API DOCUMENTS FOR BohoV2

Table of Contents

I General API Information

- The base endpoint is: http://192.168.0.55:5500
- All enpoints return eithere a JSON object, array or image in binary
- All time and timestampp related fields are follow Korea/Vietnam UTC time format

I.1 HTTP Return Codes

- `HTTP` 400 return code are bad requests
- `HTTP` 401 return code are used for malformed requests, the issue is on the sender's side.
- `HTTP` 405 return code is unauthorized request
- `HTTP` 200 return code is used for successful request
- `HTTP` 201 return code is created successful request
- `HTTP` 206 return code is the request has succeeded and the body contains the requested ranges of data
- `HTTP` 5xx return codes are used for internal errors; the issue is on server's side. It is important to NOT treat this as a failure operation

I.2 GENERAL INFORMATION ON ENDPOINTS

- For `GET` endpoints, parameter mus be sent as a `query string`
- For `POST`, `PUT` and `DELETE` endpoints, the parameters may be sent as a `query string` or in `request body` with content type `apllication/json`
- Paramaters may be sent in any order.
- If a parameter sent in both the `query string` and `request body`, the `query string` parameter will be used.

II UPDATE NOTE

`1. UPDATE NOTE (10/06/2022)`

- Implement CRUD APIs for User management module
- Implement CRUD APIs for Config management module
- Implement CRUD APIs for Node management module
- Implement CRUD APIs Device management module
- Implement Authorize process for all APIs

`2. UPDATE NOTE (20/07/2022)`

- Fixed create device error
- Updated delete method for Device & Node module
- Updated config structure module
- Updated node structure module
- Added new field data into Node table
- Updated document information
- Remove **type** field in Configure module

3. UPDATE NOTE (27/07/2022)

- Change API structure of create & update configuration for devices
- Fix some minor errors in API documents
- Update API to query detail information of node

4. UPDATE NOTE(08/08/2022)

- Re-design dataflow of create & update configurtion for devices
- Add new datafield ***preset_name*** into device configuration table
- create auto get preset information from camera
- Create sync function to map infomration from PTZ camera to device
- Add new API to get single camera configuration
- Add new snapshot API
- Add new datafield ***user and password*** into device metadata

5. UPDATE NOTE(09/08/2022)

- Update structure of snapshot API
- Update define datafield of configure

6. UPDATE NOTE(31/08/2022)

- Modify DeviceConfiguration CRUD
- Change device configure data request structure

7. UPDATE NOTE(27/09/2022)

- Update DeviceConfiguration rules when create config
- Fix API bugs
- Add Edge Controller API
- Add send/update configurate for Edge
- Add start/stop Edge system
- Add new data field for saving PTZ events

8. UPDATE NOTE(23/05/2023)

- Implement CRUD APIs for Rule & Object configurate module
- Implement CRUD APIs for Calibrate & Alarm manager module
- Remove Device Confiugration table
- Re-factor foreign key between Track, Alarm, and Device tables
- Update Device and Node key information
- Re-factor requests APIs

9. UPDATE NOTE(31/05/2023)

- Update Rule module APIs request
- Update Alarm module APIs request
- Update database construction

10. UDATE NOTE(27/06/2023)

- Implement Milestone Intergrate Module
- Update Rule request and attribute
- Update database construction

11. UDATE NOTE(21/07/2023)

- Update Rule, Alarm, and Object management request APIs
- Update attribute names of Object management APIs
- Add VMS nodes type
- Add VMS sycn API
- Update node_metadata of node management APIs
- Update snapshot support function on PTZ and Static cameras
- Update database construction

12. UDATE NOTE(27/08/2023)

- Update is_activate field for node

- Check the activate status on Start/Stop service function

13. UDATE NOTE(26/10/2023)

- Update CRUD for Node Operator module
- Update CRUD for device Group module
- Update CRUD for Touring, Patrol, Preset module
- Refactor API structure for Device, and Camera module
- Delete API list support for Camera module

13. UDATE NOTE(06/11/2023)

- Update Group Management API
- Update Node Operator API
- Update Node Management API
- Update Device Management API

14. UDATE NOTE(06/11/2023)

- Update Search event by filter API
- Refactor unused API in document
- Rule configurate refactor

15. UDATE NOTE(22/11/2023)

- Refactor endpoint in BohoV2
- Remove patrol management and merge with patrol module

16. UDATE NOTE(01/12/2023)

- Update device status response in add and get request
- Add get devices list by group id
- Add verify connection by ip and port

17. UDATE NOTE(03/12/2023)

- Add milestone service CRUD restful API
- Re-structure Touring request format
- Add node quantity to node operator request
- Re-structure Patrol request format

18. UDATE NOTE(08/12/2023)

- Re-structure Touring request format
- Re-format data return and request in json form
- Add CRUD API for Rule Schedule module

19. UDATE NOTE(09/12/2023)

- Add update or create schedule module (touring management module)
- Add update and delete API for single schedule of touring
- Re-factor delete touring API

19. UDATE NOTE(03/12/2023)

- Update the response and request value for search API
- Update search by index for search API
- Add event management API list
- Add sync preset between camera and server
- Add preset management API list
- Add sync touring process for edge service API
- Add sync rule process for edge service API

