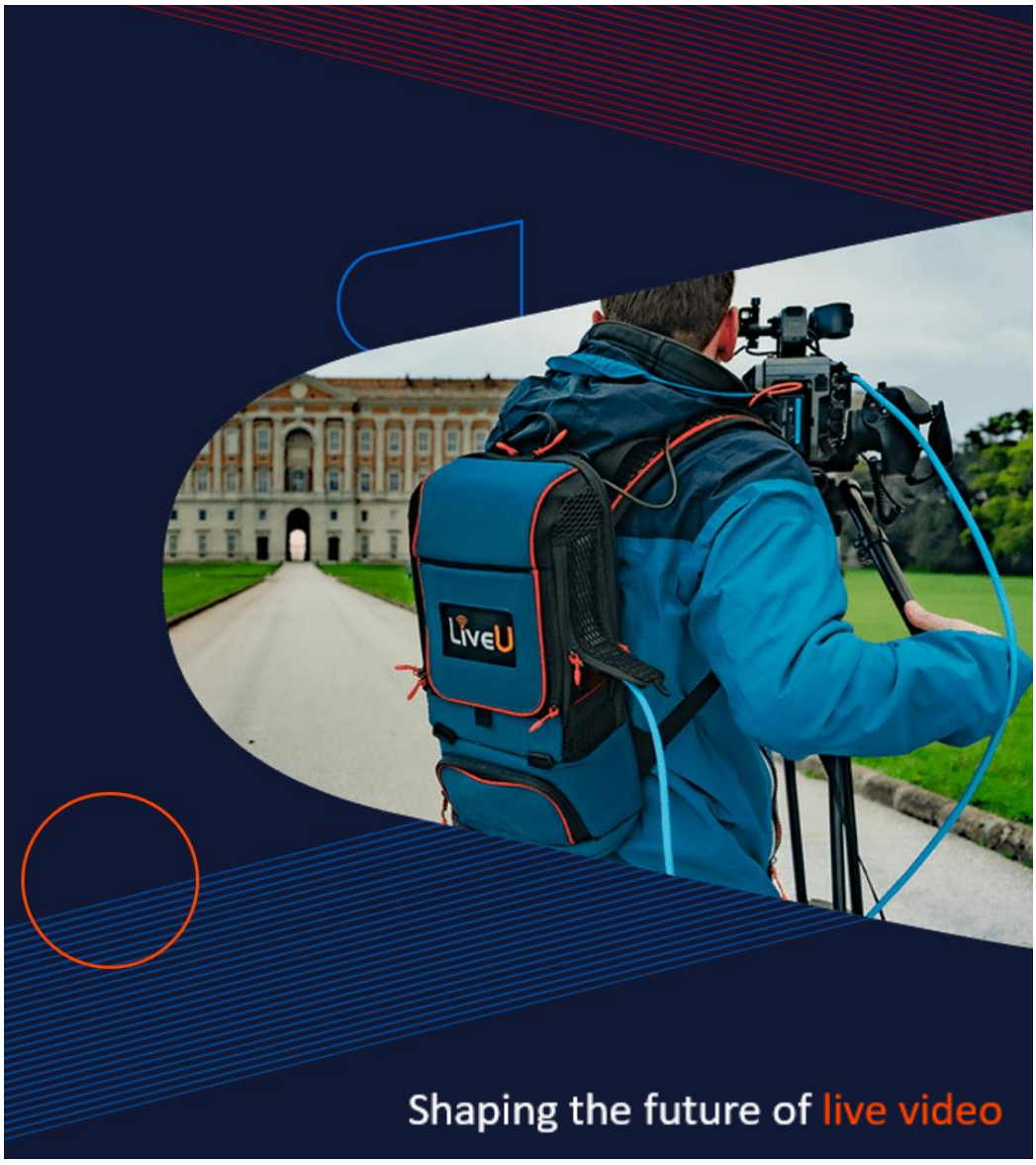
# Central Backend API Guide

## Version 10.6

LiveU Proprietary and Confidential



# Revisions

| Rev | Date | Author | Version title | Details/Comments |
|---|---|---|---|---|
| 1.0 | Jan-22-2020 | Anat | 8.2 | |
| 1.1 | Apr-05-2020 | Anat | 8.3 | 3.2 – bearer token duration is fixed<br>Base URL includes "/v2" |
| 1.2 | May-18-2021 | Anat | 9.0 | Update section 1.3 - Terms for using Central API |
| 1.3 | July-13-2021 | Anat | 9.0 | Buttery runTimeToEmpty is in minutes (not seconds) |
| 1.4 | Aug-30-2022 | Anat | 9.5 | Add newsroom integration APIs |
| 1.5 | July-18-2023 | Anat | 10.0 | Adding the X-API-Key header |
| 1.6 | Sep-11-2023 | Anat | 10.0 | Update the story integration license |
| 1.7 | Nov-02-2023 | Anat | 10.0 | Adding the X-API-Key header to the authentication APIs example |
| 1.8 | Mar-25-2024 | Anat | 10.3 | • New API for story association to unit(s)<br>• New API for story disassociation from unit(s)<br>• New API to get list of stories by inventory by unit |
| 1.9 | Jun-25-2024 | Anat | 10.5 | • Mulicamera API section<br>• List of units (per inventory, per group) |
| 1.6 | | Anat | 10.6 | • GET list of units of a story<br>• Integration Guidelines |

# Table of Contents

# 1 Document Scope

This document describes the LiveU Central Backend REST API related to release 8.5.
It is assumed that the user reading this document is familiar with basic LUC entities like inventory, unit, channel, server, group and users.

## 1.1 Restful API

The API is a RESTful API: objects are retrieved with GET, created with POST, updated with PUT, and deleted with DELETE.

## 1.2 Overview

LiveU Central (LUC) Backend controls the LiveU Field Units, Servers, Inventories, Users and more. The purpose of the API is to enable 3rd party applications to interact (manage and control) their entities.
Inventory – usually represents a customer. Inventory is the container of customer's entities.
The entities are one of:

1. Groups
2. Units: hardware units or software units (like LU-smart)
3. Mmh servers (LU2000/LU4000) and their channels (instances on server)
4. Video-return servers and their channels (instances on server)
5. Users

## 1.3 Terms for using Central API

1. Communication with Central APIs should be done via a backend application and one source communicating with Central APIs
2. Application should call the API sequentially: application should not call the next API before receiving response to the current call.
3. The application should not call LiveU Central more than 10 APIs calls per second. This is the total number of API calls for all the monitored devices
4. API calls for dynamic information should be done via subscribe/notify mechanism (over WebSocket)

Note:

- LiveU keeps the right to block API user who is overloading the system.
- LiveU keeps the right to change the overload threshold.

