Real-Time Traffic Visualizer

KMB API
REVERSE ENGINEERING

—

Calvin aka Overflow

INTRODUCTION

In the traffic visualizer playground project, one of the data source is KMB. To make a proof of concept before we get a deal with KMB, we try to find a way to make use of its publicly accessible api. However these api is non-trivial as it has some form of encapsulation which we will reverse engineer so that we can call those api from our own platform.

KMB Portal

http://search.kmb.hk/KMBWebSite/

We start with the web portal. The portal provide a way to select route and can list all stations and estimated arrival time. We found that the site operated with ajax model with the following endpoints:

- List bus stops for a particular route
- Get schedule for a particular route
- Get a list of route for a particular stop
- Get arrival estimation for a particular route and stop
- Announcement and other informative service

1. Route Bound

To retrieve a list of bound for a particular route, make a GET on

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getroutebound&route=12A

Cookies does not matter.

It returns a list of bound.

Sample output:

```
{
  "data": [
    {
      "SERVICE_TYPE": 1,
      "BOUND": 1,
      "ROUTE": "12A"
    },
    {
      "SERVICE_TYPE": 1,
      "BOUND": 2,
      "ROUTE": "12A"
    }
  ],
  "result": true
}
```

2. Bus Stops

To retrieve a list of bus stop for a particular route, make a GET on

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getstops&route=12A&bound=1&serviceType=1

Cookies does not matter.

It returns a list of bus stop. One important field is BSICode, e.g. *WH01-T-1000-0*, which will be used on *Arrival Estimation API*.

Sample output:

```
{
  "data": {
    "basicInfo": {
      "Racecourse": "N",
      "DestEName": "CHEUNG SHA WAN (SHAM MONG ROAD)",
      "OriCName": "黃埔花園",
      "ServiceTypeENG": "",
      "DestCName": "長沙灣(深旺道)",
      "BusType": null,
      "Airport": "N",
      "ServiceTypeTC": "",
      "Overnight": "N",
      "ServiceTypeSC": "",
      "OriSCName": "黄埔花园",
      "DestSCName": "长沙湾(深旺道)",
      "Special": "N",
      "OriEName": "WHAMPOA GARDEN"
    },
    "routeStops": [
      {
        "CName": "黃埔花園總站",
        "Y": "818438.31250000",
        "ELocation": "WHAMPOA GARDEN B\/T",
        "X": "837691.56250000",
        "AirFare": "5.10000000",
        "EName": "WHAMPOA GARDEN B\/T",
        "SCName": "黄埔花园总站",
        "ServiceType": "1",
        "CLocation": "黃埔花園巴士總站",
        "BSICode": "WH01-T-1000-0",
        "Seq": "0",
        "SCLocation": "黄埔花园巴士总站",
        "Direction": "F         ",
        "Bound": "1",
        "Route": "12A"
      },
      {
        "CName": "紅樂道",
        ...
```

3. Bus Schedule

To retrieve the bus schedule for a particular route, make a GET on

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getschedule&route=12A&bound=1

Cookies does not matter.

It returns a list of bus schedule.

Sample output:

```
{
  "data": {
    "01": [
      {
        "DayType": "MF        ",
        "BoundTime1": "12",
        "ServiceType_Eng": "",
        "BoundText1": "06:10 - 06:58",
        "Origin_Eng": "WHAMPOA GARDEN",
        "ServiceType": "01   ",
        "Destination_Chi": "長沙灣(深旺道)",
        "OrderSeq": "1",
        "Route": "12A",
        "Destination_Eng": "CHEUNG SHA WAN (SHAM MONG ROAD)",
        "BoundTime2": "12",
        "Origin_Chi": "黃埔花園",
        "BoundText2": "05:40 - 07:04",
        "ServiceType_Chi": ""
      },
      {
        "DayType": "MF        ",

        ...
```

4. List route passing on particular stop

To retrieve the bus route passing a particular route, make a GET on

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getRoutesInStop&bsiCode=WH01-T-1000-0

Cookies does not matter.

