Overview

- Why Open Data?
- Anatomy of Transit Data Sharing
- Being Developer-Friendly
WHY OPEN DATA?

What is open data?

• Transit data that is shared with the public
  – Typically shared via website/FTP site/web services
    • No login should be required (may use API key)
  – Should be updated regularly, with any changes in schedule/routes/stops
Open [Data ≠ Architecture ≠ Source]

- **Open architectures** mostly focus on:
  - Standards *within* an agency’s software/hardware systems
  - Interconnectivity with other government systems
- **Open source** means software source code is available
- **Open data** is the sharing of data with *external public parties*

![Diagram of data flow]

Why is open data important?

- Allows public to contribute services that are cost/time-prohibitive for the public sector
  - e.g., many mobile platforms
- Vendors are unpredictable
  - Some agencies have shared data only with Google
  - When Apple dropped Google Maps, iPhone users lost transit directions
  - Apple relied on 3rd party apps to fill the gap – only possible if open data was available
Why is open data important (to developers)?

• Developers want to create innovative apps that meet a need!
  – Some are monetized, some are not

• If you don’t provide open data, developers will often improvise
  – …via website scraping, etc.
  – Prone to breaking
  – Not beneficial to agency or rider

THE ANATOMY OF TRANSIT DATA SHARING
Two Types of Open Data

1. Static
   - e.g., Transit schedules / routes / stops
   - Change only a few times a year

2. Real-time
   - e.g., Estimated arrival times / vehicle positions / service alerts
   - Can change every few seconds

Two Magnitudes of Open Data

A. “Fire hose”
   - A dump of the complete state of the transit system
   - Not directly suitable for mobile devices
     - Static -> All transit schedules/routes/stops
     - Real-time -> All estimated arrivals/vehicle positions/service alerts

B. “Faucet”
   - Precise subset of transit data
   - Suitable for mobile devices
     - Static -> “Stop ID 10 is served by Route 5”
     - Real-time -> “It is 2 minutes until Route 5 bus arrives at Stop ID 10”
Transit Data Flow Architecture

Producer

Aggregator/Filter

Consumer

Open Data ("Fire hose")

Open Data ("Faucet")
Commonly-used “fire hose” formats

- static
- realtime

General Transit Feed Spec. (GTFS)
GTFS-realtime
Service Interface for Real-time Information (SIRI)

Produce
Aggregator/Filter

Open Data
(“Firehose”)
GTFS/GTFS-realtime format: http://goo.gl/tmwv8
SIRI format: http://goo.gl/Vnpyv
TCIP format: http://goo.gl/vd6kY

Transit Data Flow Architecture

Producer
Aggregator/Filter
Consumer

Open Data
(“Fire hose”)
Open Data
(“Faucet”)
Common “faucet” formats still emerging

- static
- realtime

Vendor/Agency-specific formats
OneBusAway API
SIRI (REST/JSON format)
Vendor/Agency-specific formats
OneBusAway API

Aggregator/Filter
Consumer

Open Data (“faucet”)

Vendor/Agency formats - http://goo.gl/NtNJ0
OneBusAway format - http://goo.gl/QXXyj
SIRI REST format - http://goo.gl/lOPzT

Example – Google Transit

Bay Area Rapid Transit (BART) Vehicles/Servers
Google Servers
Google Transit Mobile App

Static - GTFS
Realtime - GTFS-realtime
Example – Google Transit

BART Vehicles/Servers → Google Servers → Google Transit Mobile App

Static - GTFS
Realtime - GTFS-realtime Open to Public

Any 3rd party

Example – HART in Tampa, FL

HART Vehicles/Servers → USF Server → OneBusAway 3rd Party mobile apps

Static – GTFS (HART) Static & Real-time - OneBusAway API
Realtime - GTFS-realtime (USF)

More at http://goo.gl/iqHD2
Example – MTA BusTime in NY

MTA Vehicles

MTA BusTime Servers

3rd Party Mobile Apps

Static - GTFS

Static - OneBusAway API

Real-time - SIRI REST API for mobile

Successful Open Data Formats Are...

