



NYSERDA

**Department of
Transportation**

Real-time Subway Information for Improving Transit Ridership

Final Report

August 2016

Report Number 16-24

Real-time Subway Information for Improving Transit Ridership

Final Report

Prepared for
New York State Energy Research and Development Authority
Albany, NY

Joseph D. Tario, P.E.
Senior Project Manager

and

New York State Department of Transportation

Albany, NY
Robert Ancar,
Project Manager

Prepared by:

**Albany Visualization and Informatics Lab (AVAIL)
Lewis Mumford Center, University at Albany, SUNY**

Albany, NY

Catherine T. Lawson, Ph.D.
Chair, Geography and Planning Department
Director, Lewis Mumford Center

Notice

This report was prepared by the Albany Visualization and Informatics Lab, Lewis Mumford Center, University at Albany, SUNY in the course of performing work contracted for and sponsored by the New York State Energy Research and Development Authority and the New York State Department of Transportation (hereafter the "Sponsors"). The opinions expressed in this report do not necessarily reflect those of the Sponsors or the State of New York, and reference to any specific product, service, process, or method does not constitute an implied or expressed recommendation or endorsement of it. Further, the Sponsors, the State of New York, and the contractor make no warranties or representations, expressed or implied, as to the fitness for particular purpose or merchantability of any product, apparatus, or service, or the usefulness, completeness, or accuracy of any processes, methods, or other information contained, described, disclosed, or referred to in this report. The Sponsors, the State of New York, and the contractor make no representation that the use of any product, apparatus, process, method, or other information will not infringe privately owned rights and will assume no liability for any loss, injury, or damage resulting from, or occurring in connection with, the use of information contained, described, disclosed, or referred to in this report.

NYSERDA makes every effort to provide accurate information about copyright owners and related matters in the reports we publish. Contractors are responsible for determining and satisfying copyright or other use restrictions regarding the content of the reports that they write, in compliance with NYSERDA's policies and federal law. If you are the copyright owner and believe a NYSERDA report has not properly attributed your work to you or has used it without permission, please email print@nyserda.ny.gov

Information contained in this document, such as web page addresses, are current at the time of publication.

Abstract

In recent years, the standardization of transit schedule information has yielded a dramatic increase in the accessibility of computerized transit schedules and given rise to real-time service schedules. Two such real-time service schedules are the General Transit Feed Specification Real-time (GTFS-R) and the Service Interface for Real-time Information (SIRI). Due to system/vendor limitations, the Metropolitan Transportation Authority (MTA) uses GTFS-R for its real-time subway schedule and SIRI for its real-time bus schedule. In order to promote development of applications using real-time scheduling data, MTA needed a single real-time scheduling format. The purpose of this project is to create an Application Programming Interface (API) that is capable of converting GTFS-R into SIRI. The GTFS-R to SIRI real-time data feed will unify their data streams to provide developers and travelers with more complete access to real-time transit data in the largest mass transit market in the country. Furthermore, in keeping with the spirit of Open Data, Open Source, AVAIL developed a configurable server for use with the converter. By uploading a General Transit Feed Specification (GTFS) zip file or providing a URL pointing to a GTFS feed source, and specifying a URL for the GTFS-R feed, this server can be easily used to convert GTFS-R data to SIRI by any transit agency or researcher throughout the world.

Acknowledgements

Thanks to MTA staff, NYSEDA, and NYS Department of Transportation (DOT); AVAIL staff members Eric Krans and Alex Muro; and graduate students Paul Tomchik, Joel Tirado and Laura Trivison.

Table of Contents

Notice.....	ii
Abstract	iii
Acknowledgements	iii
List of Figures	v
List of Tables.....	vi
Acronyms and Abbreviations	vi
Executive Summary	ES-1
1 Introduction.....	1
1.1 Background	1
1.1.1 Automatic Vehicle Location (AVL).....	2
1.1.2 Open Data	3
1.1.3 API.....	4
1.1.4 Open Source Code.....	4
1.1.5 The General Transit Feed Specification (GTFS)	4
1.1.6 GTFS-R.....	4
1.1.7 SIRI	5
2 Project Budget	6
2.1 Analysis of actual cost incurred in relation to the budget.....	6
2.1.1 Project Budget - NYSERDA	6
2.1.2 MTA Match	6
3 Research Method	7
3.1 Data Analysis	7
3.1.1 General Transit Feed Specification (GTFS).....	7
3.1.2 General Transit Feed Specification Real-Time (GTFS-R)	8
3.1.3 Service Interface for Real-Time Information (SIRI).....	9
3.2 Methods.....	11
3.3 Project Timeline.....	13
4 Findings.....	14
4.1 Caching	15
4.2 Event-based Logging	15
4.3 System Architecture	16
4.4 Performance.....	17
4.5 Future Research	19

5	Statement on Implementation	21
5.1	Technical Aspects of Implementation	21
5.1.1	Admin Configuration Console	21
5.2	API Calls.....	22
5.2.1	Monitoring Calls.....	22
5.2.1.1	SIRI Stop Monitoring Call.....	23
5.3	SIRI StopMonitoring Requests.....	23
5.3.1	SIRI StopMonitoring Responses	24
5.3.2	The SIRI Vehicle Monitoring call	25
5.4	SIRI VehicleMonitoring Requests	25
5.4.1	SIRI VehicleMonitoring Responses.....	26
5.5	Links to the GTFS-R to SIRI API.....	27
5.6	Marketing Strategies Recommended.....	27
6	Conclusion	28
7	Bibliography.....	29
	Appendix A.....	A-1

List of Figures

Figure 1.	Metadata Visualization for NYC Subway GTFS-R.....	8
Figure 2.	Metadata Visualization for Vehicle SIRI Data	9
Figure 3.	Metadata Visualization for Bus Stop SIRI Data	10
Figure 4.	Output Data 1.....	18
Figure 5.	Output Data 2.....	18
Figure 6.	Map Visualization of Real-time Subway Train Positions	19
Figure 7.	GTFS-R to SIRI Conversion API Control Console. Uploading Static GTFS Data by URL	21
Figure 8.	GTFS-R to SIRI Conversion API Control Console. API Key and Retry Settings	22
Figure 9.	GTFS-R to SIRI Conversion API Control Console. Logging Settings.....	22

List of Tables

Table 1. Project Budget.....	6
Table 2. MTA Match.....	6
Table 3. GTFS Features.	7
Table 4. Project Gantt Chart.....	13

Acronyms and Abbreviations

AFC	Automatic Fare-box Collection
APC	Automatic Passenger Counters
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AVL	Automatic Vehicle Location
CAD	Computer Aided Dispatch
GIS	Geographic Information Systems
GTFS	General Transit Feed Specification
GTFS-R	General Transit Feed Specification Real-time
MTA	Metropolitan Transportation Authority
DOT	New York State Department of Transportation
NYSERDA	New York State Energy Research Development Authority
SIRI	Service Interface for Real-Time Information
SM	Stop Monitoring
Tri-MET	Tri-County Metropolitan Transportation District of Oregon
URL	Uniform Resource Locator (Web-address)
XML	Extensible Markup Language

Executive Summary

In recent years, the standardization of transit schedule information has yielded a dramatic increase in the accessibility of computerized transit schedules, and has brought about the availability of real-time service schedule. Currently, as a result of system/vendor limitations, the MTA publishes its real-time transit data in two different formats. The purpose of this project is to create an API that is capable of converting MTA's Subway system data, GTFS-R, on the fly into the format used to publish their bus data, known as SIRI. The GTFS-R to SIRI real-time will unify their data streams to provide developers and travelers more complete access to real-time transit data in New York City, which has the largest mass transit market in the country.

The General Transit Feed Specification (GTFS) is a common format for public transportation schedules and associated geographic information. GTFS is an open data spatial format that contains transit scheduling information that accommodates Automatic Vehicle Location (AVL) data-feeds. GTFS-R is a feed specification that extends GTFS by integrating AVL data allowing public transportation agencies to provide real-time updates about their fleet to application developers. SIRI is an Extensible Markup Language (XML) protocol that provides a similar service to GTFS-R, allowing distributed computers to exchange real-time information about public transport services and vehicles.

AVAIL mapped GTFS-R fields to SIRI fields. To provide geographic coordinates in the SIRI feed, AVAIL developed algorithms that use GTFS scheduling and spatial data as a "spatial backbone," along with the information available in the streaming GTFS-R feed, to smoothly plot the progress of a vehicle along its route.

The performance demands of a real-time system required the building of numerous auxiliary data structures atop the GTFS and GTFS-R feed data that provide instant retrieval of any required piece of information to perform the GTFS-R to SIRI conversion.

For ease of deployment, a configurable server was created around the converter. By uploading a GTFS zip file or providing a URL pointing to a GTFS feed source, and specifying a URL for the GTFS-R feed, this server can be easily setup to convert GTFS-R data to SIRI by the MTA or other transit agencies.

After completion of the GTFS-R to SIRI conversion tool, AVAIL tested the server over the course of two months. The tool converts the GTFS-R data directly to SIRI with 100 percent fidelity. The SIRI

translation feed contains an associated field that holds all data from the MTA GTFS-R. The translation tool also offers information beyond what is contained in the MTA GTFS-R feed including the addition of longitude and latitude during the conversion process. To test the accuracy of the conversion tool in developing longitude and latitude of trains in the subway system, AVAIL integrated a data analysis feature that logs information on trips in each GTFS-R to SIRI message. This logging feature can be used by MTA to assess the quality of their GTFS-R data.

