

ODJEZDY MHD V PRAZE

Teorie?

Data o odjezdech se v Praze dají načíst z městského webu ze systému Golemio. Je to obsáhlý repozitář spousty zajímavých dat. Ta data jsou relativně snadno dostupná a dobře popsána. Bylo pro mne překvapení zjistit, co je k dispozici. Ještě větší šok byl tento, cituji: „By default, each API key has a rate-limit set of 10 000 requests per 10 seconds.“

Tak se tam chovejte pěkně, nebo nám to vypnou. :) Myslím si, že to je server, který normálně celé MHD používá (viz třeba nové odjezdové tabule v zastávkách) a pořád nevěřím tomu, že je to normálně dostupný.

Popis dat a interfaců je na Swaggeru, který najdete na: <https://api.golemio.cz/pid/docs/openapi/>

The screenshot shows the Swagger UI interface for the Golemio API. At the top, there is a dropdown menu for 'Servers' set to 'https://api.golemio.cz/v2 - Main (production) server' and a green 'Authorize' button. Below this, the API structure is listed:

- GTFS Static**: General Transit Feed Specification data about the city's public transportation schedules in form of an API.
- GTFS Realtime**: Protobuf feeds of GTFS Realtime data. The feeds are updated every 20 seconds. Proto definitions can be found at [gtfs-realtime-OVapi.proto](#).
- PID Realtime Positions**: Response comes with special header `X-Last-Modified` with timestamp value of the latest resource update. When you use it as `?updatedSince` query parameter you will obtain only newer data than this value.
- PID Departure Boards**:
 - Method: GET /pid/departureboards
 - Description: GET Departure Boards
 - Note: You have to use query parameters `ids`, `cisIds`, `aswIds` or `names` in array format - eg. `ids[]={1,2,3}`. At least one of these IDs is required. The maximum is 100 stops combined in one request.
 - Parameters:
 - Try it out

Aby se s vámi městský server bavil, musíte získat API key na URL: <https://api.golemio.cz/api-keys/auth/sign-in> (To bylo moje velké překvapení, že to šlo)

The screenshot shows a login form titled 'API Keys Management'. The form is labeled 'Přihlaste se' (Log in). It contains fields for 'EMAIL' (with the value 'vas@email.cz') and 'HESLO' (password), and a blue 'Přihlásit se' (Log in) button. Below the form, there is a note: 'Ještě nemáte účet? Nový si můžete založit zde. Zapomněli jste heslo?' (Don't have an account? You can create a new one here. Forgotten password?). At the bottom, there are links for 'Podmínky užití API & Souhlas se zpracováním osobních údajů' (API usage terms & consent to processing personal data) and language options 'CS / EN'.

Systém funguje tak, že pošlete https request na golemio. V dotazu specifikujete zastávku nebo zastávky (blíže viz dále) a počet nadcházejících spojů, které chcete vypsat jako odjíždějící z dané zastávky nebo clusteru zastávek. Součástí každého dotazu je váš API klíč v headeru.

Systém vrátí JSON objekt, kde jsou vypsány údaje o všech spojích, které od okamžiku dotazu přijedou (a doufejme že i odjedou) a to v počtu položek, na který jste se ptali. Takže například pokud je v dotazu limit=10 vypíše se 10 dalších spojů, které mají z dané zastávky, nebo clusteru zastávek odjet.

Seznam zastávek je jako soubor ke stažení na <http://data.pid.cz/stops/xml/StopsByName.xml>

A má 8MB, takže doporučuji nějaký dobrý XML prohlížeč....

