Videoloft API for viewing live video, recording and playing video events

Version 1.0.8

1.1Introduction

An IP camera or NVR/DVR channel set up with Videoloft can be used to stream live video and record video to the cloud.

Live Stream
To minimise network usage, a camera won't stream live video all the time, so you need to use our API to tell it to start streaming. Once this has been done, you can then view live video for the next minute. To view for longer, just keep calling the API every 30 seconds to keep the live stream running.

Recorded video
When videos are recorded to the Videoloft cloud in response to motion or triggered by an API call, we call these 'Events' in the Videoloft app and webviewer UI. As you'll see, in the API we also refer to these as Alerts, so hopefully that's not too confusing.

It's also worth noting that we use the word 'devices' in the API, and this generally means a camera or recorder channel.

A device is always owned by a specific user, so we can identify a device using a uid.deviceid, e.g. 12345.17. In this example the user id is 12345 and the device id is 17. Users can be given access to other user's devices, so for example, user 4321 may have access to view the camera 12345.17. At the moment, you need to contact Videoloft to set up multi-user access, but in the future, we'll have a UI to allow management of this.

Recorded video is delivered as H.264/AAC (codec) over HLS (transport protocol).

Some notes here are written assuming the API is used from the web because that's how we use it, but hopefully it can be used from a Windows application too.

1.1.1 Server types

There are three main types of server required to view live and play events. These servers are hosted in the AWS cloud.

Authenticator
Calls are made to the authenticator to login as a user and get an authentication token. Once you have a valid token, information about the cameras can also be retrieved from the authenticator. *Note that tokens expire after 20 minutes.*

There is a single authenticator DNS address per region which is linked to an auto scaling group of servers through a load balancer. We currently have US and EU regions in production. Going to auth1.manything.com will automatically take you to the 'nearest' region and redirect you to the region specified for the user account. Once you know the specific authenticator to use, you should use that one directly - e.g. in the US, use useast1-auth-1.manything.com

Logger
Loggers are general purpose servers used to access data, images and video for cameras. Loggers aren't currently load balanced, so a camera is allocated to a specific logger and will generally stay on that logger unless that server develops a problem, or there is a significant change in system load. Each logger holds state information about the cameras which are allocated to it.

Calls are made to the camera's logger to get a list of events for that camera. Once you have a list of events, you can then call the logger again to get thumb images and videos for specific events. You can also call the logger to send messages to the camera, e.g. to tell the camera to start streaming live video. You need to supply a valid token to talk to loggers. The token must be valid for the camera(s) that you want to access and the action(s) that you want to perform on those cameras.

Wowza

Wowza servers are used for viewing live video streams. As with Logger servers, the Wowza servers aren't load balanced so you need to use the Wowza server that is assigned to a specific camera to view live video from that camera. Cameras sometime move to different Wowza servers, so before viewing live video, you should check that you have the latest information about which Wowza server to use.

1.1.2 Overview for viewing live video

Calls to Authenticator:

1. Login as a user, get an authentication token **/login**
2. Get a summary list of devices that this user can view, including the camera names **/devices**
3. Work out which camera you want to view live on. e.g. based on user and camera name
4. Get detailed data for this device including the Wowza server to use and video stream name **/devices/viewerInfo**

Calls to Logger

1. Tell a camera to start streaming live video **/sendcameratask** with action=livecommand (keep calling this every 30 seconds to continue the video streaming)
2. Poll camera status e.g. to find out if it's streaming live video **/cameras/status** (when camera is streaming, status live value will be set to true)

Calls to Wowza

1. Play live video from Wowza. Pass the Wowza URL with /playlist.m3u8 to your video player to play the HLS video.

1.1.3 Overview for recording an event video

Event videos are recorded automatically, e.g. when motion is detected, but you can also trigger the recording using the API.

Keep sending a triggerevent action to a camera and it will continuously record video until about 20 seconds after you stop sending the action.

Calls to Authenticator:

1. Login as a user, get an authentication token **/login**
2. Get a summary list of devices that this user can access **/devices**
3. Decide which device you want to record the event on
4. Get detailed data for that device including the logger server to use **/devices/viewerInfo**

Calls to Logger

1. Tell a camera to start recording video **/sendcameratask** with action=triggerevent - keep calling this every 15 seconds to continue the video recording
2. Just stop calling it when you want the recording to finish and it will finish 20 seconds after the last call. Note that if there is motion detected after the last call, the recording will run on for longer and will finish about 20 seconds after the motion stops. However, when you come to view the event later – the play-out will end 5 seconds after the last motion trigger or the last cameraTask trigger.