It returns a list of bus route.

Sample output:

```
{
  "data": [
    "5D       ",
    "8A       ",
    "8P       ",
    "115      ",
    "12A      ",
    "30X      ",
    "115P     ",
    "230X     "
  ],
  "result": true
}
```

5. Announcement and Traffic News

This api return a list of traffic news and announcement

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getAnnounce&route=1&bound=1

Cookies does not matter.

It returns a list of announcement.

Sample output:

```
{
  "data": [
    {
      "kpi_title": "@ 15:32 Broken Vehicle at Ferry Street Flyover",
      "kpi_referenceno": "PEQ17080409",
      "kpi_title_chi_s": "@ 15:32 渡船街天桥坏车阻路",
      "krbpiid_routeno": "1",
      "kpi_noticeimageurl": "1501833265.html",
      "krbpiid_boundno": "1",
      "krbpiid": "1520867",
      "kpi_title_chi": "@ 15:32 渡船街天橋壞車阻路"
    },
    {
      "kpi_title": "La Salle Primary School Bus Stop Temporary Relocation",
      "kpi_referenceno": "BS1708-037",
      "kpi_title_chi_s": "喇沙小学巴士站暂时迁移",
      "krbpiid_routeno": "1",
      "kpi_noticeimageurl": "1501833287_4551_0.jpg",
      "krbpiid_boundno": "1",
      "krbpiid": "1520721",
      "kpi_title_chi": "喇沙小學巴士站暫時遷移"
    }
  ],
  "result": true
}
```

6. Arrival Estimation

Arrival estimation API is a bit tricky, you need a POST on

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx/?action=get_ETA&lang=1

Cookies does not matter, but with form data (application/x-www-form-urlencoded; charset=UTF-8)

token: *TOKEN*
t: *TIMESTAMP*

It returns arrival estimations on next few buses.

Sample output:

```
{
  "data": {
    "updated": 1501822984000,
    "generated": 1501823015644,
    "responsecode": 0,
    "response": [
      {
        "w": "Y",
        "ex": "2017-08-04 13:24:32",
        "eot": "E",
        "t": "13:14",
        "ei": "N",
        "bus_service_type": 1,
        "wifi": null,
        "ol": "N"
      },
      {
        "w": "Y",
        "ex": "2017-08-04 13:44:46",
        "eot": "E",
        "t": "13:34",
        "ei": "N",
        "bus_service_type": 1,
        "wifi": null,
        "ol": "N"
      },
      {
        "w": "Y",
        "ex": "2017-08-04 13:55:43",
        "eot": "E",
        "t": "13:45",
          "ei": "N",
          "bus_service_type": 1,
          "wifi": null,
          "ol": "N"
        }
      ]
    },
    "result": true
}
```

Parameters

Tricky part is the token is encapsulated. We have to reverse engineer and produce our own token to make meaningful request.

The *TIMESTAMP* is obvious, for example 2017-08-04 04:07:02.50. Is just the current time as of the request, note the last full stop.

The *TOKEN* part is encapsulated with a obfuscated javascript function. A sample looks like this:

```
EAMTJBLS0zMTIwMTctMDgtMDQgMDQ6MDc6MDIuNTAuMTMtLTEtLTMxMjAxNy0wOC0wNCAwNDowNzowMi41MC4xMy0tMS0tMzE
yMDE3LTA4LTA0IDA0OjA3OjAyLjUwLjEzLS1XSDAxVDEwMDAwLS0zMTIwMTctMDgtMDQgMDQ6MDc6MDIuNTAuMTMtLTAtLTMx
MjAxNy0wOC0wNCAwNDowNzowMi41MC4xMy0tMTUwMTgxOTYyMjM1MA==
```