III. CODE RETURN

> code `1000` : Request Done
>
> code `1001` : Request Fail
>
> code `1002` : Data is invalid in database
>
> code `1003` : Video is processing
>
> code `1004` : Partial or empty data return

> code `1005`: Unauthorize return

IV. API USAGE

---

1 User Management API

ADD USER

- Create POST request to create new user(only admin user can add the user)

```
POST /api/rest/v1/user/add_user
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
    "role" : "str",
    "password": "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

Note : The password rule: password must contain both lower and upper letter and at least special sysbol is required (!,@,#,$,%,^,&,)

GET USERS

- Create GET request to get the list of user(only admin user can use this function)

```
POST /api/rest/v1/user/get_users
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message": "str",
  "data": [{
    "id": "str",
    "name" : "str",
    "email": "str"
  }],
  "success": "boolean"
}
```

LOGIN

- Create POST request to login the system

```
POST /api/rest/v1/login
```

```
Request Header
Authorization: No
Type: application/json
```

```
Request: {
    "name" : "str",
    "password": "str"
}

Response: {
  "data": "str", //authorize token
  "message": "str",
  "success": "boolean"
}
```

Note : Each token will have expired time 24 hours. After 24 hours the user will need to login again to access another APIs

---

CHANGE PASSWORD

- Create POST request to change the password of user (current user or admin user can update the password)

```
POST /api/rest/v1/user/update_password
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "user_id" : "str", //only require when change by admin user
    "password": "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

UPDATE ROLE

- Create POST request to change the role of user (admin user can update the password)

```
POST /user/update_role
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "user_id" : "str",
    "role": "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

---

2 Group Management API

ADD GROUP

- Create POST request to create group for camera group

```
POST /api/rest/v1/group
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request : {
  "name" : "str",
  "describle" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

GET GROUPS

- Create GET request to get the list of groups

```
GET /api/rest/v1/group
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data": [
      {
          "id" : "str",
          "name" : "str",
          "describle" : "str"
      }
    ]
  },
  "message": "str",
  "success": false
}
```

GET GROUP IN DETAIL

- Create GET request to get the group by id

```
GET /api/rest/v1/group/<group_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data":
      {
        "id" : "str",
        "name" : "str",
        "describle" : "str"
      }
  },
  "message": "str",
  "success": false
}
```

UPDATE GROUP INFO

- Create PATCH request to update node group information

```
PATCH /api/rest/v1/group/<group_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
  "describle" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE GROUP

- Create DELETE request to delete group information

```
DELETE /api/rest/v1/group/<group_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

ADD GROUP MANAGEMENT

- Create POST request to create group for camera group management

```
POST /api/rest/v1/group_management
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request : {
  "group_id" : "str",
  "device_id" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

GET GROUPS MANAGEMENT

- Create GET request to get the list of groups management

```
GET /api/rest/v1/group_management?<group_id>='str'
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Note: - not add group_id the query will return group management of that user in list
      - add group_id the query will return  group management of that group in list
```

```
Request: None

Response:{
  "data": {
    "data": [
      {
          "id" : "str",
          "device_id" : "str",
          "group_id" : "str"
      }
    ]
  },
  "message": "str",
  "success": false
}
```

UPDATE GROUP MANAGEMENT INFO

- Create PATCH request to update node group information

```
PATCH /api/rest/v1/group_management/<group_management_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "device_id" : "str",
  "group_id" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE GROUP MANAGEMENT

- Create DELETE request to delete group information

```
DELETE /api/rest/v1/group_management/<group_management_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

3 Node Operator Management API

ADD NODE OPERATOR

- Create POST request to create node operator for edge serivce

```
POST /api/rest/v1/node_operator
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request : {
  "name" : "str",
  "describle" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

GET NODE OPERATORS

- Create GET request to get the list of node group

```
GET /api/rest/v1/node_operator
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data": [
      {
          "id" : "str",
          "name" : "str",
          "describle" : "str",
          "node_count" : "int"
      }
    ]
  },
  "message": "str",
  "success": false
}
```

GET NODE OPERATOR IN DETAIL

- Create GET request to get the node operator by id

```
GET /api/rest/v1/node_operator/<node_operator_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data":
      {
        "id" : "str",
        "name" : "str",
        "describle" : "str",
        "node_count" : "str"
      }
  },
  "message": "str",
  "success": false
}
```

UPDATE NODE OPERATOR INFO

- Create PATCH request to update node operator information

```
PATCH /api/rest/v1/node_operator/<node_operator_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "node_id" : "str",
  "name" : "str",
  "describle" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE NODE OPERATOR

- Create DELETE request to delete node operator information

```
DELETE /api/rest/v1/node_operator/<node_operator_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

4 Milestone Management API

ADD MILESTONE SERVICE

- Create POST request to create milestone device

```
POST /api/rest/v1/milestone
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "communication_port": "int",
    "login_info" : {
      "host" : "str",
      "port" : "int",
      "user" : "str",
      "password" : "str"
    },
    "authen_type" : "str"
    }

Response: {
  "message": "str",
  "success": "boolean"
}
```