1 Introduction

In order to catalyze ongoing development of applications that utilize real-time transit data, data scientists at AVAIL contracted with the MTA, and funded by the DOT and NYSERDA, to create a GTFS-R to SIRI conversion API. SIRI is a more developer friendly real-time specification than GTFS-R. By making this data available to developers, MTA hopes to increase the use of real-time subway data in existing and emerging applications as well as transit research and visualization projects. The API was developed between July and October, 2015 and was made available to the MTA's developer community for beta testing (October–November, 2015). During this time, AVAIL subjected the API to an extended period of performance reporting. As of the date of this report, the fully operational API has been deployed by MTA.

1.1 Background

The transit industry, traditionally a data-intensive environment, is moving swiftly to a data environment enhanced with opportunities for machine-generated and machine-readable data. Technology advancements in on-board equipment (e.g., Automatic Vehicle Location [AVL], Automatic Passenger Counters [APC], Automatic Farebox Collection [AFC]). More recently, innovative mobile apps and social media (Open Trip Planner, Facebook, etc.) are gaining attention from the public. Yet, only a few transit agencies have fully exploited this emerging “pervasive data” computing environment. Additional technological advancements in computing are accelerating the opportunities to use machine-readable data including: cloud-based storage; cloud hosting of spatial and temporal tools suites for transit modeling software; visualizations and data analytics for marketing and related analyses; and the use of Open Source and Open Data strategies developed by data scientists, particularly APIs, could rapidly transform the work environment for transit agency staff, as well as customer-focused communications, for the transit industry.

Transit agencies have a long history of collecting data (through manually administered on-board surveys, manual ridership counts, financial administrative records, etc. With the introduction of computers into the work environment, these data collection efforts included converting these manually collected data to “machine-readable” formats, making it possible to digitally store and retrieve records efficiently. However, it was still very time-consuming to manually enter these data. The concept of automatically “harvesting” machine-readable data from operations sparked the possibility of new methods for collecting important data elements from transit operations to be used for analysis for operations and planning.

1.1.1 Automatic Vehicle Location (AVL)

One of the major technologies at the information level is Automatic Vehicle Location (AVL) systems. In his 1995 Synthesis of Transit Practice, Howard Benn defined AVL as “A smart technology that monitors the real-time location of transit vehicles (generally non-rail modes) through the use of one or more of the following: global positioning systems (GPS), Loran-C, or signpost transmitters in combination with vehicle odometers. Most installations include integration of the AVL system with a geographic information systems (GIS) or computer mapping system. The monitoring station is normally located in the dispatch /radio communications center” (Benn 1995:51). AVL data contain polling records (bus location reported when queried by a central computerized system, a round-robin polling of “time-at-location”), stop records (time-at-location of a stop), and time point records (time-at-location at time points on route). AVL systems records are “harvested” by on-board computers that upload the records nightly, using automated high-speed links, or using real-time radio transmission, with the records being stored after they are received (Furth et al., 2006).

According to Okunieff (1997), AVL technologies were first deployed in 1969, in Chicago, IL. By the late 1990s, the U.S. DOT was promoting integrating transportation infrastructure (ITI), to enable transit agencies to develop strategies for sharing AVL information. AVL data provided real-time status on information on bus arrival/departments times to assist intermodal connectivity. At the time, the dissemination strategies for AVL included online, audio-dial, video monitor, variable message sign, kiosk, and speakerphone.

Even into the late 90s, no system integrated real-time travel times with computer-aided dispatch (CAD) software to predict actual bus travel times, even though the AVL data provided congestion level information to transit software control systems. Okunieff (1997) describes the data as being virtually “trapped” in its own architecture. However, for freeway operations, AVL data could provide travel times as probes. Okunieff found the TRAVLINK (MCTO) program conducted a test of bus probes using wayside AVL devices. Benn (1995) pointed out that AVL systems were becoming more prevalent and would bring real-time information down to the bus stop level and as a result, bus route evaluation would be used when initially installing new systems. There was interest in how customers might be responding to having real-time information with respect to the toleration of wait times. AVL systems are able to provide the data to answer the question, “Where is the bus?”

1.1.2 Open Data

The general purpose of Open Data is to make data available to the public, but more specifically assembling data elements to generate a standardized data product that is available to the public for free. According to Schweiger (2015), the trajectory of Open Data within the transit industry began when the Bay Area Rapid Transit released data to the public in the form of comma-separated values (.csv) files in 1998. The most widely recognized Open Data accomplishment is the project scoped by the Tri-County Metropolitan Transportation District of Oregon (Tri-Met), in partnership with Google, culminated in the creation of General Transit Feed Specification (GTFS), in 2005. The unique contribution of this project was the generation of static schedule information (e.g., stop location, route geometrics, and stop times) in a standard format.

Schweiger points out that using an Open Data policy provides benefits to public transit authorities in terms of transparency. Transit customers gain tremendous benefits when the Open Data is accessed with free applications, either as real-time or static data. The author further found private firms used the open transit data to develop innovative applications in ways not currently available within transit agencies themselves. In addition, providing Open Data to the academic community greatly increased the use of the data and facilitated the creation of analyses and data-driven discoveries. Open Data can be used in the production of decision-support tools including: data visualizations; service planning and evaluations; route layout and design; performance analysis; and travel time and capacity analysis.

Transit agencies appreciated the increased awareness of their services, empowerment of their customers, innovation outside their agency, improved perceptions of their activities, and contributions to the private sector in the form of new business development. Additionally, they benefited from the feedback they received when their data was used by both developers and customers.

The underlying technologies used in the creation of Open Data, such as GTFS, include: scheduling software, GIS, CAD systems, AVL data, and real-time arrival prediction software. By creating an Open Data approach, many of the previous issues facing individual agencies attempting to build their own version of an integrating procedure were solved by following the standard formatting conventions. Providing customers with improved information on available transit services, in easy to access formats may result in an increase transit ridership as well.

1.1.3 API

Perhaps the most powerful advantage data science brings to transportation is the use of APIs. These tools provide accessibility to data just as plumbing provides accessibility to water. One can imagine data is like water, flowing through pipes with a series of faucets. APIs in this case, are like faucets allowing data scientists to easily and repeatedly access complex data. The advent of API technology made it possible to deploy information to consumers and transit agencies in real-time. Data scientists can utilize APIs to build data analytics dashboards for transit planners to visualize and analyze integrated, complex data. The only requirement for the using APIs is that the data be machine-readable, which requires the data to be assembled with standardized data elements.

1.1.4 Open Source Code

In production and development, open source as a development model promotes universal access via a free license to a product's design or blueprint, and universal redistribution of that design or blueprint, including subsequent improvements to it by anyone. The transportation industry is becoming increasingly aware of the potential pitfalls of proprietary software: cost, dependency on a single business entity, and the slow pace of innovation. An industry-wide commitment to Open Source software would allow advancements to Open Source projects to be deployable anywhere.

1.1.5 The General Transit Feed Specification (GTFS)

The General Transit Feed Specification defines a common format for public transportation schedules and associated geographic information. GTFS “feeds” allow public transit agencies to publish their transit data and developers to write applications that consume the data in an interoperable way. GTFS feeds provide the information needed for users of a transit system to plan their trips. They consist of six to 13 CSV files, updated irregularly, on an as-needed basis. Much of the data contained in the MTA’s GTFS feeds was pertinent to this project. The GTFS-R to SIRI conversion requires information about scheduled routes, expected stop times, and the geographic representations of the paths that vehicles follow while serving their routes.

1.1.6 GTFS-R

GTFS-R is “a feed specification that allows public transportation agencies to provide real-time updates about their fleet to application developers. It is an extension to GTFS, an Open Data format for public transportation schedules and associated geographic information” (What is GTFS-real-time? 2015).

GTFS-R feeds provide information about the actual state of the transit system at given moment as opposed to the scheduled behavior of the system. Messages may contain information about trip delays or cancellations, service alerts, and vehicle positions. These feeds are designed to be used in conjunction with the GTFS data. For example, in the MTA real-time feeds, only trip and route keys are provided. To present meaningful information to riders, application developers must reference the GTFS data. The MTA Subway GTFS-R feed updates every 30 seconds and provides estimated times of arrival for the trains in the system. The Long Island Rail Road feed updates less regularly and contains the delay in seconds from the scheduled arrival time rather than ETAs. (Neither provide vehicle positions. This information is inferred by the conversion system as discussed in section 4.2.

1.1.7 SIRI

SIRI is “an XML protocol to allow distributed computers to exchange real-time information about public transport services and vehicles” (SIRI Home Page, 2011). SIRI is used in a variety of different ways, such as:

- Provide real-time departure from stop information for display on stops, internet and mobile delivery systems;
- Provide real-time progress information about individual vehicles;
- Manage the movement of buses roaming between areas covered by different servers;
- Manage the synchronization of guaranteed connections between fetcher and feeder services;
- Exchange planned and real-time timetable updates;
- Distribute status messages about the operation of the services; and
- Provide performance information to operational history and other management systems (SIRI Home Page, 2011).