## 1.4 Integration Guidelines

LiveU requires that any integrating is performed with customer's backend application.

Integration from customer's client to LiveU Central (customer's client communicating directly with LiveU Central) is not allowed.

## 1.5 Authentication(oauth2)

API user needs to authenticate.
All APIs (except grant token request) must include Bearer token in request Authorization header.

a. To grant a Bearer token application should:
   i. Grant sso token using user credentials
   ii. Grant Bearer token using sso token
   iii. When bearer token expires, a new bearer token should be granted with the existing sso token, as long as the sso token is still valid.
b. About SSO token
   i. Used for generating a bearer token
   ii. Expire after predefined time of inactivity.
   iii. Used for refreshing the Bearer token.
c. About Bearer token
   i. Should be attached to all APIs (except the grant token API)
   ii. Expire after predefined time (agnostic to activity).
   iii. When Bearer token expires, app should refresh the token.
   iv. If sso token is not valid anymore, application should navigate user to login page.
d. For more details see Authentication APIs (3)

## 1.6 Base URL

The APIs uses the following base URL ((referred as BaseURL in API examples).
The base URL is: https://lu-central-api.liveu.tv/rest-v2/v2

## 1.7 Authorization

APIs only expose data / controls entities that this user is authorized to view/edit/control, according to the user's type, and user permissions.
Examples:

1. A user TV-studio-user who is associated to inventory TV-studio-inventory, can access only entities associated to TV-Studio inventory.
2. If type of user of TV-studio-user is read-only, this user will not be able to update/create/delete entities.
3. If type of user of TV-studio-user is regular, this user will be able to update some unit related information but will not be able to update group related information or create/delete entities.

## 1.8 Application-Id

A mandatory header should be sent with all APIs: "**Application-Id**". Any application communicating with LiveU Central should receive application-id from LiveU CS.
The received application-id should be sent in the Application-Id header for all APIs.
**An API call which does not include the Application-Id header will be rejected**.

## 1.9 Customer API Keys

A mandatory header should be sent with all APIs: " **X-API-Key**". Any application communicating with LiveU Central should receive an API key from LiveU CS.
The received API key should be sent in the X-API-Key header for all APIs.
**An API call which does not include the X-API-Key header will be rejected**.

## 1.10 Terms and Acronyms

Acronyms or short-name are in brackets.

1. LiveU Central (or LUC) – LiveU's web-based GUI for device management and control.

# 2 General Functionality

API calls regarding a single entity
APIs working on single entity will not provide information related to associated entities.
To receive information on associated entities, a different request should be sent (request per entity type). For example:

a. To retrieve all objects of a specific type, call GET /{object-type}
   e.g. GET BaseURL/inventories/{inventoryId}
b. The BaseURL/inventories/{inventoryId} will provide the direct data for this inventory
c. If API user needs information regarding entity associated to this entity (e.g. units associated to a specific inventory it should issue a new query for BaseURL/inventories/{inventoryId}/units
d. Access keys to entities

| Index | Entity | Default key {id} | Additional keys for main query |
|---|---|---|---|
| 1. | V2/units/{id} | uniqueId (bossId) | /v2/units/sn/{sn} |
| 2. | V2/inventory/{dbId} | dbId | /v2/inventories/erpId/{erpId}<br>/v2/inventories/name/{name} |
| 3. | V2/channels/{id} | uniqueId (bossId) | /v2/channels/name/{name} |
| 4. | V2/groups/{dbId } | id (db) | /v2/groups/name/{name} |
| 5. | V2/users/{username} | username | |

## 2.1 API calls regarding list of entities

a. APIs working on lists will not provide information related to associated entities.
   To receive information on associated entities, a different request should be sent (request per entity type). For example:
   1. The /inventories will provide direct data for all inventories.
   2. If API call is interested with associated units of the customer's inventories the following API should called (once per each associated inventory): /inventories/{inventoryId}/units
b. Pagination - all APIs returns significant lists should support pagination.
   offset & limit parameters will be appended to the path.
c. The default offset is 0, the default limit is 100

## 2.2 Response structure

Response - Standard HTTP Status Codes are used (both for success or failure).

```
Success Example (supported success codes are 2xx):
HTTP/1.1 200 OK
Content-Type: application/json
{
  ...,
  "value1": [...],
  "value2": [...],
  }
```

```
Failure example (supported failure codes are 4xx, 5xx):
HTTP/1.1 400 Bad Request
Content-Type: application/json
 {
   "error": {
     "code": "1001",
     "message": "Unit is not idle!"
   }
 }
```

## 2.3 Headers

### 2.3.1 Request Headers

Application should call Central APIs (except the grant token API) with the following request Headers:

1. Authorization: Bearer *token received from grant token*
2. Application-Id*: application-id key is received from LiveU CS*
3. Content-Type: application/json
4. X-API-Key: key value provided by LiveU

### 2.3.2 Response header

When response includes body, it always includes response Header.
Header name: Content-Type, value: application/json

## 2.4 Parameters

Each entity provides only its direct parameters.

### 2.4.1 Request

1. The Target entity of the API appears in the path of the request
2. The API will not support query for specific parameters
3. Fields with **null** content should not appear in the request.

### 2.4.2 Response

1. Response to creation of an entity (POST) should list the created entity fields
2. Response will not include fields with **null** content

## 2.5 Partial Updates

When editing (updating) objects, not all fields are required; non-included fields leave the existing field values as is. For example:

Existing object: {"a"=1, "b"=2}
Update json data: {"a"=11}
Final updated object: {"a"=11, "b"=2}

## 2.6 General HTTP Errors

| | | | |
|---|---|---|---|
| 400 | 400 Bad Request | 405 | Method Not Allowed |
| 401 | Unauthorized user | 415 | Unsupported Media Type |
| 403 | Forbidden, Un-authorized user | | |
| 404 | Entity was not found | | |

# 3 Authentication APIs

## 3.1 Obtain an OAuth2 sso token

Request:
https://lu-central-api.liveu.tv/auth/login/token_grant
Method: POST
Request Headers:
Authorization: Basic dXNlcjohIXBhc3N3b3JkISE=
X-API-Key: {key value}
Request Payload: {"token_type": "sso"}
Note: The authorization header includes username:password encoded in base64
Response:
Content-Type: application/json
Body:

```
{
   "access_token": "f8af84cc9af64b9fbfc85c2c38e9e23c",
   "expires_in": 2700,
   "token_type": "sso"
}
```

Note: expires_in value is in seconds.
In the example above, sso token is valid for 45 min of inactivity.