1.1.4 Overview for viewing a recorded event video

Here's a summary of the process to play an event. Process will vary depending on how you want the UI to work, but here's an example where the user wants to view events for one selected device.

Calls to Authenticator:

1. Login as a user, get an authentication token **/login**
2. Get a summary list of devices that this user can view **/devices**
3. User chooses the device they're interested in
4. Get detailed data for this device including the logger server to use **/devices/viewerInfo**

Calls to Logger

1. Get a list of 20 most recent events for this device **/alert**
2. Get thumb images for each of those events **/alertthumb**
3. Display thumb image along with time and duration of the event in the UI
4. User selects event to play
5. Form HLS video URL and play the video **/alert.m3u8**
6. If event that user wants to view isn't within the 20 events displayed, call /alert again with an offset parameter to get the next 20

Other key info

- Call /login e.g. every 15 minutes to refresh the authentication token
- Call /devices/viewerInfo every 10 minutes to make sure you have latest logger server details for a device
- In our UI, we show events in reverse order (newest first), if the user scrolls down to the end of the list of events, call /alert with an offset to get the next 20 (older) events
- To reduce load on our servers and to speed up performance for the end user, you should cache the .TS video files locally
- There are other APIs which enable you to build a date-picker for viewing events, if you wish to have access to these API, please let us know

1.1.5 Notes for multi-user systems

If you've got a system where a logged in user has access to other users' cameras, then there are some extra things to be aware of.

- In a multi-user system, a logged in user can be set up to have access to other user's cameras. This can be all their cameras, or just a subset of cameras. There are two levels of access; read-only and admin. With read only access, you can have access to see camera names, recorded and live video. With admin access, you can do additional things such as change cameras settings and delete recorded video events.

- Setting up a multi-user system can save you having to login as each individual user to view their cameras. If you have a single master user, you'll then only need one username / password to access all the cameras that want your system to access. At the moment, you'll need to get in touch with Videoloft to get a multi-user system set up, but in future we'll have a UI to allow you to manage this yourself.

- When getting the summary list of devices from the Authenticator in step 2 in above examples you'll see that the devices are grouped by the uid of user that 'owns' those cameras. See /devices API spec below for more details on the data format.

- If you want your UI to allow selection of a specific camera user to focus on, you can use the 'alias' value returned in the /device data which will be the subscriber name for the camera user that was specified when creating the user in the Videoloft Portal.

- The authentication token returned by /login will give the logged in user access to any of the devices, returned from the /devices call. For example, if a user with uid 100 has read access to user 200's cameras, then you logged in as user 100, you could call <LOGGERSERVER>/alerturl.m3u8?uid=200.1… to view events from one of user 200's cameras.

1.2 Authenticator API

1.2.1 Login

Authenticates the user and returns a token

API	Returns

GET /login
 	
Initial web login with email and password
 	
https://auth1.manything.com/login?email=<USEREMAIL>&password=<USERPASSWORD>&caller=website
 	
Parameters
 	
• **email:** email address of account holder
• **password**
• **caller:** For initial web login, set to 'website'. Not required for login from an application.
 	
e.g. https://auth1.manything.com/login?email=demo@mt.com&password=manything&caller=website
 	
Subsequent web login using cookies, so no password needed
 	
https://auth1.manything.com/login?caller=cookielogin
 	
• **caller:** Set to 'cookielogin'
 	
• The cookie from last login call will automatically be sent from the browser
 	
Important: Tokens will expire after 20 minutes, so you need to call /login regularly to ensure that fresh tokens are used.
	uid: id for the logged in user
	
	authToken: Videoloft token to access resources in the APIs below (expires after 20 minutes)
	
	authenticator: The authenticator DNS name to make API calls to (based on region of user)
	
	Also returns cookies if caller='website' or 'cookielogin'.
	
	Returns a lot of other data, but we don't need that for viewing events. In particular, please ignore the devices data as we have a new way to get that data calling /devices.
	
	Example return
	
	{
	"result":{
	"uid":12345,
	"devices":[
	{
	"id":2,
	"phonename":"Elm 5 iPhone 5 iOS8",
	"streamname":"341025.2_932950-nr",
	"wowza":"uswest2-wowza-1539961359-i-05490f5803c3c88ff.manything.com",
	"logger":"uswest2-logger-1515079812-i-06551d700ae2c0ea2.manything.com",
	"is_camera":1,
	"device_type":null,
	"language":"en-GB",
	"phoneid":"0CD580FE-2F35-4FB3-8A44-8B3B2E2CA5DA"
	},
	{
	"id":10,
	"phonename":"Birch 9 iPhone 7 iOS 10",
	"streamname":"341025.10_79661-nr",
	"wowza":"uswest2-wowza-1537199246-i-00e769f0ad1f1faa4.manything.com",
	"logger":"uswest2-logger-1515079812-i-06551d700ae2c0ea2.manything.com",
	"is_camera":1,
	"device_type":null,
	"language":"en-GB",
	"phoneid":"1C883862-3C8D-46AA-A116-55BDF68B79B5"
	}
	],
	"accountInfo":{
	"maxAge":1,
	"maxDevices":1,
	"type":"free",
	"provider":"installer"
	},
	"time":1540810542812,
	"region":"uswest2",