- **Organic**
  - Created and improved by the people actually producing and consuming the data

- **Open**
  - Open process for evolution
  - Data/documentation not hidden behind log-ins

- **Easy-to-use** for app developers
  - Is documentation simple to understand?
  - Are there existing open-source software tools?
  - Is data provided via best practice web service design (e.g., using RESTful API with JSON, instead of SOAP with XML)?
General Transit Feed Specification (GTFS)

• Created by TriMet and Google in 2005
• Has become a de facto standard world-wide for static transit schedule/route/stop data

GTFS data consists of multiple text files

GTFS data powers Google Transit and other apps

General Transit Feed Specification (GTFS)

• Over 500 agencies worldwide have transit data in GTFS format[1]
  – 49 of top 50 largest U.S. transit agencies share GTFS data, over 227 worldwide
  – At least 20 Canadian agencies share open data
• Most agencies created GTFS data for Google Transit
  – But, GTFS is open-data format used by web/mobile apps, OpenTripPlanner, OneBusAway, etc.[2]
• See “GTFS Data Exchange” for list of agencies with GTFS data
  – Or, ask your local agency

[2] In such GTFS data and otherwise, see paper co-authored by Sean Barbeau and Aaron Antrim – “The Many Uses of GTFS Data” – http://goo.gl/asR96
Promoting app development with open data

BEING DEVELOPER-FRIENDLY

Create a relationship with developers

- Open your GTFS data, and share on GTFS-Data-Exchange!
  - GTFS data should not be password or login protected
- Share real-time data too (national list pending)
- Create a “Developer page” with access to resources (e.g., GTFS license, data)
- Create developer email list/group for announcements/Q&A/collaboration
- Announce resources on “Transit Developers” group[1]

HART Developer page - http://www.gohart.org/developers/

Be Developer-Friendly!

• Use a simple “Terms of Service” based on existing industry examples\[1\][2][3][4][5]

• Use GTFS naming conventions throughout
  • “Direction_ID” is 0/1 (not N/S/E/W) in real-time data too!

• Make sure IDs match among datasets
  – E.g., tripID in real-time data matches GTFS tripID

---

Be Developer-Friendly!

• Use developer/mobile-friendly formats
  1. For data – GTFS, GTFS-realtime, SIRI REST API (see MTA NY BusTime API\[1\])
  2. For mobile APIs – RESTful web services design and JSON encoding preferred (not SOAP and XML)
Be Developer-Friendly!

• Use developer/mobile-friendly formats
  1. For data – GTFS, GTFS-realtime, SIRI REST API (see MTA NY BusTime API[1])
  2. For mobile APIs – RESTful web services design and JSON encoding preferred (not SOAP and XML)

SOAP Request

HTTP-Post Request

• 3.7 times more characters using SOAP!


Be Developer-Friendly!

• Use developer/mobile-friendly formats
  1. For data – GTFS, GTFS-realtime, SIRI REST API (see MTA NY BusTime API[1])
  2. For mobile APIs – RESTful web services design and JSON encoding preferred (not SOAP and XML)

SOAP Request

HTTP-Post Request

• 3.7 times more characters using SOAP!

XML Response

JSON Response

• 1.8 times more characters using XML!

SOAP vs. HTTP

Using HTTP Increases Battery Life by 28% on Avg.

XML vs. JSON Parsing Time – Samsung Galaxy S3

- ~4.3 times longer to parse the first response using XML
- First response time is critical for mobile apps, since application state is often destroyed when user multitasks (checks email, etc.) on their phone

- See http://goo.gl/hq6nE for details
- See http://goo.gl/EHY5l for details
Get the word out!

• After developers have created mobile apps, share them with riders
• Consider an “App Center”[1-9] to showcase apps

These practices lead to great apps!

• For more open data-powered apps and specific best practices for creating/maintaining GTFS data:
Conclusions

• Open data (e.g., GTFS) makes transit apps possible
• Understand open [data vs. architecture vs. source]
• Understand the differences in data:
  – Static vs. real-time
  – “Fire hose” vs. “Faucet”
• Understand that certain formats are more appropriate than others for certain situations (e.g., mobile)
• Being developer-friendly encourages mobile app development!

Thanks!

Sean J. Barbeau, Ph.D.
barbeau@cutr.usf.edu
813.974.7208

Principal Mobile Software Architect for R&D
Center for Urban Transportation Research
University of South Florida
Glossary

• API – Application Programming Interface
• AVL – Automatic Vehicle Location
• FTP – File Transfer Protocol
• GTFS – General Transit Feed Specification
• HTTP – HyperText Transfer Protocol
• IT – Information Technology
• JSON – Javascript Object Notation
• REST – Representational State Transfer
• SIRI -Service Interface for Real time Information
• TCIP - Transit Communications Interface Profiles
• XML – Extensible Markup Language