

GTFS-Pathways

bit.ly/gtfs-pathways

This document is a collaborative working tool for a GTFS extension proposal. As a living document, content may evolve based on community feedback. Therefore, any implementation may have to be updated. The status color scheme [A1] tracks the stages at which the contents of the GTFS extension proposal evolve. If you have any questions, please reach out to specifications@mobilitydata.org.

GitHub pull requests [here](#)

Goal

The main goal is to **route somebody in a wheelchair through a station**. The secondary goal is to **route anybody through a station**, with a focus on step-free accessibility. (For information, the GTFS-Vehicles proposal shares the same goal but for vehicles, see the [GTFS-Vehicles](#) document).

For any questions, please email hello@mobilitydata.org.

Extensions Overview¹

This proposal sliced the possible needs into four distinct new extensions, so that every agency could use only the portion which fits its need. They all build on top of the GTFS-Stations & GTFS-Entrances extensions, which are already part of the specification.

The [GTFS-Pathways](#) & [GTFS-Levels](#) describes the **inside of a station schematically**, between its entrances/exits and its platforms, to route riders using trip planning software (e.g. routing somebody in wheelchair from entrance to platform, or somebody with a stroller with as few steps as possible).

The [GTFS-PathwayEvolutions](#) describes the **planned evolution** of a station through time (e.g. elevator closed for construction, or opening schedule of an entrance linked to a mall).

The [GTFS-PathwayUpdates](#) describes the **real-time evolution** of the station (e.g. an elevator is down, or entrances are closed because of a demonstration).

Document Overview

For didactic purposes, this document starts with the dry specification of the extension (listed above), then contains , and then some [Open Questions](#).

- [\[Example A\] Step-by-step Modeling](#)

¹ Two other extensions, *GTFS-StationGeometries* & *GTFS-PathwayGeometries*, have been drafted to do indoor mapping. They are now on ice, and have been removed from this file, since we may decide to use existing [indoor mapping data formats](#) for now. Their content is still available for reference [here](#).

- [\[Example B\] Split Station](#)
- [\[Example C\] Pathways Modeling](#)
- [\[Example D\] Elevator Modeling](#)
- [\[Example E\] Platform Modeling](#)
- [\[Example F\] Wayfinding Modeling](#)
- [\[Example Z\] Cluster Modeling](#)

Extensions proposals

GTFS-Pathways

Goals

The GTFS-Pathways extension describes schematically the physical layout of a station.

Requirements

Requires the **GTFS-Levels** to describe levels of elevators (pathways with pathway_mode=5).

Overview

The GTFS-Pathways extension uses a graph representation, with nodes (called locations) and edges (called pathways).

To go from the entrance (which is a node represented as a location with location_type=2) to a platform (which is a node represented as a location with location_type=0), the rider will go through walkways, fare gates, stairs, etc (which are edges represented as pathways). The proposal also adds another type of location, a generic one, to represent, for example, a walkway crossing from which different walkways can be taken.

The list of intra-station pathways specified in pathways.txt must be exhaustive for a specific station. The scope of the station is the area under the responsibility of the public transit agency, since it's the area on which it has the power to close sections, to decide construction or opening hours, etc.

Files extended or added

File Name	State	Defines
stops.txt	Extended	Adds another generic location type.
pathways.txt	Added	Provides information about the pathways in the stations.
traversal_times.txt	Added	Provides information about variable traversal times that can occur e.g. at security checks or customs.

Files definitions

stops.txt (file extended)

Field Name	Details
stop_name	<p>(Text, Conditionally Required)</p> <p>When the location is a boarding area (location_type=4), the stop_name, if defined, should contain the name of the boarding area as displayed by the agency. It could be just one letter (like on some European intercity railway stations), or text like "Wheelchair boarding area" (NYC's Subway) or "Head of short trains" (Paris' RER).</p> <p>[Extends the existing definition]</p> <p>Conditionally Required:</p> <ul style="list-style-type: none"> - Required for locations which are stops (location_type=0), stations (location_type=1), or entrances/exits (location_type=2). - Optional for locations which are generic nodes (location_type=3) or boarding areas (location_type=4).
stop_lat	<p>(Latitude, Conditionally Required)</p> <p>[Extends the existing definition]</p> <p>Conditionally Required:</p> <ul style="list-style-type: none"> - Required for locations which are stops (location_type=0), stations (location_type=1), or entrances/exits (location_type=2). - Optional for locations which are generic nodes (location_type=3) or boarding areas (location_type=4).
stop_lon	<p>(Longitude, Conditionally Required)</p> <p>[Extends the existing definition]</p> <p>Conditionally Required:</p> <ul style="list-style-type: none"> - Required for locations which are stops (location_type=0), stations (location_type=1), or entrances/exits (location_type=2). - Optional for locations which are generic nodes (location_type=3) or boarding areas (location_type=4).
location_type	<p>(Enum, Optional) The location_type field defines the type of the location:</p> <ul style="list-style-type: none"> - 0 (or blank): Stop (or 'Platform'). A location where passengers board or disembark from a transit vehicle. Is called a "platform" when defined within a parent_station. - 1: Station. A physical structure or area that contains one or more platforms. - 2: Station Entrance/Exit. A location where passengers can enter or exit a

	<p>station from the street. The entrance/exit should also specify a <code>parent_station</code> value referencing the <code>stop_id</code> of the parent station for the entrance. If an entrance/exit belongs to multiple stations, it will be linked by pathways to both, and the data provider can either pick one station as parent, or but no parent station at all.</p> <ul style="list-style-type: none"> - 3: Generic Node. A location within a station, not matching any other <code>location_type</code>, which can be used to link together pathways defined in <code>pathways.txt</code>. - 4: Boarding Area. A specific location on a platform, where passengers can board and/or alight vehicles.
<code>parent_station</code>	<p>(Enum, Conditionally Required) The <code>parent_station</code> field defines the hierarchy between the different locations defined in <code>stops.txt</code>. It contains the ID of the parent location, as follows:</p> <ul style="list-style-type: none"> - Stop/platform (<code>location_type=0</code>): the <code>parent_station</code> field contains the ID of a station. - Station (<code>location_type=1</code>): this field must be empty. - Entrance/exit or generic node (<code>location_type=2</code> or <code>3</code>): the <code>parent_station</code> field contains the ID of a station (<code>location_type=1</code>) - Boarding Area (<code>location_type=4</code>): the <code>parent_station</code> field contains the ID of a platform (<code>location_type=0</code>). <p><i>Conditionally Required:</i> Required for <code>location_type</code> 2, 3 & 4. Optional for <code>location_type</code> 0. Forbidden: <code>location_type</code> 1.</p>
<code>boarding_edge</code>	<p>(Enum, Optional) The <code>boarding_edge</code> field specifies if the platform or boarding area is protected:</p> <ul style="list-style-type: none"> - 0 or empty: No information - 1: Open platform - 2: Manual platform doors - 3: Automated platform doors <p>This field is recommended for <code>location_type=4</code> (boarding area).</p>
<code>boarding_height</code>	<p>(Float, Optional) The <code>stop_height</code> field specifies the height in meters between the floor (defined as the surface on which the wheels are rolling) and the edge of the platform or quay or sidewalk.</p> <p>This value would be negative for suspended vehicles (on rail or cable).</p> <p>The difference between this value and the <code>vehicle.floor_height</code> aims to give information about a possible step between the platform and the vehicle. The height of the step being:</p> $\text{height} = \text{floor_height} - \text{boarding_height}$

boarding_distance	<p>(Float, Optional) The <code>boarding_distance</code> field specifies the distance in meters between the edge of the platform or quay or sidewalk to the middle of the track.</p> <p>This value should be provided only if there is a guidance system (e.g. rail, cable), and should not be provided if each driver can stop at a different distance from the edge of the platform.</p> <p>This field, used with <code>vehicle.vehicle_width</code>, aims to give information about a possible gap between the platform and the vehicle. The size of the gap being:</p> $\text{gap} = \text{boarding_distance} - \text{vehicle_width}/2$
-------------------	--

pathways.txt (file extended)

Warning: Pathways must be exhaustive in a station. As a consequence, as soon as one platform (as stop), entrance or node belonging to a station has a pathway linked to it, the station is assumed to have exhaustive description of its pathways. Therefore, the following common sense rules apply:

- No dangling location: If any location within a station has a pathway, then all locations should have pathways (except for those platforms that have boarding areas).
- No locked platforms: Each platform must be connected to at least one entrance via some chain of pathways. There are very few stations in real life where you cannot go outside.
- No pathways for a platform with boarding areas: A platform that has boarding areas is treated as a parent object, not a point. It may not have pathways assigned. All pathways should be for boarding areas.

Field Name	Details
pathway_id	<p>(ID, Required) The <code>pathway_id</code> field contains an ID that uniquely identifies the pathway. The <code>pathway_id</code> is used by systems as an internal identifier of this record (e.g., primary key in database), and therefore the <code>pathway_id</code> must be dataset unique.</p> <p>Different pathways can go from the same <code>from_stop_id</code> to the same <code>to_stop_id</code>. For example, this happens when two escalators are side by side in opposite direction, or when a stair is nearby and elevator and both go from the same place to the same place.</p>
from_stop_id	<p>(ID, Required) The <code>from_stop_id</code> field defines the location at which the pathway begins. It contains a <code>stop_id</code> that identifies a platform, entrance/exit, generic node or boarding area from the <code>stops.txt</code> file.</p>
to_stop_id	<p>(ID, Required) The <code>to_stop_id</code> field defines the location at which the pathway ends. It contains a <code>stop_id</code> that identifies a platform, entrance/exit, generic node or boarding area from the <code>stops.txt</code> file.</p>
pathway_mode	<p>(Enum, Required) The <code>pathway_mode</code> field specifies the type of pathway</p>

	<p>between the specified (from_stop_id, to_stop_id) pair. Valid values for this field are:</p> <p>1: walkway 2: stairs 3: moving sidewalk/travelator 4: escalator 5: elevator</p> <p>6: fare gate (or payment gate): A pathway that crosses into an area of the station where a proof of payment is required (usually via a physical payment gate). Fare gates may either separate paid areas of the station from unpaid ones, or separate different payment areas within the same station from each other. This information can be used to avoid routing passengers through stations using shortcuts that would require passengers to make unnecessary payments, like directing a passenger to walk through a subway platform to reach a busway.</p> <p>7: exit gate: Indicates a pathway exiting an area where proof-of-payment is required into an area where proof-of-payment is no longer required.</p> <p>8: Control: A pathway on which the passenger will have to wait a variable amount of time. Examples: ticket checkpoint, ID verification, immigration crossing, custom crossing.</p>
is_bidirectional	<p>(Enum, Required) The is_bidirectional field indicates in which direction the pathway can be used:</p> <ul style="list-style-type: none"> - 0: Unidirectional pathway, it can only be used from from_stop_id to to_stop_id. - 1: Bidirectional pathway, it can be used in the two directions. <p>Fare gates (pathway_mode=6) and exit gates (pathway_mode=7) cannot be bidirectional.</p>
cover_type	<p>(Enum, Optional) The cover_type field specifies the protection that the pathway has against the weather:</p> <ul style="list-style-type: none"> - 0 or empty: No information - 1: Outside pathway - 2: Covered pathway (with a roof; protection against vertical rain) - 3: Inside pathway (with roof & walls; protection against rain & wind)
length	<p>(Non-negative Float, Optional) The length field specifies the horizontal length in meters of the pathway from the origin location (defined in from_stop_id) to the destination location (defined in to_stop_id).</p> <p>This field is recommended for pathway_mode 1 (walkway), 6 (fare gate) and 7 (exit gate).</p>

mechanical_length	<p>(Non-negative Float, Optional) The <code>mechanical_length</code> field specifies the horizontal length in meters of a travelator.</p> <p>This field is recommended for <code>pathway_mode=3</code> (travelator).</p> <p>When the travelator is functioning, the data consumer should use <code>traversal_time</code> and ignore <code>mechanical_length</code>. But during travelator downtime, the data consumer should consider the travelator as walkway, and use <code>mechanical_length</code>.</p>
traversal_time	<p>(Positive Integer, Optional) The <code>traversal_time</code> field specifies the average time needed to walk through the pathway. Valid values for this field are:</p> <ul style="list-style-type: none"> - (empty): unknown travel time. - A positive integer: number of seconds needed to traverse this pathway on foot. <p>This field is recommended for mechanical pathways, i.e. pathways with <code>pathway_mode</code> 3 (moving sidewalk), 4 (escalator) and 5 (elevator).</p>
traversal_time_id	<p>(ID from <code>traversal_times.txt</code>, Optional) Override the <code>traversal_time</code> on specific timeframes.</p> <p>This field is recommended for Controls (i.e. pathways with <code>pathway_mode=9</code>).</p>
stair_count	<p>(Non-null Integer, Optional) The <code>stair_count</code> field specified the number of stairs of the pathway.</p> <p>Best Practices: one could use the approximation of 1 floor = 15 stairs (or 12 for an escalator) to generate approximative values.</p> <p>A positive <code>stair_count</code> implies that the rider walks up from <code>from_stop_id</code> to <code>to_stop_id</code>. And a negative <code>stair_count</code> implies that the rider walks down from <code>from_stop_id</code> to <code>to_stop_id</code>.</p> <p>This field is recommended for stairs & escalator pathways, i.e. pathways with <code>pathway_mode</code> 2 (stairs).</p>
max_stair_flight	<p>(Non-null Integer, Optional) The <code>max_steps_flight</code> field specified the maximum number of steps in a row of the pathway (between landings).</p>
mechanical_stair_count	<p>(Integer, Optional) The <code>mechanical_stair_count</code> field specified the number of stairs of an escalator. Valid values for this field are:</p> <ul style="list-style-type: none"> - (empty): unknown number of stairs. - An integer: number of stairs of the escalator. <p>This field is recommended for <code>pathway_mode</code> 4 (escalator).</p>

	<p>When the escalator is functioning, the data consumer should use <code>traversal_time</code> and ignore <code>mechanical_stair_count</code>. But during escalator downtime, the data consumer should consider the escalator as stairs, and use <code>mechanical_stair_count</code>.</p>
<code>max_slope</code>	<p>(Float, Optional) The <code>max_slope</code> field specified the maximum slope ratio of the pathway. Valid values for this field are:</p> <ul style="list-style-type: none"> - 0 or (empty): no slope. - A float: slope ratio of the pathway, positive for upwards, negative for downwards <p>This field should be used only with the <code>pathway_type</code> 1 (walkway) and 3 (travelator).</p> <p>Example: In the US, 0.083 (also written 8.3%) is the maximum slope ratio for a hand-propelled wheelchair, which means an increase of 0.083m (so 8.3cm) for each 1m.</p>
<code>max_cross_slope</code>	<p>(Float, Optional) The <code>max_cross_slope</code> field specified the maximum cross slope ratio of the pathway. Valid values for this field are:</p> <ul style="list-style-type: none"> - 0 or (empty): no slope. - A float: slope ratio of the pathway, positive for upwards, negative for downwards <p>This field should be used only with the <code>pathway_type</code> 1 (walkway) and 3 (travelator).</p>
<code>min_width</code>	<p>(Positive Float, Optional) The <code>min_width</code> field contains the minimum width of the pathway in meters.</p> <p>This field is highly recommended if the minimum width is less than 1 meter.</p>
<code>pathway_name</code>	<p>(Text, Optional) The <code>pathway_name</code> field contains the name of the pathway. Please use a name that people will understand in the local and tourist vernacular, if any.</p>
<code>pathway_code</code>	<p>(Text, Optional) The <code>pathway_code</code> field contains short text or a number that uniquely identifies the pathway for passengers. The <code>pathway_code</code> can be the same as <code>pathway_id</code> if it is passenger-facing. This field should be left blank for pathway without a code presented to passengers.</p> <p>Example: Elevator "A"</p>
<code>signposted_as</code>	<p>(Text, Optional) The <code>signposted_as</code> field is an exact string of text from physical signage visible to transit riders. The string can be used to provide text directions to users, such as 'follow signs to '. The language text should appear in this field exactly how it is printed on the signs - it should not be translated.</p>