	"**authToken**":"MzQxMDI1OjE1NDA4MTA1NDI6MTIwMDowOkwsVyxB*do-tI6nPBpo9evgs4dVQkkZhJFXizpAm32KlFqNWL_cK_oOTGi2Obr-gl-1YLrdMGRrRSgX6lzHxf4RxsBzoNq7B5yXivlYiEU6f1oBcV7_NHrpjDlWKW7d0UDEOLSTA1HJqy8_srMl0_0UoniPawAXuNKs6QJC-ncMPz7oTw.", "**authenticator**":"https://uswest2-auth-1.manything.com", "**disableUpgrades**":false, "**paymentServer**":"https://uswest2-payment.manything.com", "**forceRom**":true, "**installerid**":null, "**distributorid**":null, "**ipcamDiscoveryEnabled**":true, "**managementType**":null, "**managementScope**":null, "**chunkLengthInBC**":60, "**chunkLengthInAlert**":60, "**backlogThresholdStopRecording**":1000000, "**backlogThresholdDropLevel**":1000000, "**backlogThresholdDropToRoM**":5, "**minAppVersion**":"4.0.0 (788)", "**romAlertAdjustment**":4, "**videoUploadOrder**":"ASC", "**keepRecordingDefault**":1, "**chunkLengthPreIOS8**":120, "**showUpdatePopup**":true, "**hideUpdatePopupAfter**":12, "**continuousAlertAdjustment**":4, "**defaultToRoM**":true, "**timeOfNoTriggersToEndAlert**":20, "**emailWhenOffline**":1, "**notificationWhenOffline**":1, "**maxLengthOfTalkbackAudio**":10, "**minGapBetweenTalkbackAudioPlayback**":1, "**fallbackToLoggerUploads**":false, "**showWorkForManythingLink**":false, "**numOfFreeMotionEvents**":0, "**email**":"a@b.com" } }

1.2.2 Get list of all devices that a logged in user has access to

Get basic info for all devices that a specific user has viewer access to.

A logged in user will always have access to cameras created directly under their account. They may also have been given access to other users' cameras. A user may therefore have access to hundreds or even thousands of cameras, so we initially just get the basic information for those cameras, like the id, phone name and name (alias) of the user that owns a camera.

API	**Returns**
GET /devices curl -w '\n' -v "<AUTHSERVER>/devices" -H	*All the devices that the user (from the authToken) has access to.* Looking at the example return data below, this is what that data means:

"Authorization: ManythingToken

<AUTHTOKEN>"

Parameters

- **AUTHSERVER:** can use the value of 'authenticator' that was returned from /login, e.g. For test users, this is usually 'https://uswest2-auth-1.manything.com'
- **AUTHTOKEN:** is the value of authToken returned from /login

e.g.

curl -w '\n' -v "https://euwest1-auth-

1.manything.com/devices" -H

'Authorization: ManythingToken

ODI3MDI5OjE1NjU4NzM2MTQ6MTIwMDozO

kwsVyxB*KfSCwnMCUa1g63_RotjgCU20T

cMsf-T8JYTYDPcNjb19_7SJkSLoTuVTkR

pERMFkBofex-AicMKZWXyRHlTHrKWZDsn

gePI2xuwd0T8795sG0gab8Ekjjs5NMfXI

fwuwwz6pZXTSbrNBz69HvbmNXF7vppkG6

Gjzco32aoPFI.'