V seznamu si najdete svoji oblíbenou zastávku, do dotazu se dává **kód** zastávky označený ve výpisu zastávek jako gtfSIds - vypadá to třeba takto:

```
<stop id="1040/2" platform="B" altIdsName="Anděl (ul. Plzeňská)" lat="50.0719528" lon="14.4028082" jtskX="-744352.938" jtskY="-1044541.94" zone="P" wheelchairAccess="possible" gtfSIds="U1040Z2P">
    <line id="7" name="7" type="tram" direction="Radlická"/>
    <line id="9" name="9" type="tram" direction="Sidliště Řepy"/>
    <line id="10" name="10" type="tram" direction="Sidliště Řepy"/>
    <line id="15" name="15" type="tram" direction="Kotlářka"/>
    <line id="16" name="16" type="tram" direction="Kotlářka" direction2="Sidliště Řepy"/>
    <line id="21" name="21" type="tram" direction="Radlická"/>
    <line id="98" name="98" type="tram" isNight="true" direction="Sidliště Řepy"/>
    <line id="99" name="99" type="tram" isNight="true" direction="Sidliště Řepy"/>
    <line id="904" name="904" type="bus" isNight="true" direction="Sidliště Řepy"/>
    <line id="908" name="908" type="bus" isNight="true" direction="Jinonice"/>
</stop>
```

Tato konkrétní zastávka má kód gtfSIds="U1040Z1P" a je to zastávka, odkud jedou spoje uvedené v popisu a jedou směrem, který je tam napsaný. (kdo to tam zná, je to ta "tramvajová" naproti Centru Nový Smíchov, odkud se jede směrem "ven" z Prahy.

Zastávek je na Andělu hodně, takže třeba ta naproti, směrem na Palackého most má kód gtfSIds="U1040Z2P". Dotazy mohou obsahovat více zastávek, pak je syntax dotazu třeba: ?ids=U1040Z1P&ids=U1040Z2P

Pokud je vše v pořádku, vrátí se vám JSON, ve kterém je spousta údajů, zkuste si ho prohlédnout, je to docela zábava. (třeba kde vůz teď právě je, zda má zpozdění, nebo jestli je v něm klimatizace, co je napsáno na ceduli vozu (headsign), jestli je to noční spoj, zda má zpozdění a kolik atd. Nás zajímá, jak se jmenuje (čili jaké má číslo linky) a kdy je jeho příjezd na zastávku. Případně, jestli má tu klimu :) Pro živý obraz potřebujete čas příjezdu nebo odjezdu, dále jak se linka jmenuje, případně kam jede (někdy jede třeba do vozovny, tak je dobrý si to nechat ukázat)