<code>reversed_signposted_as</code>	(Text, Optional) Same than the <code>signposted_as</code> field, but when the pathways is used backward, i.e. from the <code>to_stop_id</code> to the <code>from_stop_id</code> .
<code>instructions</code>	(Text, Optional) The <code>instructions</code> field indicates instructions for the pathway taken from the <code>from_stop_id</code> to <code>to_stop_id</code> (e.g. "Go downstairs to the station concourse").
<code>tts_instructions</code>	(Text, Optional) Same as the <code>instructions</code> field, but intended for reading aloud by text-to-speech screen readers and voice assistants.
<code>reversed_instructions</code>	(Text, Optional) Same as the <code>instructions</code> field, but when the pathways is used backward, i.e. from the <code>to_stop_id</code> to the <code>from_stop_id</code> .
<code>tts_reversed_instructions</code>	(Text, Optional) Same as the <code>reversed_instructions</code> field, but intended for reading aloud by text-to-speech screen readers and voice assistants.
<code>visually_impaired_instructions</code>	<p>(Text, Optional) Step-by-step navigation instructions describing a pathway taken from <code>from_stop_id</code> to <code>to_stop_id</code>. This field is intended for visually-impaired riders and must therefore be readable by text-to-speech software. See "Text-to-speech field" in the Term Definitions.</p> <p><i>Example: "Turn left and go down the corridor."</i></p>
<code>reversed_visually_impaired_instructions</code>	(Text, Optional) Functions in the same way as <code>visually_impaired_instructions</code> but when the pathway is taken in the opposite direction (i.e., from <code>to_stop_id</code> to <code>from_stop_id</code>).
<code>wheelchair_assistance</code>	<p>(Enum, Optional) The assistance field indicates if using this pathway requires additional human assistance.</p> <ul style="list-style-type: none"> - 0: This pathway do not require assistance - 1: Using this pathway requires assistance from the staff without prior notice (e.g. wheelchair/stroller door near turnstiles, ramp to install) - 2: Using this pathway requires assistance from staff with prior notice
<code>wheelchair_assistance_phone</code>	(Phone number, Optional) Indicates the phone number to call to notice about assistance needs
<code>tactile_strip</code>	<p>(Enum, Optional)</p> <ul style="list-style-type: none"> - 0 or empty: The pathway has no guiding strips. - 1: The pathway has guiding strips for visually impaired and blind people
<code>report_phone</code>	(Phone number, Optional) The <code>report_phone</code> field indicates the phone number to call to report a downtime for the pathways (e.g. broken elevator).
<code>report_url</code>	(URL, Optional) The <code>report_url</code> field indicates the URL on which one can report a downtime for the pathways (e.g. broken elevator).
<code>commands_max_</code>	(Non-negative Float, Optional) The <code>commands_max_height</code> field indicates the

height	<p>maximum height of the command in meters. This field is recommended if the pathway requires some action from the rider.</p> <p>Examples: fare gate validation zone, button to automatically open a door, button to call an elevator, button to choose the level in the elevator.</p>
manual_activation	<p>(Enum, Optional) Defines if the pathways requires a maneuvering action like pushing or pulling to pass through it:</p> <ul style="list-style-type: none"> - 0 or empty: no information - 1: Do not requires manual activation - 2: Requires manual activation <p>This can apply for example to elevators, fare gates, exit gates, walkway with manual, automatic or power-assisted door.</p>

traversal_times.txt (file added)

Field Name	Details
traversal_time_id	(ID, Required) Identify the traversal time. A traversal time can have multiple timeframes defined by multiple rows in <code>traversal_times.txt</code> sharing the same <code>traversal_time_id</code> (e.g. to define different waiting time through the day). Those timeframes cannot overlap.
service_id	(ID from calendar.txt, Optional) Defines the days on which this timeframe applies. If not provided, will be assumed to be everyday.
start_time	(Time, Required) Beginning of the timeframe.
end_time	(Time, Required) End of the timeframe.
mean_traversal_time	(Non-negative integer, Required) Average traversal time (in minutes) for the passenger to go through the pathway.
safe_traversal_time	<p>(Non-negative integer, Required) Safe traversal time (in minutes) for the passenger to go through the pathway.</p> <p>“Safe” here means there is a strong likelihood that the traversal time will be less than this value. As rule of thumb, the traversal time should be less than this value at least 95% of the times.</p>

GTFS-Levels

Goals

Describe the different levels of the station. Is mostly useful with the GTFS-Pathways extension, and is required for elevator (`pathway_mode=5`) to ask the user to take the elevator to the “Mezzanine” or the “Platform” level.

Requirements

No requirement.

Files extended or added

File Name	State	Defines
<code>stops.txt</code>	Extended	Adds links to the levels.
<code>levels.txt</code>	Added	Provides information about the levels of the locations.

Files definitions

`stops.txt` (file extended)

Field Name	Required	Details
<code>level_id</code>	Optional	The <code>level_id</code> field defines the level for the location. The same level can be used by multiple stations located far away from each other.

`levels.txt` (file added)

Field Name	Details
<code>level_id</code>	(ID, Required) Id of the level that can be referenced from <code>stops.txt</code>
<code>level_index</code>	(Float, Required) Numeric index of the level that indicates relative position of this level in relation to other levels (levels with higher indices are assumed to be located above levels with lower indices). Ground level should have index 0, with levels above ground indicated by positive indices and levels below ground by negative indices.

	Any entrance or exit (location_type=1) with a level_index different than 0 will be assumed to not be on the street grid (they can link to a mall, another public transit network, a university...).
level_name	(Text, Optional) Optional name of the level (that matches level lettering/numbering used inside the building or the station). Is useful for elevator routing (e.g. "take the elevator to level "Mezzanine" or "Platforms" or "-1").
elevation	(Float, Optional) Elevation above ground, in meters (with negative values indicating underground locations)

GTFS-PathwayEvolutions

Goals

The GTFS-PathwayEvolutions extension describes the evolution of pathways, like planned closures of pathways (e.g. long-term construction, opening hours of an entrance via a mall, opening of a new subway entrance) or a reversing of an elevator in evening peak hours.

Requirements

Requires the GTFS-Pathways.

Files extended or added

File Name	State	Defines
pathway_evolutions.txt	Added	Provides information about the evolutions of the pathways in the stations.

Files definitions

pathway_evolutions.txt (file added)

Field Name	Details
pathway_id	(ID, Required) The pathway_id field contains an ID that uniquely identifies a pathways. This value is referenced from the pathways.txt file.
service_id	(ID, Required) The service_id fields contains an ID that uniquely identifies a set of dates when this pathway is closed on the timeframe defined below. This value is referenced from the calendar.txt or calendar_dates.txt file.
start_time	(Time, Required) The start_time and end_time fields specify the timeframe during which the pathway evolution applies.

	<p>Please note that outside of the timeframes explicitly defined in this file, the pathways can be assumed to be open & functional during transit operation hours.</p> <p>The time is measured from "noon minus 12h" (effectively midnight, except for days on which daylight savings time changes occur) at the beginning of the service day. For times occurring after midnight, enter the time as a value greater than 24:00:00 in HH:MM:SS local time for the day on which the trip schedule begins. E.g. 25:35:00.</p>
end_time	(Time, Required) See start_time details.
is_closed	(Enum, Optional) The is_closed field defines if a pathway is closed: <ul style="list-style-type: none"> - 0 or (empty): pathway is open - 1: pathway is closed
direction	(Enum, Optional) The direction field allows to overwrite the is_bidirectional value of the pathways. It has three states (instead of 2 for is_bidirectional) to allow to provide information on escalator or travelator changing direction during the day. <ul style="list-style-type: none"> - 0: Unidirectional forward - 1: Bidirectional - 2: Unidirectional backward

GTFS-PathwayUpdates

Goals

The GTFS-PathwayUpdates extension is the real-time version of the GTFS-PathwayEvolutions extension. It allows to describe unplanned information about closure of pathways (e.g. an elevator downtime, a pathway is flooded).

If used with the GTFS-PathwayEvolutions, this extension will override any information provided in pathway_closures.txt.

Requirements

Requires the **GTFS-Pathways** extension.

Entities extended or added

Entity Name	State	Defines
FeedEntity	Extended	Protobuf feed to update the GTFS-Pathways, contains a FeedMessage, containing a FeedEntity, containing StationUpdate entities.

StationUpdate	Added	Data about the realtime pathways status.
PathwayDescriptor	Added	A descriptor that identifies an instance of a GTFS pathway.
PathwayStatus	Added	The status of a pathway.
PathwayDirection	Added	The direction of a pathway.

Entities definitions

message FeedEntity (entity extended)

Field Name	Type	Required	Cardinality	Description
station_update	Station Update	Conditionally required	One	Data about the realtime pathways status. At least one of the fields trip_update, vehicle, alert or station_update must be provided - not all these fields can be empty.

message StationUpdate (entity added)

Field Name	Type	Required	Cardinality	Description
pathway	Pathway Descriptor	Required	Many	The Pathways that this message applies to. There can be at most one StationUpdate entity for each pathway record in pathways.txt. If there is no StationUpdate for a pathway listed in pathways.txt, it means there is no real-time status available. It does not mean that the pathway can be used.
status	Pathway Status	Required	One	The status of the pathways of this message.
alert_id	string	Optional	One	Id of the entity of the alert giving information about this downtime.
active_period	TimeRange	Optional	Many	Time when the update should be applied to the pathways. If missing, the StationUpdate will be applied as long as it appears in the feed. If multiple ranges are given, the StationUpdate will be applied during all of them.
direction	Pathway Direction	Optional	One	The direction field enables overwriting the is_bidirectional value of the pathways. It has three states to enable

				providing information on an escalator or travelator changing directions during the day: FORWARD, BIDIRECTIONAL and BACKWARD.
--	--	--	--	--

message PathwayDescriptor (entity added)

Field Name	Type	Required	Cardinality	Description
pathway_id	string	Required	One	Must be the same as in pathways.txt in the corresponding GTFS feed.

enum PathwayStatus (entity added)

Values	Description
0: UNKNOWN	Status of the pathway is unknown.
1: OPERATIONAL	Pathway is fully operational for all users.
2: OPERATIONAL WITH LIMITATIONS	Pathway should be used with care, or maybe unusable for some users. See text of alerts for more information (e.g. snow, ice, no light).
3: NOT_RUNNING	(Allowed only for travelator or escalator) The travelator or escalator is stopped but not closed like a walkway, and the escalator behave like a stair.
4: CLOSED	Pathway is closed for all users.

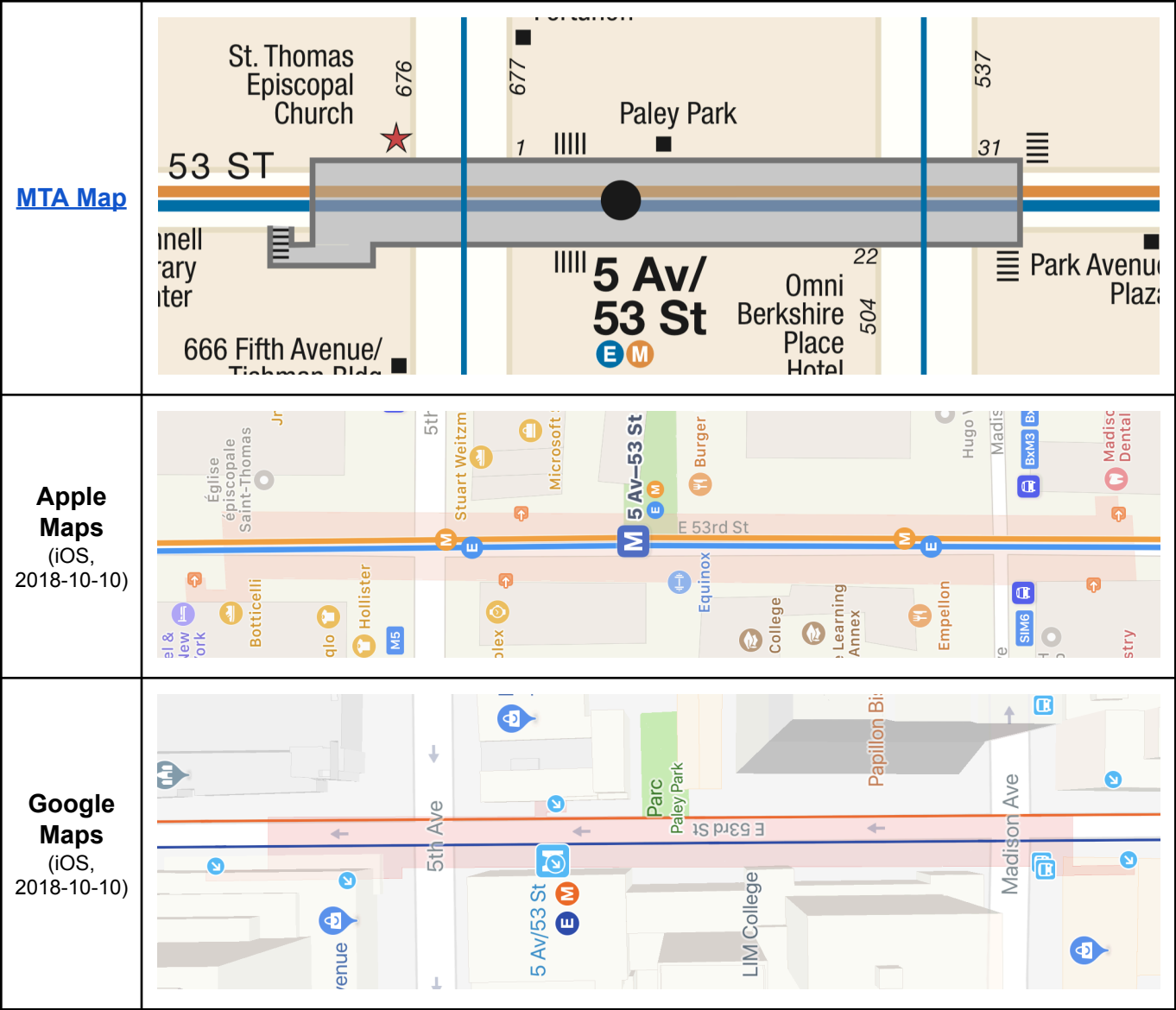
enum PathwayDirection (entity added)

Values	Description
0: UNCHANGED	(Default)
1: FORWARD	
2: BIDIRECTIONAL	
3: BACKWARD	

Examples

[Example A] Step-by-step Modeling [↑](#)

Let's start with a simple subway station so see how the modelization works. The station is 5th Avenue & 53rd Street (E,M) in NYC (US-NY). Below is the representation of this subway station as displayed by the MTA, and by two transit apps:

