

# DISPLAY API DESCRIPTION V2

*Traveller timetable information*

# DISPLAY API DESCRIPTION V2

16.09.2022

## INDEX

1	Introduction .....	5
2	Interfaces .....	6
2.1	API return values .....	6
2.2	API call common parameters .....	6
2.3	API common datatypes .....	6
2.3.1	MultiLanguageText .....	6
2.3.2	DepartureStatus enumeration .....	6
2.3.3	DepartureAttribute .....	7
2.3.4	AnnouncementData .....	7
2.4	API call common return values .....	7
2.5	Common datatypes .....	8
2.5.1	Timestamp .....	8
2.5.2	Color .....	8
2.6	TimetableView interface .....	8
2.6.1	API address .....	8
2.6.2	GetTimetables .....	9
2.6.3	GetAnnouncements .....	9
2.6.4	PostScreenCapture .....	11
2.7	Display interface .....	12
2.7.1	API address .....	12
2.7.2	PostDisplayStatus .....	12
2.7.3	GetDisplayStatusList .....	13
2.7.4	GetDataUpdateTimes ( not implemented ) .....	14
2.7.5	GetLayoutConfiguration .....	14
2.7.6	GetDisplayConfiguration .....	15
3	Real time interface .....	17
3.1	GetDisplayEvents .....	17
3.2	Realtime interface event messages .....	17
3.2.1	UpdateDeparture .....	17

# DISPLAY API DESCRIPTION V2

16.09.2022

3.2.2	DataUpdated .....	18
3.2.3	Action .....	18
4	Interface usage examples .....	20
4.1	GetTimetables .....	20
4.2	GetAnnouncements .....	21
4.3	PostScreenCapture .....	22
4.4	PostDisplayStatus .....	22
4.5	GetDisplayStatusList .....	23
4.6	GetLayoutConfiguration .....	23
4.7	GetDisplayConfiguration .....	24
4.8	SSE Events.....	26
4.8.1	DepartureUpdate .....	27
4.8.2	DataUpdated .....	27
4.8.3	Action .....	27

# DISPLAY API DESCRIPTION V2

16.09.2022

## Revision history

Revision	Author	Date	Status and description
0.14	Jarkko Pulkkinen	2022-09-16	Moved "title" to GetLayoutConfiguration, fixed typos
0.13	Jarkko Pulkkinen	2022-09-02	Added new values in methods: PostDisplayStatus, GetLayoutConfiguration, GetDisplayConfiguration
0.12	Jukka Tarkkonen	2021-09-01	DisplayRestart added to SSE actions.
0.11	Marko Jeskanen	2021-20-07	Presentation state added to display configuration. Shock_low and shock_high display statuses added.
0.10	Marko Jeskanen	2021-07-02	Display status list updated. Added GetDisplayStatusList method.
0.9	Marko Jeskanen	2021-03-17	Changed layout configuration capital letters.
0.8	Marko Jeskanen	2021-03-17	Air pollution device configuration added
0.7	Marko Jeskanen	2021-03-03	Fix example URL
0.6	Marko Jeskanen	2021-02-25	Device methods moved to device interface
0.5	Marko Jeskanen	2021-02-16	Added display configuration methods
0.4	Jukka Tarkkonen	2020-02-06	Added screen capture methods.
0.3	Marko Jeskanen	2019-08-12	Added interface usage examples.
0.2	Marko Jeskanen	2019-08-08	Added missing fields. Added specifications to language and Basic Authentication
0.1	Marko Jeskanen	2019-05-01	Initial specification