## 3.2 Obtain an OAuth2 bearer token

Request:
https://lu-central-api.liveu.tv/auth/login/token_grant
Method: POST
Request Headers:
Authorization:luc-sso f8af84cc9af64b9fbfc85c2c38e9e23c
X-API-Key: {key value}
Request Payload : { "token_type" : "bearer" }
Response:
Content-Type: application/json
Body:

```
{
    "access_token": ".....",
    "expires_in": 900,
    "token_type": "bearer"
}
```

Note: expires_in value is in seconds.
In the example above, bearer token is valid for 15 min.

# 4 Units related APIs

## 4.1 Get list of units in inventory

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/inventories/{dbId}/units?offset=0&limit=1000"
Note: Inventory must be associated to API user

Response-body - get list of units in inventory:
The response is a list of units in a specific inventory

```
[{
    "dbId": int,
    "uid": "string",
    "serialNumber": "string",
    "swVersion": "string",
    "product": "string",
    "bossId": "string",
    "alias": "string",
    "name": "string"
}]
```

Response parameters - get list of units in inventory:

| Parameter | description | Examples |
|---|---|---|
| dbId | Database entry if unit | 48841 |
| uid | Unique id of unit | "Boss100_lu600_ 323531393431643465366664346237 64" |
| serialNumber | Serial number of unit | "501616-52657" |
| swVersion | Software version of unit | "8.1.0.C17453.G82bf8d523" |
| product | Unit type | "LU-600", "LU-Smart", "LU-800" |
| appVersion | Application version of LU-Smart and LU-Lite | "6.5.41" |
| alias | Alias of unit (can be changed by customer) | |
| name | Name of unit (set by LiveU) | |

## 4.2 Get unit status

User permissions: read-only, regular, admin, media-group admin
Method: GET BaseURL/units/{*uniqueId*}/status"

Response-body - get unit status:

```
{
  "unitUniqueId": "string",
  "upTime": 0,
  "availability": "string",
  "applications": {
    "video": {
      "state": "string",
      "channel": "string",
      "usedLinks": [
        "string"
      ]
```

```
    },
    "databridge": {
      "state": "string",
      "channel": "string",
      "usedLinks": [
        "string"
      ]
    },
    "upgrade": {},
    "fetchlogs": {},
    "videoReturn": {
      "state": "string",
      "channel": "string",
      "usedLinks": [
        "string"
      ]
    },
    "store": {
      "fileName": "string",
      "fileSize": 0
    },
    "forward": {
      "uploadSpeedKbps": 0,
      "fileName": "string",
      "fileSize": 0,
      "etaSeconds": 0,
      "bytesSent": 0,
      "channel": "string",
      "usedLinks": [
        "string"
      ]
    }
  },
  "videoMode": "string",
  "battery": {
    "connected": true,
    "percentage": 0,
    "runTimeToEmpty": 0,
    "discharging": true,
    "charging": true
  },
  "onAir": true,
  "tallyLight": "connected",
  "xtenders": [
    {
      "uniqueId": "string",
      "unitLink": "string",
      "usedLinks": [
        "string"
      ]
    }
  ],
  "totalDownlinkKbps": 0,
  "totalUplinkKbps": 0
}
```


Response parameters - get unit status:

| Parameter | description | Examples |
|---|---|---|
| unitUniqueId | The unique id of the unit | "Boss100_lu600_ 32353139343164346536 666434623764" |
| upTime | The time (in seconds) the unit is up (online) | |
| availability | indicates unit's availability | "online", "offline" |

| | | |
|---|---|---|
| Applications | Activities the unit may be occupied with.<br>only active applications are reported. Unit may be occupied with multiple applications (like video and video return), with single application (like upgrade) or without any application (while the unit is idle).<br>Each application provides its own information | see<br><br>List of the unit's applications) |
| videoMode | video resolution, interlace/progressive, fps | "1080i50" |
| battery - connected | Indicates if battery is connected | |
| battery - percentage | percentage of battery charged | |
| battery - runTimeToEmpty | time in minutes till battery becomes empty.<br>The value is relevant only if discharging is true. | |
| battery - discharging | • Discharging (true): unit is not connected to power supply and currently discharging the buttery.<br>• Discharging (false): unit is connected to power supply and not using the battery power. | |
| battery - charging | • Charging (true): indicated when the unit is currently charging (meaning it is connected to power supply).<br>• Charging (false): indicated when the unit is not charging (either charging was stopped since battery fully charged, or device is not connected to power supply). | |
| onAir | indicates if unit stream is currently on-air | |
| tallyLight | indicates whether tally device is connected to the unit | |
| xtenders | indicates whether extender is connected to the unit (extender's uniqueId & unit's eth) and list of the extender interfaces | |
| totalDownlinkKbps | downlink bandwidth used by all interfaces | |
| totalUplinkKbps | uplink bandwidth used by all interfaces | |

### 4.2.1 List of the unit's applications

The applications currently supported by units are:

| application | description | States |
|---|---|---|
| video | Active when the unit is streaming live | connecting - initiation of streaming<br>streaming - while streaming<br>stopping - termination of streaming |
| databridge | Active when the unit is in databridge mode | connecting - initiation of streaming<br>bridging - while sending data<br>stopping - termination of streaming |
| upgrade | Active when unit is in process of downloading a new version | State is not applicable |
| fetchlogs | Active when unit is aggregating its logs and sending it to remote storage (S3 bucket) | State is not applicable |
| videoReturn | Active when unit is receiving video from control room | connecting - initiation of video return<br>collecting - while receiving video return<br>stopping - termination of video return |
| Store | Active while unit is storing video locally, either store&forward or Live&Store | State is not applicable |
| forward | Active while unit in store&forward or only forward (upload of file to server) | connecting - initiation of forwarding<br>forwarding – forwarding file or store&forward stream |
| ifb | One direction ifb over the stream connection | "state": "inactive" or "active" |
| audioCallBySelectedChannel | Two-way IFB. Active when unit is streaming | connecting, connected |
| audioRoom | Active when unit is connected to audio room. | connecting, connected |
| l2ControlPipe | Active when unit activates Ip Pipe | connecting or bridging or stopping |
| multiCamera | Active when multi-camera HQ is active | connecting, streaming |
| switcherAndProxy | Active when station switcher is active | connecting, streaming |
| switcher | Active when field switcher is active | connecting, streaming |