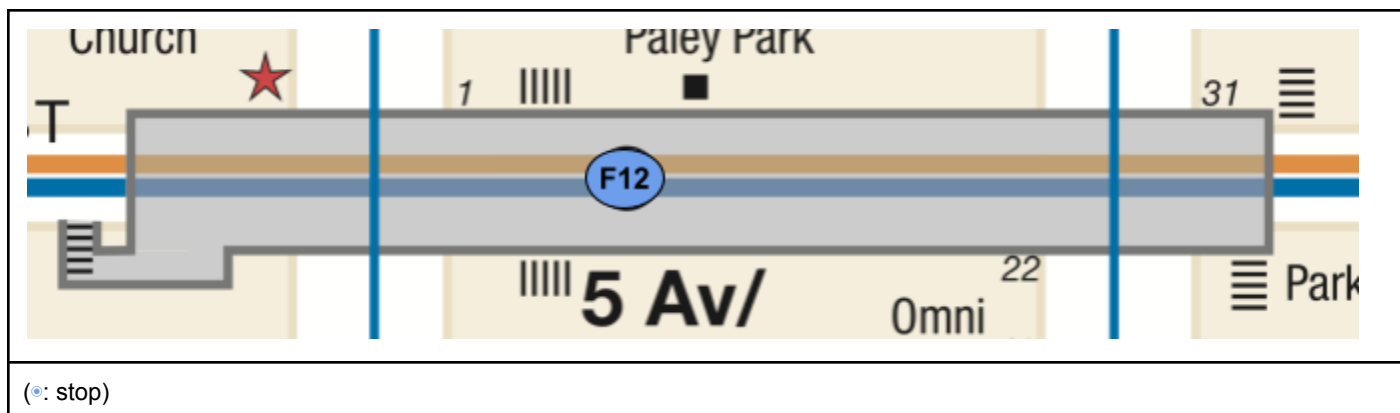


Core GTFS

With just the minimum required to make a valid GTFS, this station would be represented as just one stop:

	stop_id,stop_name,stop_lat,stop_lon
•	F12,5 Av/53 St,40.760167,-73.975224
(•: stop)	

Which can be represented by one dot:



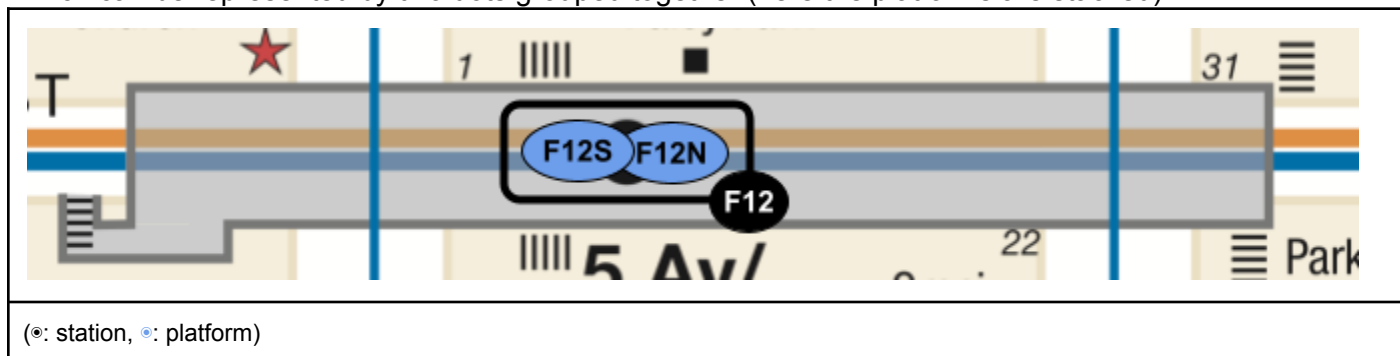
Core GTFS & GTFS-Stations

Using GTFS-Stations, it would be modeled as 2 platforms (represented by stops) and 1 station (this is the state of the current NY MTA GTFS) (new data in bold):

	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
●	F12,5 Av/53 St,40.760167,-73.975224,1,
●	F12S,5 Av/53 St,40.760167,-73.975224,0,F12
●	F12N,5 Av/53 St,40.760167,-73.975224,0,F12

(●: station, ●: platform)

Which can be represented by two dots grouped together (here the platforms are stacked):



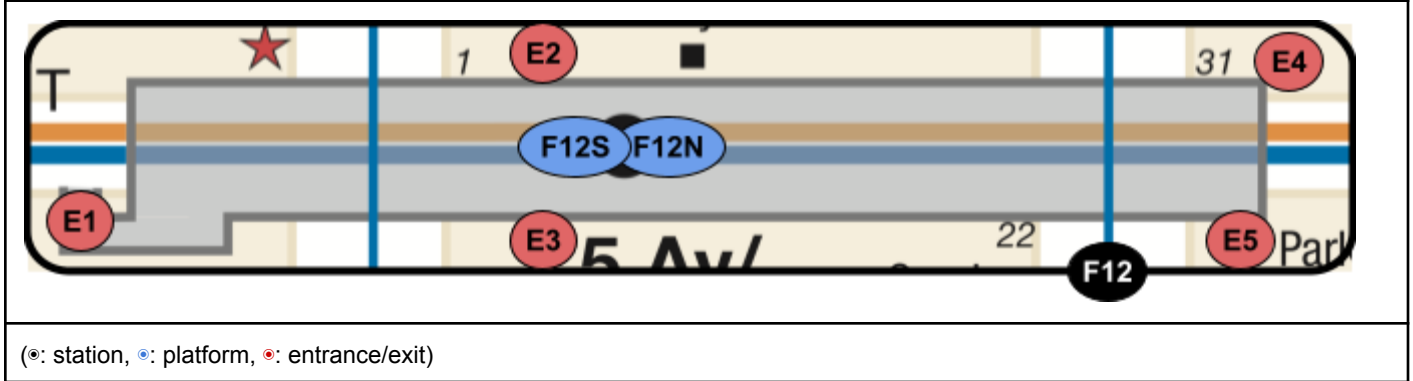
Core GTFS, GTFS-Stations & GTFS-Entrances

Using GTFS-Stations & GTFS-Entrances, it would be modeled as 2 platforms, 1 station and 5 entrances/exits (new data in bold):

	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
●	F12,5 Av/53 St,40.760167,-73.975224,1,
●	E1,5 Av/53 St SW,40.760474,-73.976099,2,F12
●	E2,5 Av/53 St NE,40.76035,-73.97546,2,F12
●	E3,5 Av/53 St SE,40.760212,-73.975512,2,F12
●	E4,Madison/53 St NE,40.759612,-73.973731,2,F12
●	E5,Madison/53 St SE,40.759491,-73.973820,2,F12
●	F12S,5 Av/53 St,40.760167,-73.975224,0,F12

<ul style="list-style-type: none"> F12N, 5 Av/53 St, 40.760167, -73.975224, 0, F12
(⦿: station, ⦿: platform, ⦿: entrance/exit)

This is the best description we can do with the current GTFS specification.



We see that we don't know if each entrance gives access to each platform. Furthermore there are long escalators linking the mezzanine to the platforms. If I know that a given escalator is broken, which entrance should I use?

Core GTFS, GTFS-Stations, GTFS-Entrances & GTFS-Pathways

Let's start by drawing this time, we'll get to the data later. First, have a look at this drawing of the station to get a sense of the real world situation.

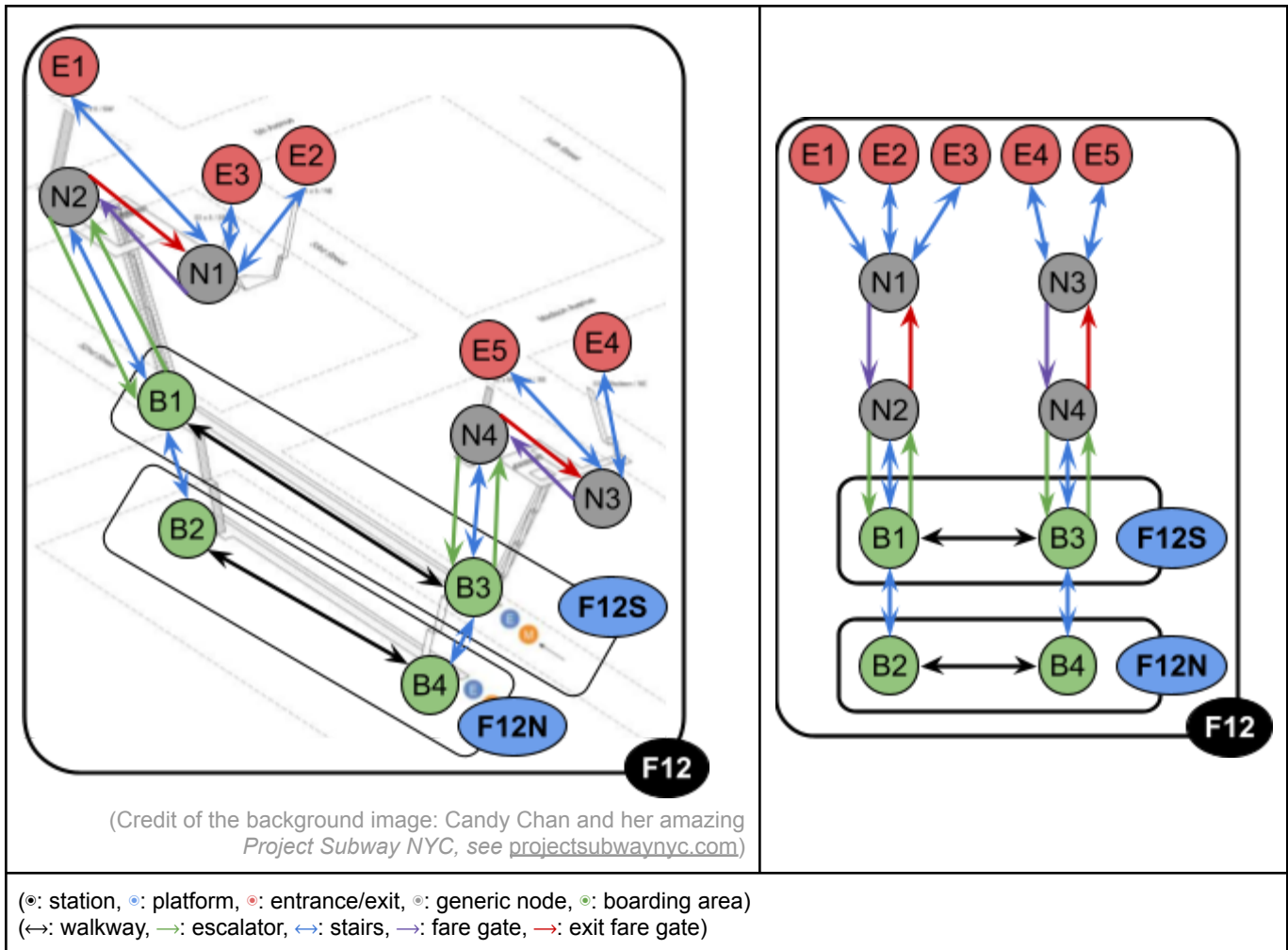
We see that:

The **three western entrances (E1, E2 & E3)** are linked with stairs to the western fare gate, which is linked with escalator and stairs to the **Southbound platform (S)**, which is linked with stairs to the **Northbound platform (N)**.

The **two eastern entrances (E4 & E5)** are linked with stairs to the eastern fare gate, which is linked with escalators and stairs to the **Southbound platform (S)**, which is linked with stairs to the **Northbound platform (N)**.

(Credit of the image: Candy Chan and her amazing *Project Subway NYC*, see projectsubwaynyc.com)

Using GTFS-Pathways, we get a graph, represented below on the 3D map & in a logical layout:



We see now that if the escalator N4->B3 is broken, the rider should avoid using the entrances E4 & E5, and instead use entrance E3 (if he/she is on the south sidewalk) or E2 (if she/he is on the north sidewalk).

Please note that GTFS-Pathways isn't meant to do indoor mapping. Each turning point of each walkway isn't described.

The **minimum** data for this station would be (new data in bold):

stops.txt	
stop_id	stop_name,stop_lat,stop_lon,location_type,parent_station
● F12	5 Av/53 St,40.760167,-73.975224,1,
● E1	5 Av/53 St SW,40.760474,-73.976099,2,F12
● E2	5 Av/53 St NE,40.76035,-73.97546,2,F12
● E3	5 Av/53 St SE,40.760212,-73.975512,2,F12
● E4	Madison/53 St NE,40.759612,-73.973731,2,F12
● E5	Madison/53 St SE,40.759491,-73.973820,2,F12
● N1	,,3,F12
● N2	,,3,F12

<ul style="list-style-type: none"> ● N3,,,,3,F12 ● N4,,,,3,F12 ● F12N,5 Av/53 St,40.760167,-73.975224,0,F12 ● B2,,,,4,F12N ● B4,,,,4,F12N ● F12S,5 Av/53 St,40.760167,-73.975224,0,F12 ● B1,,,,4,F12S ● B3,,,,4,F12S
(●: station, ●: platform, ●: entrance/exit, ●: generic node, ●: boarding area)

pathways.txt	
	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional ↔ E1N1,E1,N1,2,1 ↔ E2N1,E2,N1,2,1 ↔ E3N1,E3,N1,2,1 → N1-N2,N1,N2,6,0 → N2-N1,N2,N1,7,0 ↔ N2N3,N2,N3,2,1 → N2-B1,N2,B1,4,0 → B1-N2,B1,N2,4,0 ↔ B1B2,B1,B2,2,1 ↔ B1B3,B1,B3,1,1 ↔ B2B4,B2,B4,1,1 ↔ E4N3,E4,N3,2,1 ↔ E5N3,E5,N3,2,1 → N3-N4,N3,N4,6,1 → N4-N3,N4,N3,7,1 ↔ N4B3,N4,B3,2,1 → N4-B3,N4,B3,4,0 → B3-N4,B3,N4,4,0 ↔ B3B4,B3,B4,2,1
(↔: walkway, →: escalator, ↔: stairs, →: fare gate, →: exit fare gate)	

The **recommended** data for this station, using GTFS-Pathways would be the following (with extra spaces to increase the readability) (new data in bold).