Výsek z JSONu:

```

> [ ] stops
< [ ] departures
  < { }
    < { } arrival_timestamp
      123 predicted 12/2/2023 12:31:47 PM
      123 scheduled 12/2/2023 12:31:00 PM
    < { } delay
      123 is_available True
      123 minutes 0
      123 seconds 47
    > { } departure_timestamp
    > { } last_stop
    < { } route
      abc short_name 176
    123 type 3
      123 is_night False
      123 is_regional False
      123 is_substitute_transport False
    > { } stop
    < { } trip
      null direction
        abc headsign Karlovo náměstí
        abc id 176_1644_231202
        123 is_at_stop False
        123 is_canceled False
        123 is_wheelchair_accessible True
        123 is_air_conditioned False
      null short_name

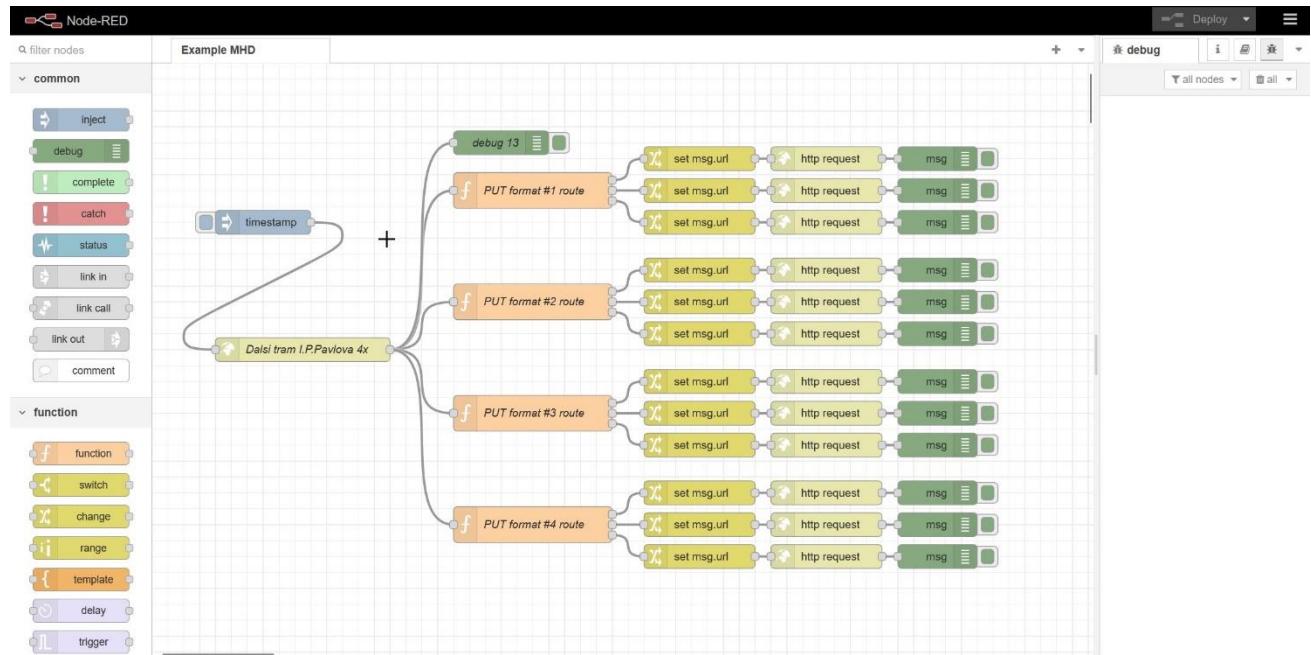
```

Když vytáhnete z JSONu údaje, které vás zajímají, do živého obrazu se to ukládá http GET requestem jako proměnná v sekci vlastní hodnoty, vždycky na jeden GET request je to dvojice "název" a "hodnota". Tady si zvolte, jak se vám líbí, já používám syntaxi `Zx_Poradi_y_Odjezd`, `Zx_Poradi_y_Link`a `Zx_Poradi_y_Smer`, kde Zx je zastávka x (já si zobrazuji na epaperu 3 různé zastávky v našem okolí, jsou nazvány Z1, Z2 a Z3 a na každé zobrazuji 3 nadcházející odjezdy (`_Poradi_1_` až `_Poradi_3_`), každý jeden spoj má 3 údaje, to je 27 hodnot). Jak si to nazvete je na Vás. Dál už by to měla být hračka.