## 4.3 Get unit status/full (including interfaces)

User permissions: read-only, regular, admin, media-group admin
Method: `GET BaseURL/units/{uniqueId}/status/full"`

Response-body - get unit's status/full:
The below dictionary is sent per interface on top of the response to API: GET BaseURL/units/{uniqueId}/status

```
"links": [
    {
      "id": "string",
      "name": "string",
      "port": "string",
      "technology": "string",
      "activeSim": "string",
      "enabled": true,
      "connected": true,
      "downSignalQuality": 0,
      "upSignalQuality": 0,
      "downlinkKbps": 0,
      "uplinkKbps": 0,
      "currentlyRoaming": true
    }
  ]
```

Response parameters - get unit's status/full:
In status/full unit report per each of the interfaces the following data:

| Parameter | description | Examples |
|---|---|---|
| id | Unique id of the interface | ETH: "eth0", "eth0"<br>Wifi: "wlan0"<br>Modem: "0"…"n"<br>Extender: "x1-0", "x1-eth0", |
| name | Operator name | "Pelephone" |
| port | The port of the interface.<br>Extender ports may be redundant to unit ports | ETH: "eth0", "eth0"<br>Wifi: "wlan0"<br>Modem: "0"…"n" |
| technology | indicates the cellular technology | 4G |
| activeSim | some modems support 2 SIMS.<br>This parameter indicates if the active SIM is SIM-A or SIM-B.<br>When this is a single SIM this parameter indicates empty string "". | "A", "B", "" |
| enabled | indicates whether the interface is enabled | true, false |
| connected | indicates whether the interface is connected | true, false |
| downSignalQuality | a number between 1-5 indicating the downlink quality | 3 |

| | | |
|---|---|---|
| upSignalQuality | a number between 1-5 indicating the downlink quality | 5 |
| downlinkKbps | Downlink data transfer speed | 0 |
| uplinkKbps | Uplink data transfer speed | 1070 |
| currentlyRoaming | indicates whether the SIM is currently roaming | true, false |

## 4.4 Get Streaming Bandwidth

Stream bandwidth is part of the unit status payload.
The relevant fields are: `totalDownlinkKbps, totalUplinkKbps.`
See Get unit status and Get unit status/full (including interfaces)

## 4.5 Get unit configuration

Method: `GET BaseURL/units/{uniqueId}/configuration"`
User permissions: read-only, regular, admin, media-group admin
Response - get unit configuration:


```
{
  "unitUniqueId": "string",
  "audio": {
    "ifbAlwaysOn": true,
    "audioChannels": "string",
    "audioEncodingBitrate": "string",
    "ifbEndToEndDelay": 0,
    "audioChannelsBitrateMap": {
      "2": "string",
      "4": "string"
    }
  },
  "video": {
    "operationalMode": "live",
    "selectedChannel": "string",
    "selectedDestination": "string",
    "linksPriority": {
      "free links": "string",
      "chargeable links": "string"
    },
    "linksPriorityV2": {
      "mode": "string",
      "priorities": {
        "cellular": {
          "priority": 0,
          "stopWhenBwReach": 0
        },
        "eth0": {
          "priority": 0,
          "stopWhenBwReach": 0
        },
        "eth1": {
          "priority": 0,
          "stopWhenBwReach": 0
        },
        "wifi": {
          "priority": 0,
          "stopWhenBwReach": 0
        }
      }
    },
```

```
    "storeAndForwardEncoder": "string",
    "delay": 0,
    "storeAndForwardQuality": "string"
  },
  "databridge": {
    "dataBridgeMode": "string"
  },
  "metadata": {
    "eventName": "string",
    "storyID": "string",
    "crewName": "string",
    "comments": "string",
    "usageTerms": "string",
    "provider": "string",
    "reporter": "string",
    "keywords": "string"
  },
  "metadataPromptEnabled": true,
  "metadataEnabled": true,
  "videoReturnChannelId": "string",
  "videoComplexityLevel": "low"
}
```


Response parameters - get unit configuration

| Parameter | description | Examples |
|---|---|---|
| unitUniqueId | The unique id of the unit | "Boss100_lu600_ 3235313934316434653666643462376 4" |
| audio- ifbAlwaysOn | Indicates if unit is configured to ifb always on (ready for one-direction ifb over stream path) | true, false |
| audio – audioChannels | Indicates number of audio channel that are used.<br>Options are derived from unit capability | 2, 4, 8 |
| audio - audioEncodingBitrate | The audio bitrate.<br>Options are derived from unit capabilities.<br>The response is I Kbps. | 96, 128, 192 |
| audio - ifbEndToEndDelay | The delay in ifb between the studio and the unit | 0 |
| audio - audioChannelsBitrateMap | The bitrate to use per number of audio channels | {"2": "96","4": "200"} |
| video- operationalMode | Unit's operational mode | "live", "storeandforward" |
| video - selectedChannel | Name of selected video channel | "Moon_Instance2" |
| video - selectedDestination | Applicable only for Solo units | |

| | | |
|---|---|---|
| video – linksPriority<br>video - linksPriorityV2 | least cost bonding settings.<br>linksPriorityV2 depends on unit capabilities. | |
| video - storeAndForwardEncoder | Encoder for STORE & FORWARD.<br>Note: H.265 requires matching video card | "H.264/AVC", "H.265/HEVC" |
| video - delay | unit delay in milliseconds | 2500 |
| video - storeAndForwardQuality | Quality of store&forward streaming | "high", "standard", "low" |
| databridge - dataBridgeMode | Defines if databridge mode is gateway ("bonding") or multipath ("wanRouter") | "bonding", "wanRouter" |
| metadata | details metadata parameters.<br>Video return is not applicable with all unit types. | |
| metadataEnabled | indicates if metadata is enabled for unit | true, false |
| metadataPromptEnabled | indicates whether metadata screen is prompted when user START streaming | true, false |
| videoReturnChannelId | id of selected video return channel.<br>Video return is not applicable with all unit types. | "Boss1100_189250945696878_ VideoReturnInstance2" |
| videoComplexityLevel | Defines if encoder is optimized for low or high video complexity.<br>The availability of this feature is based on unit capabilities | "low", "high" |