# DISPLAY API DESCRIPTION V2

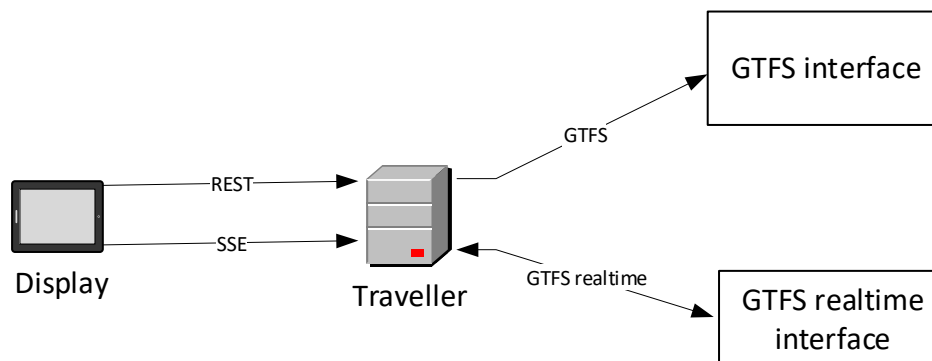
16.09.2022

## 1 INTRODUCTION

This document describes Traveller Display API. The document is written for display software developers. Using this document, it is possible to write software for timetable displays which can read timetable information using Traveller Display API.

Traveller system normally reads scheduled timetable information from GTFS interface. GTFS file will contain all scheduled information which will be shown on displays. Getting scheduled timetables is the minimum requirement for Traveller system. Display can read scheduled information through Traveller Display API REST interface.

Traveller can also read real time information through GTFS real time interface. GTFS real time interface will provide forecasts to bus arrival times to stops. Display can receive real time updates through Traveller Display API SSE interface.



# DISPLAY API DESCRIPTION V2

16.09.2022

## 2 INTERFACES

Traveller Display API is a REST interface. Communication between a display and the API is encoded using HTTPS-protocol. All information in the REST interface is transferred using JSON (JavaScript Object Notation). Traveller Display API must be used with HTTP Basic Authentication.

### 2.1 API return values

Some of the fields in the API return values are optional. In the case that such an optional field does not have a value to be returned, the field may be omitted from the returned response.

### 2.2 API call common parameters

All calls to the API need to contain the parameters described in this chapter. Username and password are to be included in an HTTP header as it is described in the Basic Authentication specification:

<https://tools.ietf.org/html/rfc7617>.

DisplayId is passed as an HTTP query parameter.

Parameter	Type	Explanation
DisplayId	String	Display identification.
Username	String	Username for Basic Authentication.
Password	String	Password for Basic Authentication.

Table 1 Common parameters for API calls

### 2.3 API common datatypes

#### 2.3.1 MultiLanguageText

MultiLanguageText contains texts for a departure with different languages.

Attribute	Type	Explanation
Language	String	Language identifier: fi, sv, pl, en... Language codes are defined in ISO 639-1 specification: <a href="https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes">https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes</a>
Text	String	Destination of a line shown on display.

Table 2 MultiLanguageText datatype

#### 2.3.2 DepartureStatus enumeration

Departure status enumeration shows different statuses for departures.

Value	Explanation
Removed	Departure is removed from the display.

## DISPLAY API DESCRIPTION V2

16.09.2022

<b>Canceled</b>	Departure is canceled on the display.
<b>Normal</b>	Normal departure
<b>AtStop</b>	Bus has arrived at the stop.
<b>LeftStop</b>	Bus has left the stop.
<b>Overridden</b>	Departure has been delayed for some reason. e.g. bus is waiting train to arrive.

### 2.3.3 DepartureAttribute

Attributes of a departure. These are the current attributes which a departure can have. One departure may have multiple attributes.

Value	Type	Explanation
<b>LowFloor</b>	String	Bus with low floor.
<b>TicketMachine</b>	String	Bus has ticket vending machine.

### 2.3.4 AnnouncementData

Announcement data contains necessary information for an announcement. This data varies between different type of announcements.

## 2.4 API call common return values

Return values of API calls always contain the status of the call. The API calls can also return other data which is described in the method call descriptions.

Attribute	Type	Explanation
<b>Status</b>	Int	Status code
<b>StatusMsg</b>	string	Status explanation.

If an API call returns information, these two parameters are returned with the API data. The returned data is described in the method call descriptions. Most common values are described below:

Status	StatusMsg	Explanation
<b>0</b>	OK	Method execution was successful
<b>-1</b>	NO_SUCH_DISPLAY_ID	The given displayid is not found in Traveller

## DISPLAY API DESCRIPTION V2

16.09.2022

### 2.5 Common datatypes

#### 2.5.1 Timestamp

Timestamp will be in a string format: "YYYY-MM-DDThh:mm:ssTZD" as described in <https://www.w3.org/TR/NOTE-datetime>. Timestamps are always in UTC.