The protocol is a standard based on the transmodel abstract model for public transport information.

2 Project Budget

2.1 Analysis of actual cost incurred in relation to the budget

Costs incurred on this project conformed directly to the budget.

2.1.1 Project Budget - NYSERDA

Table 1. Project Budget

Project Budget - NYSERDA					
	Budgeted	Total Project Costs	% of Budget	Remaining Costs	% of Budgeted Available
Direct Labor	\$9,282.00	\$9,526.65	103%	\$(244.65)	-3%
Labor Overhead	\$1,392.00	\$1,429.02	103%	\$(37.02)	-3%
Other Direct Costs	\$51,826.00	51,543.07	99%	\$282.93	1%
Facilities and Administration	\$12,500.00	\$12,499.75	100%	\$0.25	0%
Total	\$75,000.00	\$74,998.49	100%	\$1.51	0%

2.1.2 MTA Match

Table 2. MTA Match

MTA Match			
	Budgeted	Actual	Remaining
Financial	\$19,500.00	\$19,499.26	\$0.74
In-kind	\$5,500.00	\$5,500.00	\$ -
Total	\$25,000.00	\$24,999.26	\$ 0.74

3 Research Method

3.1 Data Analysis

3.1.1 General Transit Feed Specification (GTFS)

As previously described, GTFS is “a common format for public transportation schedules and associated geographic information” (What is GTFS?, 2015). A GTFS feed is a series of text files that models different aspects of transit—routes, stops, trips, etc. and is used in a diverse range of applications relating to public transportation. This data was invaluable in building the GTFS-R to SIRI conversion tool. As noted below, GTFS contains geospatial information required in the SIRI format but lacking in GTFS-R. Figure 1 displays required .txt files used in GTFS as well as some optional and required fields for the “stops.txt” file.

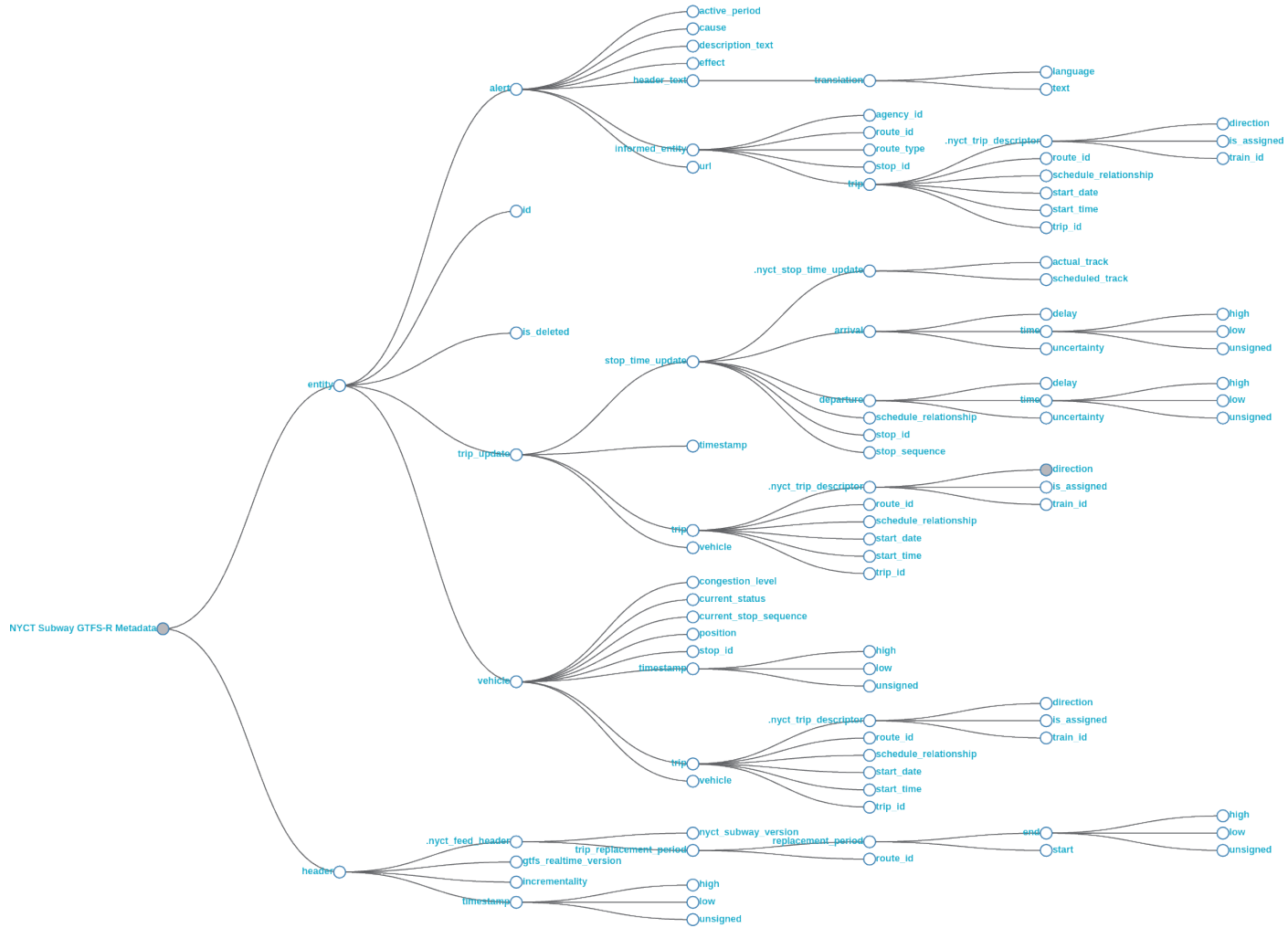
Table 3. GTFS Features.

Filename/Field Name	Required	Defines/Details
Agency.txt	Required	One or more transit agencies that provide the data in this feed.
Stops.txt	Required	Individual locations where vehicles pick up or drop off passengers.
Routes.txt	Required	Transit routes. A route is a group of trips that are displayed to riders as a single service.
Trips.txt	Required	Trips for each route. A trip is a sequence of two or more stops that occurs at specific time.
Stop_times.txt	Required	Times that a vehicle arrives at and departs from individual stops for each trip.
Calendar.txt	Required	Dates for service IDs using a weekly schedule. Specify when service starts and ends, as well as days of the week where service is available.
Stop_id	Required	The stop_id field contains an ID that uniquely identifies a stop or station. Multiple routes may use the same stop. The stop_id is dataset unique.
Stop_code	Optional	The stop_code field contains short text or a number that uniquely identifies the stop for passengers. Stop codes are often used in phone-based transit information systems or printed on stop signage to make it easier for riders to get a stop schedule or real-time arrival information for a particular stop. The stop_code field should only be used for stop codes that are displayed to passengers. For internal codes, use stop_id. This field should be left blank for stops without a code.
Stop_name	Required	The stop_name field contains the name of a stop or station. Please use a name that people will understand in the local and tourist vernacular.
Stop_desc	Optional	The stop_desc field contains a description of a stop. Please provide useful, quality information. Do not simply duplicate the name of the stop.
Stop_lat	Required	The stop_lat field contains the latitude of a stop or station. The field value must be a valid WGS 84 latitude.
Stop_lon	Required	The stop_lon field contains the longitude of a stop or station. The field value must be a valid WGS 84 longitude value from -180 to 180.

3.1.2 General Transit Feed Specification Real-Time (GTFS-R)

Figure 1 is a metadata visualization of GTFS-R created by the AVAIL research team to facilitate mapping of GTFS-R to SIRI.

Figure 1. Metadata Visualization for NYC Subway GTFS-R



3.1.3 Service Interface for Real-Time Information (SIRI)

The MTA uses SIRI to provide geospatial and temporal information on buses in its public transit network. Figures 2 and 3 are metadata visualizations of SIRI Bus Vehicles and Bus Stops, respectively.

Figure 2. Metadata Visualization for Vehicle SIRI Data.