- "**329475**": or "**400123**": The uids of the users that own the devices contained within this section. In this example the logged in user has access to cameras within two users account.
- **devices*:*** Contains one or more devices owned by the above users
- "**56**" The device id
- ***uidd:*** uid.deviceid for the device
- **phonename:** The Videoloft specified phone name for this device, can be different from the Hikvision specified camera name
- **permissions:** r= read only, a= admin
- **alias:** Subscriber name of the user that owns the devices

Example return


```
{
 "result":{
  "329475":{
   "devices":{
    "56":{
     "uidd":"329475.56",
     "phonename":"Room 1",
     "permissions":[
      "r",
     ]
    },
    "59":{
     "uidd":"329475.59",
     "phonename":"Garden",
     "permissions":[
      "a"
     ]
    }
   },
   "alias":"Customer 1"
  }
 },
{
  "400123":{
   "devices":{
    "1":{
     "uidd":"400123.1",
     "phonename":"Kitchen",
     "permissions":[
      "r",
     ]
    },
    "2":{
     "uidd":"400123.2",
     "phonename":"Hall",
     "permissions":[
```

	"a"
	]
	}
	},
	"alias":"Customer 2"
	}
	},
	"errors":[
	{
	"1.1":"Unauthorized"
	}
	]
	}

1.2.3 Get more details for a subset of devices

When viewing events, our current UI allows you to explore the events for one device at a time. In this case, you would put up a list of available cameras and let the user select a single camera to view events for. You can then get more details about that specific camera.

For event viewing, the key thing we need to know about the selected device is which logger server the device is using. So for this case, call /devices/viewerInfo with a single uidd.

Call this every 10 minutes to get latest values, e.g. to see if a camera has been moved to a new logger.

API	**Returns**

GET /devices/viewerInfo
 	
curl -v -w "\n" -H "Authorization: ManythingToken
<AUTHTOKEN>"
<AUTHSERVER>/devices/viewerInfo
?uidd=329475.56&uidd=329475.59&
uidd=400123.1
 	
Parameters
 	
• **AUTHSERVER:** can use the value of 'authenticator' that was returned from /login, e.g. For test users, this is usually 'https://uswest2-auth-1.manything.com'
 	
• **AUTHTOKEN:** is the value of authToken returned from /login
 	
• **uidd:** You can specify 1, or n uidds (up to the URL length limit)
 	
e.g.
 	
curl -v -w "\n" -H 'Authorization:
ManythingToken ODI3MDI5OjE1NjU4NzM
2MTQ6MTIwMDozOkwsVyxB*KfSCwnMCUa1g
	Need to get initial values for Logger from here
	
	Similar format to /devices as above, but has additional data, e.g. logger, Wowza and streamname.
	
	Example return
	
	Has data for two camera from user 329475 and one camera from user 400123
	
	{
	"result":{
	"329475":{
	"devices":{
	"56":{
	"uidd":"329475.56",
	"phonename":"Room 1",
	"permissions":[
	"r",
	"w"
	],
	"wowza":"uswest2-wowza-1539961359-i-05490f5803c3c88ff.manything.com",
	"logger":"uswest2-logger-1540310790-i-0f24403219cd83749.manything.com",
	"cloudRecordingEnabled":1,
	"appVersion":"2.24.2",
	"deviceType":"ipcam201701",
	"streamname":"329475.56_39039-nr",
	"language":"en-GB",
	"phoneid":"DS-2CD2042WD-I20161005BBWR653544863"

63_RotjgCU20TcMsf-1T8JYTYDPcNjb19_
7SJkSLoTuVTkRpERMFkBofex-AicMKZWX
yRHlTHrKWZDsngePI2xuwd0T8795sG0ga
b8Ekjjs5NMfXIfwuwwz6pZXTSbrNBz69H
vbmNXF7vppkG6Gjzco32aoPFI.'
https://euwest1-auth-
1.manything.com/devices/viewerInfo
?uidd=329475.56&uidd=329475.59&
uidd=400123.1
	},
	"59":{
	"uidd":"329475.59",
	"phonename":"Garden",
	"permissions":[
	"a"
	],
	"wowza":"uswest2-wowza-1539961359-i-05490f5803c3c88ff.manything.com",
	"logger":"uswest2-logger-1540310790-i-0f24403219cd83749.manything.com",
	"cloudRecordingEnabled":1,
	"appVersion":"2.24.2",
	"deviceType":"ipcam201701",
	"streamname":"329475.59_94213-nr",
	"language":"en-GB",
	"phoneid":"DS-7204HUHI-K10420180126CCWRC04758466WCVU@01"
	}
	}
	},
	"400123":{
	"devices":{
	"1":{
	"uidd":"400123.1",
	"phonename":"Kitchen",
	"permissions":[
	"r"
	],
	"wowza":"uswest2-wowza-1539961359-i-05490f5803c3c88ff.manything.com",
	" logger":"uswest2-logger-1540310790-i-0f24403219cd83749.manything.com",
	"cloudRecordingEnabled":1,
	"appVersion":"2.24.2",
	"deviceType":"ipcam201701",
	"streamname":"400123.1_67012-nr",
	"language":"en-GB",
	"phoneid":"DS-2CD2077WD-I20161005BBWR653542311"
	}
	},
	}
	}