GET MILESTONES

- Create GET request to get the list of milestone by user id

```
GET /api/rest/v1/milestone
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data": [
      {
        "id": "str",
        "user_id" : "str",
        "communication_port": "int",
        "login_info" : {
          "host" : "str",
          "port" : "int",
          "user" : "str",
          "password" : "str"
        },
        "authen_type" : "str"
      }
    ],
    "message": "str"
  },
  "message": "str",
  "success": false
}
```

GET MILESTONE IN DETAIL

- Create GET request to get the milestone information

```
GET /api/rest/v1/milestone/<milestone_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data":
      {
        "communication_port": "int",
        "login_info" : {
          "host" : "str",
          "port" : "int",
          "user" : "str",
          "password" : "str"
        },
        "authen_type" : "str"
      }
    ,
    "message": "str"
  },
  "message": "str",
  "success": false
}
```

UPDATE MILESTONE SERVICE INFO

- Create PATCH request to update milestone information

```
PATCH /api/rest/v1/milestone/<milestone_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "communication_port": "int",
    "login_info" : {
      "host" : "str",
      "port" : "int",
      "user" : "str",
      "password" : "str"
    },
    "authen_type" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE MILESTONE

- Create DELETE request to delete milestone information

```
DELETE /api/rest/v1/milestone/<milestone_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

5 Node Management API

ADD NODE

- Create POST request to create edge device

```
POST /api/rest/v1/node
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "location" : {
        "lat": "str",
        "lon": "str",
    },
    "name": "str",
    "type": "str", /// DeepStream or TensortRT
    "ip": "str",
    "node_metadata": {
      "user" : "str", ///  required for ssh connection
      "password" : "str", /// required for ssh connection
    },
    "connection_metadata": {
        "kafka" : {
            "port": "str",
            "topic": "str"

        },
        "mqtt": {
            "port": "str",
            "topic": "str"
        },
        "socket": {
            "port": "str"
        }
    },
    "engine_metadata" : {
      "resolution" : {
        "width" : "int",
        "height" : "int"
      },
      "frame_rate" : "int", // Maximum value is 30
      "sensitive": {
        "detection" : "int", //Range value from 1 to 5
        "tracking" : "int" //Range value from 1 to 5
      },
      "frame_step" : "int"
    }
    }

Response: {
  "message": "str",
  "success": "boolean"
}
```

GET NODES

- Create GET request to get the list of nodes by user id

```
GET /api/rest/v1/node
```

```
 Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data": [
      {
          "id": "str",
          "node_operator_id": "str",
          "location" : {
              "lat": "str",
              "lon": "str",
          },
          "name": "str",
          "type": "str", /// DeepStream or TensortRT
          "ip": "str",
          "is_active" : "boolean", /// The atribute to validate the status process of node service
          "node_metadata": {
            "user" : "str", /// required for ssh connection
            "password" : "str", /// required for ssh connection
          },
          "connection_metadata": {
              "kafka" : {
                  "port": "str",
                  "topic": "str"

              },
              "mqtt": {
                  "port": "str",
                  "topic": "str"
              },
              "socket": {
                  "port": "str"
              }
          },
          "engine_metadata" : {
            "resolution" : {
              "width" : "int",
              "height" : "int"
            },
            "frame_rate" : "int", // Maximum value is 30
            "sensitive": {
              "detection" : "int", //Range value from 1 to 5
              "tracking" : "int" //Range value from 1 to 5
            },
            "frame_step" : "int"
          }
      }
    ],
    "message": "str"
  },
  "message": "str",
  "success": false
}
```

GET NODES BY NODE OPERATOR

- Create GET request to get the list of nodes by user id

```
GET /api/rest/v1/nodes
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
End point : /api/rest/v1/node?npi=<str>
npi : node operator id
```

```
Response:{
  "data": {
    "data": [
      {
          "id": "str",
          "node_operator_id": "str",
          "location" : {
              "lat": "str",
              "lon": "str",
          },
          "name": "str",
          "type": "str", /// DeepStream or TensortRT
          "ip": "str",
          "is_active" : "boolean", /// The atribute to validate the status process of node service
          "node_metadata": {
            "user" : "str", /// required for ssh connection
            "password" : "str", /// required for ssh connection
          },
          "connection_metadata": {
              "kafka" : {
                  "port": "str",
                  "topic": "str"

              },
              "mqtt": {
                  "port": "str",
                  "topic": "str"
              },
              "socket": {
                  "port": "str"
              }
          },
          "engine_metadata" : {
            "resolution" : {
              "width" : "int",
              "height" : "int"
            },
            "frame_rate" : "int", // Maximum value is 30
            "sensitive": {
              "detection" : "int", //Range value from 1 to 5
              "tracking" : "int" //Range value from 1 to 5
            },
            "frame_step" : "int"
          }
      }
    ],
    "message": "str"
  },
  "message": "str",
  "success": false
}
```