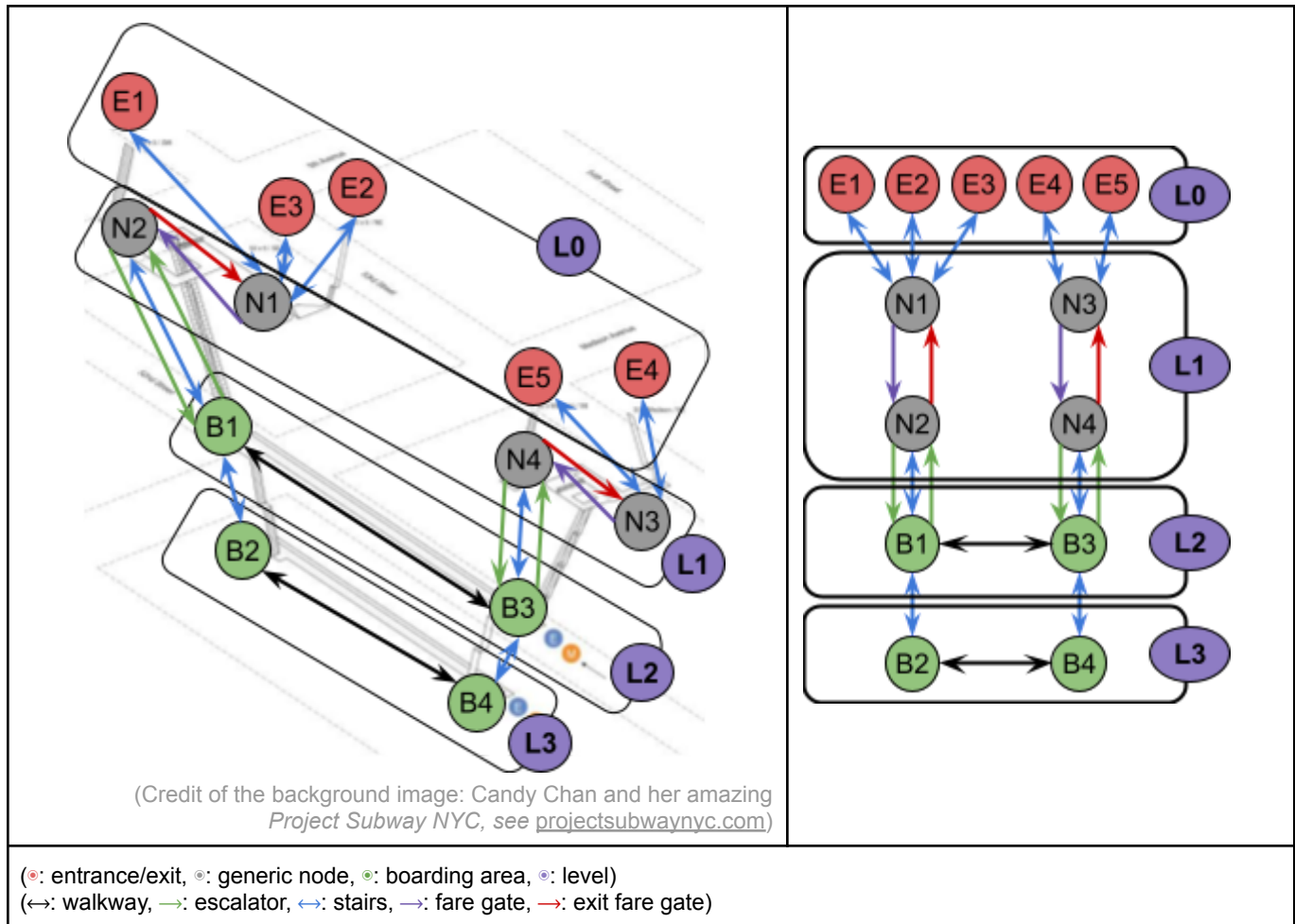
stops.txt	
	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station ● F12,5 Av/53 St,40.760167,-73.975224,1, ● E1,5 Av/53 St SW,40.760474,-73.976099,2,F12 ● E2,5 Av/53 St NE,40.76035,-73.97546,2,F12

●	E3,5 Av/53 St SE,40.760212,-73.975512,2,F12
●	E4,Madison/53 St NE,40.759612,-73.973731,2,F12
●	E5,Madison/53 St SE,40.759491,-73.973820,2,F12
●	N1,,40.760457,-73.975912,3,F12
●	N2,,40.760531,-73.976111,3,F12
●	N3,,40.759746,-73.974203,3,F12
●	N4,,40.759679,-73.974064,3,F12
●	F12S,5 Av/53 St,40.760167,-73.975224,0,F12
●	B1,,40.759746,-73.974203,4,F12S
●	B3,,40.759828,-73.974442,4,F12S
●	F12N,5 Av/53 St,40.760167,-73.975224,0,F12
●	B2,,40.760457,-73.975912,4,F12N
●	B4,,40.760375,-73.975729,4,F12N
(●: station, ●: platform, ●: entrance/exit, ●: generic node, ●: boarding area)	

pathways.txt	
	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,cover_type,length,max_slope,min_width,stair_count,signposted_as,reversed_signposted_as
↔	E1N1,E1,N1,2,1,3, 50,0,3 ,30,E&M,Exit: 5 Av/53 St SW
↔	E2N1,E2,N1,2,1,3,100,0,3 ,60,E&M,Exit: 5 Av/53 St NE
↔	E3N1,E3,N1,2,1,3,100,0,3 ,60,E&M,Exit: 5 Av/53 St SE
→	N1-N2,N1,N2,6,1,3, 1,0,0.8, ,E&M,
→	N2-N1,N2,N1,7,1,3, 1,0,0.8, ,Exit: 5 Av/53 St,
↔	N2N3,N2,N3,2,1,3, 3,0,3 ,60,E&M,Exit: 5 Av/53 St
→	N2-B1,N2,B1,4,0,3, 3,0,0.8,30,E&M,Exit: 5 Av/53 St
→	B1-N2,B1,N2,4,0,3, 3,0,0.8,30,Exit: 5 Av/53 St,E&M
↔	B1B2,B1,B2,2,1,3, 2,0,3 ,30,E&M Northbound,E&M Southbound & Exit: 5 Av/53 St
↔	B1B3,B1,B3,1,1,3,200,0,3 , ,Exit: Madison/53 St,Exit: 5 Av/53 St
↔	B2B4,B2,B4,1,1,3,200,0,3 , ,Exit: Madison/53 St,Exit: 5 Av/53 St
↔	E4N3,E4,N3,2,1,3, 50,0,3 ,30,E&M,Exit: Madison/53 St SE
↔	E5N3,E5,N3,2,1,3, 50,0,3 ,30,E&M,Exit: Madison/53 St NE
→	N3-N4,N3,N4,6,1,3, 1,0,0.8, ,E&M,
→	N4-N3,N4,N3,7,1,3, 1,0,0.8, ,Exit: Madison/53 St,
↔	N4B3,N4,B3,2,1,3, 3,0,3 ,60,E&M,Exit: Madison/53 St
→	N4-B3,N4,B3,4,0,3, 3,0,0.8,30,E&M,Exit: Madison/53 St
→	B3-N4,B3,N4,4,0,3, 3,0,0.8,30,Exit: Madison/53 St,E&M
↔	B3B4,B3,B4,2,1,3, 2,0,3 ,30,E&M Northbound,Exit: Madison/53 St
(↔: walkway, →: escalator, ↔: stairs, →: fare gate, →: exit fare gate)	

Core GTFS, GTFS-Stations, -Entrances, -Pathways & -Levels

The GTFS-Levels extension is mainly useful when there is an elevator, but also helps us to sort the station by layer. GTFS-Levels add information on each location (i.e. on each entrance, generic node and boarding area) to group them by level. Here is how they would be grouped on this station:



In the data, this would be represented by adding a `levels.txt` file and an extra column in `stops.txt` (new data in bold):

`levels.txt`

```
level_id,level_index,level_name,elevation
L0,0,Street,0
L1,-1,Mezzanine,-6
L2,-2,Southbound,-18
L3,-3,Northbound,-24
```

stops.txt	
	stop_id,level_id,stop_name,stop_lat,stop_lon,location_type,parent_station
•	F12,,5 Av/53 St,40.760167,-73.975224,1,
•	E1,L0,5 Av/53 St SW,40.760474,-73.976099,2,F12
•	E2,L0,5 Av/53 St NE,40.76035,-73.97546,2,F12
•	E3,L0,5 Av/53 St SE,40.760212,-73.975512,2,F12
•	E4,L0,Madison/53 St NE,40.759612,-73.973731,2,F12
•	E5,L0,Madison/53 St SE,40.759491,-73.973820,2,F12
•	N1,L1,,,40.760457,-73.975912,3,F12
•	N2,L1,,,40.760531,-73.976111,3,F12
•	N3,L1,,,40.759746,-73.974203,3,F12
•	N4,L1,,,40.759679,-73.974064,3,F12
•	F12S,,5 Av/53 St,40.760167,-73.975224,0,F12
•	B1,L2,,,40.759746,-73.974203,4,F12S
•	B3,L2,,,40.759828,-73.974442,4,F12S
•	F12N,,5 Av/53 St,40.760167,-73.975224,0,F12
•	B2,L3,,,40.760457,-73.975912,4,F12N
•	B4,L3,,,40.760375,-73.975729,4,F12N
(•: station, •: platform, •: entrance/exit, •: generic node, •: boarding area)	

Core GTFS, -Stations, -Entrances, -Pathways, -Levels & -PathwayEvolutions

For the sake of the example, let's say that half the elevators are closed for repair, and that the two remaining ones are from platform to exit all day, except in evening peak hour when they are from entrance to platform (since this is a downtown station).

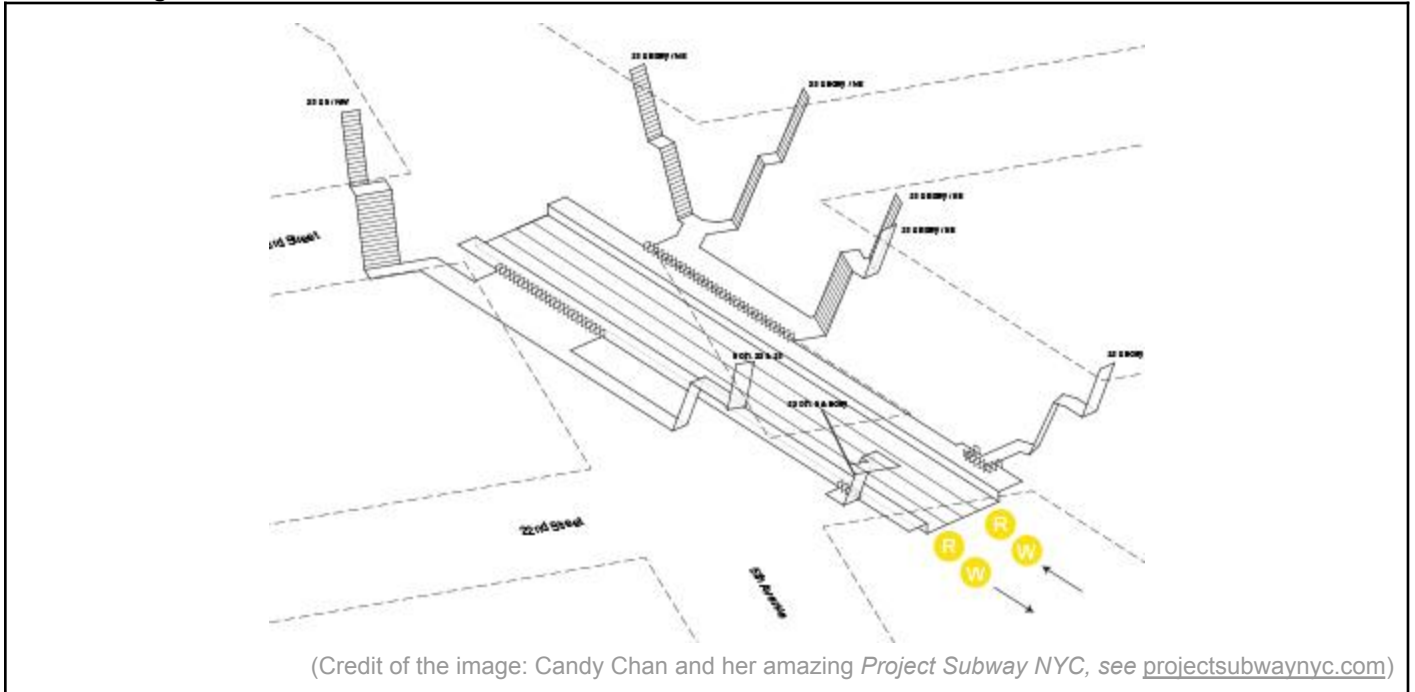
pathway_evolutions.txt	
	pathway_id,service_id,start_time,end_time,is_closed,direction
x	N2-B1,si1,,,1,
x←	B3-N4,si1,,,1,
←	B1-N2,si1,16:00:00,19:00:00,,2
←	N4-B3,si1,00:00:00,16:00:00,,2
	N4-B3,si1,19:00:00,24:00:00,,2
(x: closed pathway, ←: reversed pathway)	

Explanations:

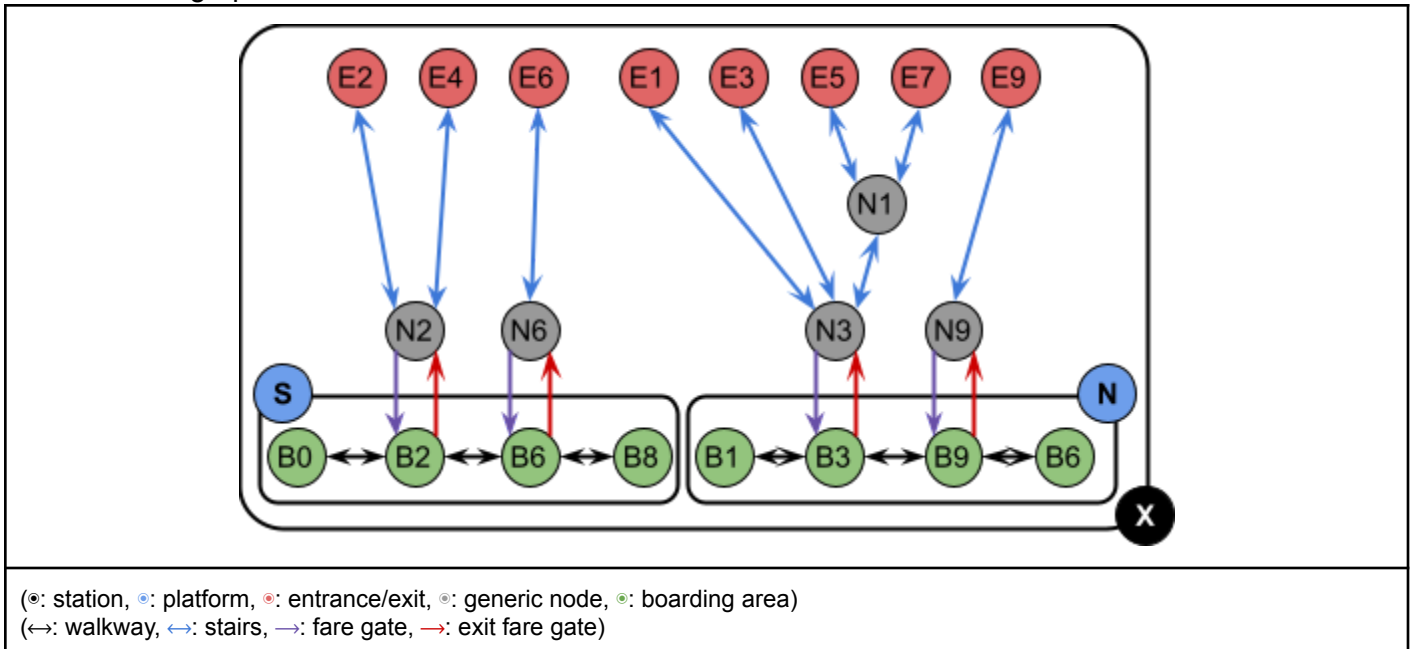
- Service with id si1 is said to describe the duration of the construction.
- B1-N2 is already from platform to exit, so it just has to be reversed during evening peak hour (so 6PM to 7PM).
- N4-B3 is already from entrance to platform, so it has to be reversed all the time except during evening peak hour (so midnight to 4PM, and from 7PM to midnight).

[Example B] Split Station

In this station (23rd Street NRW NYC (US-NY)), you have to exit to change direction. The station layout is the following:



Therefore the graph will be disconnected:


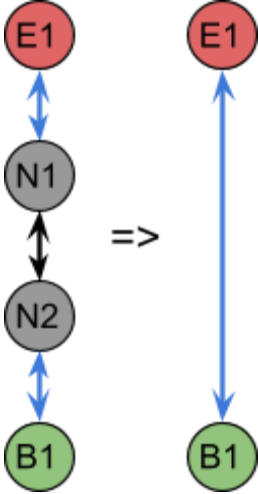


[Example C] Pathways Merging

All the examples given below are guidelines not rules. You are always allowed to increase the granularity of your modeling.

Successive stairs **should** be merged

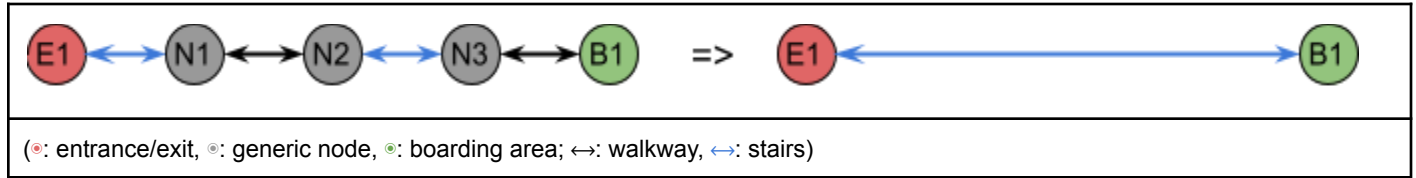
Successive stairs and rest-area without intersection should be merged.

