MBTA-REALTIME QUICK-START GUIDE

1. MBTA OPEN DATA OVERVIEW

This document is an introduction to the MBTA-realtime API, a full-featured easy-to-use RESTful API with schedule, alert, vehicle location, and arrival prediction data. The MBTA also provides GTFS, GTFS-realtime, and the RSS feeds, documented at http://realtime.mbta.com ; and the NextBus API, documented by NextBus.

1.1 Use of MBTA data

Access to MBTA data is governed by the language in the MassDOT Developers License Agreement (http://www.eot.state.ma.us/developers/) in addition to the following conditions:

- The MBTA reserves the right to suspend the data feed, modify the feed, or modify elements of the feed at any time at the MBTA's sole and absolute discretion.
- The MBTA does not guarantee any technical support of any kind to users.
- No user may execute the same polling command more often than every 10 seconds. A user that polls more often than that or otherwise overtaxes the MBTA's system may be suspended or terminated from the data feed.

2. QUICK TOUR OF THE MBTA-REALTIME API

MBTA-realtime provides RESTful web services to provide data about MBTA services. Data are provided in XML, JSON, and JSONP formats. IDs and concepts match GTFS and GTFS-realtime wherever possible. You will need your own API key to put anything in production, but we provide an “open key” that anyone can use to start development. Let’s use that key to run a few queries to show what the API can do.

The examples below run through a hypothetical smartphone app using the API to get information for a user. Both queries and responses are shown, but you’re encouraged to drop these queries into any browser and run them yourself – you will get live production data. Responses are in json; to see them in XML replace **&format=json** with **&format=xml** in the call.

2.1 stopsbylocation

A user opens the app. The app checks the phone's GPS antenna for a location – latitude 42.346961, longitude -71.076640 – and retrieves a list of nearby stations.

stopsbylocation query

http://realtime.mbta.com/developer/api/v2/stopsbylocation?api_key=wX9NwuHnZU2ToO7GmGR9uw&lat=42.346961&lon=-71.076640&format=json

Response (abridged)


```
{
  "stop":[
    {
      "stop_id":"Back Bay",
      "stop_name":"Back Bay",
      "parent_station":"place-bbsta",
      "parent_station_name":"Back Bay
Station",
      "stop_lat":"42.347158",
      "stop_lon":"-71.075769",
      "distance":"0.0463778711855412"
    },
    {
      "stop_id":"23391",
      "stop_name":"Back Bay Station",
      "parent_station":"place-bbsta",
      "parent_station_name":"Back Bay
Station",
      "stop_lat":"42.34735",
      "stop_lon":"-71.075727",
      "distance":"0.0535630360245705"
    },
    {
      "stop_id":"70014",
      "stop_name":"Back Bay Station -
Outbound",
      "parent_station":"place-bbsta",
      "parent_station_name":"Back Bay
Station",
      "stop_lat":"42.34735",
      "stop_lon":"-71.075727",
      "distance":"0.0535630360245705"
    },
    {
      "stop_id":"70015",
      "stop_name":"Back Bay Sta -
Inbound",
      "parent_station":"place-bbsta",
      "parent_station_name":"Back Bay
Station",
      "stop_lat":"42.34735",
      "stop_lon":"-71.075727",
      "distance":"0.0535630360245705"
    },
    {
      "stop_id":"place-bbsta",
      "stop_name":"Back Bay Station",
      "parent_station":"",
      "parent_station_name":"",
      "stop_lat":"42.34735",
      "stop_lon":"-71.075727",
      "distance":"0.0535630360245705"
    },
      .
      .
      .
    {
      "stop_id":"175",
      "stop_name":"Boylston St @
Dartmouth St",
      "parent_station":"place-coecl",
      "parent_station_name":"Copley
Station",
      "stop_lat":"42.349974",
      "stop_lon":"-71.077447",
      "distance":"0.212157785892487"
    }
  ]
}
```


The query returns 15 stops in total (not all are shown above), starting with the closest. It includes their ID's, names to be shown to the user, their location in latitude / longitude and their distance away in miles. The first few are all part of the "parent station" named "Back Bay" (with ID "place-bbsta"), which means they are different platforms in the same station. The app shows the user the closest stops, maybe consolidating all the "Back Bay" stops together.

2.2 routesbystop

The user requests more information about Back Bay from the app: What are the routes that serve it?

routesbystop query

```
http://realtime.mbta.com/developer/api/v2/routesbystop?api_key=wX9NwuHnZU2ToO7GmGR9uw&stop=place-bbsta&format=json
```

Response


```
{
  "stop_id":"place-bbsta",
  "stop_name":"Back Bay Station",
  "mode":[
    {
      "route_type":"1",
      "mode_name":"Subway",
      "route":[
        {
          "route_id":"903_",
          "route_name":"Orange Line"
        },
        {
          "route_id":"913_",
          "route_name":"Orange Line"
        }
      ]
    },
    {
      "route_type":"2",
      "mode_name":"Commuter Rail",
      "route":[
        {
          "route_id":"CR-Worcester",
"route_name":"Framingham/Worcester Line"
        },
        {
          "route_id":"CR-Franklin",
          "route_name":"Franklin Line"
        },
        {
          "route_id":"CR-Needham",
          "route_name":"Needham Line"
        },
        {
          "route_id":"CR-Providence",
"route_name":"Providence/Stoughton Line"
        }
      ]
    },
    {
      "route_type":"3",
      "mode_name":"Bus",
      "route":[
        {
          "route_id":"10",
          "route_name":"10"
        },
        {
          "route_id":"39",
          "route_name":"39"
        },
        {
          "route_id":"170",
          "route_name":"170"
        }
      ]
    }
  ]
}
```


The query returs all lines serving the station, with their “mode” (subway, bus, commuter rail), route_id, and route_name.