GET NODE IN DETAIL

- Create GET request to get the node information

```
GET /api/rest/v1/node/<node_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
    "data":
      {
          "node_id": "str",
          "node_operator_id" : "str",
          "location" : {
              "lat": "str",
              "lon": "str",
          },
          "name": "str",
          "type": "str", /// DeepStream or TensortRT
          "ip": "str",
          "is_active" : "boolean", /// The atribute to validate the status process of node service
          "node_metadata": {
            "user" : "str", /// Not required only for VMS
            "password" : "str", /// Not required only for VMS
            "end_point": "str" /// Not required only for VMS
          },
          "connection_metadata": {
              "kafka" : {
                  "port": "str",
                  "topic": "str"

              },
              "mqtt": {
                  "port": "str",
                  "topic": "str"
              },
              "socket": {
                  "port": "str"
              }
          },
          "engine_metadata" : {
            "resolution" : {
              "width" : "int",
              "height" : "int"
            },
            "frame_rate" : "int", // Maximum value is 30
            "sensitive": {
              "detection" : "int", //Range value from 1 to 5
              "tracking" : "int" //Range value from 1 to 5
            },
            "frame_step" : "int"
          }
      }
    ,
    "message": "str"
  },
  "message": "str",
  "success": false
}
```

UPDATE NODE INFO

- Create PATCH request to update node information

```
PATCH /api/rest/v1/node/<node_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "node_id" : "str",
    "location" : {
        "lat": "str",
        "lon": "str",
    },
    "name": "str",
    "type": "str", /// DeepStream or TensortRT
    "ip": "str",
    "node_metadata": {
        "user" : "str", /// Not required only for VMS
        "password" : "str", /// Not required only for VMS
        "end_point": "str" /// Not required only for VMS
    },
    "connection_metadata": {
        "kafka" : {
            "port": "str",
            "topic": "str"

        },
        "mqtt": {
            "port": "str",
            "topic": "str"
        },
        "socket": {
            "port": "str"
        }
    },
    "engine_metadata" : {
      "resolution" : {
        "width" : "int",
        "height" : "int"
      },
      "frame_rate" : "int", // Maximum value is 30
      "sensitive": {
        "detection" : "int", //Range value from 1 to 5
        "tracking" : "int" //Range value from 1 to 5
      },
      "frame_step" : "int"
    }
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE NODE

- Create DELETE request to delete node information

```
DELETE /api/rest/v1/node/<node_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

SYNC NODE INFORMATION (SUPPORT VMS ONLY)

- Create POST request to synchoize VMS information with other node

```
POST /api/rest/v1/node/<node_id>/sync
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "node_id" : "str" /// Select the node that user want to sync with
}

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

6. Device Management API

CREATE DEVICE

- Create POST request to create new device

```
POST /api/rest/v1/node/<node_id>/device
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "device_metadata" : {
        "manufacture" : "str",
        "describle" : "str"
    }, // Do not need to have,
    "location" : {
        "lat" : "str",
        "long" : "str"
    },
    "type" : "str", // Device Type : Camera, Radar , ....
    "camera" : {
      "driver" : "str", // camera driver (RTSP, Milestone, Onvif)
      "type" : "str", // camera type can be Static or PTZ
      "connection_metadata" : {
        "onvif" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "profile" : "str", // camera profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "rtsp" : {
          "rtsp_url" : "str",
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "milestone" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "authen_type" : "str",
          "profile" : "str", //camepra profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        }
      } // depend on driver type to declear different connection metadata format
    }
}

Response: {
  "data": "str", //device_id
  "message": "str",
  "success": "boolean"
}
```

VERIFY DEVICE CONNECTION

- Create POST request to verify new device

```
POST /api/rest/v1/node/<node_id>/device/status
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "ip" : "str",
  "port": "str"
}

Response: {
  "data": {"status":"str"}, //status of device can be ONLINE,OFFLINE,ERROR
  "message": "str",
  "success": "boolean"
}
```

GET DEVICES LIST

- Create GET request to get the list of devices

```
GET /api/rest/v1/node/<node_id>/device
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
    "device_metadata" : {
        "manufacture" : "str",
        "describle" : "str"
    }, // Do not need to have,
    "location" : {
        "lat" : "str",
        "long" : "str"
    },
    "type" : "str", // Device Type : Camera, Radar , ....
    "camera" : {
      "driver" : "str", // camera driver (RTSP, Milestone, Onvif)
      "type" : "str", // camera type can be Static or PTZ
      "connection_metadata" : {
        "onvif" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "profile" : "str", // camera profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "rtsp" : {
          "rtsp_url" : "str",
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "milestone" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "authen_type" : "str",
          "profile" : "str", //camepra profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        }
      } // depend on driver type to declear different connection metadata format
    },
    "status": "str" // ONLINE, OFFLINE, ERROR status can return
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET DEVICES LIST BY GROUP ID

