Can be measured for the actual roadway, an area within ¼ mile, or other.
`indirectness`	O	float, ratio. Measured as the ratio of the manhattan distance to crow-fly distance.

`drive_access.txt` - *Implementation Specific Requirements*

Field Name	Required	Details
`taz`	+	TAZ ID
`lot_id`	+	Lot ID
`direction`	+	String. Can have values of: ● access ● egress
`dist`	+	float, miles.
`cost`	+	integer, cents.
`travel_time`	+	float, minutes.
`start_time`	+	HHMMSS from midnight. If blank, it is assumed that this is the base condition and other time of days will override it.
`end_time`	+	HHMMSS from midnight. If blank, it is assumed that this is the base condition and other time of days will override it.

`pnr.txt` - *Implementation Specific Requirements*

Field Name	Required	Details
`lot_id`	+	Lot ID
`lot_lat`	+	Float. Lot location latitude.
`lot_long`	+	Float. Lot location longitude
`name`	O	String.
`capacity`	O	Integer. Represents number of parking spaces at park and ride. If not specified, assumed to be infinite
`overflow_capacity`	O	Integer. Represents "hide and ride" or unofficial parking availability in surrounding area. If not specified, assumed to be zero.
`hourly_cost`	O	Integer, cents. Hourly cost to park.
`max_cost`	O	Integer, cents. Maximum daily cost to park.
`type`	O	String, with possible values of: ● surface ● underground ● structure

`knr.txt` - *tion Specific Requirements*required by Fast-Trips if have kiss and ride access Represents the characteristics of the kiss and ride lot itself and is connected to a stop or station by a transfer link in `transfers.txt`.

Field Name	Required	Details
`lot_id`	+	Lot ID
`lot_lat`	+	Float. Lot location latitude.
`lot_long`	+	Float. Lot location longitude
`name`	O	String.

Other Required Files

The following files are required because they are required in GTFS and we do not want to break any GTFS reader's expectations.

agency.txt

Field Name	Required	Details
agency_id	O	ID that uniquely identifies the transit agency.
agency_name	+	Contains full name of transit agency.
agency_url	+	String. Fully qualified URL of agency.
agency_timezone	+	String. List of valid values: httpp//en.wikipedia.org/wiki/List_of_tz_database_time_zones
agency_lang	O	String. Two-letter, ISO 639-1 code for primary language used by agency. Case-insensitive (both EN and en are accepted)
agency_phone	O	String. Phone number for agency.
agency_fare_url	O	String. URL of where fares are defined.

calendar.txt

Field Name	Required	Details
service_id	O	ID that uniquely identifies the transit agency.
monday	+	0 or 1. Binary value on whether this service pattern is available on Mondays.
tuesday	+	0 or 1. Binary value on whether this service pattern is available on Tuesdays.
wednesday	+	0 or 1. Binary value on whether this service pattern is available on Wednesdays.
thursday	+	0 or 1. Binary value on whether this service pattern is available on Thursdays.

friday	+	0 or 1. Binary value on whether this service pattern is available on Fridays.
saturday	+	0 or 1. Binary value on whether this service pattern is available on Saturdays.
sunday	+	0 or 1. Binary value on whether this service pattern is available on Sundays.
start_date	+	String, YYYYMMDD. Start date for service.
end_date	+	String, YYYMMDD. End date for service.