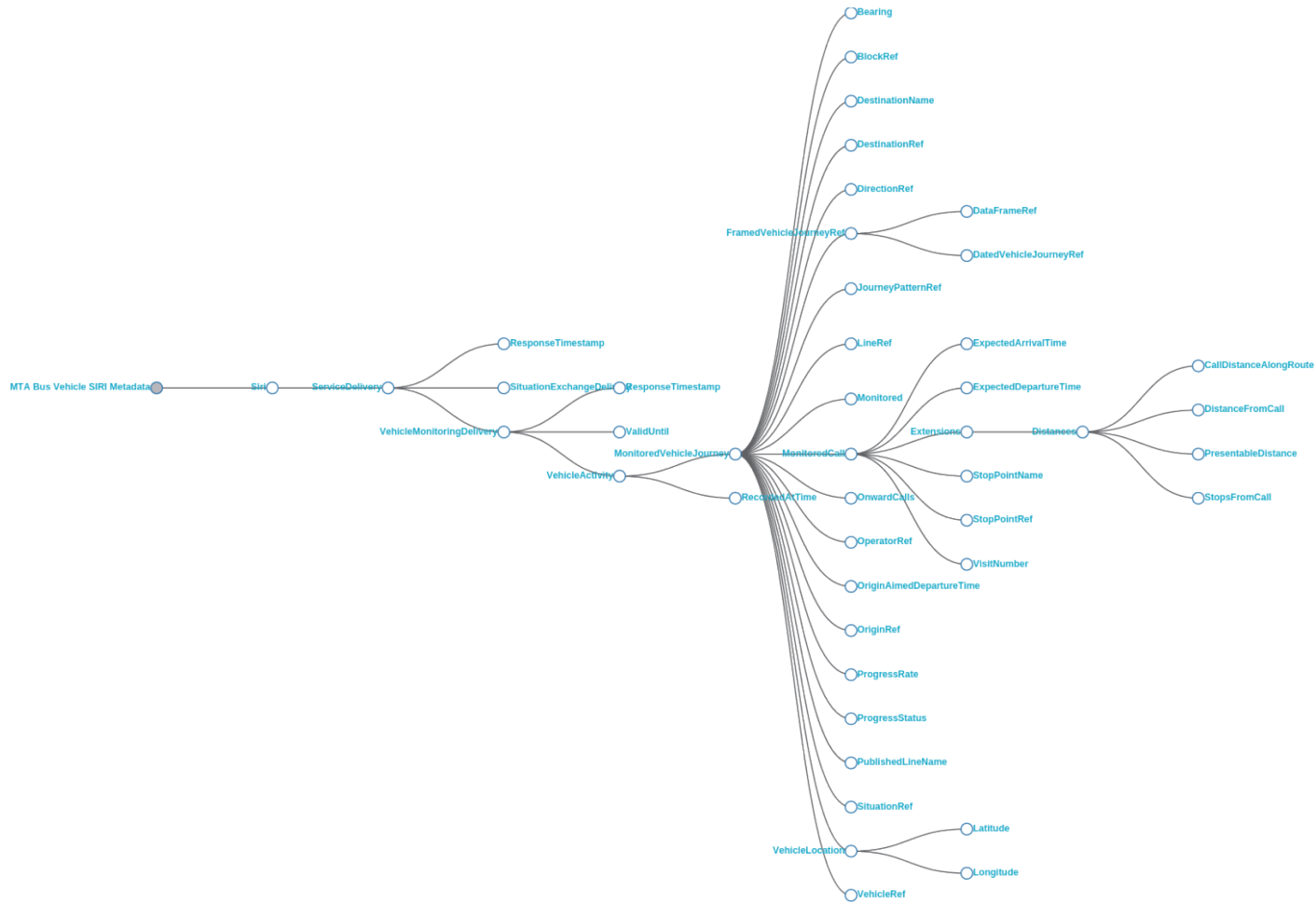
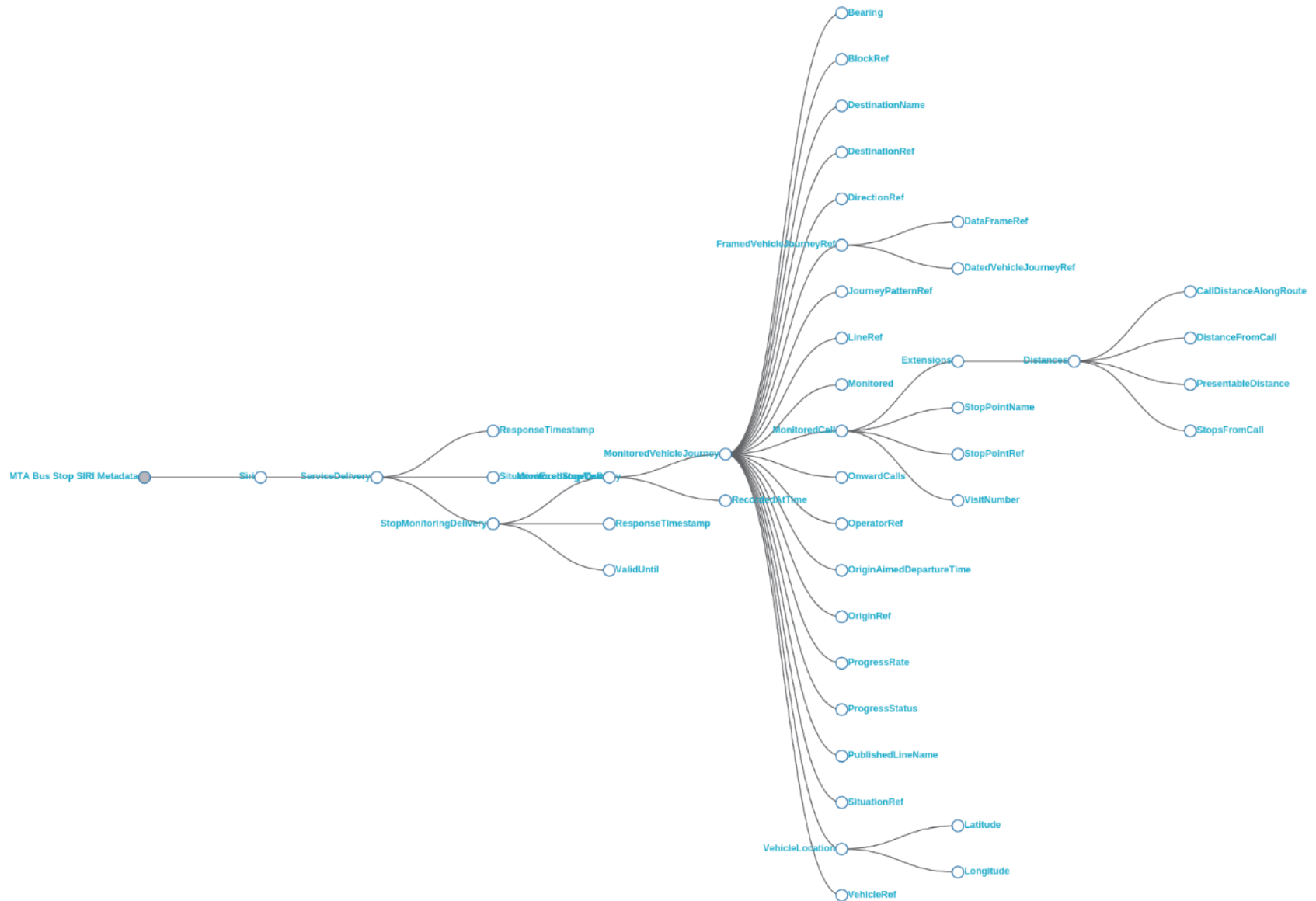


Figure 3. Metadata Visualization for Bus Stop SIRI Data



3.2 Methods

While all GTFS data needed to complete the conversion is available in tabular format that could be either loaded into a database or scanned programmatically, the resulting API server provides real-time information about the subway system in one of the world's largest cities. There can be hundreds of trains in the system in a given moment. The server must handle concurrent queries from app developers about these trains. Responses to these queries can be over a megabyte in size. These requirements place very steep demands on the server for high throughput with low latency, ruling out simple solutions to the straight-forward, one-to-one information mappings.

AVAIL identified a number of functional and nonfunctional system requirements. As such, AVAIL designed the server to handle schedule data and configuration updates seamlessly. The server is designed to be continuously available, as requiring restarts after updates would not be acceptable. Furthermore, the server is designed to make configuration and maintenance as easy as possible. This required creating a simple yet complete administrative console that is available even if the server is either not configured or misconfigured. The console provides feedback to guide administrators during the configuration process. The server is capable of being re-configured on the fly, and can handle GTFS data updates while continuing to service user requests.

AVAIL designed the server towards supporting any transit agency that adheres to the GTFS and GTFS-R standards, as well as be configurable in order to support agencies with GTFS-R feeds similar to the NYC MTA's Subway or the Long Island Rail Road feeds. These features required AVAIL to design the server to be universal, MTA agency specific information is hard-coded into the application but is only activated when MTA's `agency_id` is specified in the GTFS `agency.txt` file. The code base is also organized into modules to allow for clean extensibility and to support dissimilar feeds.

After visualizing both data structures, AVAIL mapped GTFS-R fields to SIRI fields. The vast majority of fields in GTFS-R and SIRI could be directly mapped, requiring only a formatting conversion. Since GTFS-R is encoded in protocol buffers, a binary data format, we used an existing open source library, `protobuf.js`, to decode the GTFS-R feed data for mapping to SIRI, an ASCII.

The functional requirements of the project were rather simple: convert the GTFS-R Subway feed into the SIRI format of the Bus Time system. Many elements in the conversion are straight-forward, one-to-one mappings that simply require looking up data in the GTFS feed. One key piece of information missing in the Subway's GTFS-R feed that is present in the Bus Time system's SIRI feed is vehicle location.

