CamTrace

vidéosurveillance

Explications sur le fonctionnement du Player CamTrace | CamTrace Player explanations

Pré-requis : N/A

Version du document : v1.0

Date de création du document : 30/03/2022
Date de dernière modification du document : 30/03/2022 par KDU

Pour tous renseignements complémentaires :

www.camtrace.com

Table des matières

1 But du document

Explications sur le fonctionnement du Player CamTrace.

2 Principle of use of Camtrace Player (internal use)

2.1 Connection to the player

There are 2 websocket connections to establish to the server. One for video and statuses stream and one for commands and timeline information.

2.1.1 Getting the two players urls with camtrace http api

route : **/api/v1.2/users/login**
Player control url is under “services” / “player_control”.
Player video url is under “services” / “player_video”.
You will also need Camtrace stream protocol which is under "services" / “mobile” / “stream_protocol”.

Then you first have to open a new player control connection with camera id as a parameter.
For example, opening a player control connection for camera 1 :
ws://server_url/player_control?id=1
Websocket must be in text mode.

When control player connection is established, you will receive a player id. Then you will be able to open the connection to the record stream.
For example, opening a player video connection for player id 25 specifying the stream protocol “v1b” :
ws://server_url/player_control?id=25&accept=v1b
Websocket must be in binary mode.

Player connections code can be found in files players/Player.vue, players/records/Camera.vue, players/helpers/services.js and players/helpers/classes/SimpleService.js

2.2 Using player commands

Camtrace player sends for you the timeline to display, so you will have to initialize it correctly while sending the command “init” in the player control websocket.

See *players/records/Camera.vue*, *players/records/Overlay.vue*

You can send text commands for the player like “play forw”, “goto” in the player control connection websocket.

See *decoder/record/Control.js*

2.2.1 List of Player commands

Most commands take effect without generating a return answer through "in".

init Initializes the player for a given camera and range of time
---

out:"init <type> <cam> <begin> <end> <barlen> <itime>\0"
where
<type> is "regul" or "alarm" depending on the record type
<cam> is the internal camera id
<begin> begin UNIX-timestamp (# of seconds since jan-1-1970 at 0:00 UTC) of range
<end> end UNIX-timestamp of range
<barlen> width of the navigation bar (in pixels)
<itime> Javascript time (like UNIX but in ms) of the initial position wanted
(must be between <begin>*1000 and <end>*1000). WARNING: > 32 bits!

in: "span <begin> <end>\0"
"indexed <i>\0"
"bar regul\0"
"bar regul <pos> <regul bar contents>\0"
"bar alarm\0"
"bar alarm <pos> <alarm bar contents>\0"
"bar index <pos> <index bar contents>\0"
"init\0"
"rfreq <f>\0"
"width <w>\0"
"height <h>\0"
where
<begin> is the begin time as found by the player
<end> is the end time as found by the player
<pos> is the position the bar starts with (normally 0)
<* bar contents> is the bar contents (<barlen> positions max.)
there are 17 levels "A" -> "Q" for each position
if a position is indexed, then the letter is lowercase ("a" -> "q")
<i> is the number of indexed "slices"
<f> is the noted frequency of recording
<w> is the width in pixels of the image sent to the Viewer
<h> is the height in pixels of the image sent to the Viewer

At the same time, the Player control websocket receives the corresponding image and through OnStatus:

"pfreq <f>\0" (only when the value changes)
"play <state> <itime>\0" (before the image)
"image <pos> <itime> <len> <eperiod> <event>\0" (after the image)
where
<f> is the actual physical frequency
<state> state of the player ("forw", "back" or "stop")
<pos> position in the bar the image belongs to
<itime> JavaScript-timestamp (as UNIX-timestamp, but in ms)
<len> size > 0 of the image in bytes or error code as below
<eperiod> effective period of images sent (in ms)
<event> event number (for indexation)

The error code in place of <len> can have the following values:

0	corresponds to a temporarily missing image ("black" icon)
-1	corresponds to a complete absence of images in the selection ("noimage" icon)
-2	corresponds to an image loading error ("onerror" icon)

In case of an error, the "image ..." status is preceded by the corresponding icon instead of the image (type "I" packet as described in "Direct access to image streams" below)

play Starts playing images in the selected range (from the current position)

---

out:"play <what>\0"

where

<what> "forw" starts displaying in the given direction

"back"

"stop" stop displaying

"first" display thee requested image, then stop

"last"

"next"

"prev"

step Adjusts the image increment (number of "skipped" server images)

---

out:"step <n>\0"

where

<n> is the increment between:

images of a "JPEG" sequence: 1 by 1 (default), 2 by 2, etc

"I" images (key-frames) of an "MPG4" or "H264":

"I" by "I", 2 "I" by 2 "I" or 0 for all images (default)

goto Changes the current position within the selected time range

---

out:"goto <pos>\0"

where

<pos> is a position between 0 and <barlen>-1

"goto <epos>\0" is sent back to the Viewer

where

<epos> is the position effectively reached

time Changes the current time within the selected time range

---

out:"time <itime>\0"

reserved for external synchronisation between servers of a cluster

type Changes the type of recording whilst keeping the time range

---
out:"type <state> <itime>\0"
where
<type> is "regul" or "alarm" depending on the recording type wanted
<itime> Javascript time (like UNIX but in ms) of the initial position wanted

zoom Reduces the selected time range (zoom in)
---
out:"zoom <bpos> <epos>\0"
where
<bpos> position (in the bar) of the new time range beginning
<epos> position (in the bar) of the new time range end

in: "span <btime> <etime>\0"
<btime> new UNIX timestamp of range beginning
<etime> new UNIX timestamp of range ending
or
in: "span -1\0" if no image or syntax error

span Changes the selected time range
---
out:"span <what>\0"
where
<what> is the type of change wanted

"next" same time span, starting at the end of the current range
"prev" same time span, ending at the beginning of the current range
"back" zoom out: add same time span before AND after the current range
"time <len>" time range of <len> seconds starting modulo <len>
"hour <len> <tz>" same, but <len> is in hours and the
timezone <tz> in minutes is taken into account
"all" all images available for the given camera

mask Retrieve or save the indexation mask
---
out:"mask\0" returns "mask <mask>\0" over OnStatus
or
out:"mask <mask>" saves the mask <mask> (base64 encoded bitmap)

freq Changes the display frequency (unrelated to the recording frequency)
---
out:"freq <f>"
where
<f> is the desired display frequency in images per second

index Enable/disable the addition of "index" Status
---
out:"index <idx>\0" start or stop sending indexing data for each image

where
<idx> is "1" to enable, "0" to disable (default)

sync Toggles the player into synchronized mode
---
out:"sync <ratio>\0" enables or disables synchronized mode
where
<ratio> is the ratio wanted between current time and the time of images (decimal part accepted after a ".") or "0" to toggle back to non-synchronized mode

In this mode, images are displayed with strict following of the current time (possibly modified by <ratio>). It follows that some images can be skipped if the display rate is insufficient and also that the "black" icon can appear when there is nothing to display for a given time ("0" image length)

quit Quits the player
---
out:"quit\0" closes sockets and terminates the associated player on the server-side

2.3 Receiving video stream packets

Encoded video streams packets arrive on the player video socket. They are encapsulated in a specific Camtrace protocol and have to be demultiplexed before being decoded and/or displayed, depending of the video codec used by the camera. (h264, h265, mjpeg, MPEG4)

See *decoder/Decoder.js* and *decoder/live/Old.js*

You also receive status packets on the player video socket. They are mostly used to manage the player's current position on the timeline.

See *players/records/Overlay.vue*

Pour tous renseignements complémentaires :

www.camtrace.com