1.3Logger API

1.3.1 Get event data for a device

API	Returns
GET /alert There may be hundreds or thousands of events recorded by a camera, so we request the details for these in small batches, requesting more data as the user scrolls through the list. Get data for the first 20 events	Return value you'll need for event viewing: • **alert:** Id of the alert • **startt:** Start time of the event in milliseconds (UTC) • **endt:** End time of the event in milliseconds (UTC)

<LOGGERSERVER>/alert?uid=599.63&startt=1&offset=&limit=20&maxage=604800000&token=<AUTHTOKEN>

Parameters

- **LOGGERSERVER:** is the Logger server that a specific device is assigned to. Get the value from /devices/viewerInfo call above.
- **AUTHTOKEN:** is the value of authToken returned from /login
- **uid:** uid.deviceid for the device. e.g.599.63
- **startt:** start time in milliseconds of the event range (to keep things simple, set to 1)
- **offset:** Only return results after this offset in the overall list, e.g. '0' if want to get events from start of list, '20' if want to get next 20 results
- **limit:** How many events to return from the overall list
- **maxage:** Use this so that the request only returns events that the user's plan allows, this value is set to the user's cloud storage allowance in milliseconds, e.g. 604800000 would be used for a 7 day cloud storage plan

  For maxage, first get accountInfo.maxAge from the /login response, which will be in days. Multiply by 1000x60x60x24 to make it milliseconds.
- **token:** Videoloft token from login

Get next 20

<LOGGERSERVER>/alert?uid=599.63&startt=1&offset=20&limit=20&maxage=604800000&token=<AUTHTOKEN>

Note: We use max age to make sure we don't show events where the video has been auto deleted in line with the user's storage plan. For a 7 day storage plan, we actually store 9 days of video so the user can pay to upgrade and see an additional 2 days. We auto delete the database entries and the actual videos on separate schedules, so we may have 10 days of events in the database, but only 9 days of video.

- **type:** What triggered the event, can be 'motion' or 'audio'
- **smx:** Magnitude of the motion. Hex number, ranges from 00 to 0a
- **aud:** Magnitude of the sound. Hex number, ranges from 00 to 0a
- **tz:** Time zone offset for this device, measured in hours. Can be +ve or -ve. Can be used to adjust time displayed in UI to camera's local time.
- **rom:** Set to 1 if event was recorded in 'Record on motion mode', which is the default. Will be 0 if camera was in 24/7 continuous recording mode.

Also returns these, but not needed for Event viewing.

- **deviceid:** Device id, so in this example will just be the device id passed in as part of uid.
- **startval:** Raw data received from camera.

Example return

Two events are returned in the example below.


```
{
 "result": [
  {
   "startt": 1540567150000,
   "deviceid": 63,
   "startval": "{"tz":1,"t":"motion","smx":"07",
"id":1540567150,"hasvideo":0,"rom":1,"e":"start",
"max":"00000000000000000000000000000000
00000000000000000000000000000000000001
e00000000000000000050000000000000000
000000000000","aud":null}",
   "alert": 1540567150,
   "rom": 1,
   "smx": "07",
   "aud": null,
   "tz": 1,
   "type": "motion",
   "has_hdstill": 0,
   "endt": 1540567157091
  },
  {
   "startt": 1540567090000,
   "deviceid": 63,
   "startval":
"{"tz":1,"t":"motion","smx":"07","id":1540567090,"hasvide
o":0,
"rom":1,"e":"start","max":"000000010000000
00000000000001010000010000000000001
00000000000000001e0000000000000000
00800000000000000000000000010000","aud":null}",
   "alert": 1540567090,
```

	"rom": 1, "smx": "07", "aud": null, "tz": 1, "type": "motion", "has_hdstill": 0, "endt": 1540567097000 }] }

1.3.2 Get alert thumbs

Returns a thumb image for the specified alert

API	Returns
GET /alertthumb <LOGGERSERVER>/alertthumb/uidd/alertid/<AUTHTOKEN> e.g. <LOGGERSERVER>/alertthumb/12345.63/1540565930/ <AUTHTOKEN> **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo call above • **AUTHTOKEN**: is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63 • **alertid:** is the id of the alert, which is returned from /alert	This call just returns the image, there is no additional data. Call can be used from the web, so the image is displayed directly into a HTML element.