- Create GET request to get the list of devices by group id

```
GET /api/rest/v1/group/<group_id>/device
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
    "device_metadata" : {
        "manufacture" : "str",
        "describle" : "str"
    }, // Do not need to have,
    "location" : {
        "lat" : "str",
        "long" : "str"
    },
    "type" : "str", // Device Type : Camera, Radar , ....
    "camera" : {
      "driver" : "str", // camera driver (RTSP, Milestone, Onvif)
      "type" : "str", // camera type can be Static or PTZ
      "connection_metadata" : {
        "onvif" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "profile" : "str", // camera profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "rtsp" : {
          "rtsp_url" : "str",
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "milestone" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "authen_type" : "str",
          "profile" : "str", //camepra profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        }
      } // depend on driver type to declear different connection metadata format
    },
    "status": "str" // ONLINE, OFFLINE, ERROR status can return
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET DEVICE

- Create GET request to get current device information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
    "device_metadata" : {
        "manufacture" : "str",
        "describle" : "str"
    }, // Do not need to have,
    "location" : {
        "lat" : "str",
        "long" : "str"
    },
    "type" : "str", // Device Type : Camera, Radar , ....
    "camera" : {
      "driver" : "str", // camera driver (RTSP, Milestone, Onvif)
      "type" : "str", // camera type can be Static or PTZ
      "connection_metadata" : {
        "onvif" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "profile" : "str", // camera profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "rtsp" : {
          "rtsp_url" : "str",
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "milestone" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "authen_type" : "str",
          "profile" : "str", //camepra profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        }
      }, // depend on driver type to declear different connection metadata format
      "status": "str" // ONLINE, OFFLINE, ERROR status can return
    }
  },
    "message" : "str",
    "success" : "boolean"
}
```

GET DEVICE STATUS

- Create GET request to get current device status information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/status
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "status": "str" // ONLINE, OFFLINE, ERROR status can return
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT DEVICE

- Create PATCH request to edit the current device

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "device_metadata" : {
        "manufacture" : "str",
        "describle" : "str"
    }, // Do not need to have,
    "location" : {
        "lat" : "str",
        "long" : "str"
    },
    "type" : "str", // Device Type : Camera, Radar , ....
    "camera" : {
      "driver" : "str", // camera driver (RTSP, Milestone, Onvif)
      "type" : "str", // camera type can be Static or PTZ
      "connection_metadata" : {
        "onvif" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "profile" : "str", // camera profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "rtsp" : {
          "rtsp_url" : "str",
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        },
        "milestone" : {
          "ip" : "str",
          "http_port" : "str",
          "rtsp_port" : "str",
          "authen_type" : "str",
          "profile" : "str", //camepra profile can be 0,1,2
          "user" : "str", // In case camera dont have security step let it empty
          "password": "str" // In case camera dont have security step let it empty
        }
      } // depend on driver type to declear different connection metadata format
    }
}

Response: {
  "data" : {
    "status": "str" // ONLINE, OFFLINE, ERROR status can return
  },
  "message": "str",
  "success": "boolean"
}
```

DELETE DEVICE

- Create DELETE request to delete device information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>
```

```
 Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

GET SNAPSHOT BY DEVICE ID

- Create GET request to get snapshot from camera

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/snapshot
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
    "size":"array", // [width,height] resolution of camera
    "format": "jpg",
    "img" : "str" // base64 image format
  },
  "message": "str",
  "success": "boolean"
}
```

---

7. Patrol Management API

CREATE PATROL

- Create POST request to create new patrol

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/patrol
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "name" : "str" // name of patrol
}

Response: {
  "data": "str", //camera_id
  "message": "str",
  "success": "boolean"
}
```

GET PATROL LIST

- Create GET request to get the list of patrols

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
  "name" : "str"
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET PATROL

- Create GET request to get current patrols information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
  "name": "str"
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT PATROL

- Create PATCH request to edit the current patrol

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "name" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE PATROL

- Create DELETE request to delete patrol information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

CREATE PATROL MANAGEMENT

- Create POST request to linkage between preset and patrol together

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_management
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "preset_ids" : "list" // list of preset ids
}
Response:{
  "data": {
  "id" : "str"
  },
    "message" : "str",
    "success" : "boolean"
}
```

GET PATROL MANAGEMENT LIST

- Create GET request to get the list of patrol managements

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_management
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
  "patrol_id" : "str",
  "preset_id" : "str"
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET PATROL MANAGEMENT

- Create GET request to get current patrol management information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_management/<patrol_management_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
  "patrol_id" : "str",
  "preset_id" : "str"
  },
    "message" : "str",
    "success" : "boolean"
}
```

DELETE PATROL MANAGEMENT

- Create DELETE request to delete patrol management information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_management/<patrol_management_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

CREATE PATROL SCHEDULE

- Create POST request to create a schedule for patrol

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_schedule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "touring_id" : "str",
  "color": "str", //should store in hexa or any color format value
  "schedule": {
    "start_time": "str", // time format hh:mm
    "end_time": "str", // time format hh:mm
    "day": "str" // the integer value preset for day start Monday = 0 and Sunday = 6
  }
}
Response:{
  "data": {
  "id" : "str"
  },
    "message" : "str",
    "success" : "boolean"
}
```

GET PATROL SCHEDULE LIST

- Create GET request to get the list of patrol schedules

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_schedule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
    "touring_id" : "str",
  "color": "str", //should store in hexa or any color format value
  "schedule": {
    "start_time": "str", // time format hh:mm
    "end_time": "str", // time format hh:mm
    "day": "str" // the integer value preset for day start Monday = 0 and Sunday = 6
  }
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET PATROL SCHEDULE