AVAIL developed a conversion tool that successfully translates every data field of the GTFS-R feed. The conversion tool also produces additional information based on the GTFS-R feed. For instance, SIRI includes latitude and longitude fields that are optional in GTFS-R data feeds. The MTA GTFS-R feeds, in particular, exclude these fields. Vehicle location is clearly very important for user facing apps, and luckily location is easy to infer from previous stop, next stop, elapsed travel time, and estimated time of arrival. The logic used to infer train locations is rather simple. When a train is at a stop, its location is clear. Otherwise, trains are moved along the route path specified in the GTFS schedule data using following simple formulas:

Equation 1.
$$D_{ratio_covered} = \frac{T_{cur} - T_{prev}}{T_{eta} - T_{prev}}$$

Where $D_{ratio_covered}$ is the ratio covered out of what distance remained between the train and its next stop at the time of the previous GTFS-R message T_{prev} . T_{cur} is the timestamp of the current message and T_{eta} is the estimated arrival time per the current message. If we let D_{along_prev} be the distance traveled from the previous stop at T_{eta} , and D_{prev_rem} be the distance that remained between a train and its next stop at T_{eta} , the formula for the current distance along the path, D_{along_cur} becomes

Equation 2.
$$D_{along_cur} = D_{along_prev} + (D_{prev_rem} * D_{ratio_covered})$$

The resulting location information is added to the GTFS-R feed during the conversion process. This mapping of trains is dependent upon completeness of the GTFS data. If spatial data and/or schedule data does not exist for a trip, we do not speculate concerning the future path of the train. The translation API includes a logging feature to collect data completeness information for MTA as a tool for assessing the quality of their GTFS-R data.

In most instances, locations of stops do not fall directly onto the lines defined in the GTFS paths.txt file. Stops must therefore be projected onto the path lines before computing the position of a vehicle. A least squares regression was used to make this projection under strict constraints that preserve the smoothness of a vehicle's journey. The projected point stands in for the stop when the system infers a vehicle's location. This path-fitting algorithm has the added benefit of finding GTFS data anomalies, which it logs when they exceed a threshold. Inspection of these logs will indicate where the GTFS data deserves data quality improvements.

The performance demands of this real-time system required the construction of numerous auxiliary data structures atop the GTFS and GTFS-R feed data. These provide instant retrieval of any required piece of information, there is no scanning for data. Furthermore, all information required to perform the GTFS-R to SIRI conversion is held entirely in fast access memory.

To understand the importance of this fast information retrieval, consider that inferring a MTA subway train's geographic position requires inspecting multiple entries in the GTFS-R feed, jumping to the relevant information from the GTFS stops.txt and shapes.txt files, then updating previously inferred data about the train. This must be done for the hundreds of vehicles in the transit system at any given moment, all while serving large amounts of data to the system users.

For ease of deployment, AVAIL wrapped a configurable server around the converter. By uploading a GTFS zip file or providing a URL pointing to a GTFS feed source, and specifying a URL for the GTFS-R feed, this server can be easily setup to convert GTFS-R data to SIRI by the MTA or other transit agencies.

3.3 Project Timeline

Included below, including percentage completed and projected percentage of completion by month

Table 4. Project Gantt Chart

	Project Timeline (July 1, 2015 - August 31, 2016)													
Task	Jul y	Au g	Sep t	Oc t	No v	De c	Ja n	Fe b	Ma r	Ap r	Ma y	Ju n	Ju l	Au g
1: Project Management	█	█	█	█	█	█	█	█	█	█	█	█	█	█
2: GTFS-Real Time to Siri API	█	█	█	█	█							█	█	
3: Processing Protocol Buffers		█	█											
4: GTFS-R To Siri Beta Availability			█	█	█									

4 Findings

The first major problem encountered while developing the conversion server was projecting the stops in the GTFS data onto the route shapes. While tracking trains along their route, stops are used as points of location certainty. Since stop coordinates do not necessarily fall on the shape of the routes that visit them, AVAIL needed to project stops onto the routes. In most cases, the stop is only a couple of feet from the shape, making the projection is simple. In some extreme cases, the stop coordinates are quite a distance away from the route. Stop to route path projections are made using a least square regression with the strict constraint that a train never backtracks.

AVAIL needed to develop a dynamic programming algorithm that performs the projections. It runs in $O(SW^2)$ time, where S is the number of stops along the route and W is the number of points in the route's geographic shape representation. This work is done in a separate process so as not to slow down the server. As the algorithm runs, any stop whose distance from the route serving it exceeds a threshold is logged to an output file to aid in anomaly correction. These logs have already led to the correction of couple of problematic routes.

The GTFS data is in tabular format and could easily be uploaded to database tables and queried using SQL. However, each of the trains in the system require dozens of queries to create the SIRI response. This would unavoidably result in thousands of disk seeks, regardless of what indices are built. The GTFS-R message also presents challenges. It arrives as an array of nested objects, containing arrays themselves. Without indices, nested scans would be required to find the information necessary to make the GTFS-R to SIRI conversion.

To provide the performance required for this server, all relevant GTFS and GTFS-R information is kept in memory. Indices are built atop this information to make it queryable by trip key, route key, and/or stop key. The expected $O(1)$ search time for all pieces of information sometimes required building lookup tables whose values are also lookup tables. To avoid redundancy, some indices are interconnected. Wrappers around the indices provide interfaces that abstract away the sometimes complicated implementation details and instead present an interface to developers with meaningful method names. The code that builds these indices was developed as separate projects, each with its own GitHub repository, so that developers could use these tools for other transit Projects.

4.1 Caching

Building the object representing the API query response and serializing it for transport to clients proved to be a critical bottleneck. When this was done for each response, the server could handle only a handful of concurrent users. To overcome this limitation, a custom caching layer was implemented.

For each GTFS-R message received, a SIRI response containing all the information for each train in the system is built. Then it is broken down into the largest possible chunks, which are serialized in both JSON and XML format and stored as raw binary data in contiguous memory blocks (buffers). Indices are built over these that enable expected $O(1)$ lookup of any information chunk. Ad hoc JSON or XML responses can then be generated by first using the indices themselves to determine what information belongs in a response, then using the indices to retrieve the relevant information from the buffers and sewing it together to create valid JSON or XML syntax. These responses are “memoized,” and simply timestamped if a matching query arrives before the information expires.

4.2 Event-based Logging

The GTFS-R to SIRI conversion system was built as a hierarchy of highly decoupled component modules. These components interact through interface APIs. This created a bit of a problem when it came to logging. Configuration needed to be centralized in the component atop the component hierarchy, but AVAIL did not want to complicate the inter-component interfaces by passing around loggers.

Instead, each component contains an event-emitter on which other components can register listeners. This allows decoupling of system components while allowing components to observe the state of other components. This event-based inter-component messaging system is currently used for logging statistics about the completeness of, and anomalies discovered within, the GTFS and GTFS-R data feeds, as well as the train tracking errors arising from these anomalies. The SIRI server—the component atop the system hierarchy—listens for events and writes them to log files. This setup allows administrators to control logging through the administrative console.

These logs are in machine readable format to allow automated analysis of the GTFS and GTFS-R data sources and the system’s behavior. These logs reveal what types of errors are degrading the information presented to transit system users. Because of the low coupling between components, this system is easily extensible and other loggers can be added if deemed necessary.

4.3 System Architecture

The GTFS-R to SIRI conversion system was built to serve a high volume of requests with low latency and high throughput. Dozens of data structures are built atop the GTFS, GTFS-R, and generated SIRI data to provide the fastest possible service to the system's users. For each system component, these data structures are stored entirely in memory and are centralized into a single object that we call the component's Data Store. The Data Stores encapsulate the state of the system and provide immediate access to all the information used to perform the GTFS-R to SIRI conversion.

Care was taken while designing the system's components to adhere, as much as possible, to the functional programming paradigm. This paradigm requires that the output value of a function depends only on the arguments that are input to the function, so calling a function f twice with the same value for an argument x will produce the same result $f(x)$ each time.

The GTFS and GTFS-R input feeds are treated as input streams, and the output of the respective system components depends only on the latest data received from these input streams. In fact, with exception of the SIRI message timestamps and the train tracking component that stores and updates train locations from one GTFS-R message to the next, all components throughout the system are pure functions.

Beyond the well-known theoretical and engineering benefits of the functional paradigm, the functional nature of the system components provides a huge benefit for testing and analysis: we can isolate the core system components and, using archived input messages, replay the system with complete access to each component's Data Store. These archived messages can be filtered to include only the information for a specific train or route.

This combination of cleanly organized data structures, thorough logging, and the ability to replay the system starting from a point in time for a specific train creates a powerful analysis tool with which we can identify, isolate, and inspect anomalous input as it moves through the system. Anomaly handling code can then be implemented to make the system more robust. The ability to isolate the system core and run archived messages through it also enables us to generate reports on the aggregate level. Weeks of data can be run through the system in hours, all while having access to the internal data structures and logging capabilities of the system.

4.4 Performance

After completion of the GTFS-R to SIRI conversion tool, AVAIL tested the server over the course of two months. The tool converts the GTFS-R data directly to SIRI with 100 percent fidelity. The SIRI translation feed contains an associated field that holds all data from the MTA GTFS-R.

As stated above, in the Methods section, the translation tool also offers information beyond what is contained in the GTFS-R feed. The most important of these is the addition of longitude and latitude during the conversion process. SIRI developers typically expect longitude and latitude in their data feed, but GTFS-R keeps these fields optional and MTA does not include longitude and latitude. AVAIL's conversion algorithm omits the latitude and longitude fields for unscheduled trips and unmapped scheduled trips because the locations of the vehicles cannot be inferred with an acceptable level of confidence.

To test the accuracy of the conversion tool in developing longitude and latitude of trains in the subway system, AVAIL integrated a data analysis feature that logs information on trips in each GTFS-R to SIRI message. This logging feature can be used by MTA to assess the quality of their GTFS-R data.