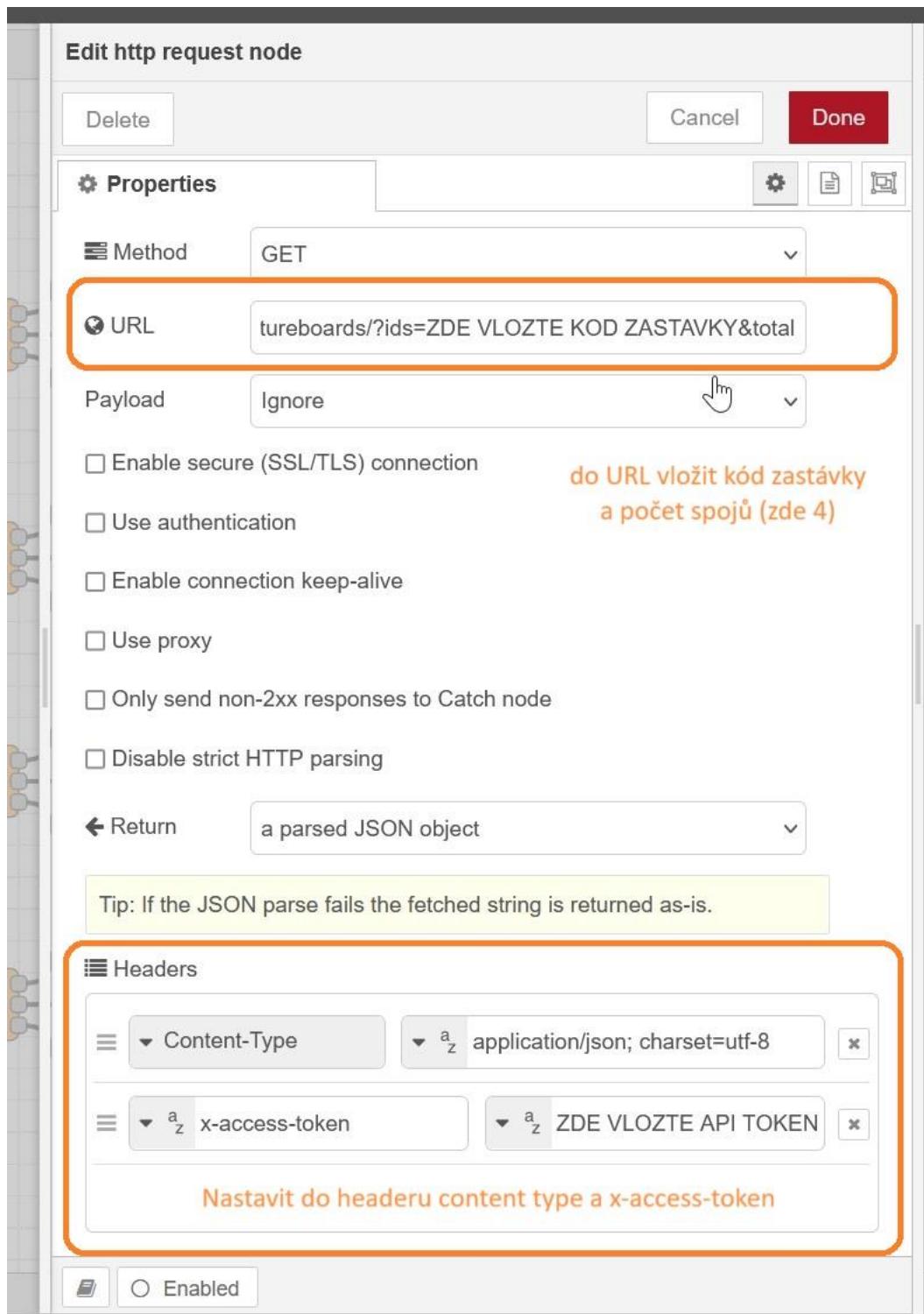
Realizace?

Protože mne zaujalo a nakoplo to, že jsem od města dostal bez keců na první dobrou API key, začal jsem být netrpělivý a nechtělo se mi to dlouze programovat. Použil jsem tedy Node Red, běží mi v Dockeru na Synology. Ale dá se pustit i na Raspberry. Dál popisuju jak to mám uděláno na Node Red. Vzorový flow je v souboru, importujte ho do Node Red.

Měli byste vidět:



Do http requestu si zeditujte zastávku, nebo kombinaci zastávek, počet budoucích spojů, které chcete vypsat a do nastavení nodu vložte nastavení headerů - API token a encoding.



Do funkčních nódů s názvem "PUT format..." vložte váš token pro Živý Obraz.