#### 2.5.2 Color

Color will be in a string in hexadecimal format specified with: #RRGGBB, where the (RR) red, GG (Green) and BB (Blue) hexadecimal integers specify the components of the color. All values must be between 00 and FF.

For example, #FF0000 is rendered as red, because red component is set to its highest value FF and others are set to lowest value 00.

### 2.6 TimetableView interface

TimetableView interface is used to get information for one timetable view. Timetable view is a part of display, which shows departure and announcements. Display hardware might consist of two screens and both screen can display different timetables. Timetableview interface is used to get data for one timetable view. For example display hardware with two screens would have two timetableview ids, which are used to get timetable data for each of the screen.

#### 2.6.1 API address

Timetableview API base address is `https://<server>/traveller/rest/timetableview/v2/<method parameters>`. For example, test server API call URL to get timetables for display test\_id would be:

[https://test.server/traveller/rest/timetableview/2/gettimetables?displayid=test\\_id](https://test.server/traveller/rest/timetableview/2/gettimetables?displayid=test_id)

V2 in api URL is version of the interface. This document describes version 2 of this interface.



## DISPLAY API DESCRIPTION V2

16.09.2022

### 2.6.2 GetTimetables

GetTimetables is used to get timetables in this timetable view. Timetable view is configured in Traveller. Timetables are returned for the next 48h or less. The first departure time is returned for the time the call has been made. Subsequent departures are returned if there are any.

*Departure[] getTimetables( )*

Attribute	Type	Explanation
Route	String *	Route id. e.g. A1
Stop	String *	Stop id.
DepartureTime	Timestamp *	Scheduled departure time from this stop.
Line	String *	Short name of line. e.g. 1. Shown to passengers.
DepartureStatus	DepartureStatus *	Status of this departure. e.g. canceled
Destination	MultiLanguageText[]	Destination of the route from this stop.
Via	MultiLanguageText[]	Intermediate destination if any.
Operator	MultiLanguageText[]	Operator of departure if any.
Info	MultiLanguageText[]	Departure information e.g. 'Stop 3 is skipped due to construction'
ArrivalTime	Timestamp	Scheduled arrival time to this stop.
Attributes	String[]	Attributes of this departure. E.g. LowFloor = this departure is driven with a low floor bus.
Platform	String	Platform

Table 3 Departure data type

### 2.6.3 GetAnnouncements

GetAnnouncements is used to get active announcements for this timetable view. A call will return announcements which are currently active or will be active within the next 48h.

*Announcement[] GetAnnouncements( )*

Value	Type	Explanation
Priority	AnnouncementPriority*	Priority for the announcement.
ViewOrder	int	View ordering
Playlist	PresentationTime[]*	List of times when the announcement is shown on the display.
AnnouncementType	AnnouncementType *	Type of announcement. This field sets which type of data field (TextData, ImageData, TrafficData,

## DISPLAY API DESCRIPTION V2

16.09.2022

		StateTextData or StateImageData) contains data. Other type of data fields are empty.
<b>TextData</b>	MultiLanguageText[]	Announcement text in different languages.
<b>ImageData</b>	ImageData	Image announcement data.
<b>TrafficData</b>	TrafficData	This is used to describe what kind of traffic announcement this is. For example Tram or Bus announcement. Traffic type can be used to show different icons for announcement.
<b>StateTextData</b>	StateTextData	
<b>StateImageData</b>	StateImageData	
<b>AnnouncementType</b>	TextData   ImageData   TrafficData   StateTextData   StateImageData	

Table 4 Announcement data type

Value	Type	Explanation
<b>StartTime</b>	Timestamp*	Beginning time for showing announcement on display. UTC time.
<b>EndTime</b>	Timestamp*	Last time when announcement is shown on display. UTC time.

Table 5 Presentationtime data type

AnnouncementPriority	Explanation
<b>NORMAL</b>	Normal priority for announcement.

AnnouncementType	Explanation
<b>TXT</b>	Text announcement
<b>IMAGE</b>	Image announcement
<b>TRAFFIC</b>	Traffic announcement
<b>STATETEXT</b>	State text announcement.
<b>STATEIMAGE</b>	State image announcement.