We found the token is generated with getETA() function from

http://search.kmb.hk/KMBWebSite/function.js?r=2017050102

Put the code to deobfuscator gives a hint on the token format:

```
var pY = against[_0x1c1a("0x2")]() + "-" + ("00" + (against["getUTCMonth"]() +
1))[_0x1c1a("0x3")](-2) + "-" + ("00" + against[_0x1c1a("0x4")]())[_0x1c1a("0x3")](-2) + " " +
("00" + against[_0x1c1a("0x5")]())["slice"](-2) + ":" + ("00" +
against["getUTCMinutes"]())[_0x1c1a("0x3")](-2) + ":" + ("00" +
against[_0x1c1a("0x6")]())[_0x1c1a("0x3")](-2) + "." + ("00" +
against[_0x1c1a("0x7")]())["slice"](-2) + ".";

sep = _0x1c1a("0x8") + pY + _0x1c1a("0x9");

var oauth_token = "EA" + Base64[_0x1c1a("0xa")](arr[_0x1c1a("0xb")][_0x1c1a("0xc")]() + sep + val
+ sep + arr[_0x1c1a("0xd")] + sep + arr[_0x1c1a("0xe")][_0x1c1a("0xc")]()[_0x1c1a("0xf")](/-/gi,
"") + sep + arr["Seq"] + sep + (new Date)[_0x1c1a("0x10")]());
```

pY is the timestamp field we see on form data *T*, *oauth_token* is the form data *Token*.

Token looks like EA + base64(something), we decode the above token and bingo, we got:

```
12A--312017-08-04 04:07:02.50.13--1--312017-08-04 04:07:02.50.13--1--312017-08-04
04:07:02.50.13--WH01T10000--312017-08-04 04:07:02.50.13--0--312017-08-04
04:07:02.50.13--1501819622350
```

Next we need to figure out the components of the above string. One obvious repeat pattern is the *sep*, which is [something][pY][something], or `--312017-08-04 04:07:02.50.13--`

Let's break down the token:

```
12A--312017-08-04 04:07:02.50.13--1--312017-08-04 04:07:02.50.13--1--312017-08-04
04:07:02.50.13--WH01T10000--312017-08-04 04:07:02.50.13--0--312017-08-04
04:07:02.50.13--1501819622350
```

There are a few components we don't know, we make a few more request on:

1. Same route to see any time-dependent field
2. Different route
3. Different stop

Two requests on 12A first bus stop:

```
12A--312017-08-04 04:07:02.50.13--1--312017-08-04 04:07:02.50.13--1--312017-08-04
04:07:02.50.13--WH01T10000--312017-08-04 04:07:02.50.13--0--312017-08-04
04:07:02.50.13--1501819622350
```

```
12A--312017-08-04 04:07:11.02.13--1--312017-08-04 04:07:11.02.13--1--312017-08-04
04:07:11.02.13--WH01T10000--312017-08-04 04:07:11.02.13--0--312017-08-04
04:07:11.02.13--1501819631702
```

The only changing factor is the highlighted part, which look very like a unix timestamp in ms.

Next, we make request on the first 4 bus stops of 12A

```
12A--312017-08-04 04:07:02.50.13--1--312017-08-04 04:07:02.50.13--1--312017-08-04
04:07:02.50.13--WH01T10000--312017-08-04 04:07:02.50.13--0--312017-08-04
04:07:02.50.13--1501819622350
```

```
12A--312017-08-04 04:07:25.82.13--1--312017-08-04 04:07:25.82.13--1--312017-08-04
04:07:25.82.13--HU06N10000--312017-08-04 04:07:25.82.13--1--312017-08-04
04:07:25.82.13--1501819645182
```

```
12A--312017-08-04 04:07:54.98.13--1--312017-08-04 04:07:54.98.13--1--312017-08-04
04:07:54.98.13--HU01W09900--312017-08-04 04:07:54.98.13--2--312017-08-04
04:07:54.98.13--1501819674398
```

```
12A--312017-08-04 04:57:18.68.13--1--312017-08-04 04:57:18.68.13--1--312017-08-04
04:57:18.68.13--GI02N10000--312017-08-04 04:57:18.68.13--3--312017-08-04
04:57:18.68.13--1501822638368
```

The red part looks like the *BSIcode* we got in (1), blue part look like the bus stop index.