(The **routes** query returns all routes in the system.)
(The **route_id** value is used for the **&route=** parameter in many queries, like **stopsbyroute**, **schedulebyroute, vehiclesbyroute,** and **predictionsbyroute.** Pick a route_id like “CR-Providence” from above and give them a try.)

2.3 predictionsbystop

The user requests more information about Back Bay from your app: When will service be arriving?

predictionsbystop query

http://realtime.mbta.com/developer/api/v2/predictionsbystop?api_key=wX9NwuHnZU2ToO7GmGR9uw&stop=place-bbsta&format=json

Response (very abridged)

```
{ {
  "stop_id":"place-bbsta",
  "stop_name":"Back Bay",
  "mode":[
    {
      "route_type":"1",
      "mode_name":"Subway",
      "route":[
        {
          "route_id":"903_",
          "route_name":"Orange Line",
          "direction":[
            {
              "direction_id":"0",
              "direction_name":"Outbound",
              "trip":[
                {
                  "trip_id":"23460627",
                  "trip_name":"2:54 pm from Oak Grove to Forest Hills",
                  "trip_headsign":"Forest Hills",
                  "sch_arr_dt":"1407179940",
                  "sch_dep_dt":"1407179940",
                  "pre_dt":"1407180751",
                  "pre_away":"403",
                  "vehicle":{
                    "vehicle_id":"1250",
                    "vehicle_lat":"42.35346",
                    "vehicle_lon":"-71.06241",
                    "vehicle_bearing":"195",
                    "vehicle_timestamp":"1407180333"} } ] } ] } ] } ],
      "alert_headers: [{
             "alert_id": "23123",
             "header_text": "Orange Line experiencing moderate delays due to a
disabled train",
             "effect_name": "Delay"
      }]
 }
```

(The actual response will include several upcoming predictions for every service serving the station, organized by mode, route, direction, and trip. We've included just one here for clarity.)

The query returns all upcoming predictions for service that will serve the stop, for each prediction including the route, the direction, trip destination (trip_headsign), scheduled arrival and departure time (sch_arr_dt and sch_dep_dt), predicted time (pre_dt), and predicted number of seconds away (pre_away). The vehicle's current location is included as well. Finally, there's summary information about an alert. Much more information is available using the alert_id.

(**predictionsbyroute** and **predictionsbytrip** take &route=<route_id> and &trip=<trip_id> respectively.)

2.4 alertheaders

Seeing an alert makes the user decide to look at a summary of all the alerts in the system.

alertheaders query

```
http://realtime.mbta.com/developer/api/v2/alertheaders?api_key=wX9NwuHnZU2ToO7GmGR9uw&format=json
```

Response


```
{
  "alert_headers":[
    {
      "alert_id":23123,
      "header_text":"Orange Line experiencing moderate delays due to a disabled train",
      "effect_name":"Delay"
    },
    {
      "alert_id":33398,
      "header_text":" Buses replacing Fairmount Line Trains 781 (9:40 p.m. from South Station) and 782 (10:20 p.m. from Readville) on August 8 due to construction.",
      "effect_name":"Shuttle"
    },
    {
      "alert_id":33417,
      "header_text":"Beginning Mon Apr 14, the Church St @ Lexington St (layover) bus stop is temporarily closed due to construction.",
      "effect_name":"Stop move"
    },
    {
      "alert_id":33448,
      "header_text":" Buses replacing Red Line service between JFK/UMass and North Quincy Stations Aug 16-17 and Aug 23-24 due to construction",
      "effect_name":"Shuttle"
    },
    {
      "alert_id":33811,
      "header_text":"Route 1 experiencing minor delays",
      "effect_name":"Delay"
    },
    {
      "alert_id":33828,
      "header_text":"Savin Hill Station closed for maintenance",
      "effect_name":"Station closure"
    }
  ]
}
```


The query returns alert “headers,” a short summary of the information available about each alert. This includes information about disruptions happening now and upcoming disruptions, all mixed together. It does not include anything about elevators or escalator outages, athough it would if we had set &include_access_alerts=true. Your application shows a summary list to the user, using “stop move”, “shuttle”, and “delay” to mark the alerts with different icons.

(Alert headers and full alerts are available as a full list or by stop or by route, with **alertheadersbystop**, **alertsbyroute**, **alerts,** etc. Full information about an individual alert is also available by alert_id. Since it includes every alert there is without much metadata to sort them with, **alertheaders** is not the most useful call in practice, but it is useful for this demonstration.)