- Create GET request to get current patrol schedule information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_schedule/<patrol_schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
    "touring_id" : "str",
  "color": "str", //should store in hexa or any color format value
  "schedule": {
    "start_time": "str", // time format hh:mm
    "end_time": "str", // time format hh:mm
    "day": "str" // the integer value preset for day start Monday = 0 and Sunday = 6
  }
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT PATROL SCHEDULE

- Create PATCH request to edit the current patrol

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_schedule/<patrol_schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "touring_id" : "str",
  "color": "str", //should store in hexa or any color format value
  "schedule": {
    "start_time": "str", // time format hh:mm
    "end_time": "str", // time format hh:mm
    "day": "str" // the integer value preset for day start Monday = 0 and Sunday = 6
  }
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE PATROL SCHEDULE

- Create DELETE request to delete patrol management information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/patrol/<patrol_id>/patrol_schedule/<patrol_schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

8. Preset Management API

SYNCHRONIZE PRESET

- Create GET request to get all preset information from camera

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/preset/sync
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": [{
    "id" : "str", // preset id
    "name" : "str" // preset name
  }],
  "message": "str",
  "success": "boolean"
}
```

GET PRESET LIST

- Create GET request to get the list of preset

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/preset
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
  "token" : "int",
  "pos" : "json",
  "name" : "str"
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET PRESET

- Create GET request to get current preset information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/preset<preset_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
  "token" : "int",
  "pos" : "json",
  "name" : "str"
  ,
    "message" : "str",
    "success" : "boolean"
}
```

EDIT PRESET

- Create PATCH request to edit the current touring

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/preset<preset_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE PRESET

- Create DELETE request to delete all linking information to preset

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/preset<preset_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

PTZ TOUR

- Create GET request to move ptz at direct preset

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/preset/<preset_id>/control
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message" : "str",
  "success" : "boolean"
}
```

---

9. Tour Management API

CREATE TOURING

- Create POST request to create new touring

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/touring
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "active" : "boolean",
  "patrol_setting" : [{
    "patrol_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  }],
  "preset_setting" : [{
    "preset_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  }]
}

Response: {
  "data": {
    "id" : "str"
  },
  "message": "str",
  "success": "boolean"
}
```

GET TOURING LIST

- Create GET request to get the list of touring

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/touring
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [{
  "id" : "str",
  "device_id" : "str",
  "active" : "boolean",
  "preset_schedule" : {
    "preset_id" : "int",
    "color": "str",
    "schedule" : [{
      "days" : "str",
      "start_time": "str",
      "end_time": "str"
    }]
  },
  "patrol_schedule" : {
    "patrol_id" : "int",
    "color": "str",
    "schedule" : [{
      "days" : "str",
      "start_time": "str",
      "end_time": "str"
    }]
  }
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET TOURING

- Create GET request to get current touring information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {
  "id" : "str",
  "device_id" : "str",
  "active" : "boolean",
  "preset_schedule" : {
    "preset_id" : "int",
    "color": "str",
    "schedule" : [{
      "days" : "str",
      "start_time": "str",
      "end_time": "str"
    }]
  },
  "patrol_schedule" : {
    "patrol_id" : "int",
    "color": "str",
    "schedule" : [{
      "days" : "str",
      "start_time": "str",
      "end_time": "str"
    }]
  }
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT TOURING

- Create PATCH request to edit the current touring

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "active" : "boolean",
  "patrol_setting" : [{
    "patrol_schedule_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  }],
  "preset_setting" : [{
    "preset_schedule_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  }]
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE TOURING

- Create DELETE request to delete all linking information to touring

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

CREATE OR UPDATE TOURING SCHEDULE

- Create POST request to create/update touring schedule

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>/schedule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "patrol_setting" : {
    "patrol_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  }, // if the schedule is patrol schedule
  "preset_setting" : {
    "preset_id" : "str",
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }]
  } // If the schedule is preset schedule
}

Response: {
  "data": {
    "id" : "str", // the scheudle id
    "schedule_tpye" : "str" // the type of schedule can be preset/patrol
  },
  "message": "str",
  "success": "boolean"
}
```

UPDATE TOURING SCHEDULE BY ID

- Create PATCH request to update touring schedule

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>/schedule/<schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "schedule_type" : "str", // the type of schedule can be preset/patrol
    "color" : "str",
    "schedule" : [{
      "start_time" : "str",
      "end_time" : "str",
      "day" : "str"
    }],
}

Response: {
  "data": {
    "id" : "str", // the scheudle id
    "schedule_tpye" : "str" // the type of schedule can be preset/patrol
  },
  "message": "str",
  "success": "boolean"
}
```

DELETE TOURING SCHEDULE BY ID

- Create DELETE request to delete touring schedule

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/touring/<touring_id>/schedule/<schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
    "schedule_type" : "str", // the type of schedule can be preset/patrol
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