# 4.6 GET/update unit's operational mode

### 4.6.1 Get operational mode:

User permissions: read-only, regular, admin, media-group admin
see Get unit configuration

### 4.6.2 Update operational mode

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/operationalMode"`

Request-body - update unit operational mode

```
{"operationalMode": "string"}
```

Request parameters - update unit operational mode

| Parameter | description | Examples |
|---|---|---|
| operationalMode | Indicates if the unit is streaming live or STORE & FORWARD | "live", “storeandforward” |

## 4.7 GET/SET unit delay

### 4.7.1 Get unit’s delay:

User permissions: read-only, regular, admin, media-group admin
see Get unit configuration

### 4.7.2 Update delay

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/delay"`
Request-body - update unit’s delay request body

```
{"delay": int}
```

Request parameters - update unit’s delay

| Parameter | description | Examples |
|---|---|---|
| delay | Delay is set in milliseconds | 2500 |

## 4.8 GET/SET unit store&forward quality

### 4.8.1 Get store&forward quality

User permissions: read-only, regular, admin, media-group admin
see Get unit configuration

### 4.8.2 Update store&forward quality

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/storeandforwardQuality"`
Request-body - update unit store&forward quality

```
{"quality”:  "string", "encoder”: "string"}
```

Request parameters - update unit’s store&forward quality

| Parameter | description | Examples |
|---|---|---|
| quality | Set quality of store&forward stream | "high", "standard" or "low" |
| encoder | Set encoder of store&forward stream<br>"h265" encoder is enabled only for units with AVIC video card | "h264" or "h265" |

## 4.9 GET/SET unit selected video channel

### 4.9.1 Get selected video channel:

User permissions: read-only, regular, admin, media-group admin
see Get unit configuration

### 4.9.2 Update selected video channel:

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/channels/selected"`
Request-body - update unit's selected video channel

```
{"selectedChannel": "string"}
```

Request parameters - update unit's selected video channel

| Parameter | description | |
|---|---|---|
| selectedChannel | uniqueId of the selected channel | "Boss1100_161334323476_Instance4" |

## 4.10 Start/Stop video streaming

### 4.10.1 Start video stream:

User permissions: regular, admin, media-group admin
Method: `POST BaseURL/units/{uniqueId}/stream"`

### 4.10.2 Stop video stream:

User permissions: regular, admin, media-group admin
Method: `DELET BaseURL/units/{uniqueId}/stream"`

## 4.11 GET/SET unit selected video return channel

### 4.11.1 Get selected video return channel:

User permissions: read-only, regular, admin, media-group admin
see Get unit configuration

### 4.11.2 Update selected video return channel:

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/vrChannels/selected"`
Request-body - update unit's selected video-return channel

```
{"selectedChannel": "string"}
```

Request parameters - update unit selected video-return channel

| Parameter | description | Examples |
|---|---|---|
| selectedChannel | the uniqueId of the selected video return channel | "Boss1100_233845177955412_VideoReturnInstance1" |

Note: supported only for unit version 8.0.1 and up

## 4.12 Start/Stop video return

### 4.12.1 Start video return stream:

User permissions: regular, admin, media-group admin
Method: `POST BaseURL/units/{uniqueId}/vrChannels/stream"`
Permission by user type: regular, admin, media-group admin

### 4.12.2 Stop video return stream:

User permissions: regular, admin, media-group admin

Method: `DELETE BaseURL/units/{uniqueId}/vrChannels/stream"`
Permission by user types: regular, admin, media-group admin
Note: unit version should be 8.0.1 and up

# 4.13 GET unit location

User permissions: read-only, regular, admin, media-group admin
Method: `GET BaseURL/units/{uniqueId}/geolocation"`
Permission by user types: read-only, regular, admin, media-group admin

Response-body – get unit location:

```
{
  "latitude": "string",
  "longitude": "string",
  "address": "string"
}
```

Response parameters – get unit location:

| Parameter | description | Examples |
|---|---|---|
| latitude | Norh -south coordinates | 40.785091 |
| longitude | East–west coordinates | -73.968285 |
| address | address | 89 West 86th Transverse Road, New York, NY 10024<br>New York, Central Park New York, Manhattan New York United States |

# 4.14 GET/SET bandwidth limit

## 4.14.1 Get unit's bandwidth limit

User permissions: read-only, regular, admin, media-group admin
Method: `GET BaseURL/units/{uniqueId}/bwLimitation"`
Response body – get unit bandwidth limitation

```
{"bwLimitInKbps": int}
```

Response parameters - get unit bandwidth limitation

| Parameter | description | Examples |
|---|---|---|
| bwLimitInKbps | maximum bandwidth (in Kbps) unit can use for streaming. Value of zero (0) indicates there is no bandwidth limit | 0, 3500 |

## 4.14.2 Set unit's bandwidth limit of unit

User permissions: regular, admin, media-group admin
Method: `PUT BaseURL/units/{uniqueId}/bwLimitation"`
Request-body - update unit's bandwidth limit

```
{"bwLimitInKbps": int}
```

Request parameters - update unit's bandwidth limit
Same parameters as in the GET response (see Get unit's bandwidth limit)

# 4.15 GET/SET unit's metadata

## 4.15.1 Get unit's metadata configuration

User permissions: read-only, regular, admin, media-group admin
Method: GET BaseURL/units/{*uniqueId*}/metadata"

Response-body – get unit metadata:

```
{
  "enabled": true,
  "promptEnabled": true,
  "eventName": "string",
  "storyId": "string",
  "crewName": "string",
  "comments": "string",
  "usageTerms": "string",
  "provider": "string",
  "reporter": "string",
  "keywords": "string"
}
```