2.5 alertbyid

The user requests more information about a specific alert.

alertbyid query

```
http://realtime.mbta.com/developer/api/v2/alertbyid?api_key=wX9NwuHnZU2ToO7GmGR9uw&id=33448&format=json
```

Note: The above will not work in production unless there happens to be an alert_id 33448 active right now. If you are following along take an alert_id that was returned in the last example and use that as the parameter.

Response


```
{
  "alert_id":33448,
  "effect name":"Shuttle",
  "effect":"DETOUR",
  "cause_name":"construction",
  "cause":"CONSTRUCTION",
  "header_text":"Buses replacing Red Line
service between JFK/UMass and North Quincy
Stations Aug 16-17 and Aug 23-24 due to
construction",
  "short_header_text":"Buses replacing Red Line
service between JFK/UMass and North Quincy
Stations Aug 16-17 and Aug 23-24 due to
construction",
  "description_text":"Due to necessary track
and signal work, buses will replace Red Line
trains between JFK/UMass and Quincy Center
Stations in both directions from start to end
of service beginning Saturday, August 16, 2014,
through Sunday, August 17, 2014, as well as
Sunday, August 23, 2014, through Sunday, August
24, 2014. Regular Red Line train service will
resume at the start of service on Mondays.",
  "severity":"Severe",
  "created_dt":"1405360489",
  "last modified dt":"1405606519",
  "service_effect_text":"Red Line (Braintree
branch) shuttle",
  "timeframe text":"starting August 16",
  "alert_lifecycle":"Upcoming",
  "recurrence text":"weekends",
  "effect_periods":[
    {
      "effect_start":"1408177800",
      "effect end":"1408257000"
    },
    {
      "effect_start":"1408264200",
      "effect_end":"1408343400"
    },
    {
      "effect start":"1408782600",
      "effect end":"1408861800"
    },
    {
      "effect_start":"1408869000",
      "effect end":"1408948200"
    }
  ],
  "affected_services":{
    "services":[
      {
        "route_type":"1",
        "mode name":"Subway",
        "route_id":"933_",
        "route_name":"Red Line",
        "stop_id":"70095",
        "stop name":"JFK/UMASS Braintree -
Outbound"
      },
      {
        "route_type":"1",
        "mode_name":"Subway",
        "route id":"933 ",
        "route_name":"Red Line",
        "stop id":"70097",
        "stop_name":"North Quincy Station -
Outbound"
      },
      {
        "route_type":"1",
        "mode name":"Subway",
        "route_id":"933_",
        "route_name":"Red Line",
        "stop id":"70098",
        "stop_name":"North Quincy Station -
Inbound"
      },
      {
        "route_type":"1",
        "mode name":"Subway",
        "route_id":"933_",
        "route name":"Red Line",
        "stop_id":"70102",
        "stop_name":"Quincy Center Station -
Inbound"
      }
    ],
    "elevators":[
    ]
  }
}
```


The response includes a wealth of phrases – text that summarized the alert even further (“Red Line (Braintree branch) shuttle”) or expound on it (“due to track work…”), phrases to describe when it happens (“starting August 16”) and how it repeats (“weekends”). It also includes a lot of metadata – the effect (“Shuttle”), the extend of the disruption to those affected (“severe”), the fact that it represents a future event (“Upcoming”), individual time periods in epoch time, and a list of affected routes and stations. The application uses it to inform the user.

3. WHAT'S NEXT

3.1 Getting Your Own Account and API Key

Before you go any further you'll want to get your own API key.

3.1.1 REGISTER FOR AN ACCOUNT

To register for an account, visit the Developer Portal (http://realtime.mbta.com/Portal/) and click the "Register" link on the upper right-hand corner. Enter a username, password, email address, and phone number, and then click the "Register" button.

The Developer Portal will send back an email acknowledging the request for registration, along with a confirmation token, and a confirmation URL. Click the URL or visit http://realtime.mbta.com/Portal/Account/Confirmation and enter the token to complete the registration process. The account will be confirmed in the system.

3.1.2 LOG IN

To login to a registered developer account, visit the Developer Portal (http://realtime.mbta.com/Portal/) and click the "Log in" link on the upper right-hand corner. Enter the username and password, and then click the "Log in" button. The "Manage API Keys" page will open.

3.1.3 REGISTER FOR AN API KEY

To register for an API key, visit the "Manage API Keys" page, enter the name and description of the application which will use the API key, and then click the "Register" button.

The Developer Portal will send an email once the API Key has been granted. Note: this may take up to a day.

3.2 Learn More

http://realtime.mbta.com/Portal/Home/Documents has several relevant documents, including "MBTA-realtime API Documentation," a thorough review of every query you can call, every parameter you can specify, and every field the API can return. Documentation on other data sources are also available.

3.3 Getting help

More documentation is available at http://realtime.mbta.com .

The MBTA is happy to answer developer questions at developer@mbta.com . Developers are also encouraged to join the MBTA Developers discussion forum at https://groups.google.com/forum/?fromgroups#!forum/massdotdevelopers .