JSON API

This brief document equips you with what you need to get started with the Vision APIs. As a valued member of the Vision fraternity, you should have already obtained the credentials to access the platform APIs. If not, please contact your local Vision provider/dealer.

Each API call is listed with what is required as its input and what that API call will provide as its output.

1. GET LOCATIONS

This API call provides you list of locations in your customer account. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint : *GET: {host_name}/api/external/locations*

Request should have Authorization header with your customer token.

Ex: *ajax({ type: "GET",*
url: `/api/external/locations`,
headers: {
'Authorization': `Bearer ${customer_token}`
}),

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format: The data token contains the result of the execution.

data.locations:

```
[ {
  "locationId": "location Id",
  "locationName": "location Name",
  "address": {
      "geo": {
          "latitude": xx.xxxxxxx,
          "longitude": xx.xxxxxxx
       },
      "street": "xxxxx",
      "street2": " xxxxx ",
      "city": " xxxxx ",
       "state": " xxxxx ",
       "postal": " xxxxx ",
       "country": " xxxxx "
   },
   "phone": " xxxxxxxxxx ",
   "timezone": "America/NewYork"
},…….] // end of locations
```

Result in case of failure

If the request fails, the server sends the response data in the following JSON format. The err code member contains the error code of the execution.

Err code	Description
401	**When No authorization header found in request headers** *{ error:* *{* *reason: "authorizationRequired", message: "No request headers for authorization".* *code: 401,* *}* *}* **When no token found in authorization header** *{ error:* *{* *reason: "noToken",* *message: "No token in request headers for authorization".* *code: 401,* *}* *}*
500	*{ error:* *{* *reason: "internalError", message: "Encountered an internal error. Please try again.",* *code: 500,* *}* *}*
404	*{* *error: {* *reason: "notFound",* *message: "No customer found with token",* *code: 404,* *}* *}*

The failure messages will be similar for all the API calls.

2. GET GATEWAYS IN A LOCATION

This API call provides you list of gateways in the specified location. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is: **API endpoint:** *GET: {host_name / api/external /location/gateways/{locationId}*

Request should have Authorization header with your customer token .

Ex: *ajax({ type: "GET", url: `/api/external/ location/gateways/{locationId}`, headers: { 'Authorization': `Bearer ${customer_token}` }),*

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format:
The data token contains the result of the execution. data.gateways : [*{*

"gatewayId": "gateway Id",

"gatewayName": "gateway Name",

"locationId": "location Id"

},.....] // end of gateways

Result in case of failure

For the error message format – refer to the section above in the document.

3. GET FEEDS IN A GATEWAY

This API call provides you list of feeds in the specified gateway. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint: *GET: {host_name / api/external /gateway/feeds/{gatewayId}*

Request should have Authorization header with your customer token.

Ex: *ajax({ type: "GET", url: `/api/external/gateway/feeds/{gatewayId}`, headers: { 'Authorization': `Bearer ${customer_token}`*

}),

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format: The data token contains the result of the execution.

```
data.feeds: [{
                    "feedId": "feed id",
                   "feedName": "feed name",
    "locationId": "location Id",
                    "gatewayId": " gateway Id ",
                    "audio":  true/false,  //  audio  enabled or not on this feed
                    "recording": {
                            "mode": "manual / schedule",
                             "days": 3, //no of days of cloud storage
                            "status": true/ false, //current status of recording or not
                            "schedule": [{ // if mode is scheduled
                                              "hours":  1, // number of hours to record from start
                                             "daysOfWeek":  [1,   2,   3,   4,   5  ],
                                              //days of week starts from Monday
                                            "start":  "12:00 PM" // recording start time
                            } .....]
                     },
}..... ] // end of feeds
```

Result in case of failure

For the error message format – refer to the section above in the document.

4. GET FEEDS IN A LOCATION

This API call provides you list of feeds in the specified location. Note that this includes all the feeds in the location including feeds added through all gateways and feeds added directly to cloud. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint: *GET: {host_name / api/external /location/feeds/{locationId}*

Request should have Authorization header with your customer token.

Ex:
```
ajax({ type: "GET",   url: `/api/external/ location /feeds/{
        locationId }`,
        headers: {
                'Authorization':  `Bearer ${customer_token}`
}),
```

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format: The data token contains the result of the execution.

```
data.feeds: [{
                    "feedId": "feed Id",
                   "feedName": "feed name",
    "locationId": "location Id",
                    "gatewayId": " gateway Id ", //for cameras added to cloud gatewayId is null
                   "audio":  true/false,  //  audio  enabled or not on this feed
          "recording": {
                        "mode": "manual or schedule",
                         "days": 3,
                        "status": true/ false, //  recording happening  or not
                        "schedule": [{ // if mode is schedule
                                          "hours":  1, // number of hours to record from start
                                         "daysOfWeek":  [1,   2,   3,   4,   5  ],
                                          //days of week starts from Monday
                                        "start":  "12:00 PM" // recording start time
                        } .....]
                    },
}..... ] // end of feeds
```

Result in case of failure

For the error message format – refer to the section above in the document.