ImageData	Explanation
<b>Filename</b>	Name of the image file.
<b>URL</b>	URL where the announcement image can be downloaded.
<b>ImageHash</b>	Hash calculated from image. This can be used to determine if image needs to be loaded from the server.

## DISPLAY API DESCRIPTION V2

16.09.2022

TrafficData	Explanation
AnnouncementText	MultiLanguageText[]
TrafficType	This is used to describe what kind of traffic announcement this is. For example Tram or Bus announcement. Traffic type can be used to show different icons for announcement.

StateTextData	Explanation
AnnouncementText	MultiLanguageText[]
AnnouncementState	State for announcement

StateImageData	Explanation
Filename	Name of the image file.
AnnouncementState	State for announcement
URL	URL where the announcement image can be downloaded.
ImageHash	Hash calculated from image. This can be used to determine if image needs to be loaded from the server.

### 2.6.4 PostScreenCapture

PostScreenCapture is used to send a capture of the current view visible on the display to the server. This method is called using an HTTP POST request with the Content-Type of the request body as "multipart/form-data".

Status postScreenCapture( deviceid , timestamp, filedata )

Value	Type	Explanation
deviceid	String *	In addition to the DisplayId that is included as a query parameter to all the API requests, screen captures also require an identifier that uniquely identifies the specific device that is sending the image. This is because there is nothing preventing multiple physical displays from showing the same display contents, but the server must be able to differentiate between them. Deviceid is also a query parameter.
timestamp	Timestamp *	Time when the screen capture was taken. This will be included in the query parameters of the request.
filedata	application/octet-stream *	The screen capture is expected to be a JPEG file encoded in the multipart request body as a binary part with the form-data name "filedata".

## DISPLAY API DESCRIPTION V2

16.09.2022

### 2.7 Display interface

Display interface is used to get data to one display. This data is related to one physical display, which will have one or more screens. For example one PC could have two screen attached. Both screens would show different timetables and so there is one deviceId and two displayIds.

#### 2.7.1 API address

Display API base address is `https://<server>/traveller/rest/display/<method parameters>`. For example getting display configuration for device:

<https://test.server/traveller/rest/display/GetDisplayConfiguration>

#### 2.7.2 PostDisplayStatus

PostDisplayStatus is used to send the current status of a display to the server. This can happen periodically or after certain events. For example, a shock sensor has been triggered. DisplayStatus list can contain one or more statuses.

**Status postDisplayStatus( deviceId, displayStatus[] )**

Value	Type	Explanation
StatusId	String*	Identifier for the status
StatusValue	String*	Status value
StatusChangeTime	Timestamp	Status change time. Can be left null -> Server will use method execution time for change time

Table 6 DisplayStatus datatype

StatusId	Type and values	Explanation
Door_1	String: open, closed	Door 1 status values
Door_2	String: open, closed	Door 2 status values
Shock_low	String: normal, triggered	Shock sensor status: normal , triggered. If sensor is triggered and then it sets to normal state. Both statuses should come server: triggered and then normal
Shock_high	String: normal, triggered	Shock sensor status: normal , triggered. If sensor is triggered and then it sets to normal state. Both statuses should come server: triggered and then normal
IP	String:	Current IP address of the display
Hostname	String	Current hostname of the display
SIM_ID	String	Sim card id of the display

## DISPLAY API DESCRIPTION V2

16.09.2022

Startup_time	Timestamp	Display startup time. The time when the display was powered on.
Button_announcement	String: pressed	Indicates that TTS button has been pressed
Air_quality_pm2_5	String	PM 2.5 air quality value
Air_quality_pm10	String	PM 10 air quality value
Device_temperature	String	Device temperature

Table 7: StatusId values

### 2.7.3 GetDisplayStatusList

GetDisplayStatusList is used to get current display statuses for given display. Returned display statuses are the latest ones from given device. Statuses which can be queried with this interface are described in table below.

