# GTFS-realtime introduction

Kurt Raschke
kurt@kurtraschke.com

# Why do standards for transit data matter?

- Standards serve as a force multiplier for civic software
- Producers and consumers benefit from standardized, interoperable formats
- Agencies can use open standards to drive procurement

# What is GTFS-realtime?

- Real-time complement to GTFS
- Open standard
- Extensible
- Synoptic
- Packaged in Protocol Buffers

# What is GTFS-realtime not?

- GTFS-realtime is **not** for resource-constrained consumers who only need data concerning a slice of the overall transit network

- GTFS-realtime is **not** inherently a source of historical data, unless it has been archived somewhere

# Synoptic vs. piecewise

**Synoptic feeds:**

- communicate the *entire* state of a transit network
- are efficient for system-to-system communication (e.g. trip planner, digital signage headend)

**Piecewise feeds:**

- communicate the state of a particular slice of the transit network
- are ideal for individual consumers (e.g. mobile apps, individual digital signs)

- Easy to slice up a synoptic feed for piecewise clients; harder (potentially *much* harder) to go the other way

# Re-packaging GTFS-realtime feeds for consumers

- OneBusAway and OpenTripPlanner (both open source) can be used to consume a GTFS-realtime feed and provide a REST API for downstream consumers

- Some agencies also provide bespoke piecewise feeds (see, e.g. https://www.mbta.com/developers/v3-api)

# What do GTFS-realtime feeds contain?

- Trip Updates
- Vehicle Positions
- Alerts

# What do GTFS-realtime feeds contain?

- **Trip Updates**
- Vehicle Positions
- Alerts

```
entity {
  id: "40166022"
  trip_update {
    trip {
      trip_id: "40166022"
      start_date: "20190329"
      route_id: "80"
    }
    stop_time_update {
      stop_sequence: 30
      arrival {
        time: 1553909756
      }
      departure {
        time: 1553909756
      }
      stop_id: "2411"
    }
    stop_time_update {
      stop_sequence: 31
      arrival {
        time: 1553909816
      }
      departure {
        time: 1553909816
      }
      stop_id: "2412"
    }
    vehicle {
      id: "y1401"
      label: "1401"
    }
  }
}
```

**What trip is this?**

**When is the vehicle arriving at and departing from this stop?**

(Times are POSIX timestamps, that is, seconds since midnight UTC, January 1, 1970)

**What vehicle is serving this trip?**
(Optional!)

# What do GTFS-realtime feeds contain?

- **Trip Updates**
- Vehicle Positions
- Alerts

```
entity {
  id: "5131858WKDY"
  trip_update {
    trip {
      trip_id: "5131858WKDY"
    }
    stop_time_update {
      stop_sequence: 10
      arrival {
        delay: 6
        time: 1553913876
        uncertainty: 30
      }
      departure {
        delay: 6
        time: 1553913909
        uncertainty: 30
      }
      stop_id: "EMBR"
    }
    stop_time_update {
      stop_sequence: 11
      arrival {
        delay: 0
        time: 1553913972
        uncertainty: 30
      }
      departure {
        delay: 0
        time: 1553914002
        uncertainty: 30
      }
      stop_id: "MONT"
    }
  }
}
```

**Arrival and departure times can also be expressed relative to the schedule, and can include uncertainty.**

# What do GTFS-realtime feeds contain?

- Trip Updates
- **Vehicle Positions**
- Alerts

```
entity {
  id: "y1642"
  vehicle {
    trip {
      trip_id: "39996581"
      schedule_relationship: SCHEDULED
      route_id: "34E"
    }
    position {
      latitude: 42.30271
      longitude: -71.109924
      bearing: 0.0
    }
    current_status: IN_TRANSIT_TO
    timestamp: 1553913346
    vehicle {
      id: "y1642"
      label: "1642"
    }
  }
}
```

**What trip is the vehicle operating?**
(Optional!)

**Where is the vehicle?**

**What is the vehicle's identity?**

# What do GTFS-realtime feeds contain?

- Trip Updates
- Vehicle Positions
- **Alerts**

```
entity {
  id: "302361"
  alert {
    active_period {
      start: 1554712200
      end: 1554877800
    }
    informed_entity {
      stop_id: "70087"
    }
    informed_entity {
      stop_id: "70088"
    }
    informed_entity {
      stop_id: "door-shmnl-sydney"
    }
    cause: MAINTENANCE
    effect: OTHER_EFFECT
    header_text {
      translation {
        text: "Savin Hill Elevator 947 (Sydney Street to unpaid lobby)
unavailable from Mon Apr 8 through Tue Apr 9 due to maintenance"
        language: "en"
      }
    }
    description_text {
      translation {
        text: "To access the station, please use the main entrance on
Savin Hill Avenue by exiting the parking lot and turning right to go up
the hill to the main entrance. Please contact a station personnel for
assistance."
        language: "en"
      }
    }
  }
}
```