	
<p>[Source for image: https://www.flickr.com/photos/stevenpisano/21625349979]</p>	<p>(⊙: entrance/exit, ○: generic node, ●: boarding area; ↔: walkway, ↔: stairs)</p>

pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length,s tair_count			
<p>⊙↔⊙ ○↔○ ○↔⊙ ○↔●</p>	<p>E1N1,E1,N1,2,1,0,15 N1N2,N1,N2,1,1,1, N2N3,N2,N3,2,1,0,15 N3B1,N3,B1,1,1,5</p>	<p>=></p>	<p>⊙↔● E1B1,E1,B1,2,1,6,30</p>
<p>(⊙: entrance/exit, ○: generic node, ●: boarding area; ↔: walkway, ↔: stairs)</p>			

Successive stairs & walkways can be merged

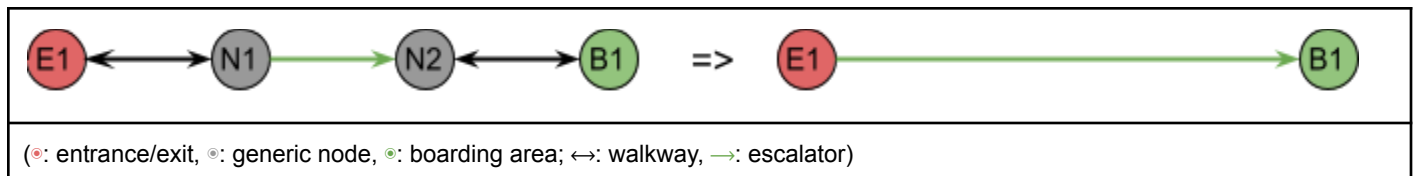
Successive stairs and walkways without intersection can be merged. The merged pathway will have the sum of the stair counts and of the length as properties.














pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length,stair_count				
<div><div><div><div><div></div><div>↔</div><div></div></div></div><div><div><div></div><div>↔</div><div></div></div></div><div><div><div></div><div>↔</div><div></div></div></div><div><div><div></div><div>↔</div><div></div></div></div></div></div>	<div>E1N1,E1,N1,2,1,0,15 N1N2,N1,N2,1,1,1, N2N3,N2,N3,2,1,0,15 N3B1,N3,B1,1,1,5</div>	<div>=></div>	<div><div><div><div><div></div><div>↔</div><div></div></div></div><div><div><div></div><div>↔</div><div></div></div></div></div></div>	<div>E1B1,E1,B1,2,1,6,30</div>
<div>(<div></div>: entrance/exit, <div></div>: generic node, <div></div>: boarding area; <div>↔</div>: walkway, <div>↔</div>: stairs)</div>				

Access walkways can be merged

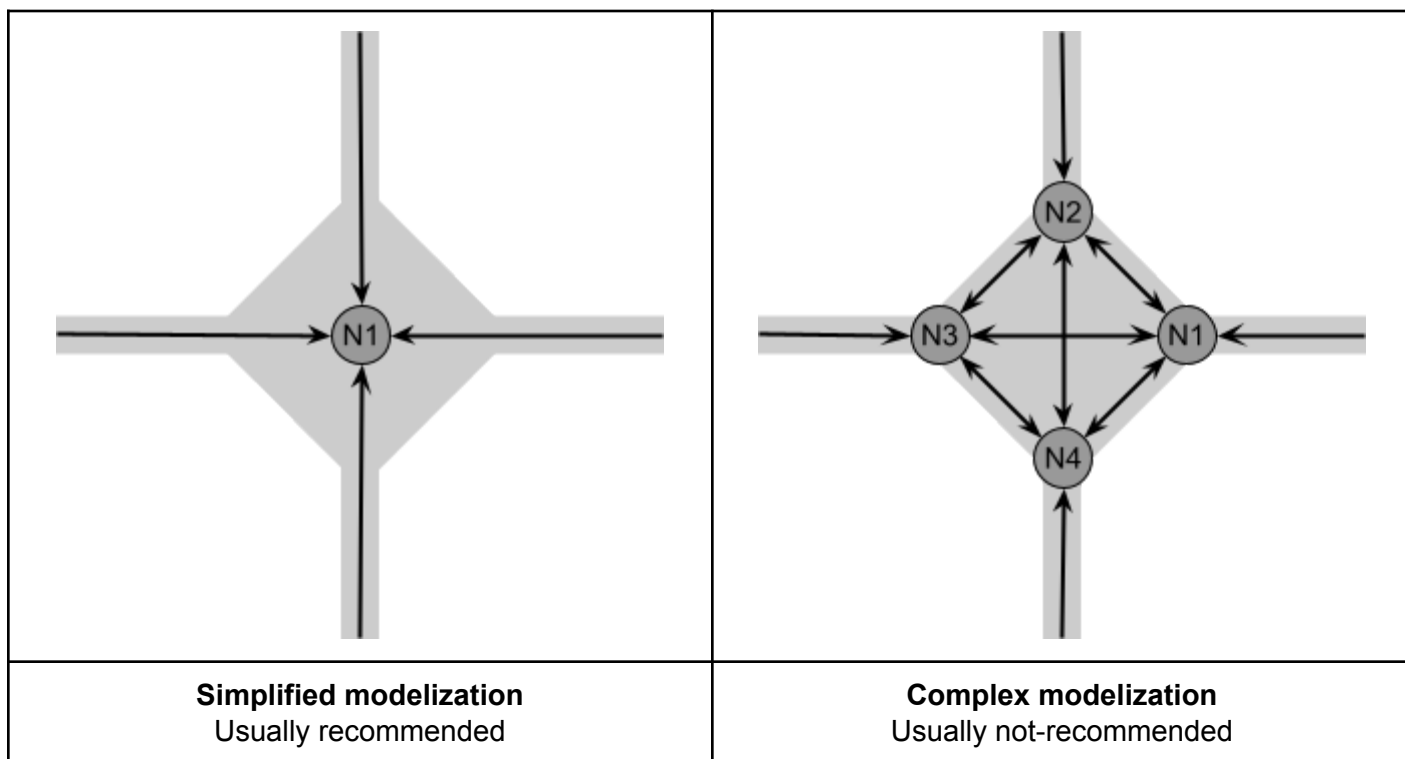
Access walkways to escalator, travelator, elevator, fare gate or exit gate can be merged with them. The merged pathway will have the sum of the length as properties.



pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length,travel_time				
 ↔ 	E1N1,E1,N1,1,1,5,	=>	 → 	E1-B1,E1,B1,4,0,10,30
 → 	N1-N2,N1,N2,4,0, ,30			
 ↔ 	N2B1,N2,B1,1,1,5,			
( : entrance/exit,  : generic node,  : boarding area; ↔: walkway, →: escalator)				

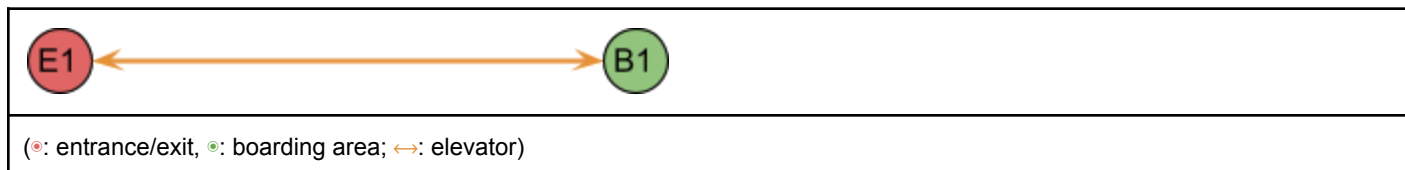
Surface should be simplified

If any path through a room can be used by the traveler (e.g. station hall), the pathways should not contain all the possible paths through the room. By default, a simplified modelization is recommended. Sometimes, a complex representation is needed (e.g. if the size of the hall is so large that the simplified modelization would give massively pessimistic travel distance). Therefore the level of granularity of the pathways will depend on the size of the room.



[Example D] Elevator Modeling [↑](#)


Elevator information is contained in `stops.txt` & `pathways.txt`, but also in `levels.txt` in which the level names are the one displayed in the elevator. For example this station:



...would be represented in GTFS as (station S & platform P aside):

stops.txt	
	stop_id,level_id,stop_name,stop_lat,stop_lon,location_type,parent_station
●	E1,L0,,42.358056,-71.063611,2,S
●	B1,L1,,42.358056,-71.063611,4,P
(● : entrance/exit, ● : boarding area)	

pathways.txt	
	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,

	length, traversal_time, signposted_as, reversed_signposted_as E1B1, E1, B1, 5, 1, 2, 30, Green Line Northbound, City Hall Plaza
(⬤: entrance/exit, ⬤: boarding area; ↔: elevator)	

levels.txt
level_id, level_index, level_name L0, 0, Street L1, -1, Platform

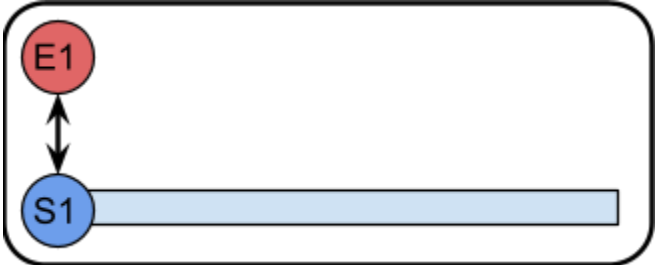
- From the street, it would be indicated as “Take the elevator signposted as “Green Line Northbound” to level “Platform””.
- From the platform, it would be indicated as “Take the elevator signposted as “City Hall Plaza” to level “Street””.

[Example E] Platform Modeling [↑](#)

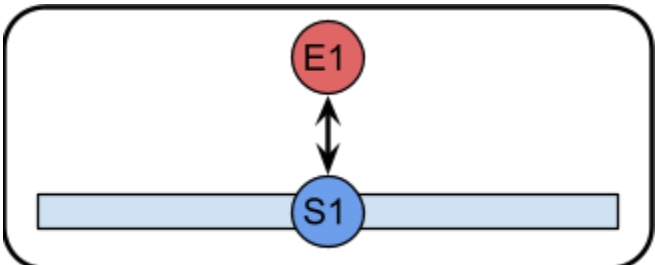
Each end of a platform should always be described as boarding areas instead of just having one platform (as stop) representing the platform. This might not be needed if the vehicles are short, but it recommended if the vehicles are long.

When a rider alights from a vehicle, a trip planner has to compute the mean distance (or time) between the portion of the platform on which the rider alighted, and the exit.

With a naive descriptions of the platforms, there is no way to make the distinction between the following three cases:



200m platform, access at one end

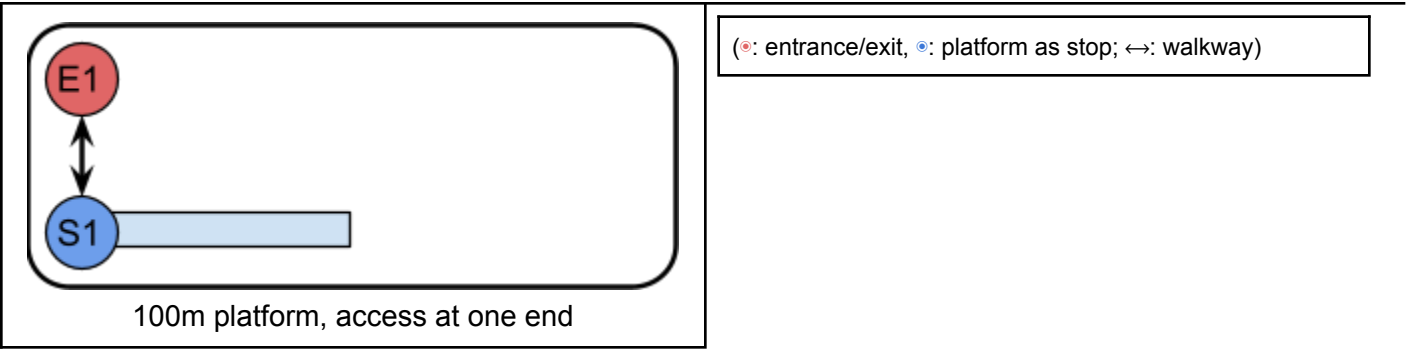


200m platform, access in the middle

All three cases on the left would be modeled as:

stops.txt	
	stop_id, stop_name, stop_lat, stop_lon, location_type, parent_station
⬤	S, , 42.358056, -71.063611, 1,
⬤	E1, , 42.358056, -71.063611, 2, S
⬤	S1, , 42.358056, -71.063611, 0, S
(⬤: station, ⬤: entrance/exit, ⬤: platform as stop)	

pathways.txt	
	pathway_id, from_stop_id, to_stop_id, pathway_mode, is_bidirectional
↔	E1S1, E1, S1, 1, 1



Therefore, some extra information is needed:

pathways.txt	
<div> <p>Case A: 200m platform, access at one end</p> </div>	<div> ●\leftrightarrow● ●\leftrightarrow● </div> <pre> pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length E1B1,E1,B1,1,1,50 B1B2,B1,B2,1,1,200 </pre>
<div> <p>Case B: 200m platform, access in the middle</p> </div>	<div> ●\leftrightarrow● ●\leftrightarrow● ●\leftrightarrow● </div> <pre> pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length E1B1,E1,B1,1,1,50 B2B1,B2,B1,1,1,100 B1B3,B1,B3,1,1,100 </pre>
<div> <p>Case C: 100m platform, access at one end</p> </div>	<div> ●\leftrightarrow● ●\leftrightarrow● </div> <pre> pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length E1B1,E1,B1,1,1,1 B1B2,B1,B2,1,1,100 </pre>
<p>(●: entrance/exit, ●: boarding area; \leftrightarrow: walkway)</p>	

