ADP Gateway API Documentation

John Machado

May 2025

Contents

1 Introduction

A REST API has been implemented to facilitate automated access to the sign's database. REST APIs are widely supported in all programming languages. For more information visit POSTMAN's REST API primer.

REST APIs facilitate sign automation.

They are analogous to user interfaces for humans.

2 Read Screens

Returns all screen data.

2.1 Endpoint

.../screens/read

2.2 Method

GET

2.3 Payload

NOT REQUIRED

2.3.1 Valid Values

NOT APPLICABLE

2.4 Returned Data

database data

```
{
  "screens": {
      "screen_id": {
        "vertical_alignment": string,
        "rows": {
          "row_id": {
            "font_size": string,
            "font_weight": string,
            "hold_time": string,
            "horizontal_alignment": string,
            "in_mode": string,
            "scroll_speed": string,
            "segments": {
              "segment": {
                "foreground_color": string,
                "background_color": string,
                "flash": string,
                "text": string
              }, ...
            }
          }, ...
        }
      }
    }
  }
}
```

3 Update Screens

A screen is a collection of rows, which in turn are a collection of segments.

- Screen creation/update database operations are inferred; this is the only endpoint needed.
- The playlist will become the list of screens added by this operation.

3.1 Endpoint

.../screens/update

3.2 Method

GET

3.3 Payload

```
{
  "screen_id": {
    "vertical_alignment": string,
    "rows":
      "row_id": {
        "font_size": int,
        "font_weight": string,
        "hold_time": int,
        "horizontal_alignment": string,
        "in_mode": string,
        "scroll_speed": string,
        "segments":
          "segment_id": {
            "foreground_color": string,
            "background_color": string,
            "flash": string,
            "text": string
          }, ...
      }, ...
  }
}
```

3.3.1 Valid Values

- screens: map of screen
- screen.id: string
  - one to four character length
  - can be any ascii character in range 32,125 inclusive
- screen.vertical_alignment: string
  - aligns all rows vertically
  - options:
    - top, middle, fill, bottom
- row.id: string
  - screen.id appended with three digit left zero padded row number
  - must be sequential
  - examples:
    - screen.id = "A":
      1. first row.id = "A000"
      2. second row.id = "A001"
      3. ...
      4. nth row.id = "An" where n $\leq$ 999 left zero padded
    - screen.id = "x434"
      1. first row.id = "x434000"
      2. second row.id = "x434001"
      3. ...
      4. nth row.id = "x434n" where n $\leq$ 999 left zero padded
- row.font_size: integer
  - integer
  - font size in pixels
  - normal options:
    - 5, 7, 9, 11, 14, 15, 16, 22, 24, 30, 32, 40, 64, 71, 80, 88
  - bold options:
    - 5, 11, 14, 15, 16, 22, 30, 32, 40
- row.font_weight: string
  - options:
    - bold, normal

- row.hold_time: integer
  - options:
    - integer in the range 1 to 99 inclusive
- row.horizontal_alignment: string
  - options:
    - left, center, right
- row.in_mode: string
  - options:
    - hold, scroll
- row.scroll_speed: string
  - options:
    - slowest, slow, normal, fast, fastest
- segment.id: string
  - row.id appended with three digit left zero padded segment number
  - must be sequential
  - examples:
    - row.id = "A000":
      1. first segment.id = "A000000"
      2. second segment.id = "A000001"
      3. ...
      4. nth segment.id = "A000{n}" where n $\leq$ 999 left zero padded
    - row.id = "x434000000"
      1. first segment.id = "x434001000"
      2. second segment.id = "x434001001"
      3. ...
      4. nth segment.id = "x434001{n}" where n $\leq$ 999 left zero padded
- segment.foreground_color: string
  - options:
    - black, red, green, blue, yellow, white
- segment.background_color: string
  - options:
    - black, red, green, blue, yellow, white, or 6 digit hex value preceded by #

- flash: string
  - options:
    - on, off
- text: string
  - options:
    - ascii characters in the range 32 to 125 inclusive