Edit function node

Delete Cancel Done

Properties

Name PUT format #1 route

Setup On Start On Message **On Stop**

```

1 msg.Depature_Event = msg.payload
2 let poradi = "0" //zde je poradove cislo spoje, na který se ptáme (0 = první)
3 let url = "http://in.zivyobraz.eu/"
4 let key = "MEZI UVODZOVKY VLOZTE VAS TOKEN PRO VSTUP DAT DO ZIVY OBRAZ"L
5 let legend_time_A = "Z3_Poradi_" // Z3 znamená že je to moje třetí zastávka, o kterou se zajímám
6 let legend_time_B = "_Odjezd"
7 let legend_line_name = "_Linka"
8 let legend_destination = "_Smer"
9 let str = " "
10 let time = ""
11 let linka = ""
12 let smer = ""
13
14 str = msg.Depature_Event.departures[poradi].departure_timestamp.scheduled
15 time = str.slice(11, 16)
16 let time_vystup = url + "?import_key=" + key + "&" + legend_time_A + poradi + legend_time_B + "=" + time
17 msg.payload = time_vystup.slice(0, 220)
18 var msg_time = { payload: msg.payload };
19
20 linka = msg.Depature_Event.departures[poradi].route.short_name
21 let linka_vystup = url + "?import_key=" + key + "&" + legend_time_A + poradi + legend_line_name + "=" + linka
22 msg.payload = linka_vystup.slice(0, 220)
23 var msg_linka = { payload: msg.payload };
24
25 smer = msg.Depature_Event.departures[poradi].trip.headsign
26 let smer_vystup = url + "?import_key=" + key + "&" + legend_time_A + poradi + legend_destination + "=" + smer
27 msg.payload = smer_vystup.slice(0, 220)

```

PUT nody by určitě šly napsat lépe, celé to ale vznikalo postupně a šlo mi o vyzkoušení. Určitě se najde někdo, kdo to napíše líp. PUT nody už také kopírují filozofii toho, jak to chci ukládat na Živý Obraz – pokud budete zobrazovat přes něco jiného, třeba HAS nebo nějaký svůj displej, budeme se tady už lišit. Zapojte se!

Až to vsecno nastavíte, zapněte si debug window. Klikněte na timestamp node a tím spusťte jeden dotaz. V debug okně byste měli vidět něco jako:

debug

i

▼ all nodes all

12/6/2023, 9:56:26 PM node: debug 12
msg.payload : Object
▶ { stops: array[1], departures: array[3], infotexts: array[0] }

12/6/2023, 9:56:28 PM node: b6eac49fd8b1a51b
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:28 PM node: 2b27c02ed84af827
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:28 PM node: b559895d251da8d4
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:28 PM node: d0f14617646a5d17
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:28 PM node: db3a3592ff5f34c1
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:28 PM node: 1668081a86ce1d7e
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023, 9:56:29 PM node: d1c3bedef0f0c44d
msg : Object
▶ { payload: "", _msgid: "5a731129f69752c5", url: "http://in.zivyobraz.eu/?import...", statusCode: 200, headers: object ... }

12/6/2023 9:56:29 PM node: 5fc31bb750a70f36

Prohlédněte si data z debug objektů (zde je to debug 12, dá se to rozbalit a je tam vidět struktura vráceného JSONu)

A pokud jsou reponce ze strany Živého Obrazu 200, pak máte data o MHD i v Živém Obrazu. Formátovat si to musíte sami 😊

Poděkování:

Michalovi @MultiTricker – za Živý Obraz

PID a městu Praha že zpřístupnili data (pořád se tomu divím, skoro se bojím, že spíš zapomněli zavřít dveře)

Uživatelům Tad a Ivondracek na fóru Home Assistant, z jejichž příspěvků jsem prvně zjistil, že PID data jsou takto luxusně k dispozici

Varianta Arduino - ESP32

Protože ne každý má k dispozici NodeRed, napadlo mne naučit extrakt ze serveru ještě nejoblíbenější bastlící platformu - ESP32. První verze už funguje, bude to za na mém GITHubu v jiném projektu.