JSON: JavaScript Object Notation

Gaston Sanchez

About

In these slides we give you a crash introduction to **JSON data**:

- JSON basics
- Toy examples

BART API

The BART API | bart.gov

bart.gov/schedules/developers/api

Incognito

Bay Area
Rapid Transit

Since 2010, the BART API (http://api.bart.gov/) has been a one-stop shop for BART:

- Schedules;
- Service advisories;
- Fares;
- Real time estimates;
- Station information and more!

Two ways to get a BART API key

- **(1) No strings attached**
  We won't make you register for BART open data. Just follow our short and simple License Agreement, give our customers good information and don't hog community resources:

  `MW9S-E7SL-26DU-VV8V`

- **(2) Strings with benefits**
  If you sign up for your very own key you'll still be able to access the API if the public key is refreshed. Plus you'll get change notices and other updates to keep your application running smoothly.

BART - API Documentation

api.bart.gov/docs/overview/index.aspx

BART API Documentation

- **Overview**
  - Examples
  - Change Log
  - Notices
  - Station Abbreviations
  - Glossary
  - Output Formats
- Advisories
- Real-Time Estimates
- Route Information
- Schedule Information
- Station Information
- Version Information

BART API

The BART API gives you access to pretty much all of the BART service and station data available on the BART website. Check out an overview or read our simple License Agreement then jump right in with your own API validation key.

Command Overview

The BART API contains several different functional areas:

Section	Description
Overview	Contains general information and help about the BART API.
Advisories	Contains commands and calls pertaining to BART service advisories (BSA), elevator outages and train counts.
Real-Time Estimates	Contains commands and calls pertaining to estimated time of departure (ETD).
Route Information	Contains commands and calls pertaining to BART routes.
Schedule Information	Contains commands and calls pertaining to trip planning, route schedules, station schedules, holiday schedules, and special messages.
Station Information	Contains commands and calls pertaining to BART stations

BART - API Documentation, Sta

api.bart.gov/docs/stn/index.aspx

Incognito

BART API Documentation

- Overview
- Advisories
- Real-Time Estimates
- Route Information
- Schedule Information
- **Station Information**
  - help
  - stnaccess
  - stninfo
  - stns

Station Information API

Command Overview

The BART Station Information feed contains commands for requesting information about the BART stations.

The following commands are available through the Station Information API calls:

Command	Description
help	Requests detailed information regarding a specific route.
stninfo	Provides a detailed information about the specified station.
stnaccess	Requests detailed information how to access the specified station as well as information about the neighborhood around the station.
stns	Provides a list of all available stations.
ver	**This command has been depricated and replaced by the API Version command.**

BART - API Documentation, Sta

api.bart.gov/docs/stn/stns.aspx

Incognito

- Overview
- Advisories
- Real-Time Estimates
- Route Information
- Schedule Information
- Station Information
  - help
  - stnaccess
  - stninfo
  - **stns**
- Version Information

Station Information API

Command: stns

Inputs

Parameter	Description
cmd=stns	Requests current API version information (Required)
key=<key>	API registration key (Required)
json=y	Returns API output in JSON format. Default output is XML if parameter not specified. (Optional)

Notes

This command provides a list of all of the BART stations with their full names, abbreviations, latitude, longitude and addresses.

Results

XML Sample

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
<uri><![CDATA[ http://api.bart.gov/api/stn.aspx?cmd=stns ]]></uri>
  <stations>
    <station>
      <name>12th St. Oakland City Center</name>
      <abbr>12TH</abbr>
```

XML Sample


```xml
<?xml version="1.0" encoding="utf-8" ?>
<root>
<uri><![CDATA[ http://api.bart.gov/api/stn.aspx?cmd=stns ]]></uri>
  <stations>
    <station>
      <name>12th St. Oakland City Center</name>
      <abbr>12TH</abbr>
      <gtfs_latitude>37.803664</gtfs_latitude>
      <gtfs_longitude>-122.271604</gtfs_longitude>
      <address>1245 Broadway</address>
      <city>Oakland</city>
      <county>alameda</county>
      <state>CA</state>
      <zipcode>94612</zipcode>
    </station>
    ...
```

JSON Sample