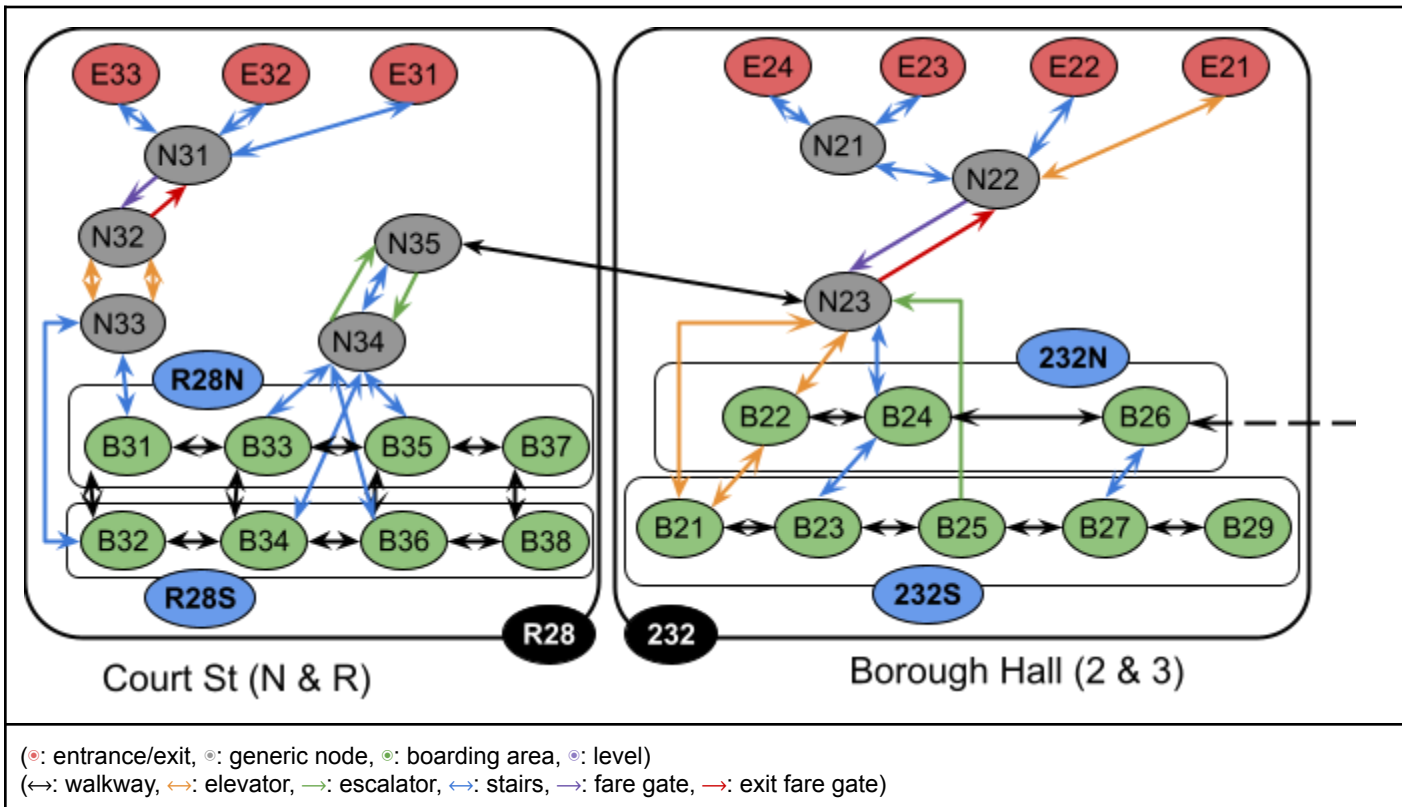
With a stops.txt as following:

stops.txt	
	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
●	S,,42.358056,-71.063611,1,
●	E1,,42.358056,-71.063611,2,S
●	S1,,42.358056,-71.063611,0,S
●	B1,,[latitude],[longitude],4,S1
●	B2,,[latitude],[longitude],4,S1
●	B3,,[latitude],[longitude],4,S1
(●: station, ●: entrance/exit, ●: platform as stop, ●: boarding area)	

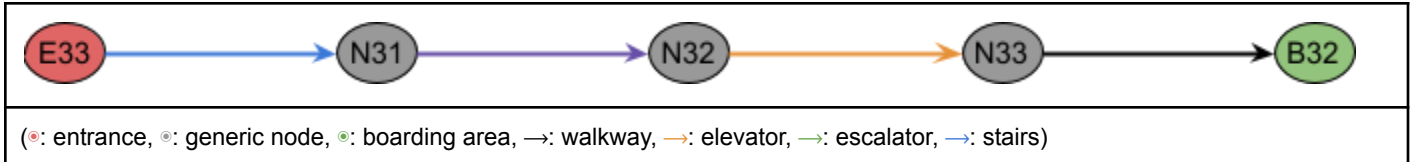
Which allows a trip planner to compute that the mean distance between alighting and exiting the platform is: 10m for case A, 50m for case B and 50m for case C.

[Example F] Wayfinding Modeling ↑

Let's look at the graph of the Court Street & Borough Hall cluster and choose one path in it (the full description of this cluster is done as [Example Z in this document](#)).



If we want to go from outside Court St station to its southbound platform (so from around E33 to any boarding area in R38N), the optimal path would be:



<pre> route.route_short_name = "R" station.stop_name = "Court St" entrance.stop_name = "Clinton & Montague NW corner" signposted_as = "" </pre>	
<pre> generic_node.stop_name = "" signposted_as = "" </pre>	
<pre> generic_node.stop_name = "" signposted_as = "Elevators to R" N32's level_name = "Upper Mezzanine" N33's level_name = "Lower Mezzanine" </pre>	



```
generic_node.stop_name = ""
signposted_as = "Bay Ridge"
```



(⦿: generic node, ●: boarding area, →: walkway, →: elevator, →: escalator, →: stairs)


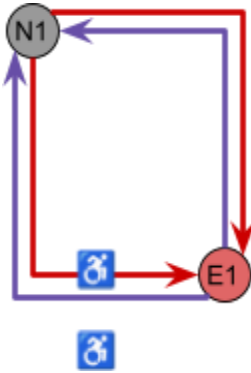
Which would produce the following wayfinding indication:

Enter “**Court St**” station at “**Clinton & Montague NW corner**” entrance. Pass the fare gates, then follow the sign “**Elevators to R**”, then take the elevator to floor “**Lower Mezzanine**”, then follow the sign “**Bay Ridge**”.

We see the value if, on a certain day, it’s not the R but the N which runs on this track, and if the destination is not “Bay Ridge” but “Coney Island” . **The values above are the one to follow in the station to get the right platform**, they are not describing which route or destination will effectively pass on this track on a particular day.

[Example G] Fare gates modeling

Accessible gates

Real life situation	Modeling
	
[Source: http://bit.ly/2RmAbc5]	(⊙: entrance/exit, ⊙: generic node, →: fare gate, →: exit fare gate)

pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length,m in_width	
⊙→⊙	E1-N1,E1,N1,6,0,1,0.3
⊙→⊙	N1-E1,N1,E1,7,0,1,0.3
⊙→⊙	E1-N1-ada,E1,N1,6,0,1,1
⊙→⊙	N1-E1-ada,N1,E1,7,0,1,1
(⊙: entrance/exit, ⊙: generic node; →: fare gate, →: exit fare gate)	

Métro & RER case

In Paris there are two networks: the Métro (subway) and the RER (suburban rail). Therefore, if the station contains interconnection between Métro and RER:

Métro & RER entry & exit gates		
As data		As graph
	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,length,min_width E1-N1,E1,N1,6,0,1,0.3 N1-E1,N1,E1,7,0,1,0.3 N2-N1,N2,N1,6,0,1,0.3 N1-N2,N1,N2,6,0,1,0.3 E2-N1,E2,N1,6,0,1,0.3 N1-E2,N1,E2,7,0,1,0.3	
(⬤: entrance/exit, ○: generic node, →: fare gate, →: exit fare gate)		

[Example Z] Cluster Modeling

Let’s look at Court Street & Borough Hall cluster in Brooklyn (US-NY). Below is the representation of those subway stations as displayed by the MTA, and in two transit apps.

Top left: **Google Maps** (2018-10-13)

Top right: **MTA Map**

Bottom right: **Apple Maps** (2018-10-13)

Core GTFS

With just the minimum required to make a valid GTFS, this would be represented as just 3 stops:

	stop_id,stop_name,stop_lat,stop_lon
•	R28,,Court St,,40.6941,-73.991777
•	232,,Borough Hall (2&3),,40.693219,-73.989998
•	423,,Borough Hall (4&5),,40.692404,-73.990151
(•: stop)	

Core GTFS & GTFS-Stations

Using GTFS-Stations, it would be 6 platforms (as stops) and 3 stations (this is the state of the current NY MTA GTFS) (new data in bold):

	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
•	R28,Court St,40.6941,-73.991777,1,
•	R28N,Court St Northbound,40.6941,-73.991777,0,R28
•	R28S,Court St Southbound,40.6941,-73.991777,0,R28
•	232,Borough Hall 2&3,40.693219,-73.989998,1,
•	232N,Borough Hall 2&3 Northbound,40.693219,-73.989998,0,232
•	232S,Borough Hall 2&3 Southbound,40.693219,-73.989998,0,232
•	423,Borough Hall 4&5,40.692404,-73.990151,1,
•	423N,Borough Hall 4&5 Northbound,40.692404,-73.990151,0,423
•	423S,Borough Hall 4&5 Southbound,40.692404,-73.990151,0,423
(•: station, •: platform)	

Two alternates representations would be possible:

- Only one station for both Borough Hall stations: this would be an as-good representation.
- Only one station for the whole cluster: this is not recommended since Court Street & Borough Hall are really branded as distinct station.

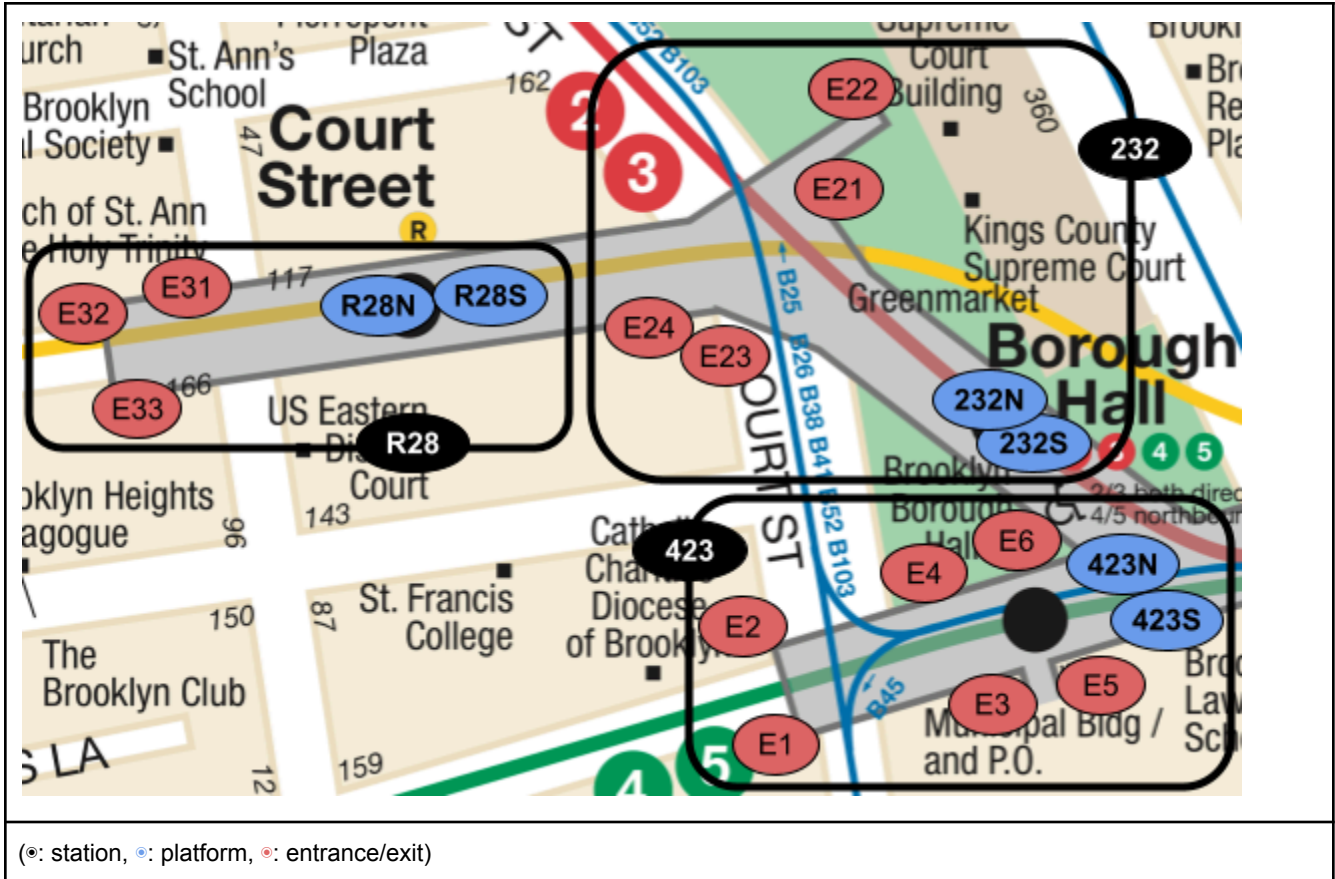
Merging by default stations because they are linked underground is a dangerous rule of thumb, for example, Park Pl (2&3) and Fulton Street Station (2&3) are linked underground, without even exiting the fare zone, but they are two subsequent stations from the same line, and therefore they cannot be merged.

Core GTFS, GTFS-Stations & GTFS-Entrances

Using GTFS-Stations & -Entrances, it would be 6 platforms, 3 station and 13 entrances/exits (new data in bold).

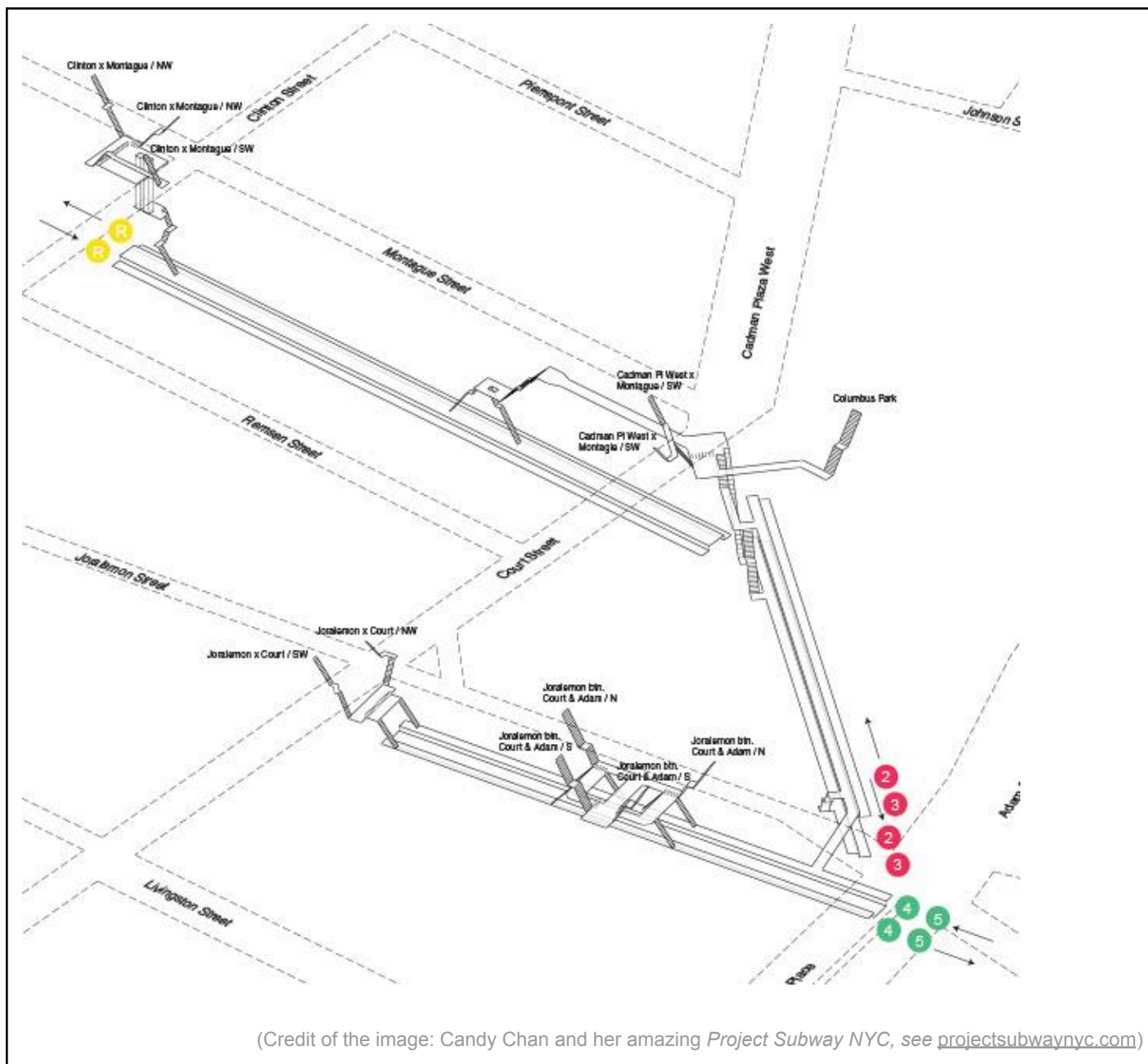
	stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station
•	R28,Court St,40.6941,-73.991777,1,
•	E31,Clinton & Montague NW,40.694445,-73.992732,2,R28