Response Parameters– get unit metadata:

| Parameter | description | Examples |
|---|---|---|
| enabled | indicates whether metadata is enabled | true, false |
| promptEnabled | indicates whether to prompt for metadata information when user starts stream | true, false |
| eventName | Name of the event | |
| storyId | Story id of the event | |
| crewName | Crew name | |
| comments | comments | |
| usageTerms | Usage tems of the event | |
| provider | Provider of the event | |
| reporter | Reporter of the event | |
| keywords | keywords of the event | |

## 4.15.2 Update unit's metadata configuration

User permissions: regular, admin, media-group admin
Method: PUT BaseURL/units/{*uniqueId*}/metadata"

Request-body - update unit's metadata:

```
{
  "enabled": true,
  "promptEnabled": true,
  "eventName": "string",
  "storyId": "string",
  "crewName": "string",
  "comments": "string",
  "usageTerms": "string",
  "provider": "string",
  "reporter": "string",
  "keywords": "string"
}
```

Request parameters - update unit's metadata:
Same parameters as in the GET response (see Get unit's metadata configuration)

Note: unit should be idle and have the capability to activate metadata (depends on unit type and version)

# 4.16 Modem/interfaces configuration

## 4.16.1 Get unit's modem port configuration

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/units/{uniqueId/interfaces/modem/{port}/config"

Response-body – get modem port configuration:

```
{
  "port": "string",
  "numberToDial": "string",
  "username": "string",
  "password": "string",
  "apns": [
    "string"
  ],
  "hardBandwidthLimitInKbps": 0,
  "disable": true,
  "forceLTE": true,
  "gracePeriod": 0,
  "randomUsernameAndPassword": true,
  "name": "string",
  "delayRunningPPPD": 0
}
```

Response Parameters - get modem interface configuration

| Parameter | description | Examples |
|---|---|---|
| port | The first modem port. First modem port is zero (0) | "6" |
| numberToDial | Specifies the phone number and/or dial-in string to be dialed by the unit. | "*99#" |
| username | Specifies the username for connecting to the service provider. Used by old modems (CDMA). | |
| password | Specifies the password for connecting to the service provider. Used by old modems (CDMA). | |
| apns | Access Point Name (APN) is the name of a gateway between mobile network and the public Internet. If empty, use LiveU defaults. | |
| hardBandwidthLimitInKbps | Indicates if modem has bandwidth limit. When modem is not limited, the value is zero (0). | 0<br>3000 |
| disable | indicates if modem is disabled or enabled | |
| forceLTE | If true, 3G is forced | |
| gracePeriod | | |
| randomUsernameAndPassword | If this value is true, it overrides the User and Password set every boot. | true,<br>false |
| name | Name of the interface | |

| | | |
|---|---|---|
| delayRunningPPPD | For very old modems, add delay before dial | |

### 4.16.2 Update unit's modem port configuration:

User permissions: regular, admin, media-group admin
Method: PUT
BaseURL/units/{uniqueId/interfaces/modem/{port}/config"

## 4.17 ETH interfaces

### 4.17.1 Get unit's eth port configuration

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/units/{uniqueId}/interfaces/eth/{port}/config"

Response-body – get eth port configuration:

```
{
  "macAddress": "string",
  "localAddress": "string",
  "netmask": "string",
  "gateway": "string",
  "staticConfiguration": true,
  "dnsServers": [
    "string"
  ],
  "enabled": true,
  "isPOESupported": true,
  "powerOverEthernet": true,
  "port": "string"
}
```

Response Parameters – get eth port configuration

| Parameter | description | Examples |
|---|---|---|
| macAddress | Eth MAC address | "78:51:0c:00:b7:03" |
| localAddress | Eth local IP address | "172.16.44.174" |
| netmask | Netmask | "255.255.252.0" |
| gateway | Gateway IP | "172.16.45.254" |
| staticConfiguration | If DHCP, static configuration=false | false, true |
| dnsServers | IP of DNS | ["192.168.2.8","8.8.8.8"] |
| enabled | indicates if eth port is disabled or enabled | false, true |
| isPOESupported | indicates unit supports POE | false, true |
| powerOverEthernet | Indicates the status of POE | false, true |
| port | The wifi port. First eth port is zero (0) | "eth1" |

### 4.17.2 Update unit's eth port configuration

User permissions: regular, admin, media-group admin
Method: PUT
BaseURL/units/{uniqueId}/interfaces/eth/{port}/config"
Request-body: update eth port configuration

```
{
  "macAddress": "string",
  "localAddress": "string",
```

```
  "netmask": "string",
  "gateway": "string",
  "staticConfiguration": true,
  "dnsServers": [
    "string"
  ],
  "enabled": true,
  "isPOESupported": true,
  "powerOverEthernet": true,
  "port": "string"
}
```

Request Parameters – update eth port configuration

| Parameter | description | Examples |
|---|---|---|
| localAddress | ETH local IP (required when DHCP-off) | "172.16.44.174" |
| netmask | Netmask (required when DHCP-off) | "255.255.252.0" |
| gateway | Gateway IP (required when DHCP-off) | "172.16.45.254" |
| staticConfiguration | If DHCP, static configuration=false | false, true |
| dnsServers | IP of DNS (required when DHCP-off) | ["192.168.2.8","8.8.8.8"] |
| enabled | disable or enable eth port | false, true |
| isPOESupported | If POE supported, use it or not.<br>It is recommended to use POE (true) when unit is connected to extender | false, true |
| port | The wifi port. First eth port is zero (0) | "eth0" |

# 4.18 WiFi interfaces

Note: The wifi port which should be used as {port} parameter in the path is "wlan0"

## 4.18.1 Get unit's wifi port configuration

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/units/{uniqueId}/interfaces/wlan/{port}/config"

Response-body – get wifi port configuration

```
{
  "enabled": true,
  "port": "string"
}
```

Response Parameters – get wifi port configuration

| Parameter | description | Examples |
|---|---|---|
| enabled | indicates if unit's wifi is enabled or disabled | "true", "false" |
| port | the unit wifi port | "wlan0" |

## 4.18.2 Enable/Disable wifi interface of unit

User permissions: regular, admin, media-group admin
Method: PUT
BaseURL/units/{uniqueId}/interfaces/wlan/{port}/config"

Request-body - enable/disable wifi interface

```
{
  "enabled": true,
}
```

Request Parameters - enable/disable wifi interface

| Parameter | description | Examples |
|---|---|---|
| enabled | enable or disable the wifi port | "true", "false" |

### 4.18.3 Scan wifi networks

User permissions: regular, admin, media-group admin
Method: `POST`
`BaseURL/units/{uniqueId}/interfaces/wlan/{port}/wifi/scan"`

Notes:

1. while scanning the currently connected wifi network is being disconnected.
2. To receive the scanned networks, application should call Get list of wifi networks detected by unit.