5. GET RECORDINGS FOR A FEED

This API call provides you list of recordings for the specified feed. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint: *GET: {host_name /api/external/feed/recordings/{feedId}/{start}/{end}*

Request should have Authorization header with your customer token.

The request parameters *start* and *end* should be in milliseconds. For example, to fetch recordings of feedId - xxxx, starting from June 1st, 2020 12:00 AM IST upto June 30th, 2020 11:59 PM IST

Ex: ajax({ type: "GET",

*url: `/api/external/* feed/recordings/ xxxx/*1590949800*/1593541740`,

headers: {

'Authorization': `Bearer ${customer_token}`

}),

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format:
The data token contains the result of the execution.
Note: Dates in below data follow ISO-8601 date representation *data.recordings:*

[{

"recordingId": "xxxx",

"duration": xx, ***// in seconds***

"locationId": "location Id",

"feedId": "feed Id",

"time": "2020-06-16T07:04:52.000Z", // start time

"end": "2020-06-16T07:06:25.000Z" // end time

}….]

Result in case of failure

For the error message format – refer to the section above in the document.

6. TO PLAY/DOWNLOAD A CAMERA'S RECORDED ARCHIVE

This API call allows you to play/download a specific recorded archive of a specified camera. You can call this API by sending an HTTP GET request to its URI.

The address and the format of the API call is:

API endpoint: *GET: api/external/ feed/recording/media/${recordingId}?token='<customer_token>'*

Ex: *ajax({ type: 'GET*

url: `/api/external/feed/recording/media/<recording-Id>? *token='<customer_token>'*

*})`

Result in case of success

If the request succeeds, the server sends the recording data in binary format.

Result in case of failure

For the error message format – refer to the section above in the document.

7. VIEW SHARED LINKS FOR A FEED

This API call provides you list of shareable links for the specified feed. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint: *GET: {host_name /api/external/feed/shares/{feedId}*

Request should have Authorization header with your customer token.

Ex: *ajax({ type: "GET", url: `/api/external/* feed/
shares/{feedId}`,
headers: {
'Authorization': `Bearer ${customer_token}`
}),

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format:
The data token contains the result of the execution.
Note: Dates in below data follow ISO-8601 date representation *data.shares:*

[{

"perpetual": true,/false // If true link never expires; otherwise expires on end

"start": "2020-06-15T08:06:00.000Z",

"end": "2020-06-15T08:07:12.076Z", // If perpetual = true ignore this

"feedId": " feed Id ",

link:
*"https://{**host_name**}/feed/embed/xxxxxxxxxxxx/?token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7InNoYXJlIjp7Il9pZCI6IjVmMWViMGMyOWU1ODhiNzg5ZTIxZmExOSIsInN0YXJ0IjoiMjAyMC0wNy0yN1QxMDoyNjowMC4wMDBaIiwicGVycGV0dWFsIjpmYWxzZSwiZmVlZElkIjoiNWYxYjIwYTgzZGVlYzI2M2Q4NTUxZDk2IiwiZW5kIjoiMjAyMC0wNy0zMVQxMDoyNjowMC4wMDBaIn19LCJleHAiOjE1OTYxOTQ3NjAsImlhdCI6MTU5NTg0Njg1MH0.pttvIOKpHNuHjUeU2ctloNX84TFmaIKP7MJ7BDc2hDY"*

}....]

Result in case of failure

For the error message format – refer to the section above in the document.

8. ADD SHAREABLE LINK FOR A FEED

This API call allows you to create a shareable link for the specified feed. You can call this API by sending an HTTP POST request to its URI. The address and the format of the API call is:

API endpoint: *POST: api/external/feed/add_share/{feedId}*

Request Body should have:

```
"share": {
        "start": "Date  Object",
         "end": "Date Object",
         "perpetual": "true or false", // if link never expires – true
        "feedId": "feedId"
 }
```

Ex: *ajax({ type: 'POST'*

```
url: `/api/external/feed/add_share/${this.new_share.feedId}`,
headers: {
        'Authorization': `Bearer ${v.customer.token}`
},

data: { share: share}

}),
```

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format:
The data token contains the result of the execution – which contains the link that was created. Note: Dates in below data follow ISO-8601 date representation.

```
Data = {
        "success":
                {
                "message":"Sharable link added successfully",
                 "code":200,
                "AddedLink":{
                    "id":"xxxxx",
                    "start":"2020-07-20T10:26:00.000Z",
                    "perpetual":false,
                    "feedId":"5f1b20a83deec263d8551d96",
                            "end":"2020-07-
                31T10:26:00.000Z",    link:
                "https://{host_name}/feed/embed/5f1b20a83deec263d8551d96/?token=eyJhb
```

GciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7InNoYXJlIjp7Il9pZCI6IjVmMWViM
GMyOWU1ODhiNzg5ZTIxZmExOSIsInN0YXJ0IjoiMjAyMC0wNy0yN1QxMDoyNjo
wMC4wMDBaIiwicGVycGV0dWFsIjpmYWxzZSwiZmVlZElkIjoiNWYxYjIwYTgzZGVl
YzI2M2Q4NTUxZDk2IiwiZW5kIjoiMjAyMC0wNy0zMVQxMDoyNjowMC4wMDBaI
n19LCJleHAiOjE1OTYxOTQ3NjAsImlhdCI6MTU5NTg0Njg1MH0.pttvIOKpHNuHjU
eU2ctloNX84TFmaIKP7MJ7BDc2hDY"

}

}

}

Result in case of failure

For the error message format – refer to the section above in the document.

9. GET LIST OF ANALYITCS CONFIGURED FOR A FEED

This API call allows you to get all the analytics configured for the specified feed. You can call this API by sending an HTTP GET request to its URI. The address and the format of the API call is:

API endpoint: *GET: api/external/ analytics/types/{Id}*

Note: Id should be the feed id. If you like to get list of all analytics in the system (not for a specific feed, use "all" instead of feed id.

Ex: *ajax({ type: 'GET*

url: `/api/external/ analytics/types/{Id}`,
headers: {
'Authorization': `Bearer ${v.customer.token}`
},

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format:

The data token contains the result of the execution. *data.analytics_types: [{*

"value": "line_crossing",

'label': 'Line crossing'

}….]

Result in case of failure

For the error message format – refer to the section above in the document.

10. GET ANALYITCS EVENTS FOR LOCATIONS

This API call allows you to get all the analytic events that were raised for all feeds of the specified locations. You can call this API by sending an HTTP POST request to its URI.

The address and the format of the API call is:

API endpoint: *POST: api/external/location/analytics/events?page={page_number}&page_size={size}`*

Note: If request does not have query string then default values for *'page'* and **'***page_size'* are *0* and *500* respectively. Page value should start from zero**.**

Request Body should have:

```
{
    "uiTypes":  [analytic types values],   // Should have atleast one value
     "locations":  [location id's],    // Should have atleast one value
    "start":  date,
     "end":  date
}
```

Note: ***start*** and ***end*** dates should follow the ISO-8601 date representation.

Ex: *ajax({ type: 'POST'*

```
 url: `/api/external/location/analytics/events?page=0&page_size=100`,
headers: {
        'Authorization': `Bearer ${v.customer.token}`
 },

 data:  {
   "uiTypes": ["line_crossing","people_count","vehicle_count","zone_intrusion"],
    "locations": ["location Id-1","location Id-2"],
    "start": "2020-07-06T07:39:00.000Z",
    "end": "2020-07-23T19:39:03.939Z"
 },
```

Result in case of success

If the request succeeds, the server sends the response data in the following *JSON* format: The data token contains the result of the execution.

```
data : {  total:  xx //total events

    count events: []

    }
```

Note: Events will have the following JSON format *events: {*

// example for People Count event type.

"_id": "event Id",

"name": "Zone 1",

"object_classification": "person, p: 0.93",
"type": "people_count",

*"time": 1594909770235, // **in milliseconds***

"feedId": "feed Id",

"locationId": "location Id",

"images": [

{

"uuid": "11c0e0b2-eb7a-496c-9b29-afdcdf5aae6b",

"name": "image",

"mimetype": "image/jpeg",

"path": 'https://<host_name>/api/external/events/<event-id>/image/<imageindex>?token=customer_token'

} …

],

"clips": [

{

"uuid": "11c0e0b2-eb7a-496c-9b29-afdcdf5aae6b",

"name": "clip",

"mimetype": "video/mp4",

"path": 'https://host_name/api/external/events/5f884af5aa420e06538a1b29/clip/0?token=customer_token'

}

],

"timezone": "America/New_York"

}, ….]

Result in case of failure

For the error message format – refer to the section above in the document.