```
{
   "?xml":{
      "@version":"1.0",
      "@encoding":"utf-8"
   },
   "root":{
      "uri":{
         "#cdata-section":"http://api.bart.gov/api/stn.aspx?cmd=stns&json=y"
      },
      "stations":{
         "station":[
            {
               "name":"12th St. Oakland City Center",
               "abbr":"12TH",
               "gtfs_latitude":"37.803768",
               "gtfs_longitude":"-122.271450",
               "address":"1245 Broadway",
               "city":"Oakland",
               "county":"alameda",
               "state":"CA",
               "zipcode":"94612"
            },
...
```

http://api.bart.gov/api/stn.aspx?cmd=stns&key=your-key

protocol resource path

http://api.bart.gov/api/stn.aspx?cmd=stns&key=your-key

host query

Usage example:

http://api.bart.gov/api/stn.aspx?cmd=stns&key=MW9S-E7SL-26DU-VV8V

How to make request from R?

You can use R package **httr**. If you are not familiar with the structure of the URI, you can use **`parse_url()`**

```
bart = "http://api.bart.gov/api/stn.aspx?cmd=stns&
key=MW9S-E7SL-26DU-VV8V"

parse_url(bart)
```

Uncovering URI

```
parse_url(bart)

$scheme
[1] "http"

$hostname
[1] "api.bart.gov"

$port
NULL

$path
[1] "api/stn.aspx"

$query
$query$cmd
[1] "stns"

$query$key
[1] "MW9S-E7SL-26DU-VV8V"
```

GET request: with httr's function GET()

```
req = GET(
  url = "http://api.bart.gov/",
  path = "api/stn.aspx",
  query = list(
    cmd = "stns",
    key = "MW9S-E7SL-26DU-VV8V"
  )
)
```

GET request: with httr's function GET()

```
# returned output
req
Response
[http://api.bart.gov/api/stn.aspx?cmd=stns
&key=MW9S-E7SL-26DU-VV8V]
  Date: 2020-06-30 01:34
  Status: 200
  Content-Type: text/xml; charset=utf-8
  Size: 13.9 kB
```

Parsing XML content with xml2()

Extracting content()

```
# extract XML content
doc = content(req)
doc
{xml_document}
<root>
[1]
<uri><![CDATA[http://api.bart.gov/api/stn.
aspx?cmd=stns]]></uri>
[2] <stations>\n  <station>\n
<name>12th St. Oakland City Center< ...
[3] <message/>
```

```
stn_name = doc %>%
  xml_find_all("//name") %>%
  xml_text()

stn_lat = doc %>%
  xml_find_all("//gtfs_latitude") %>%
  xml_text()

stn_lon = doc %>%
  xml_find_all("//gtfs_longitude") %>%
  xml_text()

stn_address = doc %>%
  xml_find_all("//address") %>%
  xml_text()
```

```
stn_city = doc %>%
  xml_find_all("//city") %>%
  xml_text()

stn_county = doc %>%
  xml_find_all("//county") %>%
  xml_text()

stn_zip = doc %>%
  xml_find_all("//zipcode") %>%
  xml_text()
```

```
dat = data.frame(
  name = stn_name,
  latitude = stn_lat,
  longitude = stn_lon,
  address = stn_address,
  city = stn_city,
  county = stn_county,
  zipcode = stn_zip,
  stringsAsFactors = FALSE
)
```

Parsing json content with jsonlite

GET request: with httr's function GET()

```
json_req = GET(
  url = "http://api.bart.gov/",
  path = "api/stn.aspx",
  query = list(
    cmd = "stns",
    key = "MW9S-E7SL-26DU-VV8V",
    json = "y"
  ),
  accept_json()
)
```

Extracting JSON content

```
# data is contained as raw Unicode
class(json_req$content)

# converting from character vector
# containing JSON into a json structure
jdat = jsonlite::fromJSON(
  (rawToChar(json_req$content))

# extract data frame of stations
dat_stns = jdat$root$stations
```