AVAIL also found minor MTA Subway GTFS data anomalies identified by our stops-to-paths fitting algorithm. These anomalies were quickly corrected by the MTA when brought to their attention. Running the algorithm on the data of other agencies led to the discovery of some interesting schedules, such as a bus line in upstate N.Y. that apparently serves Antarctica. These types of GTFS, GTFS-R, and SIRI data logging tools could be used by transit agencies to increase the accuracy of their real-time transit information.

Load testing of the server showed we are able to provide up to 1,000 simultaneous connections with response time less than one second. This scalability means that deploying this conversion service should be inexpensive for agencies looking to use it.

The combination of complete indexing and caching gives the server high throughput with low latency. With these optimizations, serving responses becomes simply a matter of low level memory copying and piping swaths of contiguous memory into the kernel's networking stack.

Figure 4 shows the server’s performance when all requests are for the largest possible response, which contains all information available about trains in the Subway System. These tests were run on an old desktop with 2G of memory and an Intel Core2 Duo CPU at 3.00GHz using Siege, an http load testing and benchmarking utility.

You can see that it handles over 550 transactions per second, with a throughput of over 371MB per second and an average latency of 0.40 seconds. Figure 5 shows the server’s performance for all types of requests, large and small. In this test, there are 3,956 unique requests being sent to the server. You can see that the server is able to handle over 1,400 requests per second with response sizes averaging around 11K. During this test the average latency was 0.01 seconds.

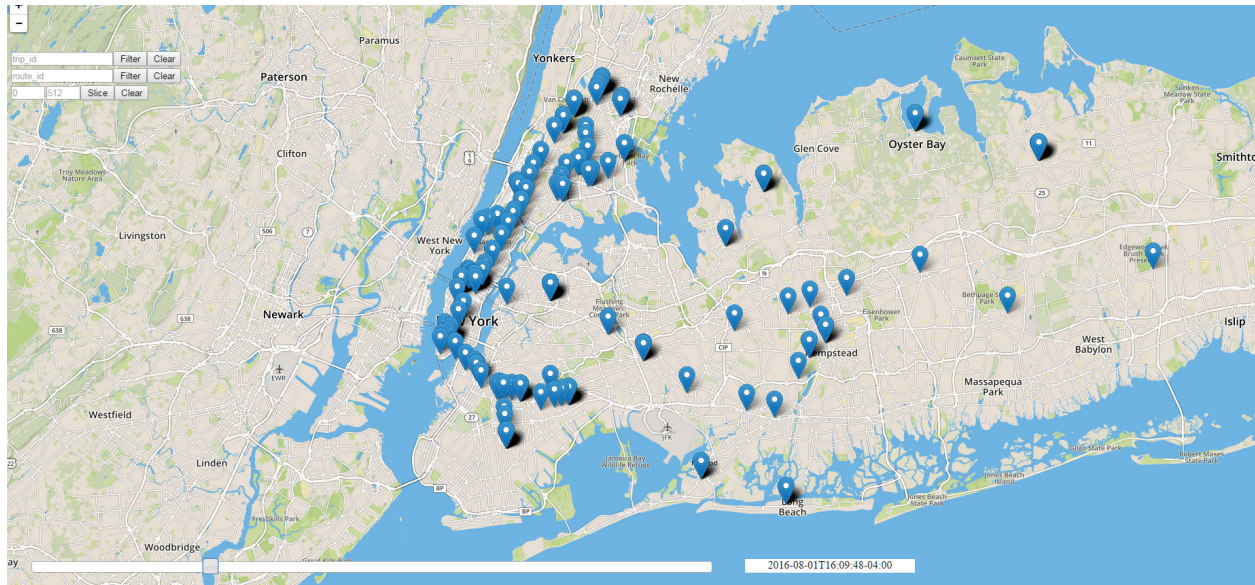
Figure 4. Output Data 1

Server Performance: Large Response Size	
Transactions:	9540734 hits
Availability:	100.00%
Elapsed time:	17298.57 secs
Data transferred:	6425950.50 MB
Response time:	0.40 secs
Transaction rate:	551.53 trans/sec
Throughput:	371.47 MB/sec
Concurrency:	223.07
Successful transactions:	9540694
Failed transactions:	281
Longest transaction:	64.54
Shortest transaction:	0.00

Figure 5. Output Data 2

A simple visualization of live trains was implemented to track and visualize real-time trains available at: <http://map.mta.availabs.org/>. This visualization shows all of the data collected from the MTA and Long Island Railroad real-time feeds using the GTFS-R to SIRI Conversion API. It updates with new data every 30 seconds, currently you need to refresh to get new data. To see a route, enter a number such as three, six, or nine into the middle filter and click “filter.”

Figure 6. Map Visualization of Real-time Subway Train Positions



4.5 Future Research

The GTFS-R feed currently contains three types of trips: scheduled with route shape, scheduled without route shape, and unscheduled. For the latter two, it is not possible to impute accurately the longitude and latitude of the train in real-time. Longitude and latitude are necessary data fields for SIRI, but are optional fields in the GTFS-R feed. Longitude and latitude are currently not included in MTA’s GTFS-R feed.

Further research is needed to correct all data quality issues in the GTFS-R feed, specifically for scheduled trips containing no route shape and unscheduled trips. Currently, 19 percent of all trips are unmapped in the SIRI conversion because of missing GTFS shape information. This is an easily correctable data quality issue. Twenty-five percent of all trips are unmapped because they are unscheduled. This problem is more difficult due to the real-time nature of these trips but it could be solved by developing a method for including route information for unscheduled trips. The logging feature of the API makes it possible

to locate and quantify these types of trips in MTA's GTFS-R feed. With this logged information, it is possible to improve the quality of the data in the GTFS-R feed and thereby improve the performance of the real-time SIRI feed. Alternatively, future research could focus on ways of imputing longitude and latitude from MTA's GTFS-R feed as it is currently compiled.

Future opportunities include the universal deployment of the conversion tool. More research is required to formally develop a truly expandable conversion API for universal deployment. Research could also be conducted into the usefulness of this conversion API to understand transit system reliability and performance under duress (e.g., blizzards and hurricanes).

5 Statement on Implementation

As previously noted, a real-time transit data API is useful to a transit agency both for internal tracking and for third party software applications that use the feed to display real-time transit information. MTA was using two different data formats to track and communicate real-time bus and subway information. Real-time bus information was communicated in the SIRI format, while real-time subway information was communicated using GTFS-R. MTA had found their SIRI API to be more popular with their developer community. The conversion of GTFS-R to SIRI is meant as a way of better implementing the real-time subway data into software applications by using the more popular format for both buses and subway trains. Therefore, the implementation of a real-time data conversion API is designed to be as close as possible to the current implementation method for SIRI real-time bus information from MTA. MTA will publicize the conversion API during its annual hackathon as well as through the press.

5.1 Technical Aspects of Implementation

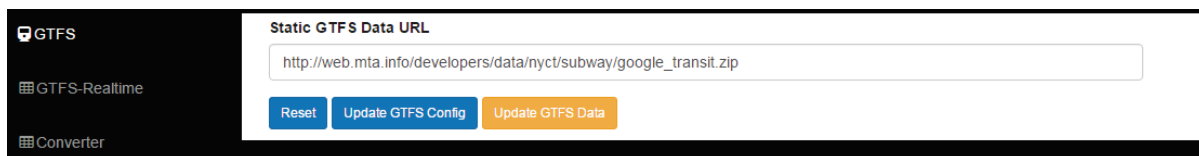
The code for this project is open source making the translation tool extendable by any agency using GTFS-R. Additionally the open source code has been abstracted (generalized) as much as possible to minimize the effort to implement at transit agencies other than MTA. The source code for the project can be found at: https://github.com/availabs/MTA_Subway_SIRI_Server.

5.1.1 Admin Configuration Console

The admin configuration console for the service can be accessed at: <http://siri.mta.availabs.org/console/>

The configuration console allows a user to set the GTFS for the conversion API by pointing the server to a URL where the data is hosted.

Figure 7. GTFS-R to SIRI Conversion API Control Console. Uploading Static GTFS Data by URL



Additionally, the configuration console allows the user to easily point the conversion API toward a GTFS-R Feed by way of a URL and an API Key.

Figure 8. GTFS-R to SIRI Conversion API Control Console. API Key and Retry Settings

The screenshot shows a configuration console with a dark sidebar on the left containing the following menu items: GTFS, GTFS-Realtime, and Converter. The main content area is titled "GTFS-Realtime Feed URL" and contains the following fields and buttons:

- GTFS-Realtime Feed URL:
- API Key:
- Read Frequency (seconds):
- Retry Frequency (seconds):
- Max number of retries:
- Buttons: "Reset" and "Update Config"

Lastly, a user can set logging flags for the converter through the configuration console.

Figure 9. GTFS-R to SIRI Conversion API Control Console. Logging Settings

The screenshot shows the same configuration console as Figure 8, but with the "Converter" menu item selected in the sidebar. The main content area is titled "Converter Logging Flags" and contains the following checked options and buttons:

- Train Locations
- Train Tracking Errors
- Train Tracking Statistics
- Unscheduled Trips
- No Spatial Data Trips
- Buttons: "Reset" and "Update Config"

5.2 API Calls

This section is a technical implementation description for software developers intending to utilize the API.

5.2.1 Monitoring Calls

The technical language in this section is exactly the same language used on MTA's website to aid software developers in implementing the SIRI real-time bus API.