**DisplayStatus2[] getDisplayStatusList( String deviceId , String[] statusList )**

Value	Type	Explanation
StatusId	String	Status identifier
StatusValue	String	Status value

Table 8: DeviceStatus2 datatype

Status identifier	Explanation
Air_quality_pm2_5	Air quality for PM 2.5 particles
Air_quality_pm10	Air quality for PM 10 particles

Table 9: Usable statuses for GetDisplayStatusList method

## DISPLAY API DESCRIPTION V2

16.09.2022

### 2.7.4 GetDataUpdateTimes ( not implemented )

GetDataUpdateTimes is used to get last data update time. Update times can be used to check if data queries to server are needed. For example, when TimetableUpdateTime is later than displays last GetTimetables-method call, display should use GetTimetables-method to get updated timetables.

*DataUpdateTime[] getDataUpdateTimes( deviceId )*

Value	Type	Explanation
<b>DatId</b>	String	Data identifier
<b>UpdateTime</b>	Timestamp	Timestamp for last update time for this data.

DatId	Explanation
<b>TimetableUpdateTime</b>	Timestamp for last timetable update
<b>AnnouncementsUpdateTime</b>	Timestamp for last announcement update
<b>DisplayLayoutUpdateTime</b>	Timestamp for last display layout update
<b>DisplayConfigurationUpdateTime</b>	Timestamp for last display configuration update

Table 10 DatId values

### 2.7.5 GetLayoutConfiguration

GetLayoutConfiguration is used to get display layout configuration. This configuration contains items that can be modified using Traveller user interface. It does not contain complete layout information. Complete display layout is outside of the scope of this interface.

*LayoutConfiguration[] getLayoutConfiguration(deviceId)*

Value	Type	Explanation
<b>Id</b>	String	Configuration identifier.
<b>Value</b>	String	Configuration value.

Configuration identifier	Type	Explanation
<b>title_foreground</b>	Color	Title text foreground color
<b>clock_foreground</b>	Color	Clock foreground color
<b>timetable_header_foreground</b>	Color	Timetable header foreground color
<b>timetable_header_line_foreground</b>	Color	Timetable header foreground color – line column

## DISPLAY API DESCRIPTION V2

16.09.2022

<b>timetable_header_destination_foreground</b>	Color	Timetable header foreground color – destination column
<b>timetable_header_time_foreground</b>	Color	Timetable header foreground color – time column
<b>timetable_foreground</b>	Color	Timetable foreground color
<b>timetable_line_foreground</b>	Color	Timetable foreground color – line column
<b>timetable_destination_foreground</b>	Color	Timetable foreground color – destination column
<b>timetable_time_foreground</b>	Color	Timetable foreground color – time column
<b>announcement_foreground</b>	Color	Default announcement color
<b>title</b>	String	Title of the display (header)

Table 11 *LayoutConfiguration* datatype

### 2.7.6 GetDisplayConfiguration

GetDisplayConfiguration is used to get general display settings.

***DisplayConfiguration[] getDisplayConfiguration( deviceId )***

Value	Type	Explanation
<b>Id</b>	String	Configuration identifier.
<b>Value</b>	String	Configuration value.

Configuration identifier	Type	Explanation
<b>Display_Brightness</b>	String	%-value of brightness. Integer number between 0-100 or string 'auto' for automatic brightness control.
<b>Speaker_Volume</b>	Integer between 0-100	%-value of speaker volume on the display. The speaker is mainly used for Text-To-Speech.
<b>air_pollution_limits</b>	AirPollutionConfiguration[]	List of AirPollutionConfiguration objects.
<b>air_pollution_limit_type</b>	String	Air pollution limit type which is used in display. Only two possible values: pm10 or pm2_5
<b>presentation_state</b>	String: normal, only_announcements, blank	Display presentation state: normal: normal functionality only_announcements: display doesn't show timetables. Only announcements are shown. blank: display should be totally blank
<b>stops</b>	String[]	List of assigned stops (e.g. "Stop1", "Stop2")

## DISPLAY API DESCRIPTION V2

16.09.2022

Value	Type	Explanation
<b>pollutionClass</b>	Integer	Air pollution class number. From 0 to 5
<b>name</b>	String	Air pollution class name.
<b>pm10</b>	AirPollutionLimit	PM10 air pollution limits
<b>pm2_5</b>	AirPollutionLimit	PM2,5 air pollution limits

Table 12: AirPollutionConfiguration datatype

Value	Type	Explanation
<b>min</b>	Float	Air pollution minimum
<b>max</b>	Float	Air pollution

Table 13: AirPollutionLimit



## DISPLAY API DESCRIPTION V2

16.09.2022

### 3 REAL TIME INTERFACE

Real time interface API is used to get updated information to a display. Real time interface is an HTML5 Server Sent Events interface. A display will listen for events which are sent from the server. These events will update real time information on the display, for example, forecast departure times. Events may also request the display to do something, for example, the server may request the display to fetch timetables again.