•	E32,Clinton & Montague NW,40.694415,-73.992621,2,R28
•	E33,Clinton & Montague SW,40.694301,-73.992628,2,R28
•	R28N,Court St Northbound,40.6941,-73.991777,0,R28
•	R28S,Court St Southbound,40.6941,-73.991777,0,R28
•	232,Borough Hall 2&3,40.693219,-73.989998,1,
•	E21,Columbus Park Elevator,40.693595,-73.990300,2,232
•	E22,Columbus Park,40.693826,-73.990115,2,232
•	E23,Cadman Plz West & Montague SW,40.693739,-73.990630,2,232
•	E24,Cadman Plz West & Montague SW,40.693642,-73.990565,2,232
•	232N,Borough Hall 2&3 Northbound,40.693219,-73.989998,0,232
•	232S,Borough Hall 2&3 Southbound,40.693219,-73.989998,0,232
•	423,Borough Hall 4&5,40.692404,-73.990151,1,
•	E1,Joralemon & Court NW,40.692657,-73.991077,2,423
•	E2,Joralemon & Court SW,40.692503,-73.991180,2,423
•	E3,Joralemon btw. Court & Adam SW,40.692394,-73.990556,2,423
•	E4,Joralemon btw. Court & Adam NW,40.692608,-73.990543,2,423
•	E5,Joralemon btw. Court & Adam SE,40.692334,-73.990180,2,423
•	E6,Joralemon btw. Court & Adam NE,40.692560,-73.990107,2,423
•	423N,Borough Hall 4&5 Northbound,40.692404,-73.990151,0,423
•	423S,Borough Hall 4&5 Southbound,40.692404,-73.990151,0,423
(•: station, •: platform, •: entrance/exit)	

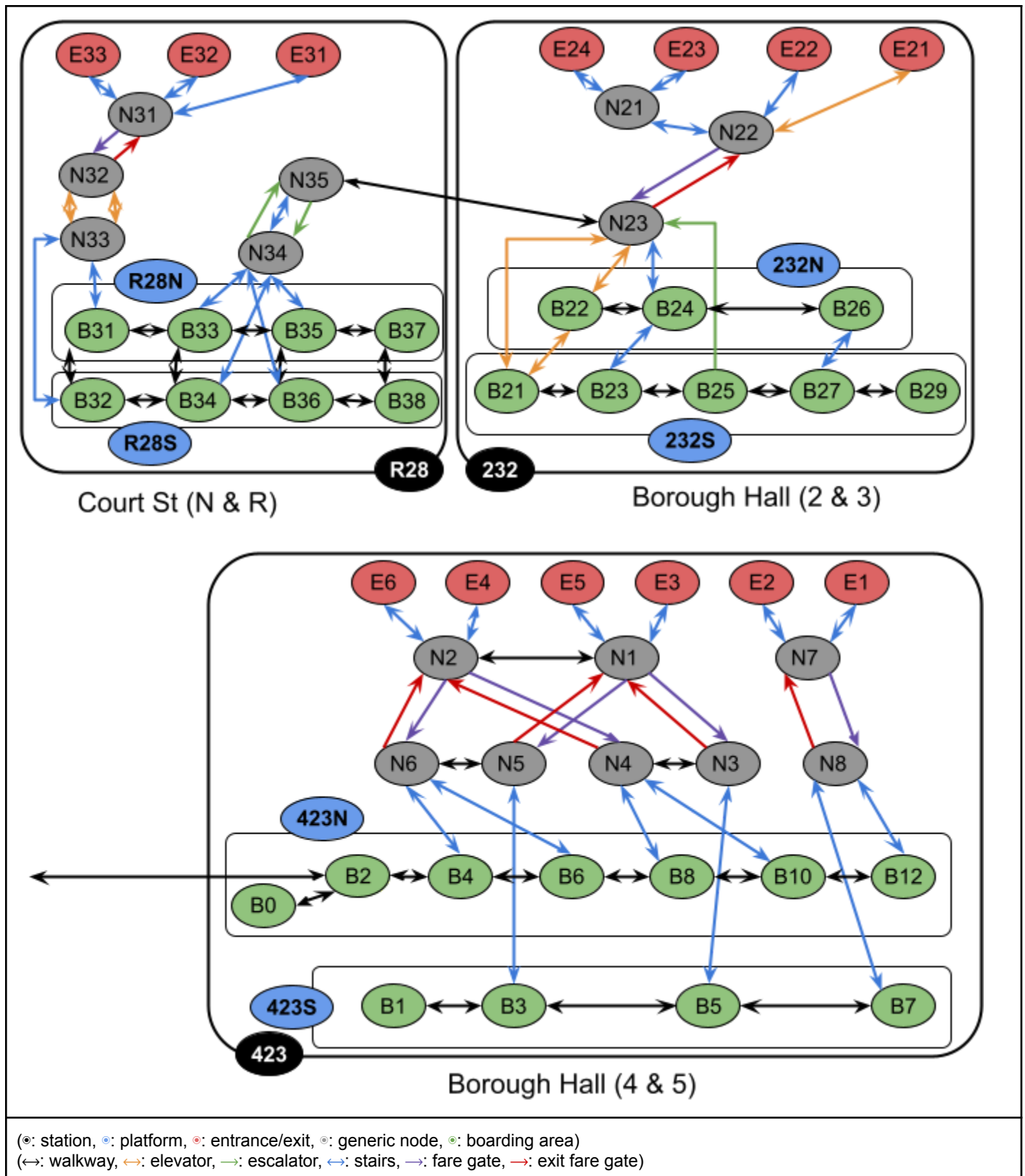


Core GTFS, -Stations, -Entrances & -Pathways

Let's have a look on the 3D view of the station go get the real world situation:



The graph of the station then is:



For the sake of the example, we are just going to ignore the Borough Hall (4&5) part of the cluster for now. The **minimum** data for this station would be (new data in bold):

stops.txt	
	<pre> stop_id,stop_name,stop_lat,stop_lon,location_type,parent_station • R28,Court St,40.6941,-73.991777,1, • E31,Clinton & Montague NW,40.694445,-73.992732,2,R28 • E32,Clinton & Montague NW,40.694415,-73.992621,2,R28 • E33,Clinton & Montague SW,40.694301,-73.992628,2,R28 • N31,,,,,3,R38 • N32,,,,,3,R38 • N33,,,,,3,R38 • N34,,,,,3,R38 • N35,,,,,3,R38 • R28N,Court St Northbound,40.6941,-73.991777,0,R28 • B31,,,,,4,R38N • B33,,,,,4,R38N • B35,,,,,4,R38N • B37,,,,,4,R38N • R28S,Court St Southbound,40.6941,-73.991777,0,R28 • B32,,,,,4,R38N • B34,,,,,4,R38N • B36,,,,,4,R38N • B38,,,,,4,R38N • 232,Borough Hall 2&3,40.693219,-73.989998,1, • E21,Columbus Park Elevator,40.693595,-73.990300,2,232 • E22,Columbus Park,40.693826,-73.990115,2,232 • E23,Cadman Plz West & Montague SW,40.693739,-73.990630,2,232 • E24,Cadman Plz West & Montague SW,40.693642,-73.990565,2,232 • N21,,,,,3,232 • N22,,,,,3,232 • N23,,,,,3,232 • 232N,Borough Hall 2&3 Northbound,40.693219,-73.989998,0,232 • B22,,,,,4,232N • B24,,,,,4,232N • B26,,,,,4,232N • 232S,Borough Hall 2&3 Southbound,40.693219,-73.989998,0,232 • B21,,,,,4,232S • B23,,,,,4,232S • B25,,,,,4,232S • B27,,,,,4,232S • B29,,,,,4,232S [...and Borough Hall 4&5 locations]</pre>
(•: station, •: platform, •: entrance/exit, •: generic node, •: boarding area)	

pathways.txt

	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional
	E31N31,E31,N31,2,1
	E32N31,E32,N31,2,1
	E33N31,E33,N31,2,1
	N31-N32,N31,N32,6,0
	N32-N31,N32,N31,7,0
	N32N33A,N32,N33,5,1
	N32N33B,N32,N33,5,1
	N33B31,N33,B31,2,1
	B31B33,B31,B33,1,1
	B33B35,B33,B35,1,1
	B35B37,B35,B37,1,1
	N33B32,N33,B32,1,1
	B31B32,B31,B32,1,1
	B33B34,B33,B34,1,1
	B35B36,B35,B36,1,1
	B37B38,B37,B38,1,1
	B32B34,B32,B34,1,1
	B34B36,B34,B36,1,1
	B36B38,B36,B38,1,1
	B33N34,B33,N34,2,1
	B35N34,B35,N34,2,1
	B34N34,B34,N34,2,1
	B36N34,B36,N34,2,1
	N34N35,N34,N35,2,1
	N34-N35,N34,N35,4,0
	N35-N34,N35,N34,4,0
	N35N23,N35,N23,1,1
	E24N21,E24,N21,2,1
	E23N21,E23,N21,2,1
	N21N22,N21,N22,2,1
	E22N22,E22,N22,2,1
	E21N22,E21,N22,5,1
	N22-N23,N22,N23,6,0
	N23-N22,N23,N22,7,0
	N23B22,N23,B22,5,1
	N23B24,N23,B24,2,1
	B22B24,B22,B24,1,1
	B24B26,B24,B26,1,1
	N23B21,N23,B21,5,1
	B22B21,B22,B21,5,1
	B24B23,B24,B23,2,1

→	N23B25,N23,B25,4,0
↔	B26B27,B26,B27,2,1
↔	B21B23,B21,B23,1,1
↔	B23B25,B23,B25,1,1
↔	B25B27,B25,B27,1,1
↔	B27B29,B27,B29,1,1
↔	B26B2,B26,B2,1,1
	[...and Borough Hall 4&5 pathways]
(•: station, •: platform, •: entrance/exit, •: generic node, •: boarding area) (↔: walkway, ↔: elevator, →: escalator, ↔: stairs, →: fare gate, →: exit fare gate)	

The **recommended** data for this station, using GTFS-Pathways would be the following (with extra spaces to increase the readability) (new data in bold):

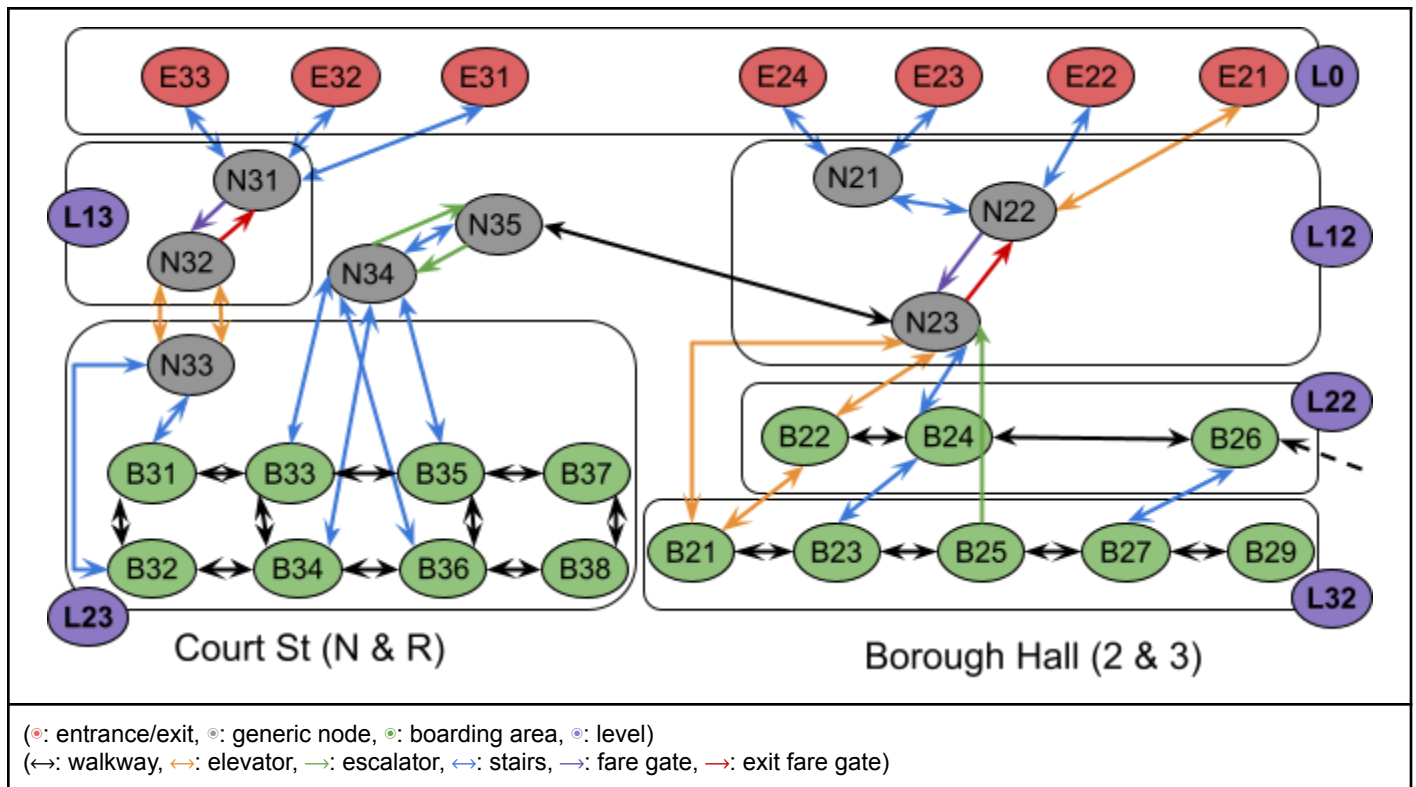
stops.txt
[Same than above but with the stop_lat & stop_lon]

pathways.txt	
	pathway_id,from_stop_id,to_stop_id,pathway_mode,is_bidirectional,cover_type, length,max_slope,min_width,stair_count,traversal_time,signposted_as,reverse d_signposted_as
↔	E31N31,E31,N31,2,1,3,5,0,1,30,,R,Exit: Clinton St & Montague St NW corner
↔	E32N31,E32,N31,2,1,3,5,0,1,30,,R,Exit: Clinton St & Montague St NW corner
↔	E33N31,E33,N31,2,1,3,5,0,1,31,,R,Exit: Clinton St & Montague St SW corner
→	N31-N32,N31,N32,6,0,3,2,0,0.8,,,R,
→	N32-N31,N32,N31,7,0,3,2,0,0.8,,,Exit: Clinton St & Montague St,
↔	N32N33A,N32,N33,5,1,3,,0,2,,30,Elevators to R,Elevator to street
↔	N32N33B,N32,N33,5,1,3,,0,2,,30,Elevators to R,Elevator to street
↔	N33B31,N33,B31,2,1,3,40,0,2,31,,Manhattan & Queens,Elevator to street
↔	B31B33,B31,B33,1,1,3,100,0,1,,,2345 & Exit: Court St & Montague St,Exit: Elevator to Clinton & Montague Street
↔	B33B35,B33,B35,1,1,3,10,0,1,,,,Exit: Elevator to Clinton & Montague Street
↔	B35B37,B35,B37,1,1,3,10,0,1,,,,Exit
↔	N33B32,N33,B32,2,1,3,40,0,2,31,,Bay Ridge,Elevator to street
↔	B31B32,B31,B32,1,1,3,1,0,1,,,Bay Ridge 95 St,Manhattan & Queens
↔	B33B34,B33,B34,1,1,3,1,0,1,,,Bay Ridge 95 St,Manhattan & Queens
↔	B35B36,B35,B36,1,1,3,1,0,1,,,Bay Ridge 95 St,Manhattan & Queens
↔	B37B38,B37,B38,1,1,3,1,0,1,,,Bay Ridge 95 St,Manhattan & Queens
↔	B32B34,B32,B34,1,1,3,40,0,1,,,2345 & Exit: Court St & Montague St,Exit: Elevator to Clinton & Montague Street
↔	B34B36,B34,B36,1,1,3,100,0,1,,,,Exit: Elevator to Clinton & Montague Street
↔	B36B38,B36,B38,1,1,3,100,0,1,,,,Exit
↔	B33N34,B33,N34,2,1,3,10,0,1,29,R,2345 & Exit: Court St & Montague St,R
↔	B34N34,B34,N34,2,1,3,10,0,1,29,R,2345 & Exit: Court St & Montague St,R