5.2.1.1 SIRI Stop Monitoring Call

The SIRI `StopMonitoring` ("SIRI SM") call allows the developer to request information about the vehicles serving a particular stop. As much as possible, the values used both in the SM request and the SM response correspond to the values in the GTFS data for the B63.

5.3 SIRI StopMonitoring Requests

- The conversion API server parameters map one to one with the MTA's SIRI bus server parameters therefore the documentation for the bus API is valid documentation for the subway SIRI conversion API. The following parameter descriptions were taken from the MTA's SIRI bus server documentation:
- **key** - TBD
- **version** - which version of the SIRI API to use (1 or 2). Defaults to 1, but 2 is preferable.
- **OperatorRef** - the GTFS agency ID to be monitored (optional). Currently, all stops have operator/agency ID of MTA. If left out, the system will make a best guess. Usage of the `OperatorRef` is suggested, as calls will return faster when populated.
- **MonitoringRef** - the GTFS stop ID of the stop to be monitored (required). For example, 308214 for the stop at 5th Avenue and Union St towards Bay Ridge.
- **LineRef** - a filter by 'fully qualified' route name, GTFS agency ID + route ID (e.g. MTA NYCT_B63).
- **DirectionRef** - a filter by GTFS direction ID (optional). Either 0 or 1.
- **StopMonitoringDetailLevel** - Level of detail present in response. In order of verbosity:
 - `basic` - only available in version 2.
 - `normal` - default.
 - `calls` Determines whether or not the response will include the stops ("calls" in SIRI-speak) each vehicle is going to make **after** it serves the selected stop (optional).
- **MaximumNumberOfCallsOnwards** - Limits the number of `OnwardCall` elements returned in the query.
- **MaximumStopVisits** - an upper bound on the number of buses to return in the results.
- **MinimumStopVisitsPerLine** - a lower bound on the number of buses to return in the results per line/route (assuming that many are available)

An example URL, incorporating some of these GET parameters, is http://bustime.mta.info/api/siri/stop-monitoring.xml?key=##KEY##&OperatorRef=MTA&MonitoringRef=308209&LineRef=MTA NYCT_B63

5.3.1 SIRI StopMonitoring Responses

The XML below illustrates the SIRI StopMonitoring response. The comments in the XML below document the meaning of each of the XML elements that are unique to the SIRI StopMonitoring call. Much of the interesting data is contained inside the `MonitoredVehicleJourney` element, which has [its own documentation](#).

Note that the `MonitoredVehicleJourney` of a SIRI SM call will always have a `MonitoredCall` element. The first `OnwardCall` element (if requested) will always be the next stop the bus will make, which in most cases is a stop before the `MonitoredCall`.

Other notes about the particularities of `StopMonitoring` data is shown after the example response.

```
<!-- SIRI container elements -->
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ServiceDelivery>
    <!-- The timestamp on the MTA Bus Time server at the time the request was
    fulfilled -->
    <ResponseTimestamp>2011-01-28T14:49:58-05:00</ResponseTimestamp>
    <!-- SIRI container for VehicleMonitoring response data -->
    <StopMonitoringDelivery version="1.3">
      <!-- Same as above. Required by the SIRI spec. -->
      <ResponseTimestamp>2011-01-28T14:49:58-05:00</ResponseTimestamp>
      <!-- The time until which the response data is valid until. -->
      <ValidUntil>2011-01-28T14:50:58-05:00</ValidUntil>
      <!-- SIRI container for data about a particular vehicle service the
      selected stop -->
      <MonitoredStopVisit>
        <!-- The timestamp of the last real-time update from the particular
        vehicle -->
        <RecordedAtTime>2011-01-28T14:49:27-05:00</RecordedAtTime>
        <!-- A complete MonitoredVehicleJourney element. See the
        MonitoredVehicleJourney page for more information-->
        <MonitoredVehicleJourney>...</MonitoredVehicleJourney>
      </MonitoredStopVisit>
      ... More MonitoredStopVisit elements
    </StopMonitoringDelivery>
  </ServiceDelivery>
</Siri>
```

5.3.2 The SIRI Vehicle Monitoring call

The SIRI Vehicle Monitoring (“SIRI VM”) call allows the developer to request information about one, some, or all vehicles monitored by the MTA Bus Time system. As much as possible, the values used both in the VM request and the VM response correspond to the values in the GTFS data.

5.4 SIRI VehicleMonitoring Requests

The SIRI VehicleMonitoring API is available at: <http://siri.mta.availabs.org/api/siri/vehicle-monitoring.json>

Please note that the calls made without either a `VehicleRef` or `LineRef` produces quite a load on the system, so use them sparingly. Any developers found to be making repeated calls (e.g., at less than 30 second intervals) for all vehicles in the system may find their key revoked.

With the following GET parameters:

- **key** - TBD
- **version** - which version of the SIRI API to use (1 or 2). Defaults to 1, but 2 is preferable.
- **OperatorRef** - the GTFS agency ID to be monitored (optional). Currently `MTA`.
- **VehicleRef** - the ID of the vehicle to be monitored (optional). This is the 4-digit number painted on the side of the bus, for example `7560`. Response will include all buses if not included.
- **LineRef** - a filter by 'fully qualified' route name, GTFS agency ID + route ID (optional).
- **DirectionRef** - a filter by GTFS direction ID (optional). Either `0` or `1`.
- **VehicleMonitoringDetailLevel** - Level of detail present in response. In order of verbosity:
 - `basic` - only available in version 2.
 - `normal` - default.
 - `calls` Determines whether or not the response will include the stops ("calls" in SIRI-speak) each vehicle is going to make after it serves the selected stop (optional).
- **MaximumNumberOfCallsOnwards** Limit on the number of `OnwardCall` elements for each vehicle when `VehicleMonitoringDetailLevel=calls`
- **MaximumStopVisits** - an upper bound on the number of buses to return in the results.
- **MinimumStopVisitsPerLine** - a lower bound on the number of buses to return in the results per line/route (assuming that many are available)

5.4.1 SIRI VehicleMonitoring Responses

The XML below illustrates the SIRI VehicleMonitoring response. The comments in the XML below document the meaning of each of the XML elements that are unique to the SIRI VehicleMonitoring call. Much of the interesting data is contained inside the `MonitoredVehicleJourney` element, which has [its own documentation](#).

```
<!-- SIRI container elements -->
<Siri version="1.3" xmlns="http://www.siri.org.uk/siri">
  <ServiceDelivery>
    <!-- The timestamp on the MTA Bus Time server at the time the request was
    fulfilled -->
    <ResponseTimestamp>2011-01-28T14:22:48-05:00</ResponseTimestamp>
    <!-- SIRI container for VehicleMonitoring response data -->
    <VehicleMonitoringDelivery version="1.3">
      <!-- Same as above. Required by the SIRI spec. -->
      <ResponseTimestamp>2011-01-28T14:22:48-05:00</ResponseTimestamp>
      <!-- The time until which the response data is valid until. -->
      <ValidUntil>2011-01-28T14:23:48-05:00</ValidUntil>
      <!-- SIRI container for data about a particular vehicle -->
      <VehicleActivity>
        <!-- The timestamp of the last real-time update from the particular
        vehicle -->
        <RecordedAtTime>2011-01-28T14:22:30-05:00</RecordedAtTime>
        <!-- A MonitoredVehicleJourney element for a vehicle in revenue
        service. Please See the MonitoredVehicleJourney page for a thorough
        discription -->
        <MonitoredVehicleJourney>
          ...
        </MonitoredVehicleJourney>
      </VehicleActivity>
      <!-- The next vehicle. -->
      <VehicleActivity>
        <RecordedAtTime>2011-01-28T14:22:43-05:00</RecordedAtTime>
        <!-- A complete MonitoredVehicleJourney element. See the
        MonitoredVehicleJourney page for more information-->
        <MonitoredVehicleJourney>...</MonitoredVehicleJourney>
      </VehicleActivity>
      ... More VehicleActivity elements
    </VehicleMonitoringDelivery>
  </ServiceDelivery>
</Siri>
```

5.5 Links to the GTFS-R to SIRI API

The server that is running a real-time SIRI feed for the MTA subway system is available here at <http://siri.mta.availabs.org/>. The full data is available at the following URLs:

- <http://siri.mta.availabs.org/api/siri/vehicle-monitoring.json>
- <http://siri.mta.availabs.org/api/siri/vehicle-monitoring.xml>
- <http://siri.mta.lirr.availabs.org/api/siri/vehicle-monitoring.json>
- <http://siri.mta.lirr.availabs.org/api/siri/vehicle-monitoring.xml>
- <http://siri.mta.lline.availabs.org/api/siri/vehicle-monitoring.json>
- <http://siri.mta.lline.availabs.org/api/siri/vehicle-monitoring.xml>
- <http://siri.mta.statenisland.availabs.org/api/siri/vehicle-monitoring.json>
- <http://siri.mta.statenisland.availabs.org/api/siri/vehicle-monitoring.xml>

5.6 Marketing Strategies Recommended

As part of testing the conversion API, MTA Developers from the Google Developers User Group were invited to test the tool in their existing applications. The MTA Facebook Developers User Group consists of software developers who are already using the GTFS-R or SIRI APIs in their transit applications. These software developers are the intended audience and will be marketing the Conversion API as a result of marketing their real-time transit applications.