Traveller uses SSE stream named events to send data to displays. Method name in the document will be the SSE Event name. Event data is in JSON format and is described in this document for each event.

#### 3.1 GetDisplayEvents

This method is used to listen for display events from the Traveller server. Messages sent by the Traveller server are described in this document.

API address /traveller/sse/display

Parameter	Type	Explanation
DisplayId	String	DisplayId in Traveller.
Username	String	Username for basic authentication. (In HTTP header)
Password	String	Password for basic authentication. (In HTTP header)

Table 14 GetDisplayEvent common parameters

#### 3.2 Realtime interface event messages

This chapter describes the messages which can be sent by the Traveller server to a display.

##### 3.2.1 UpdateDeparture

An UpdateDeparture message is sent when there is an update for a departure on the display. This update might be a forecast for a departure or a state change. For example, bus arrives to a stop.

Value	Type	Explanation
RouteId	String *	Route short name.
StopId	String *	Stop id.
DepartureTime	Timestamp *	Scheduled departure time from this stop.
DepartureStatus	String *	Status of this departure. I.e. canceled
Attributes	String[]	Departure attributes e.g. LowFloor = low floor bus,
EstimatedDepartureTime	Timestamp	Estimated departure time from the stop.
EstimatedArrivalTime	Timestamp	Estimated arrival time to the stop.
Platform	String	Platform change.

## DISPLAY API DESCRIPTION V2

16.09.2022

Info	MultiLanguageText[][]	Extra information for departure. <i>Not implemented</i>
------	-----------------------	---

### 3.2.2 DataUpdated

An UpdateData message is sent when some data has been changed on the server and the display needs to get new data from the Traveller server.

Value	Type	Explanation
UpdateTime	Timestamp*	Data update timestamp.
UpdatedData	UpdatedDataEnum*	Defines what has been updated.

Value	Explanation
Timetables	Timetables have been changed. The display should get new timetables from the server after the given update time. The update time can be in the past or in the future. If the update time is in the future, the display should get the new timetables from the server when the update time has passed.
Announcements	Announcements have been changed. Display should get announcements after update time. Update time can be in the future.
DisplayLayout	Layout of display has changed and it needs to be read with GetLayoutConfiguration-method.
DisplayConfiguration	Display configuration has changed and it needs to be read with GetDisplayConfiguration-method.

Table 15: UpdatedDataEnum

### 3.2.3 Action

An action message is sent to the display when a specific one-time action is required from the display in question. It is up to the display to see to the completion of the requested action since there is no general acknowledgement message or other confirmations in use for all Action messages.

Value	Type	Explanation
ActionType	ActionTypeEnum *	Name of the action to be performed by the display

Value	Explanation
DisplayRestart	The display is expected to restart itself.
ScreenCapture	The display is expected to send a screen capture of the view that is visible on the monitor at the moment of receiving the message using the REST interface method described in this document.

## DISPLAY API DESCRIPTION V2

16.09.2022

*Table 8: ActionTypeEnum*

## DISPLAY API DESCRIPTION V2

16.09.2022

### 4 INTERFACE USAGE EXAMPLES

This chapter contains examples of how to use the interface with cURL (<https://curl.haxx.se/>) Example data is taken from test server. Examples are to describe common use cases.

curl option -o is used to put return value to return.json -file.

Common parameters used in test cases:

Parameter	Value	Explanation
user	test	Username for interface user
password	test	Password for interface user
DisplayId	test_id	Id of the display in all interface calls.