1.3.3 Play an event video in the web using HLS

API	Returns
GET /alert.m3u8 The video is in HLS format, which can easily be played in the web as the video is transferred using a number of small video files transferred over HTTPS. Because of the small file size it is very quick to scrub through the video as you don't have to wait for the full video to download to see video near the end of the recording. HLS videos can be played in Safari natively and also in Chrome and Firefox using HLS.js. We have used 2 options for using HLS.js.	This call returns a standard HLS .m3u8 file, which can be played in the web using HLS.js. The HLS player, will make subsequent calls to the logger to get the .ts video files.

The simplest is using React.js because there is a component available for this. https://www.npmjs.com/package/react-hls In older non-React code, we use https://github.com/video-dev/hls.js/ Use this URL to specify where to get the video from. <LOGGERSERVER>/alerturl.m3u8?uid=12345.63 &startt=1540567026&endt=1540567037&rom=1 &id=1540567030&buildmp4=0&token=<AUTHTOKEN > **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to. Get the value from /devices/viewerInfo call above. • **AUTHTOKEN:** is the value of authToken returned from /login • **uid:** uid.deviceid of the device that recorded the event. e.g. 12345.63 Get the values below from data returned by /alert, but need to convert from ms to s. • ***startt*:** Start time of the event in seconds • ***endt*:** End time of the event in seconds • ***rom*:** Usually set to '1'. Will be '1' if the camera was in record on motion mode when the event was recorded. • ***id*:** id of the alert. • **buildmp4**: Set to 0 for HLS video. If set to 1, it will return an MP4 file for download. Here's a useful site that lets you test the above URL: http://streambox.fr/mse/hls.js-0.7.4/demo/. Just form the URL path to the logger as above and copy to this site's custom URL field and the video should play.	

1.3.4 Tell camera to start streaming live video

Use /sendcameratask to send a 'livecommand' action request to the camera.

Need to call this every 30 seconds while you want the camera to keep streaming live video. If you stop calling it, the camera will stop streaming live video 60 seconds after the last call.

API	**Returns**
GET /sendcameratask	If sent OK, then returns 200 and result:

<LOGGERSERVER>/sendcameratask?uid=<uidd> &action=livecommand&token=<AUTHTOKEN> e.g. <LOGGERSERVER>/sendcameratask?uid=12345.63 &action=livecommand&token=<AUTHTOKEN> **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo • **AUTHTOKEN:** is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63	{"result":"request sent to camera"}

1.3.5 Tell camera to record an event video

(Needs App version 3.15 or above installed on Cloud Adapter)

This uses the same API call as above (to tell a camera to live stream) but uses a different command 'triggerevent' instead of 'livecommand'. It also needs to be called every 15 seconds rather than every 30 seconds.

Use /sendcameratask to send a 'triggerevent' action request to the camera.

Need to call this every 15 seconds while you want the camera to keep recoding video. If you stop calling it, the camera will stop recoding video about 20 seconds after the last call.

API	**Returns**
GET /sendcameratask <LOGGERSERVER>/sendcameratask?uid=<uidd> &action=triggerevent&token=<AUTHTOKEN> e.g. https://uswest2-logger-1561647198-i-04df43ecb589ae8c3.manything.com/sendcameratask?uid=1000244.3&action=triggerevent&token=OTg2MTY4OjE1NjUxOTE0ODA6MTIwMDowOkwsVyxB*WfhFRSphnhZP6k7zbcw6mWBPEHW31cOhNLzHhV9IBhe6uazt75fKb3yKHLCzRvU7w3GnC6EHb26fK_PaNT3DA07RyrYRKs8B7hNR4VeUVZSTNZyAJP6B2JkX8UTlaHTti7B70GO_DqbeZvES8s1yiAuSIOxgosXyYa7QuVa0844. **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo	If sent OK, then returns 200 and result: {"result":"request sent to camera"}

• **AUTHTOKEN:** is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63	

1.3.6 Get camera status

Use /cameras/status to find out current state of a camera. For example. This can be used to find out of the camera is streaming live video, and which Wowza server it is currently streaming to.

Can be used to get status for a single camera or a list of cameras.

API	**Returns**

GET /cameras/status
 	
GET <logger>/cameras/status[?uidd=A.B[&uidd=C.D[&...]]]
 	
Specifying Devices: You may provide a complete list of all uidds for which you want to receive data. You may specify 1 or more uidds with 1 or more "uidd" querystring params. If you provide none, you will get all resident devices to which you have access.
 	
Authorization: send token in Authorization header.
 	
Curl Example:
- curl -w '\n' -v "<LOGGERSERVER>/cameras/status/
?uidd=329475.56&uidd=329475.22
&uidd=943584.1" -H "Authorization: ManythingToken
<AUTHTOKEN>"
 	
Parameters
 	
• **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo
• **AUTHTOKEN:** is the value of authToken returned from /login
• **uidd**: You can specify 1, or n camera uidds
	Return values of particular interest for live video.
	