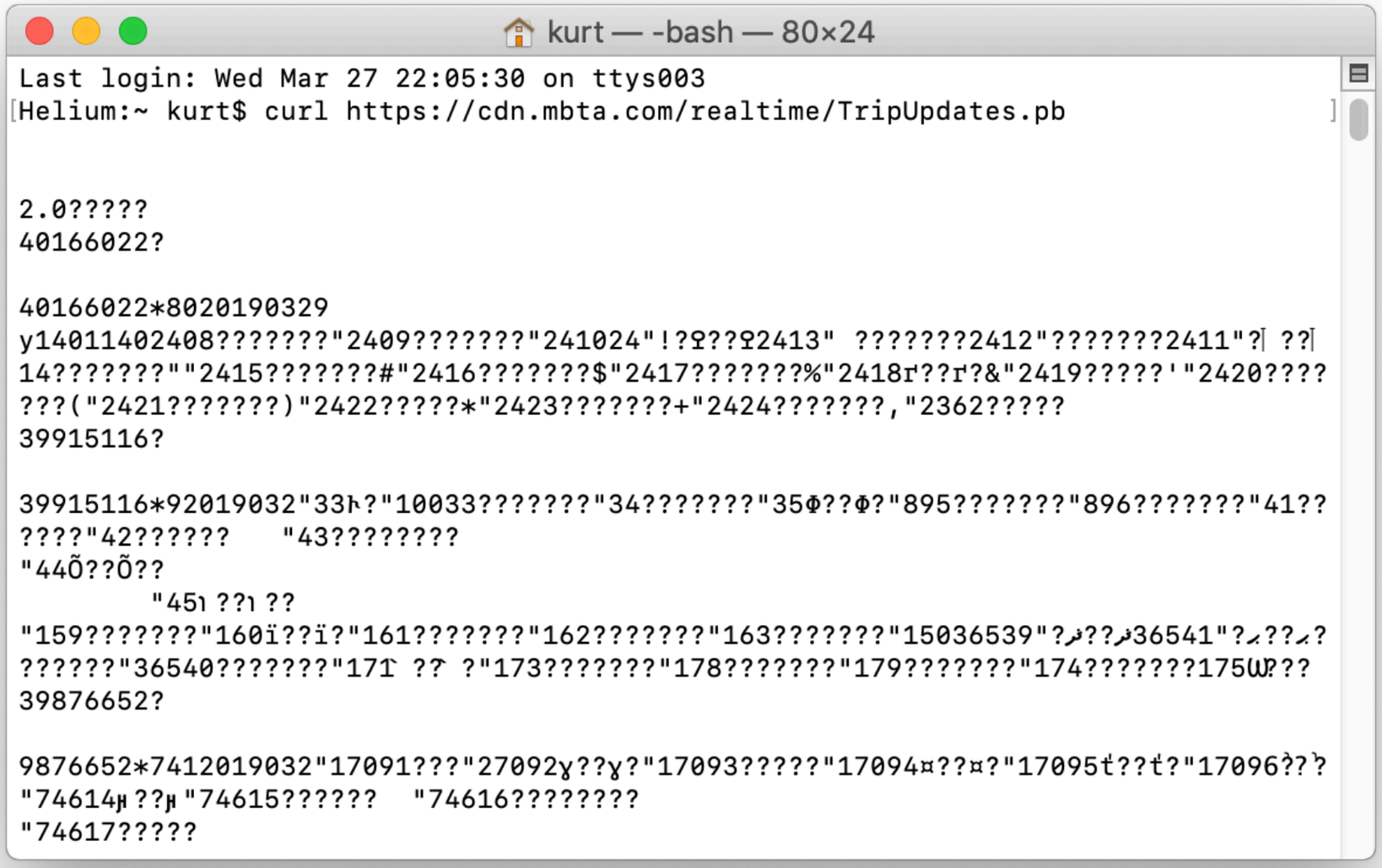
**When is this alert active?**

**Who is affected by this alert?**

# What do GTFS-realtime feeds contain?

- Trip Updates
- Vehicle Positions
- **Alerts**

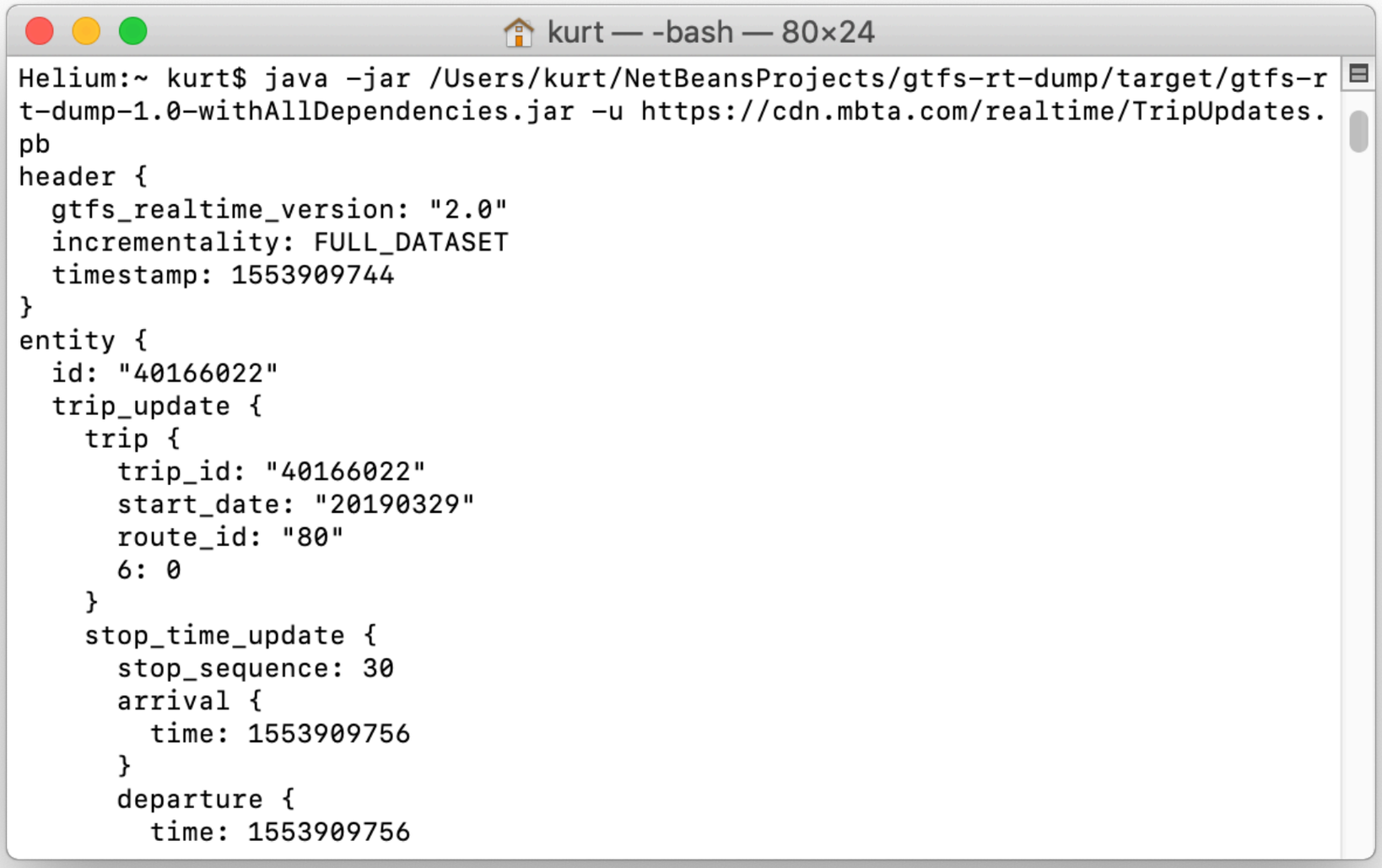
```
entity {
  id: "302490"
  alert {
    active_period {
      start: 1553907174
      end: 1553921361
    }
    informed_entity {
      agency_id: "1"
      route_id: "426"
      route_type: 3
    }
    cause: POLICE_ACTIVITY
    effect: OTHER_EFFECT
    header_text {
      translation {
        text: "Route 426 experiencing delays of up to 25 minutes due to
police action"
        language: "en"
      }
    }
    description_text {
      translation {
        text: ""
        language: "en"
      }
    }
  }
}
```




# Protocol Buffers?

# Protocol Buffers

- Binary serialization format designed by Google
- More efficient than XML or JSON
- Google-maintained libraries for Java, Python, C++; community-maintained libraries for many other languages
- Not directly human-readable, but debugging tools exist
- Pre-built GTFS-realtime bindings for .NET, Java, JavaScript, PHP, Python, Ruby, and Go: https://github.com/google/gtfs-realtime-bindings




# Use tools!

https://github.com/kurtraschke/gtfs-rt-dump
https://github.com/laidig/gtfs-rt-printer

# Producing GTFS-realtime

- Native feeds are best, but…
- Converting legacy APIs to GTFS-realtime is not impossible (and can be fairly easy in some cases) https://kurtraschke.com/2015/01/legacy-avl-export
- What if you only have vehicle positions? Use a prediction engine! https://github.com/TheTransitClock/transitime

# Consuming GTFS-realtime: passenger information

- Consuming data directly on a resource-limited endpoint (mobile device, Raspberry Pi driving a digital sign, etc.)? Consider using a piecewise API, whether provided by the transit agency, or filtered through OneBusAway or OpenTripPlanner.

- Or, build your own backend…

# Consuming GTFS-realtime: research and analysis

- Tools exist to automatically consume a GTFS-realtime feed and store its contents in a database: https://github.com/CUTR-at-USF/gtfsrdb

- How to keep an archiver running? Consider the free tier many cloud services offer, or a Raspberry Pi hiding under your desk…

- Someone may already be archiving data!

# Questions?