#### 4.1 GetTimetables

Queries the next timetables from the Traveller server for display with id "test\_id".

```
curl -o return.json --user test:test  
https://testserver/traveller/rest/timetableview/2/gettimetables?displayid=test_id
```

Response is cut after first two departures.

```
{  
  "statusMsg": "OK",  
  "status": 0,  
  "departures": [  
    {  
      "arrivalTime": "2019-08-12T11:56:00Z",  
      "destination": [  
        {  
          "text": "Kotowiecko",  
          "language": "pl"  
        }  
      ],  
      "departureStatus": "NORMAL",  
      "departureTime": "2019-08-12T11:56:00Z",  
      "stop": "229",  
      "route": "kla_17",  
      "line": "17"  
    },  
    {  
      "arrivalTime": "2019-08-12T12:08:00Z",  
      "destination": [  
        {  
          "text": "Majkowska Medix",  
          "language": "pl"  
        }  
      ]  
    }  
  ]  
}
```

## DISPLAY API DESCRIPTION V2

16.09.2022

```
    }  
  ],  
  "departureStatus": "NORMAL",  
  "departureTime": "2019-08-12T12:08:00Z",  
  "stop": "229",  
  "route": "kla_17-1",  
  "line": "17"  
}  
]  
}
```

*Response 1 GetTimetablesResponse*

### 4.2 GetAnnouncements

Queries announcements from the Traveller server for display with id "test\_id"

```
curl -o return.json --user test:test https://testserver/traveller/rest/timetableview/  
2/getannouncements?displayid=test_id
```

```
{  
  "statusMsg": "OK",  
  "status": 0,  
  "announcements": [  
    {  
      "viewOrder": 1,  
      "priority": "NORMAL",  
      "playlist": [  
        {  
          "endTime": "2019-08-31T19:00:00Z",  
          "startTime": "2019-08-08T02:00:00Z"  
        }  
      ],  
      "announcementText": [  
        {  
          "text": "Test announcement",  
          "language": "pl"  
        }  
      ]  
    }  
  ]  
}
```

*Response 2 GetAnnouncementsResponse*

## DISPLAY API DESCRIPTION V2

16.09.2022

### 4.3 PostScreenCapture

Sends a screen capture image to the Traveller server for the display with id "test\_id" and identifies the sender as device "testdevice".

```
curl -o return.json --user test:test -F filedata=@image.jpg  
"https://testserver/traveller/rest/timetableview/2/postscreencapture?displayid=test_id&deviceid=testdevice&timestamp=2020-02-06T15:00:00Z"
```

```
{  
  "status": 0,  
  "statusMsg": "OK",  
}
```

*Response 3 PostScreenCapture response*

### 4.4 PostDisplayStatus

Sends display status to the Traveller server for the display with id "test\_id" and device identifier "testdevice".

```
curl -o return.json --user test:test --request POST --header "Content-Type:  
application/json" --data ' [{"statusId": "Door_1", "statusValue":  
"closed"}, {"statusId": "Door_2", "statusValue": "open"}, {"statusId":  
"Shock_1", "statusValue": "triggered", "statusChangeTime": "2021-02-  
16T18:53:09+02:00"} ]'  
"https://testserver/traveller/rest/display/postdisplaystatus?displayid=test_id&device  
id=testdevice"
```

## DISPLAY API DESCRIPTION V2

16.09.2022

### 4.5 GetDisplayStatusList

Queries display status list for given device. In this example deviceid is SIP3. In the example below two statuses are queried: AIR\_QUALITY\_PM2\_5 and AIR\_QUALITY\_PM10

```
curl --user test:test --location --request GET
'https://testserver/traveller/rest/display/getdisplaystatuslist?deviceid=SIP3&statuslist=AIR_QUALITY_PM2_5&statuslist=AIR_QUALITY_PM10'
```

```
{
  "statusList": [
    {
      "statusId": "AIR_QUALITY_PM2_5",
      "statusValue": "23234"
    },
    {
      "statusId": "AIR_QUALITY_PM10",
      "statusValue": "10234"
    }
  ],
  "status": 0,
  "statusMsg": "OK"
}
```

*Response 4 GetDisplayStatusListResponse*

### 4.6 GetLayoutConfiguration

Queries layout configuration parameters from the Traveller server for the device with identifier "testdevice".

```
curl -o return.json --user test:test
'https://testserver/traveller/rest/display/getlayoutconfiguration?deviceid=testdevice'
"
```

```
{
  "layoutConfiguration": {
    "title_foreground": "#000000",
    "clock_foreground": "#000000",
    "timetable_header_foreground": "#000000",