---

10. Rule Management API

CREATE RULE

- Create POST request to create new rule

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/rule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
  "combine_name" : "str",
  "active" : "boolean",
  "post_action" : "json", // be defined later can ignore for current version
  "alarm_type" : "str",
  "points" : "list", // the ROI area contain list of point [[x,y],..,[x,y]]
  "level" : "str",
  "preset_id": "str",
  "schedule_id" : "str",
  "alarm_metadata" : {
    "loitering" : {"time_stand" : "str"},
    "sabotage" : "json", // be defined later can ignore for current version
    "tripwire" : {"direction": "str"}, // direction can be : left to right/right to left
  },
  "objects" : "list" // list of object id storage in list [0,1,2]
}

Response: {
  "data": "str", //rule_id
  "message": "str",
  "success": "boolean"
}
```

- Object mapping information

```
0: bike
1: car
2: bus
3: truck
4: ambulance
5: firetruck
6: people
```

GET RULE LIST

- Create GET request to get the list of rules

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/rule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {[
    {
  "id" : "str",
  "name" : "str",
  "combine_name" : "str",
  "active" : "boolean",
  "post_action" : "json", // be defined later can ignore for current version
  "alarm_type" : "str",
  "points" : "list", // the ROI area contain list of point [[x,y],..,[x,y]]
  "level" : "str",
  "alarm_metadata" : {
    "loitering" : {"time_stand" : "str"},
    "sabotage" : "json", // be defined later can ignore for current version
    "tripwire" : {"direction": "str"}, // direction can be : left to right/right to left
  },
  "preset_id" : "str",
  "schedule_id" : "str",
  "objects" : "list" // list of object id storage in list [0,1,2]
  }]
  },
    "message" : "str",
    "success" : "boolean"
}
```

GET RULE

- Create GET request to get current rule information

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/rule/<rule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
  "id" : "str",
  "name" : "str",
  "combine_name" : "str",
  "active" : "boolean",
  "post_action" : "json", // be defined later can ignore for current version
  "alarm_type" : "str",
  "points" : "list", // the ROI area contain list of point [[x,y],..,[x,y]]
  "level" : "str",
  "alarm_metadata" : {
    "loitering" : {"time_stand" : "str"},
    "sabotage" : "json", // be defined later can ignore for current version
    "tripwire" : {"direction": "str"}, // direction can be : left to right/right to left
  },
  "preset_id" : "str",
  "schedule_id" : "str",
  "objects" : "list" // list of object id storage in list [0,1,2]
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT RULE

- Create PATCH request to edit the current rule

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/rule/<rule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
  "combine_name" : "str",
  "active" : "boolean",
  "post_action" : "json", // be defined later can ignore for current version
  "alarm_type" : "str",
  "points" : "list", // the ROI area contain list of point [[x,y],..,[x,y]]
  "level" : "str",
  "preset_id": "str",
  "schedule_id" : "str",
  "alarm_metadata" : {
    "loitering" : {"time_stand" : "str"},
    "sabotage" : "json", // be defined later can ignore for current version
    "tripwire" : {"direction": "str"}, // direction can be : left to right/right to left
  },
  "objects" : "list" // list of object id storage in list [0,1,2]
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE RULE

- Create DELETE request to delete rule information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/rule/<rule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

CREATE SCHEDULE

- Create POST request to create schedule rule

```
POST /api/rest/v1/node/<node_id>/device/<device_id>/schedule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
  "time_info": [{
    "days" : "str",
    "start_time" : "str",
    "end_time" : "str"
  }]
}

Response: {
  "data": "str", //rule_id
  "message": "str",
  "success": "boolean"
}
```

GET SCHEDULE LIST

- Create GET request to get the list of schedules

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/schedule
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": [
  {
  "id" : "str",
  "name" : "str",
  "time_info" : [{
    "days" : "str",
    "start_time" : "str",
    "end_time" : "str"
    }]
  }],
    "message" : "str",
    "success" : "boolean"
}
```

GET SCHEDULE

- Create GET request to get schedule by id

```
GET /api/rest/v1/node/<node_id>/device/<device_id>/schedule/<schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
  "id" : "str",
  "name" : "str",
  "time_info" : [{
    "days" : "str",
    "start_time" : "str",
    "end_time" : "str"
  }]
  },
    "message" : "str",
    "success" : "boolean"
}
```

EDIT SCHEDULE

- Create PATCH request to edit the current schedule

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/schedule/<schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "name" : "str",
  "time_info" : [{
    "days" : "str",
    "start_time" : "str",
    "end_time" : "str"
  }]
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE SCHEDULE

- Create DELETE request to delete schedule information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/schedule/<schedule_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "data": {},
  "message" : "str",
  "success" : "boolean"
}
```

---

11. Event Management API

UPDATE EVENT

- Create PATCH request to update event

```
PATCH /api/rest/v1/node/<node_id>/device/<device_id>/event/<event_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: {
  "rule_id" : "str",
  "type": "str",
  "bounding_box": {
    "topleftx" : "int",
    "toplefty" : "int",
    "bottomrightx" : "int",
    "bottomrighty" : "int"
  },
  "event_time" : "str",
  "is_watch" : "boolean",
  "is_verify" : "boolean"
}

Response: {
  "message": "str",
  "success": "boolean"
}
```