	• **live:** Will be true if the camera is currently streaming live video. Otherwise will be false.
	• **wowza**. If camera is streaming video, and live=true, then this value will be the Wowza server that the camera is streaming to. Warning: if live=false, then this value will not be valid, it will be set to 'wowza1.manything.com'
	
	{
	"result": {
	"12345": {
	"devices": {
	"46": {
	"uidd": "12345.46",
	"permissions": [
	"a"
	],
	"session": null,
	"live": true,
	"startt": 0,
	"lastchunkend": null,
	"endt": null,
	"lastactivity": 1520250794022,
	“lastthumb”: 1579791887,
	“laststill”: 1578999321,
	"stillmode": "unknown",
	"wowza": "uswest2-wowza-1549300082-i-05967ac0e1b1b5f87.manything.com",
	"device": true,
	"status": "offline",
	"lastalert": 0,
	"battery": null,
	"power": "unknown",
	"stalepower": 1,
	"facebookStreamId": null,
	"queue_mb": null,
	"queue_sec": null
	},
	"108": {
	"uidd": "12345.108",

	"permissions": ["a"], "session": null, "live": false, "startt": 0, "lastchunkend": null, "endt": null, "lastactivity": 1547745408000, "stillmode": "normal", "wowza": "wowza1.manything.com", "device": true, "status": "offline", "lastalert": 0, "battery": 100, "power": null, "stalepower": 1, "facebookStreamId": null, "queue_mb": null, "queue_sec": null } } } } } Errors: You should receive standard sensible HTTP status codes. Additionally you will receive an "errors" array indicating any devices you requested that are not resident on the logger, or that you are not authorized to view.

1.3.7 Tell camera to capture an image

Use /sendcameratask to send a 'capturethumb' or 'capturestill' action request to the camera.

The camera can capture two size of image. Thumbs are smaller images, with smaller file sizes and should be used where possible. If a larger image is required then capture a ‘still’ image.

Thumb image: 256 x 144 (IP cam) or 192 x 144 (smartphone cam)
Still image: 640 x 360 (IP cam) or 480 x 360 (smartphone cam)

Note that this API call only tells the camera to capture the image, it doesn’t return the image. To retrieve the captured image:

- First use the /camera/status API to get the time of the most recently image captured. These values are returned in lastthumb or laststill.
- Call the /getthumb or /getstill API calls to retrieve the image, passing in lastthumb or laststill values for the time parameter.

API	**Returns**
GET /sendcameratask <LOGGERSERVER>/sendcameratask?uid=<uidd> &action=capturethumb&token=<AUTHTOKEN> e.g.	If sent OK, then returns 200 and result: {"result":"request sent to camera"}

<LOGGERSERVER>/sendcameratask?uid=12345.63 &action=capturethumb&token=<AUTHTOKEN> **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo • **AUTHTOKEN:** is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63	

1.3.8 Get a thumb or still image

Returns a thumb or still image for the specified time

API	Returns
GET /getthumb or GET /getstill <LOGGERSERVER>/getthumb/uidd/time/<AUTHTOKEN> <LOGGERSERVER>/getstill/uidd/time/<AUTHTOKEN> e.g. <LOGGERSERVER>/getthumb/12345.63/1540565930/ <AUTHTOKEN> **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo call above • **AUTHTOKEN:** is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63 • **time:** is the time of the captured image. You can get the time of the most recently captured image by calling /cameras/status and using the lastthumb or laststill. This time is a Unix timestamp in seconds, e.g. 1583229327	This call just returns the image, there is no additional data. Call can be used from the web, so the image is displayed directly into a HTML element.

1.3.9 Get real time data

Rather than calling the API each time to get the camera status (/camera/status), it's more efficient to use this real time API call. For example, if you want to update the UI to show whether a camera is live or offline without polling that API.

As with cameras/status, it can be used to get data for a single camera or a list of cameras.

API	Returns

GET /cameras/realtime/live
 	
GET <logger>/cameras/realtime/live[?uidd=A.B[&uidd=C.D[&...]]]
 	
Specifying devices: You may provide a complete list of all uidds for which you want to receive data. You may specify 1 or more uidds with 1 or more "uidd" querystring params. If you provide none, you will get all resident devices to which you have access.
 	
Authorization: send token in Authorization header.
 	