```

## DISPLAY API DESCRIPTION V2

16.09.2022

```
"timetable_foreground": "#000000",
"announcement_foreground": "#000000"
},
"status": 0,
"statusMsg": "OK"
}
```

*Response 5 GetLayoutConfiguration response*

### 4.7 GetDisplayConfiguration

Queries display configuration parameters from the Traveller server for the device with identifier "testdevice". Example configuration contains only two air pollution limits. There can be more if needed.

```
curl -o return.json --user test:test
"https://testserver/traveller/rest/display/getdisplayconfiguration?deviceid=testdevice"
```

```
{
  "displayConfiguration": [
    {
      "id": "air_pollution_limit_type",
      "value": "pm10"
    },
    {
      "id": "display_brightness",
      "value": "1"
    },
    {
      "id": "speaker_volume",
      "value": "50"
    },
    {
      "id": "presentation_state",
      "value": "normal"
    },
    {
      "id": "air_pollution_limits",
      "value": [
        {
          "pollutionClass": 0,
          "name": "Bardzo dobry",

```



## DISPLAY API DESCRIPTION V2

16.09.2022

```
    "pm10": {
      "min": 10.1,
      "max": 210.0
    },
    "pm2_5": {
      "min": 1000.0,
      "max": 3700.0
    }
  },
  {
    "pollutionClass": 1,
    "name": "Dobry",
    "pm10": {
      "min": 21.1,
      "max": 61.0
    },
    "pm2_5": {
      "min": 13.1,
      "max": 37.0
    }
  },
  {
    "pollutionClass": 2,
    "name": "Umiarkowany",
    "pm10": {
      "min": 61.1,
      "max": 101.0
    },
    "pm2_5": {
      "min": 37.1,
      "max": 61.0
    }
  },
  {
    "pollutionClass": 3,
    "name": "Dosteczny",
    "pm10": {
      "min": 101.1,
      "max": 141.0
    },
    "pm2_5": {
      "min": 61.1,
      "max": 85.0
    }
  }
}
```

## DISPLAY API DESCRIPTION V2

16.09.2022

```
    },
    {
      "pollutionClass": 4,
      "name": "Zły",
      "pm10": {
        "min": 141.1,
        "max": 201.0
      },
      "pm2_5": {
        "min": 85.1,
        "max": 121.0
      }
    },
    {
      "pollutionClass": 5,
      "name": "Bardzo zły",
      "pm10": {
        "min": 201.0,
        "max": 10000.0
      },
      "pm2_5": {
        "min": 121.0,
        "max": 10000.0
      }
    }
  ],
  "status": 0,
  "statusMsg": "OK"
}
```

*Response 6 GetDisplayConfiguration response*

### 4.8 SSE Events

SSE event stream can be listened with cURL:

```
curl -N --user test:test https://testserver/traveller/sse/display?displayid=test_id
```

## DISPLAY API DESCRIPTION V2

16.09.2022

Events which can be read from the stream are in the examples below.

### 4.8.1 DepartureUpdate

Departure update for passing time update. Forecasting system has created a new departure time from given stop.

```
event:DepartureUpdate
data:{"routeId":"kla_17","stopId":"229","departureTime":"2019-08-12T11:56:00Z","estimatedDepartureTime":"2019-08-12T11:56:05Z"}
```

Departure update for departure cancellation on given stop.

```
event:DepartureUpdate
data:{"routeId":"kla_17-1","stopId":"229","departureTime":"2019-08-12T12:08:00Z","status":"CANCELED","estimatedDepartureTime":"2019-08-12T12:08:00Z"}
```

### 4.8.2 DataUpdated

Timetables have been updated

```
event:DataUpdated
data:{"updateTime":"2019-08-12T12:47:16Z","updatedData":"Timetables"}
```

Announcements have been updated

```
event:DataUpdated
data:{"updateTime":"2019-08-12T12:49:59Z","updatedData":"Announcements"}
```

### 4.8.3 Action

Request for the display to restart itself.

```
event:Action
data:{"actionType":"DisplayRestart"}
```

Request for a screen capture.

## DISPLAY API DESCRIPTION V2

16.09.2022

```
event:Action  
data:{"actionType":"ScreenCapture"}
```