3.4 Returned Data

user submission

```
{
  "screen_id": {
    "vertical_alignment": string,
    "rows":
      "row_id": {
        "font_size": string,
        "font_weight": string,
        "hold_time": string,
        "horizontal_alignment": string,
        "in_mode": string,
        "scroll_speed": string,
        "segments":
          "segment_id": {
            "foreground_color": string,
            "background_color": string,
            "flash": string,
            "text": string
          }, ...
      }, ...
  }
}
```

4 Remove Screens

Delete screens from the signs database and playlist

4.1 Endpoint

.../screens/delete

4.2 Method

GET

4.3 Payload

```
{
    [
        string,
        ...,
        string
    ]
}
```

4.3.1 Valid Values

- Array elements must be valid screen ids.

4.4 Returned Data

user submission

```
{
    "removed_screens": [
        string,
        ...,
        string
    ]
}
```

5 Remove All Screens

Removes all screens from both the database and the playlist.

5.1 Endpoint

.../screens/delete_all

5.2 Method

GET

5.3 Payload

NOT REQUIRED

5.3.1 Valid Values

NOT APPLICABLE

5.4 Returned Data

"all screens deleted"

6 Read Playlist

Returns all playlist data.

6.1 Endpoint

.../playlist/read

6.2 Method

GET

6.3 Payload

NOT REQUIRED

6.3.1 Valid Values

NOT APPLICABLE

6.4 Returned Data

database data

```
{
    "playlist": [
        string,
        ...,
        string
    ]
}
```

7 Update Playlist

The playlist is the list of screens that will be shown sequentially. The interval between screens is determined by the HOLD TIME option and any overflow condition.

- Updating the playlist will replace the old playlist with the new one.
- Removing screen ids from the playlist will not remove the screen data from the database.

7.1 Endpoint

.../playlist/update

7.2 Method

GET

7.3 Payload

```
{
    [
        string,
        ...,
        string
    ]
}
```

7.3.1 Valid Values

- array where elements must be valid screen ids

7.4 Returned Data

user submission

```
{
    "updated_playlist": [
        string,
        ...,
        string
    ]
}
```

8 Remove from Playlist

Removes entries from the playlist.

8.1 Endpoint

.../playlist/delete

8.2 Method

GET

8.3 Payload

The screen ids of screens that should be removed from the signs database.

```
{
    [
        string,
        ...,
        string
    ]
}
```

8.3.1 Valid Values

- array where elements must be valid screen ids

8.4 Returned Data

user submission

```
{
    "removed_from_playlist": [
        string,
        ...,
        string
    ]
}
```

9 Remove All from Playlist

Removes all playlist entries.

9.1 Endpoint

.../playlist/delete_all

9.2 Method

GET

9.3 Payload

NOT REQUIRED

9.3.1 Valid Values

NOT APPLICABLE

9.4 Returned Data

"playlist-deleted"

10 Settings Read

Read values of sign settings.

10.1 Endpoint

.../settings/read

10.2 Method

GET

10.3 Payload

NOT REQUIRED

10.3.1 Valid Values

NOT APPLICABLE

10.4 Returned Data

database data

```
{
  "brightness": string,
  "dimensions": {
    "width": string,
    "height": string
  }
}
```

11 Settings Update

Update settings values.

11.1 Endpoint

.../settings/update

11.2 Method

GET

11.3 Payload

Each setting is sufficient to perform an update but all setting parameters are necessary. For example, dimensions can be updated by itself, but both width and height will need to be provided for the update to take effect.

```
{
  ?"brightness": real,
  ?"dimensions": {
    "width": integer,
    "height": integer
  }
}
```

11.3.1 Valid Values

- brightness: real
  - real number in range 0 and 1 inclusive
  - example: 0.5
- dimensions.width: integer
  - represents the width of the sign in pixels.
  - example: 100
- dimensions.height: integer
  - represents the width of the sign in pixels.
  - example: 25

11.4 Returned Data

user submission

```
{
  ? "brightness": string,
  ?"dimensions": {
    "width": string,
    "height": string
  }
}
```