DELETE EVENT

- Create DELETE request to delete event information

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/event/<event_id>
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message" : "str",
  "success" : "boolean"
}
```

GET IMAGE EVENT

- Create GET request to get event image

```
DELETE /api/rest/v1/node/<node_id>/device/<device_id>/event/<event_id>/image?type=<str>
```

type : full/crop

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: Image data
```

---

12. Search API

SEARCH EVENT BY FILTER

- Create GET request to get all events information by different filter

```
GET /api/rest/v1/node/<node_id>/device/events?dis=<list:str>&oit=<list:str>&tq=<str>&level=<list:int>&eit=<list:int>&limit=<int>[&st
```

```
Example: http://localhost:5500/api/rest/v1/node/<node_id>/device/events?limit=100&eit=1,2,3
```

```
Request Header
Authorization: Bearer
Type: application/json
```

- Parameters

Parameter	Type/Value	Description	Notes
dis	List	Device ID list	The list of device can be define by comma Example : 1,2,3,4. In case we want to query all dont add this option in paramater
ot	List	Object type list. The object type can be defined follow information: 0 - bike, 1 - car, 2 - bus, 3 - truck, 4 - ambulance, 5 - firetruck, 6 - people	The list of object can be define by comma Example : 0,1,2,3,4. In case we want to query all dont add this option in paramater
tq	String	Time query option: day,5min,10min,30min,hour,week,month,year,custom	Default = 'None'
eit	List	Event type list. The event type can be defined follow information: 0 - tresspasing, 1 - loitering, 2 - tripwire, 3 - sabotage	The list of event type can be define by comma Example : 0,1,2,3. In case we want to query all dont add this option in paramater
p	Integer	the index page want to query	Default = 1
pl	Integer	the length of the page to query	Default = 100
start	String	start time to query	Time format: %Y-%m-%d. Only change value if tq == 'custom'
end	String	end time to query	Time format: %Y-%m-%d. Only change value if tq =='custom'

```
Request: None

Response:{
  "data": {
    "events" : [{
          "device_name": "str",
          "device_location": "str",
          "device_id" : "str",
          "rule_id" : "str",
          "id" : "str", // This is tracking id
          "tracking_number" : "str",
          "start_time": "str",
          "end_time" : "str",
          "alarm_type": "str",
          "alarm_level" : "str",
          "image_infos" : {
            "recognize_result": {
              "lisence_plate" : "str",
              "color" : "str",
              "direction": "str"
            }, /// Only contain value if object is vehicle
            "event_type" : "str",
            "event_id" : "str",
            "event_time": "str",
            "bounding_box" : {
              "topleftx" : "int",
              "toplefty" : "int",
              "bottomrightx" : "int",
              "bottonrighty" : "int"
            }
          },
          "created_at": "str",
          "updated_at": "str",
          "deleted_at": "str"
      }],
    "total" : "int", // Total of the events
    "total_pages" : "int", // Total of pages based on the define condition

  },
  "message": "str",
  "success": "boolean"
}
```

---

10. Health Check Monitor API

GET DEVICE INFORMATION

- Create GET request to get the current information of the AI server

```
GET /api/rest/v1/health
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response:{
  "data": {
      "processor" : {
          "physicals" : "int",
          "logicals" : "int",
          "usage" : "str", //percent value per cpu
          "usages" : "str" //whole cpu
      },
      "memory" : {
          "total" : "int",
          "available" : "int",
          "used" : "int",
          "free" : "int",
          "percent" : "str"
      },
      "storage" : {
        "total" : "int",
        "available" : "int",
        "used" : "int",
        "free" : "int",
        "percent" : "str"
      }
   },
    "message": "str"
  },
  "message": "str",
  "success": "boolean"
}
```

---

11. Edge Controller API

Upload Edge configuration

- Create POST request to upload configuration

```
POST /api/rest/v1/node/<node_id>/uploadConfig
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message": "str",
  "success": "boolean"
}
```

Start Edge System

- Create GET request to start edge system

```
GET /api/rest/v1/node/<node_id>/start
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message": "str",
  "success": "boolean"
}
```

`Note: The service return error if the is_activate already TRUE`

Stop Edge System

- Create GET request to stop edge system

```
GET /api/rest/v1/node/<node_id>/stop
```

```
Request Header
Authorization: Bearer
Type: application/json
```

```
Request: None

Response: {
  "message": "str",
  "success": "boolean"
}
```

`Note: The service return error if the is_activate already FALSE`

12. Milestone Intergrate

Camera Control Command

- Create POST request to control camera from server side

```
POST /api/rest/v1/milestone
```

```
Request Header
Authorization: Bearer
Type: application/json
```

- Parameter

Parameter	Type/Value	Description	Notes
ms_uid	String	Milestone UID of camera	The database inside the milestone
command	String	The command request	The command type allow : ENABLE CAMERA, DISABLE CAMERA

```
Request: {
  "ms_uid" : "str", /// The milestone UID from milestone
  "command" : "str", /// The command that milestone want to process on camera
}

Response: {
  "message": "str",
  "success": "boolean"
}
```