↔	B35N34,B35,N34,2,1,3,10,0,1,29,R,2345 & Exit: Court St & Montague St,R
↔	B36N34,B36,N34,2,1,3,10,0,1,29,R,2345 & Exit: Court St & Montague St,R
↔	N34N35,N34,N35,2,1,3,5,0,1,35,,2345 & Exit,R
→	N34-N35,N34,N35,4,0,3,1,0,0.8,,30,2345 & Exit,R
→	N35-N34,N35,N34,4,0,3,1,0,0.8,,30,R,2345 & Exit
↔	N23N35,N23,N35,1,1,3,100,6,3,,,R,2345 & Exit: Court Street & Montague Street
↔	E24N21,E24,N21,2,1,3,5,0,1,16,,23 & 45R via passageway,Exit: Court Street
↔	E23N21,E23,N21,2,1,3,5,0,1,14,,23 & 45R via passageway,Exit: Montague Street
↔	N21N22,N21,N22,2,1,3,5,0,1,15,,23 & 45R via passageway,Exit: Court & Montague Sts
↔	E22N22,E22,N22,2,1,3,5,0,1,31,,23 & 45R via passageway,Exit: Cadman Plaza, Post Office & Supreme Court
↔	E21N22,E21,N22,5,1,3, 5,0, 1,,30,23 & Manhattan 45,Elevator to Court Street & Cadman Plaza
↔	N22-N23,N22,N23,6,0,3,2,0,0.8,,,23 & 45R via passageway,
↔	N23-N22,N23,N22,7,0,3,2,0,0.8,,,Exit: Court St & Montague,
↔	N23B22,N23,B22,5,1,3,5,0,0.8,,30,Elevator to 2345,Elevator to mezzanine
↔	N23B24,N23,B24,2,1,3,10,0,1,20,,2345,Brooklyn 23 & R & Exit: Court St
↔	B22B24,B22,B24,1,1,3,25,0,1,,,45,R & Exit: Court St
↔	B24B26,B24,B26,1,1,3,175,0,1,,,45,Elevator to Exit: Court St
↔	N23B21,N23,B21,5,1,3,5,0,0.8,,45,Elevator to 2345,Elevator to mezzanine
↔	B22B21,B22,B21,5,1,3,5,0,0.8,,30,Elevator to Brooklyn 23,45 & Manhattan 23
↔	B24B23,B24,B23,2,1,3,10,0,1,27,,23 Flatbush Av & New Lots Av,Manhattan 23
↔	N23-B25,N23,B25,4,0,3,,0,0.8,,30,Brooklyn 23,R & Exit: Court St
↔	B26B27,B26,B27,2,1,3,10,0,1,24,,Brooklyn 23,45
↔	B21B23,B21,B23,1,1,3,25,0,1,,,45,Elevator to Manhattan 23 & R & Exit: Court St
↔	B23B25,B23,B25,1,1,3,25,0,1,,,45,Elevator to Manhattan 23 & R & Exit: Court St
↔	B25B27,B25,B27,1,1,3,150,0,1,,,45,Manhattan 23 & R & Exit: Court St
↔	B27B29,B27,B29,1,1,3,50,0,1,,,,45 & Manhattan 23 & R & Exit
	B2B26, B2,B26,1,1,3,200,3,2,,,23 & R,45
	[...and Borough Hall 4&5 pathways]
(↔: walkway, ↔: elevator, →: escalator, ↔: stairs, →: fare gate, →: exit fare gate)	

Core GTFS, GTFS-Stations, -Entrances, -Pathways & -Levels

Since we have 4 elevators in those 2 stations, GTFS-Levels should be used to represent them:



In the data, this would be represented by adding a `levels.txt` file and an extra column in `stops.txt` (new data in bold):

levels.txt	
level_id,level_index,level_name	
L0,0,Street	
L12,-12,Mezzanine	
L22,-22,Manhattan / Upper Platform	
L32,-32,Brooklyn / Lower Platform	
L13,-23,Upper Mezzanine	
L23,-33,Lower Mezzanine	

stops.txt	
stop_id, level_id ,stop_name,stop_lat,stop_lon,location_type,parent_station	
• R28,,Court St,40.6941,-73.991777,1,	
• E31, L0 ,Clinton & Montague NW,40.694445,-73.992732,2,R28	
• E32, L0 ,Clinton & Montague NW,40.694415,-73.992621,2,R28	
• E33, L0 ,Clinton & Montague SW,40.694301,-73.992628,2,R28	
• N31, L13 ,,3,R38	
• N32, L13 ,,3,R38	
• N33, L23 ,,3,R38	

⦿	N34,,,,,3,R38
⦿	N35,,,,,3,R38
⦿	R28N,,Court St Northbound,40.6941,-73.991777,0,R28
⦿	B31,L23,,,,,4,R38N
⦿	B33,L23,,,,,4,R38N
⦿	B35,L23,,,,,4,R38N
⦿	B37,L23,,,,,4,R38N
⦿	R28S,,Court St Southbound,40.6941,-73.991777,0,R28
⦿	B32,L23,,,,,4,R38N
⦿	B34,L23,,,,,4,R38N
⦿	B36,L23,,,,,4,R38N
⦿	B38,L23,,,,,4,R38N
⦿	232,,Borough Hall 2&3,40.693219,-73.989998,1,
⦿	E21,L0,Columbus Park Elevator,40.693595,-73.990300,2,232
⦿	E22,L0,Columbus Park,40.693826,-73.990115,2,232
⦿	E23,L0,Cadman Plz West & Montague SW,40.693739,-73.990630,2,232
⦿	E24,L0,Cadman Plz West & Montague SW,40.693642,-73.990565,2,232
⦿	N21,L12,,,,,3,232
⦿	N22,L12,,,,,3,232
⦿	N23,L12,,,,,3,232
⦿	232N,,Borough Hall 2&3 Northbound,40.693219,-73.989998,0,232
⦿	B22,L22,,,,,4,232N
⦿	B24,L22,,,,,4,232N
⦿	B26,L22,,,,,4,232N
⦿	232S,,Borough Hall 2&3 Southbound,40.693219,-73.989998,0,232
⦿	B21,L32,,,,,4,232S
⦿	B23,L32,,,,,4,232S
⦿	B25,L32,,,,,4,232S
⦿	B27,L32,,,,,4,232S
⦿	B29,L32,,,,,4,232S
	[...and Borough Hall 4&5 locations]
(⦿: station, ⦿: platform, ⦿: entrance/exit, ⦿: generic node, ⦿: boarding area)	

Open Questions

Q1. Stairs upward or downward? [2018-10-16, LF]

Should we have the information of the direction of the stairs? (Up or down). This could be inferred from levels.txt if the pathway has only upward or only downward stairs. But some pathway could contains 15 stairs down & 15 stairs up.

Options:

- A: Make two fields, stair_up_count & stair_down_count

- **B: Forbid pathways with up & down stairs (aka they have to be split in multiple pathways), and allow negative values for `stair_count`.**
- Z: Ignore the issue

=> Option B added in the proposal. Easier to implement and shorter to parse.

Q2. Definition of length and traversal time for merged pathways [2018-10-16, LF]

When a escalator (resp. travelator) is functioning, the traversal time is a better value than the `stair_count` (resp. `length`). But when it is stopped but open, it turns into a stair (resp. walkway), and then the `stair_count` (resp. `length`) is needed.

Options:

- A: Keep `stair_count` (resp. `length`) empty for escalator (resp. travelator), but then what to when they are not operational but open?
- B: Fill `stair_count` (resp. `length`) with the value, but then read below.

When access walkways are merged with an escalator (resp. travelator), should the `traversal_time` (resp. `length`) contains the time (resp. `length`) of only the escalator (resp. travelator), or should it contains the total value?

Options:

- C: Forbid merging between access pathways and mechanical pathways
- D: Provide only the `length/stair_count` values for the non-mechanical part (so option A), but then what to do when they are not operational but open?
- **E: Provide distinct values, like `mechanical_stair_count` & `mechanical_length`**
- Z: Ignore the issue

=> Option E added in the proposal. Remove ambiguity and allow producer to provide it or not.

Q3. Underground entrances & exits [2018-11-09, LF]

Some entrances to public transit network are not on the street, but linked to another structure (mall, university, other public service...). They are needed to provide correct connection, but need to be matched with other data source (e.g. OSM) with a more complex algorithm than the one on the street.

Options:

- 0: We do not include those entries.
- **1: We include them as `location_type=2`, and add required level to be not 0.**
- 2: We include them as `location_type=2`, and add another field for "underground".
- 3: We add another location type for "underground entries"

=> Option 1 added in the proposal on `level_index`, as a simple assumption, not a rule.

Q4. Bus stops and stations [2018-11-09, LF]

Should close on-street bus stops be grouped as a station? IMHO it should not, since a station should be physical structure, either underground or as bus terminal. Also because this grouping could depend of the service, and two bus stops will be in the same “station” for regular bus, and at the same time subsequent stops of a local bus line.

If agency want to group bus stops that they see logically connected, I would advocate to create a “cluster” concept, instead of extending the “station” concept to a blurry definition.

=> **transfers.txt seems to be enough for now.**

Q5. Pathways Evolution: additive or subtractive [2019-05-28, LF]

Without any specific information, a pathway is assumed to be open during service hours. The `pathway_evolutions.txt` aims to allow to change that default behavior. But how should it change it?

[Option A] Subtractive: specifying timeframes during which the pathways is exceptionally closed

This works well with the default, which is that a pathway is open.

It works well for construction work or downtime.

It works badly to specify for example entrances through a mall, which are open only during business hours, M-F 8AM-5PM, since it forces to define all the time when it is *not* open.

[Option B] Additive: specifying timeframes during which the pathways is open

In this approach, as soon as a pathways evolution is defined, the pathway is now closed outside of that time. It works well for pathways with specific opening time (as describe above, link with malls), but badly for construction.

[Option C] Both subtractive and additive

Like in `calendar_dates.txt`, we could define a flag defining if we are adding or removing service, but it generates potential conflicts, which can be resolved either by [Option C1] ordering the evolutions (like in GTFS-Fares) or by [Option C2] allowing only one behaviour by pathways. Below is how it would look like with Option C2. It is more complex than the current proposal, but allows easier description of the different scenarios. Sections in blue are the one added.

`pathways.txt`

Field Name	Details
<code>closed_by_default</code>	(Enum, Optional) Defines if the pathways is either: <ul style="list-style-type: none">- 0 or empty: open by default during the service hours.- 1: closed by default, and open only during the times defined in <code>pathway_evolutions.txt</code>.

`pathway_evolutions.txt` (file added)

Field Name	Details
------------	---------

pathway_id	(ID, Required) The <code>pathway_id</code> field contains an ID that uniquely identifies a pathways. This value is referenced from the <code>pathways.txt</code> file.
service_id	(ID, Required) The <code>service_id</code> fields contains an ID that uniquely identifies a set of dates when this pathway is closed on the timeframe defined below. This value is referenced from the <code>calendar.txt</code> or <code>calendar_dates.txt</code> file.
start_time	<p>(Time, Required) The <code>start_time</code> and <code>end_time</code> fields specify the timeframe during which the pathway evolution applies.</p> <p>Please note that outside of the timeframes explicitly defined in this file, the pathways can be assumed to be open & functional during transit operation hours.</p> <p>The time is measured from "noon minus 12h" (effectively midnight, except for days on which daylight savings time changes occur) at the beginning of the service day. For times occurring after midnight, enter the time as a value greater than 24:00:00 in HH:MM:SS local time for the day on which the trip schedule begins. E.g. 25:35:00.</p>
end_time	(Time, Required) See <code>start_time</code> details.
is_closed	<p>(Enum, Conditionally Required) The <code>is_closed</code> field defines if a pathway is closed:</p> <ul style="list-style-type: none"> - 0 or (empty): pathway is open - 1: pathway is closed <p>Conditionally Required:</p> <ul style="list-style-type: none"> - If the pathway referred by <code>pathway_id</code> is closed by default (i.e. has <code>closed_by_default=1</code>), then <code>is_closed</code> must be set to 0. - If the pathway referred by <code>pathway_id</code> is open by default (i.e. has <code>closed_by_default=0</code> or empty), then <code>is_closed</code> can be either 1 to define closing time, or set to 0 (or empty) to define for example evolution in the direction (see field <code>direction</code>).
direction	<p>(Enum, Optional) The <code>direction</code> field allows to overwrite the <code>is_bidirectional</code> value of the pathways. It has three states (instead of 2 for <code>is_bidirectional</code>) to allow to provide information on escalator or travelator changing direction during the day.</p> <ul style="list-style-type: none"> - 0: Unidirectional forward - 1: Bidirectional - 2: Unidirectional backward

Appendices

[A1] Status color scheme

This document is a working tool for a GTFS extension proposal. As a living and collaborative document, content may evolve based on community feedback. The scheme below tracks the stages at which the contents of the GTFS extension proposal evolve. If you have any questions, please reach out to specifications@mobilitydata.org.

Working draft	<ul style="list-style-type: none">- This content has not yet been adopted or put to a GitHub pull request (PR) on google/transit.- It may evolve based on community feedback, and is therefore subject to change.- Any implementation may have to be updated with iterations or official adoption.
Proposal	<ul style="list-style-type: none">- This content is currently a GTFS extension proposal via a GitHub pull request (PR) on google/transit.- The hyperlinked to its relevant PR is embedded in the text.- The PR should serve as the sole venue for discussion and iteration.- Text of this color may no longer be up to date.
Adopted	<ul style="list-style-type: none">- This content has been put to a GitHub pull request (PR) on google/transit as a GTFS extension proposal, and has been officially adopted.- Text of this color may no longer be up to date.- Any references to this material must be made from the official reference.md.

[A2] Evolution history

2021-02-08

- Archived GTFS-Stations and GTFS-StationEntrances (adopted in the spec)

2019-05-28

- Adds `boarding_edge` in `stops.txt`
- Adds `max_stair_flight`, `manual_activation` and `max_cross_slope` in `pathways.txt`
- Adds Q5 "Pathways Evolution: additive or subtractive".

2019-02-08

- Adds `pathway_mode=8` **“Control”** and `traversal_times.txt`.

2019-01-29

Resolve option questions:

- Add `pathways.mechanical_length` and `pathways.mechanical_stair_count`.
- Change type of `pathways.stair_count` to allow downward stairs.
- Define entrances/exits with `level≠0` as being not on the street grid.

2018-11-28

GTFS-Pathways

- `pathways.assistance` renamed into `pathways.wheelchair_assistance`.
- Improve `pathway_evolution.start_time` definition to handle direction evolutions (and not only `is_closed`).