NOTE:

1. Face_match has detected person details. It has the following format

face_match: {
person: {
email: "email of person", // will contain email id of the recognized person or empty for unrecognized
engineId: "-999/id1" //-999 for unrecognized, and any integer value for recognized externalId: "", //empty for unknown, entered value for known
name: {
first: " unknown/Firstname", // "unknown" for unrecognized, or name for recognized
last: " " //empty for unrecognized, or entered last name for recognized
}
}
}

2. License Plate has detected license plate number. It has the following format

license_plate_number: "TS07EA8099/TS07E8099" // it will be empty if number is not recognized from the license plate OR license number(s) as recognized by the system each recognized number is delimited by a "/"

vehicle: { // this will be "null" for unknown vehicles. For known vehicles the following data will be provided – whatever was added in the system for that vehicle.
country: "India",
internal_id: "Z-1301",
make: "", model: "4W",
plate: "AP11AP1111",
state: "Andhra Pradesh",
year: 2021
}

3. Image or clip for each event will have the path from where it needs to be fetched with a separate call.

To fetch the image of an event:

You can get analytics events image by sending an HTTP GET request to its URI as mentioned in the "*path*" field of the response.

GET: *api/external/ events/<event-Id>/image/<image-index>?token='<customer_token>'*
Ex: ajax({ type: 'GET

url: `/api/external/events/<event-id>/image/<image-index>?token='<customer-token>`,

}) //if there are multiple images for the same event, use the appropriate "image-index" value in the above url to fetch specific image.

To fetch the clip of an event:

You can get analytics events clip/video by sending an HTTP GET request to its URI as mentioned in the "*path*" field of the response.

GET: *api/external/ events/<event-Id>/clip/<clip-index>?token='<customer_token>* `

Ex: ajax({ type: 'GET url: `/api/external/events/<event-id>/clip/<clip-

index>?token='<customer-token>`,

}) //if there are multiple clips for the same event, use the appropriate "clip-index" value in the above url to fetch specific clip.

11. GET ANALYITCS EVENTS FOR FEEDS

This API call allows you to get all the analytic events that were raised for the specified feeds. You can call this API by sending an HTTP POST request to its URI.

The address and the format of the API call is:

API endpoint: *POST: api/external/ feed/analytics/events?page={page_number}&page_size={size}*`

Note: If request does not have query string then default values for *'page'* and **'***page_size'* are *0* and *500* respectively. Page value should start from zero**.**

Request Body should have:

```
{
  "uiTypes": [analytic type values],   // Should have atleast one value
  "feeds": [feed Id-1, feed Id-2],    // Should have atleast one value
  "start": start date,
```

```
    "end":  end date
}
```

Note: start and end dates should follow the ISO-8601 date representation.

Ex: *ajax({ type: 'POST'*

```
url: `/api/external/feed/analytics/events?page=0&page_size=100`,
headers: {
        'Authorization': `Bearer ${v.customer.token}`
},

data:  {
        "uiTypes": ["line_crossing","people_count","vehicle_count","zone_intrusion"],
        "feeds": ["xxxxx","xxxxxx"],
        "start": "2020-07-06T07:39:00.000Z",
        "end": "2020-07-23T19:39:03.939Z"
},
```

Result in case of success

For the result format will be similar to the result of "GET ANALYITCS EVENTS FOR LOCATIONS" API call – refer to the section above in the document.

Result in case of failure

For the error message format – refer to the section above in the document.

12. FETCH HEALTH STATUS OF FEED(S)

This API call allows you to get health status of specified feeds by sending an HTTP POST request to its URI

API endpoint: *POST: 'api/external/ feeds/status'*

Request Body should have:

```
{
    "feeds":  ["feed Id-1","feed Id-2"],    // Should have at least one value
}
```

Ex: *ajax({ type: 'POST'*

```
url: `/api/external/feeds/status`,
headers: {
```

```
            'Authorization': `Bearer ${customer.token}`
        },
        data:  {
            "feeds": ["feed Id-1","feed Id-2”]
        },
```

Result in case of success

If the request succeeds, the server sends the response data in the following format:

```
data : {
    feeds_status: [
            { feedId: “feed Id”,
            feedName: “feed name”,
        status: “online/offline”,
        locationId: “location Id” } ….. ]
}
```

Result in case of failure

For the error message format – refer to the section above in the document.

13. FETCH FEED(S) HEALTH STATUS BY LOCATION(S)

This API call allows you to get health status of all feeds from the specified locations by sending an HTTP POST request to its URI

API endpoint: *POST: ‘api/external/ location/feeds/status `*

Request Body should have:

```
{       "locations":  [location id’s],    // Should have at least one value
}
```

Ex: *ajax({   type: ‘POST’*

```
        url: `/api/external/location/feeds/status`,
        headers: {
            'Authorization': `Bearer ${customer.token}`
        },
        data:  {
            " locations": ["location Id1","location Id2"]
        },
```

Result in case of success

If the request succeeds, the server sends the response data in the following format:

```
        data : {
feeds_status: [
```

*{ feedId: “*feed Id”*,*
feedName: “feed name”,
status: “online/offline”,
*locationId: “*location Id” *}*
…..]
}

Result in case of failure

For the error message format – refer to the section above in the document.