### 4.18.4 Get list of wifi networks detected by unit

User permissions: read-only, regular, admin, media-group admin
Method: `GET`
`BaseURL/units/{uniqueId}/interfaces/wlan/{port}/wifi"`

Response-body – get list of wifi networks

```
[
  {
    "status": "string",
    "ssid": "string",
    "bssid": "string",
    "signalLevel": 0,
    "frequency": "string",
    "flags": "string",
    "reception": "string",
    "networkType": "string"
  }
]
```

Response Parameters – get list of wifi networks

| Parameter | description | Examples |
|---|---|---|
| status | indicates if wifi port is connected to wifi network | "Connected", "Unknown" |
| networkType | Network type | WPA2-PSK", "NONE" |
| bssid | Basic service set identifiers (BSSID) is used to describe sections of a wireless local area network or WLAN. Most of the time it is associated with MAC address of the AP | 80:ce:62:90:b1:a0 |
| signalLevel | Level of wifi signal | -78 |
| frequency | Eifi network frequency | "2412" |
| flags | Wifi network flags | " [WPA2-PSK-CCMP-preauth][ESS]" |

| | | |
|---|---|---|
| reception | Wifi network reception level | "two", "three" |
| ssid | Network name | LiveU-Guest |

Notes:
1. Scan for networks (see Scan wifi networks) is require before applying this API

### 4.18.5 Connect unit to wifi network

User permissions: regular, admin, media-group admin
Method: `PUT`
`BaseURL/units/{uniqueId}/interfaces/wlan/{port}/wifi/{ssid}"`

Request-body - connect to wifi network

```
{
  "networkType": "string",
  "password": "string",
  "onlyConnect": true,
  "ssid": "string"
}
```

Request Parameters - connect to wifi network

| Parameter | description | Examples |
|---|---|---|
| networkType | Network type | WPA2-PSK", "NONE" |
| password | Network password.<br>must be provided if network is secured. | |
| onlyConnect | Unit connect to wifi but does not “remember” network & password for future use | true, false |
| ssid | Network name | |

### 4.18.6 Delete (forget) wifi network

User permissions: regular, admin, media-group admin
Method: `DELETE`
`BaseURL/units/{uniqueId}/interfaces/wlan/{port}/wifi/{ssid}`

# 5 List of units

## 5.1 List of units in an Inventory

### 5.1.1 Get the inventories the API user is associated to

1. Method: GET
   BaseURL/inventories

### 5.1.2 Get list of units in inventory

This API is described in 4.1, Get list of units in inventory.

### 5.1.3 Get list of units in inventory, enhanced

1. When querying for a list of units the query should include the offset and limit params (see 2.1, API calls regarding list of entities).
2. Method: GET
   BaseURL/inventories/{dbId}/units/enhanced
3. Example of the current payload when the list includes a single streaming unit:


```
[
  {
  "dbId": 135912,
  "uid": "Boss100_lu800_xxxx",
  "serialNumber": "nnnn",
  "swVersion": "10.1.0.C3743.G6bfc23d12",
  "vicType": "HEVC HDR",
  "vicSerialNumber": "ESEnnnn",
  "product": "LU-800",
  "saleType": "",
  "status": {
    "unitUniqueId": "Boss100_lu800_xxxxx",
    "upTime": 781565.5908114329,
    "availability": "online",
    "applications": {
      "video": {
        "state": "streaming",
        "channel": "Boss1100_xxxxx",
        "usedLinks": null,
        "inputSdi": null,
        "uploadKbps": 20595
      },
      "store": {
        "fileName": "/storage/../xxx.mkv",
        "fileSize": 5119667851
      }
    },
    "videoMode": "1080p59.94",
    "battery": {
      "connected": true,
      "percentage": 96,
      "runTimeToEmpty": 306,
      "discharging": false,
      "charging": false
    },
    "onAir": false,
    "tallyLight": "disconnected",
    "totalDownlinkKbps": 0,
    "totalUplinkKbps": 20595,
    "videoModeList": [
      {
        "sdiPort": "0",
        "videoMode": "1080p59.94"
```

```
      }
    ]
  },
  "freelancer": false,
  "lastReboot": 1718542583000,
  "appVersion": "",
  "bossId": "Boss100_lu800_30323045363144364633324546363238",
  "alias": "nnn-alias",
  "name": " nnn-alias-name"
}]
```


## 5.2 List of units in a Group

### 5.2.1 Get the groups the API user is associated to

1. Method: GET
   BaseURL/groups

### 5.2.2 Get list of units in a group

1. When querying for a list of units the query should include the offset and limit params (see 2.1, API calls regarding list of entities).
2. Method: GET
   BaseURL/groups/{dbId}/units
3. Payload is similar to the payload described in 5.1.2.

### 5.2.3 Get list of units in a group, enhanced

1. When querying for a list of units the query should include the offset and limit params (see 2.1, API calls regarding list of entities).
2. Method: GET
   BaseURL/groups/{dbId}/units/enhanced
3. Payload is similar to the payload described in 5.1.2.

# 6 Newsroom integration

## 6.1 Overview

1. To support Newsroom integration customer should purchase LU-INGEST-NRCS license.
   This license will enable the customer to manage stories.
2. The list of stories can be seen by Central user or unit user.
3. Central user or unit user can select a specific story for the unit.
   This should be done BEFORE the unit starts transmitting
4. The customer should set the external story-id
5. LiveU Central manages the stories based on internal story's db-id.