Next, we make request on the reverse direction of bus stops of 12A

```
12A--312017-08-04 05:01:56.35.13--2--312017-08-04 05:01:56.35.13--1--312017-08-04
05:01:56.35.13--SH37W11800--312017-08-04 05:01:56.35.13--0--312017-08-04
05:01:56.35.13--1501822916436
```

We notice the highlight number 2 means the reverse direction.

Let’s summarize what we found

```
12A--312017-08-04 04:07:02.50.13--1--312017-08-04 04:07:02.50.13--1--312017-08-04
04:07:02.50.13--WH01T10000--312017-08-04 04:07:02.50.13--0--312017-08-04
04:07:02.50.13--1501819622350
```

```
[Route]--312017-08-04 04:07:02.50.13--[Bound/Direction]--312017-08-04
04:07:02.50.13--1--312017-08-04 04:07:02.50.13--[BSICode]--312017-08-04
04:07:02.50.13--[StopIndex]--312017-08-04 04:07:02.50.13--[UTC]
```

We still don’t know what the remaining 1 mean, but look at other API endpoint suggest it may be *serviceType*, but we don’t care much on it.

Put it to the test

Now we have completely reverse engineered the token encapsulation, let craft a request with 2nd stop on route 12A.

First we make a request to get the BSIcode:

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getstops&route=12A&bound=1&serviceType=1

The code for 2nd bus stop (index=1) is HU06-N-1000-0

We craft the token with current timestamp 2017-08-04 07:02:21.00. and UTC 1501830161000

```
12A--312017-08-04 07:02:21.00.13--1--312017-08-04 07:02:21.00.13--1--312017-08-04
07:02:21.00.13--HU06N10000--312017-08-04 07:02:21.00.13--1--312017-08-04
07:02:21.00.13--1501830161000
```

```
token:
EAMTJBLS0zMTIwMTctMDgtMDQgMDc6MDI6MjEuMDAuMTMtLTEtLTMxMjAxNy0wOC0wNCAwNzowMjoyMS4wMC4xMy0tMS0tMzE
yMDE3LTA4LTA0IDA3OjAyOjIxLjAwLjEzLS1IVTA2TjEwMDAwLS0zMTIwMTctMDgtMDQgMDc6MDI6MjEuMDAuMTMtLTEtLTMx
MjAxNy0wOC0wNCAwNzowMjoyMS4wMC4xMy0tMTUwMTgzMDE2MTAwMA==
```

```
t: 2017-08-04 07:02:21.00.
```

We got it

Let’s try another one, 3rd stop on route 30X.

http://search.kmb.hk/KMBWebSite/Function/FunctionRequest.ashx?action=getstops&route=30X&bound=1&serviceType=1

The code for 3rd bus stop (index=2) is TS11-S-1200-0

We craft the token with current timestamp 2017-08-04 07:19:55.00. and UTC 1501831195000

```
30X--312017-08-04 07:19:55.00.13--1--312017-08-04 07:19:55.00.13--1--312017-08-04
07:19:55.00.13--TS11S12000--312017-08-04 07:19:55.00.13--2--312017-08-04
07:19:55.00.13--1501831195000
```

```
token:
EAMzBYLS0zMTIwMTctMDgtMDQgMDc6MTk6NTUuMDAuMTMtLTEtLTMxMjAxNy0wOC0wNCAwNzoxOTo1NS4wMC4xMy0tMS0tMzE
yMDE3LTA4LTA0IDA3OjE5OjU1LjAwLjEzLS1UUzExUzEyMDAwLS0zMTIwMTctMDgtMDQgMDc6MTk6NTUuMDAuMTMtLTItLTMx
MjAxNy0wOC0wNCAwNzoxOTo1NS4wMC4xMy0tMTUwMTgzMTE5NTAwMA==
```

```
t: 2017-08-04 07:19:55.00.
```

We got it too. This pretty much reveal the whole logic. Let’s start coding :)