6 Conclusion

In order to offer their developers a harmonized real-time transit information system, MTA needed to be able to convert GTFS-R for the subway system to SIRI, the real-time format used for their bus system. The solution included: mapping GTFS-R fields to SIRI fields using GTFS as a “spatial backbone”; the analysis and algorithmic correction of anomalous data and unscheduled trips; the development of a configurable server for implementation; and published as abstracted open source code for generalizable deployment.

More research is required to formally develop a truly expandable conversion API for universal deployment. One area of research in this regard is to determine best practices for mapping the longitude and latitude of all trips - scheduled and unscheduled - in the SIRI feed. This research requires mapping unscheduled trip IDs to corresponding GTFS data.

Additionally, research should be conducted into the usefulness of this conversion API to understand transit system reliability and performance under duress (e.g., blizzards and hurricanes).

When developers utilize the GTFS-R to SIRI Conversion API in the development of transit apps, transit riders will have the benefit of seamless multi-modal trip planning in real-time. In addition, there will be new opportunities for agencies to monitor, analyze, and visualize system-wide operations performance. This should yield cost savings, efficiencies, and service improvements.

7 Bibliography

- Benn, H. P. *TCRP Synthesis 10 Bus Route Evaluation Standards*. Transportation Research Board, National Research Council, Washington, D. C. 1995.
- Furth, P. G. *TCRP Synthesis 34 Data Analysis for Bus Planning and Monitoring*. Transportation Research Board, National Research Council, Washington, D. C. 1998.
- No Author. (2015). *What is GTFS-realtime?*. Retrieved from <https://developers.google.com/transit/gtfs-realtime/?hl=en>
- No Author. (2011). *SIRI Home Page*. Retrieved from <http://user47094.vs.easily.co.uk/siri/overview.htm>
- No Author. (2015). *What is GTFS?*. Retrieved from <https://developers.google.com/transit/gtfs/?hl=en>
- No Author. (2015). *General Transit Feed Specification Reference*. Retrieved from <https://developers.google.com/transit/gtfs/reference>
- Okunieff, P.E., *TCRP Synthesis 24 AVL Systems for Bus Transit*, Transportation Research Board, National Research Council, Washington, D. C. 1997.
- Schweiger, C. L. *TCRP Synthesis 48 Real-Time Bus Arrival Information Systems*. Transportation Research Board, National Research Council, Washington, D. C. 2015.


```
"departure": {
  "delay": null,
  "time": {
    "low": 1457434170,
    "high": 0,
    "unsigned": false
  },
  "uncertainty": null
},
"schedule_relationship": 0,
".nyct_stop_time_update": {
  "scheduled_track": "4",
  "actual_track": null
},
"_departure": {
  "time": 1457434170
}
},
{
  "stop_sequence": null,
  "stop_id": "501N",
  "arrival": {
    "delay": null,
    "time": {
      "low": 1457439090,
      "high": 0,
```

```
        "unsigned": false
      },
      "uncertainty": null
    },
    "departure": null,
    "schedule_relationship": 0,
    ".nyct_stop_time_update": {
      "scheduled_track": "2",
      "actual_track": null
    },
    "_arrival": {
      "time": 1457439090
    }
  }
],
  "timestamp": null
},
  "vehicle": null,
  "alert": null
}
"cause": 1,
  "effect": 8,
  "url": null,
  "header_text": {
    "translation": [
      {
```

```

        "text": "Train delayed",
        "language": null
    }
]
},
"description_text": null
}

```

A.2 SIRI Feed

```

"Siri": {
  "ServiceDelivery": {
    "ResponseTimestamp": "2016-03-21T23:11:18-04:00",
    "VehicleMonitoringDelivery": [
      {
        "ResponseTimestamp": "2016-03-21T23:11:18-04:00",
        "ValidUntil": "2016-03-08T05:16:38-05:00",
        "VehicleActivity": [
          {
            "MonitoredVehicleJourney": {
              "LineRef": "MTA NYCT_1",
              "DirectionRef": null,
              "FramedVehicleJourneyRef": {
                "DataFrameRef": "2016-03-08",
                "DatedVehicleJourneyRef": "MTA
NYCT_A20151206WKD_026550_1..S02R"
              },
              "JourneyPatternRef": "MTA NYCT_1..S02R",
              "PublishedLineName": "1",

```

```
"OperatorRef": "MTA NYCT",
"OriginRef": null,
"DestinationRef": "MTA_140S",
"DestinationName": "SOUTH FERRY LOOP",
"OriginAimedDepartureTime": "2016-03-08T04:25:30-05:00",
"SituationRef": null,
"Monitored": true,
"VehicleLocation": {
  "Latitude": null,
  "Longitude": null
},
"Bearing": null,
"ProgressRate": null,
"ProgressStatus": null,
"VehicleRef": "MTA NYCT_01 0425+ 242/SFY",
"MonitoredCall": {
  "Extensions": {
    "Distances": {
      "PresentableDistance": null,
      "DistanceFromCall": null,
      "StopsFromCall": 0,
      "CallDistanceAlongRoute": null
    }
  }
},
"ExpectedArrivalTime": "2016-03-08T05:16:38-05:00",
"ExpectedDepartureTime": "2016-03-08T05:18:38-05:00",
```

```

"StopPointRef": "MTA_137S",
"StopPointName": "Chambers St",
"VisitNumber": 1
},
"OnwardCalls": {
  "OnwardCall": [
    {
      "Extensions": {
        "Distances": {
          "PresentableDistance": null,
          "DistanceFromCall": null,
          "StopsFromCall": 0,
          "CallDistanceAlongRoute": null
        }
      },
      "ExpectedArrivalTime": "2016-03-08T05:16:38-05:00",
      "ExpectedDepartureTime": "2016-03-08T05:18:38-05:00",
      "StopPointRef": "MTA_137S",
      "StopPointName": "Chambers St",
      "VisitNumber": 1
    },
    {
      "Extensions": {
        "Distances": {
          "PresentableDistance": null,
          "DistanceFromCall": null,

```

```
        "StopsFromCall": 1,
        "CallDistanceAlongRoute": null
    }
},
"ExpectedArrivalTime": "2016-03-08T05:20:08-05:00",
"ExpectedDepartureTime": "2016-03-08T05:20:08-05:00",
"StopPointRef": "MTA_138S",
"StopPointName": "Cortlandt St",
"VisitNumber": 1
},
{
"Extensions": {
    "Distances": {
        "PresentableDistance": null,
        "DistanceFromCall": null,
        "StopsFromCall": 2,
        "CallDistanceAlongRoute": null
    }
},
"ExpectedArrivalTime": "2016-03-08T05:21:38-05:00",
"ExpectedDepartureTime": "2016-03-08T05:22:38-05:00",
"StopPointRef": "MTA_139S",
"StopPointName": "Rector St",
"VisitNumber": 1
},
{
```

```
    "Extensions": {
      "Distances": {
        "PresentableDistance": null,
        "DistanceFromCall": null,
        "StopsFromCall": 3,
        "CallDistanceAlongRoute": null
      }
    },
    "ExpectedArrivalTime": "2016-03-08T05:25:08-05:00",
    "ExpectedDepartureTime": null,
    "StopPointRef": "MTA_140S",
    "StopPointName": "South Ferry Loop",
    "VisitNumber": 1
  },
],
```

```
"SituationExchangeDelivery": [
  {
    "Situations": {
      "PtSituationElement": {
        "SituationNumber": null,
        "PublicationWindow": {
          "StartTime": null,
          "EndTime": null
        },
        "Severity": "undefined",
        "Summary": "GTFS-Realtime train delay alert.",

```

"Description": "The following trains are delayed: 05 0512 180/FLA.The following routes are affected: 5.",

```
"Affects": {  
  "VehicleJourneys": [  
    [  
      {  
        "LineRef": "MTA 5",  
        "DirectionRef": "1"  
      }  
    ]  
  ]  
},  
"Consequences": {  
  "Consequence": [  
    {  
      "Condition": "DELAYED"  
    }  
  ]  
}
```

}

NYSERDA, a public benefit corporation, offers objective information and analysis, innovative programs, technical expertise, and support to help New Yorkers increase energy efficiency, save money, use renewable energy, and reduce reliance on fossil fuels. NYSERDA professionals work to protect the environment and create clean-energy jobs. NYSERDA has been developing partnerships to advance innovative energy solutions in New York State since 1975.

To learn more about NYSERDA's programs and funding opportunities, visit nyserdera.ny.gov or follow us on Twitter, Facebook, YouTube, or Instagram.

**New York State
Department of Transportation**

50 Wolf Road
Albany, NY 12232

telephone: 518-457-6195

dot.ny.gov

**New York State
Energy Research and
Development Authority**

17 Columbia Circle
Albany, NY 12203-6399

toll free: 866-NYSERDA
local: 518-862-1090
fax: 518-862-1091

info@nyserdera.ny.gov
nyserdera.ny.gov



NYSERDA
Department of
Transportation

State of New York

Andrew M. Cuomo, Governor

New York State Energy Research and Development Authority

Richard L. Kauffman, Chair | John B. Rhodes, President and CEO

New York State Department of Transportation

Joan McDonald, Commissioner