## 6.2 APIs

### 6.2.1 Get list of stories by inventory

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/inventories/{dbId}/stories?offset=0&limit=100

- This API is paginated, if offset & limit are not indicated, the default will be offset=0 and limit=100.
  The maximum limit is 1000
- It is possible to search by text in the slugline using query param searchKey=slugline&searchValue=test
- It is possible to sort by last modified (asc or desc) query param sortBy=lastModified&sortDir=desc

### 6.2.2 Create new story

User permissions: regular, admin, media-group admin
Method: POST
`BaseURL/stories`

### 6.2.3 Get a specific story

User permissions: read-only, regular, admin, media-group admin
Method: GET
`BaseURL/stories /{dbId}`

### 6.2.4 Update a specific story

User permissions: regular, admin, media-group admin
Method: PUT
`BaseURL/stories /{dbId}`

### 6.2.5 Delete a specific story

User permissions: regular, admin, media-group admin
Method: DELETE
`BaseURL/stories /{dbId}`

### 6.2.6 Get unit current story

User permissions: read-only, regular, admin, media-group admin
Method: GET
`BaseURL/units/{uniqueId}/metadata`

### 6.2.7 Update unit with a story

User permissions: regular, admin, media-group admin
Method: PUT
`BaseURL/units/{uniqueId}/metadata`
Note: updating the unit metadata (story) is relevant only when the unit is idle.

### 6.2.8 Associate a story to unit(s)

User permissions: regular, admin, media-group admin
Method: POST
`BaseURL/stories/{dbId}/units/association`

### 6.2.9 Disassociate story from unit(s)

User permissions: regular, admin, media-group admin
Method: POST
`BaseURL/stories/{dbId}/units/association`

### 6.2.10 Get list of stories by inventory by unit

User permissions: regular, admin, media-group admin
Method: GET

BaseURL/inventories/{dbId}/units/{uniqueId}/stories?offset=0&limit=100

- This API is paginated, if offset & limit are not indicated, the default will be offset=0 and limit=100.
  The maximum limit is 1000
- It is possible to search by text in the slugline using query param searchKey=slugline&searchValue=test
- It is possible to sort by last modified (asc or desc) query param sortBy=lastModified&sortDir=desc

### 6.2.11 Get list of units associated to a story

User permissions: read-only, regular, admin, media-group admin
Method: GET
BaseURL/stories/{dbId}/units

## 6.3 Examples

1. Get story list
   GET
   BaseURL/inventories/514/stories?offset=0&limit=20&searchKey=slugline&searchValue=test&sortBy=lastModified&sortDir=desc
   Response payload:

```
[
	{
		"id": 368,
		"slugline": "tesr External story",
		"crewName": "abcd",
		"storyId": "12345678",
		"comments": "",
		"lastModified": 1654672321000
	},
	{
		"id": 366,
		"slugline": "gg",
		"crewName": "abcd",
		"storyId": "abcdefg",
		"comments": "",
		"lastModified": 1654611828000
	}
]
```

2. Create a story
   POST BaseURL/stories
   Request payload:

```
{"slugline": "Elections",
 "storyId": "123456789-",
 "reporter": "John.M",
 "inventory": 514
}
```

3. Get unit's matadata
   GET BaseURL/units/Boss100_lu300_xxxx/metadata
   Response payload:

```
{
      "enabled": false,
      "promptEnabled": true,
      "eventName": "Elections",
      "storyId": "123456789-",
      "crewName": "John.M",
      "comments": "",
      "usageTerms": "",
      "provider": "",
      "reporter": "",
      "keywords": ""
}
```


4. Associate a story to unit(s)
   POST BaseURL/stories/1088/units/association
   Request payload:
   `["Boss100_lu800_3141314432314436463643393"]`

5. Disassociate a story from unit(s)
   POST BaseURL/stories/1088/units/disassociation
   Request payload:
   `["Boss100_lu800_3141314432314436463643393"]`

6. Get list of stories by inventory by unit
   GET BaseURL**/**inventories/514/units/Boss100_lu800_**xxx**/stories?offset=0&limit=20
   Response payload:
   The response payload includes the assigned field.


```
[
  {
    "id": 1088,
    "slugline": "Elections",
    "crewName": "John.M",
    "storyId": "0ef0b4ae:008d4e49:65300269",
    "comments": "\n\n",
    "lastModified": 1697693651000,
    "assigned": true
  }
]
```


7. Get list of units associated to a story
   GET BaseURL**/**stories/10673/units
   Response Payload:


```
[
      {
               "dbId": 74309,
               "uid": "Boss100_xxx",
               "bossId": "Boss100_xxx",
               "alias": "alias of unit"
      },
      {
               "dbId": 44239,
               "uid": "Boss100_yyy",
               "bossId": "Boss100_yyy",
               "alias": " alias of unit"
      }
]
```

# 7 Multi-camera integration

## 7.1 Overview

1. To support multi-camera the customer should have the hardware unit supporting it (currently LU-800).
2. To support multi-camera the customer should have multi-camera service (currently pro2 or pro4)
3. In order to stream to multi-camera all the channels (based on the configured number of channers) must be idle.

## 7.2 APIs

### 7.2.1 Get the unit's groups (which are shared with the API user)

1. Method: GET
   BaseURL/units/{uniqueId}/groups.

### 7.2.2 Get a list of multi-camera destinations (per group)

1. Method: GET
   BaseURL/groups/{dbId}/multiChannels

### 7.2.3 Get the current multi-camera configuration

1. The multi-camera configuration is included in the unit configuration
2. Method: GET
   BaseURL/units/{uniqueId}/configurationV2

### 7.2.4 Set the multiple camera channel:

1. Method: PUT
   BaseURL/units/{uniqueId}/multiplecamera/channel
2. Payload: {"channel": "alias of channel"}

### 7.2.5 Set the multi-camera number of channels:

1. Method: PUT
   BaseURL/units/{uniqueId}/multiplecamera
2. payload: {"numberOfFeeds": 3}

### 7.2.6 Set the delay of multi-camera:

1. Method: PUT
   BaseURL/units/{uniqueId}/multiplecamera
   payload: {"streamingInformation":
   {"sdi0": {"video": {"delay": 700}},
   "sdi1": {"video": {"delay": 700}},
   "sdi2": {"video": {"delay": 700}},
   "sdi3": {"video": {"delay": 700}}
   }

### 7.2.7 Start a multi-camera stream:

1. To be able to start streaming the number of relevant channel must be idle.
2. Method: POST
   BaseURL/units/{uniqueId}/stream/multiplecamera

### 7.2.8 Stop a multi-camera stream

1. Method: DELETE
   DELETE BaseURL/units/{uniqueId}/stream/multiplecamera