Curl Example:
 	
curl -w '\n' -v "<LOGGERSERVER>/cameras/realtime/live?uidd=827029.20428&uidd=827029.20429" -H "Authorization: ManythingToken <AUTHTOKEN>"
 	
Parameters
 	
• **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo
 	
• **AUTHTOKEN:** is the value of authToken returned from /login
 	
• **uidd**: You can specify 1, or n camera uidds
	The connection will be kept open for up to 5 minutes. During that data will be returned for each update triggered by a camera. To keep getting data, you could disconnect and recall the API every 4 minutes. Or you could check for disconnection and remake the connection.
	
	Each set of data returned will be for a specific camera. You can see which camera it is in the uidd key.
	
	Other keys will be similar to those returned from /cameras/status.
	
	Particular fields of interest include:
	
	• **status**: Indicates the camera status e.g. 'live', 'novideo' or 'offline'. If camera is disconnected from the Adapter, then the value will be 'novideo'.
	
	An example of data returned for one camera update:
	
	Note that there are different types of data returned from this API. For data relating to camera status, the 'type' key will be set to 'liveSession'.
	
	"data":{
	"type":"liveSession",
	"data":{
	"session":1613158412,
	"live":true,
	"startt":1613741672313,
	"lastchunkend":1613741672311,
	"endt":null,
	"lastactivity":1613741844000,
	"lastthumb":1613741823,
	"laststill":1613741020,
	"stillmode":"video_only",
	"wowza":"euwest1-wowza-1611946206-i-0be12753c9156d4e3.manything.com",
	"uidd":"827029.20428",
	"device":true,
	"status":"live",
	"lastalert":1613741020,
	"battery":null,
	"power":null,
	"stalepower":1,
	"facebookStreamId":null,
	"queue_mb":null,
	"queue_sec":null
	}
	
	For data about new motion events / alerts, the 'type' key will be set to 'alert'.
	
	data: {

	"type": "alert",
	"data": {
	"uidd": "250843.27",
	"id": 1613745105,
	"rom": 1,
	"startt": 1613745105000,
	"endt": 1613745117000,
	"smx": "0a",
	"aud": null,
	"tz": 0,
	"type": "motion",
	"has_hdstill": 0
	}
	}
	
	See section 1.3.1 (Get event data for a device) for more information on events. The data returned above is the same format as historical event data, the main difference is that the above data is received for newly creation motion events.

1.4Wowza API

1.4.1 Play live video from Wowza

Live video can be played using the HLS protocol.

API	**Returns**
https://<WOWZA-SERVER>/manything/<STREAMNAME>/playlist.m3u8?userToken=<AUTHTOKEN> Use the above path in your video player to play the live video. e.g. https://euwest1-wowza-1557390870-i-00eee9526fed3c30d.manything.com/manything/827029.4_984533-nr/playlist.m3u8?userToken=<AUTHTOKEN> **Parameters** • **WOWZA-SERVER:** is the Wowza server that a specific device is assigned to. Get the value from /devices/viewerInfo. Note that cameras often move to a new Wowza server, so make sure you have a recent value. • **AUTHTOKEN:** is the value of authToken returned from /login	Returns the video stream As with general HLS spec, this will return an m3u8 file, which is an index containing video fragments to play. If you pass this API call to a video player that accepts a m3u8 file, it should manage everything for you i.e. getting latest m3u8 file, playing the ts file video fragments.

• **STREAMNAME:** is the live video stream name for the camera to be viewed, get the value from /devices/viewerInfo	

1.5Other APIs

1.5.1 PTZ Controls

From your viewer application, you may want to control the camera's PTZ settings. At the moment, we only support this for Hikvision cameras and we're currently working on this for Axis cameras.

The control is done using the /sendcameratask API.

The following actions can be used:

Move or zoom in the specified direction for 1 second, and then stop automatically:
`ptz_up`, `ptz_down`, `ptz_left`, `ptz_right`, `ptz_in`, `ptz_out`

Move or zoom in the specified direction until told to stop:
`ptzstart_up`, `ptzstart_down`, `ptzstart_left`, `ptzstart_right`, `ptzstart_in`, `ptzstart_out`

Stop moving:
`ptzstop`

Example for the `ptz_up` action

API	**Returns**
GET /sendcameratask <LOGGERSERVER>/sendcameratask?uid=<uidd> &action=ptz_up&token=<AUTHTOKEN> e.g. <LOGGERSERVER>/sendcameratask?uid=12345.63 &action=ptz_up&token=<AUTHTOKEN> **Parameters** • **LOGGERSERVER:** is the Logger server that a specific device is assigned to, get the value from /devices/viewerInfo • **AUTHTOKEN:** is the value of authToken returned from /login • **uidd:** uid.device id for a specific device e.g. 12345.63	If sent OK, then returns 200 and result: {"result":"request sent to camera"}