Chapter 1 - Appendix 2 – ATEAS API

1.1. Communication basics

It is possible to use a TCP or HTTP protocol based channel. Using the TCP protocol may be easier and may be supported by a wider range of external devices. The transmission, however, cannot be secured, nor is it possible to require authentication. Using the HTTP protocol, TLS can be enforced for maximum security and it is also possible to require authentication.

Using this API, the administration and camera servers are listening on the specified ports, whereas external devices are responsible for creating and maintaining connections.

The administration server listens on the API ports 8504 (TCP channel) and 80 (HTTP) which is default HTTP port of the administration server and may be changed in the configuration file. The camera server listens on the API ports 8505 (TCP channel) and 8080 (HTTP) which is default HTTP port of the camera server and may be changed in the configuration file.

The HTTP protocol uses standard HTTP response codes. Sending data must be performed with the HTTP POST method. The HTTP protocol may return data in JSON format.

Using the TCP protocol, unsolicited messages may be sent (e.g. system events). For this purpose, WebSocket protocol is used on HTTP protocol level. Data can have the JSON or XML format.

Text encoding can be configured for TCP protocol when opening the channel. HTTP protocol defaults to UTF8 encoding.

TIP

You can find some useful links to TCP and HTTP implementation examples on the homepage of your administration server.

1.2. ATEAS API of the administration server

1.2.1. External events

Receiving an external event

Protocol	API
TCP	[ATEAS EVENT (status) (objectid) (elementid)]
HTTP	POST /api-base/event DATA: status=(status)&objectid=(objectid)&elementid=(elementid)

Parameters:

Parameter	Value	Default	Description
status	START, STOP		Start or stop
objectid	1 – 10 000		Object number
elementid	1 – 99 999		Element number

Example:

Protocol	API
TCP	[ATEAS EVENT START 1 1]
HTTP	DATA: status=START&objectid=1&elementid=1

1.2.2. Video wall

<u>Switch content to video wall</u>

Protocol	API
TCP	[ATEAS VIDEOWALL (monitor) (submonitor) (serverid) (deviceid) (wallid) (meta)]
HTTP	POST /api-base/videowall DATA: monitor=(monitor)&submonitor=(submonitor)& serverid=(serverid)&deviceid=(deviceid)&wallid=(wallid)&meta=(meta)

Parameters:

Parameter	Value	Default	Description

monitor	1 – 192		Monitor number
submonitor	0 – 16	0	Submonitor number
serverid	1 – 9 999		Server number
deviceid	1 – 999		Device number
wallid	1 – 1000	1	Video wall number
meta	0 – 1	0	Metadata yes or no

Example:

Protocol	API
TCP	[ATEAS VIDEOWALL 1 1 1 10 1 0]
HTTP	DATA: monitor=1&submonitor=1&serverid=1&deviceid=10&wallid=1&meta=0

NOTE

For monitors of type 4, 9 or 16, a submonitor value must be passed.

NOTE

If both serverid and deviceid values are zero, the monitor will be turned off (video disappears and the default ATEAS logo shows up).

NOTE

If serverid is zero, a positive deviceid will be interpreted as a URL number of a web content link created by the administrator).

NOTE

For TCP protocol, if using the optional wallid parameter, the submonitor parameter must be passed as well to enable the server to parse the message.

1.2.3. License plates

LP list assignment

Protocol	API
TCP	
HTTP	POST /api-base/plate DATA: plate=(plate)&list=(list)

Parameters:

Parameter	Value	Default	Description
plate	(plate)		License plate
list	none, white, black, user1, user2		List assignment

Example:

Protocol	API
TCP	
HTTP	DATA: plate=2A56217&plate=white

JSON response:

```
{
    “Plate”: ”2A56217”,
    “List”: “white”
}
```

NOTE

License plates can be provided in both decorative and normalized form.

NOTE

The list parameter is optional. If omitted, the current list of the LP will be returned.

1.2.4. Face database

Adding a group of people

Protocol	API
TCP	
HTTP	POST /api-base/face/group/add DATA: name=(name)

Parameters:

Parameter	Value	Default	Description
name	(name)		Group name

Example:

Protocol	API
TCP	
HTTP	DATA: name=group

JSON response:

```
{
    “result”: ”ok”,
    "id": "2"
    "name": "group"
}
```

POZNÁMKA

Groups with duplicate names disregarding any case differences cannot be created.

<u>Removing a group of people</u>

Protocol	API
TCP	
HTTP	POST /api-base/face/group/remove DATA: id=(id)

Parameters:

Parameter	Value	Default	Description
id	(id)		Group number

Example:

Protocol	API
TCP	
HTTP	DATA: id=2

JSON response:

```
{
    "result": "ok"
}
```

Renaming a group of people

Protocol	API
TCP	
HTTP	POST /api-base/face/group/name DATA: id=(id)&name=(name)

Parameters:

Parameter	Value	Default	Description
id	(id)		Group number
name	(name)		New group name

Example:

Protocol	API
TCP	
HTTP	DATA: id=2&name=group

JSON response:

```
{
    “result”: ”ok”
}
```

Adding a person

Protocol	API
TCP	

HTTP	POST /api-base/face/person/add DATA: name=(name)&group=(group)&uuid=(uuid)

Parameters:

Parameter	Value	Default	Description
name	(name)		Person name
group	(id)	1	Group number
uuid	(uuid)		Custom identifier

Example:

Protocol	API
TCP	
HTTP	DATA: name=person&group=&uuid=

JSON response:

```
{
    “result”: ”ok”,
    “id”: ”1”,
    “name”: ”person”,
    “uuid”: ””
}
```

POZNÁMKA

The uuid parameter can be used as an external identifier e.g. in an access system. It will also be part of the face recognition event.

Removing a person

Protocol	API
TCP	
HTTP	POST /api-base/face/person/remove DATA: id=(id)

Parameters:

Parameter	Value	Default	Description
id	(id)		Person number

Example:

Protocol	API
TCP	
HTTP	DATA: id=1

JSON response:

```
{
    “result”: ”ok”
}
```

<u>Edit a person</u>

Protocol	API
TCP	
HTTP	POST /api-base/face/person/update DATA: id=(id)&name=(name)&group=(group)&uuid=(uuid)

Parameters:

Parameter	Value	Default	Description
id	(id)		Person number
name	(name)		New person name
group	(id)		Group number
uuid	(uuid)		Custom identifier

Example:

Protocol	API
TCP	
HTTP	DATA: id=1&name=person&group=2&uuid=

JSON response:

```
{
    “result”: ”ok”,
    “name”: ”person”,
    “uuid”: ””
}
```

Adding a face

Protocol	API
TCP	
HTTP	POST /api-base/face/person/image DATA: person=(person)&serverid=(serverid)&index=(index)&data=(data)

Parameters:

Parameter	Value	Default	Description

person	(id)		Person number
serverid	(id)		Server number for analytics
index	0 – 9	0	Image index
data	(uuid)		Image data

Example:

Protocol	API
TCP	
HTTP	DATA: person=1&serverid=1&index=0&data=

JSON response:

```
{
    “result”: ”ok”,
    “image”: ”/9j/4AA…”
}
```

POZNÁMKA

The data parameter is expected to contain a base64 encoded jpeg or bmp image with a 24 or 32-bit pixel color. If the data parameter is empty, the image will be removed.

POZNÁMKA

After a face image has been successfully saved in the database, the response contains the canonical face representation as a base64 encoded jpeg image data blob.

1.2.5. Event notifications

Subscription

Protocol	API
TCP	automatic
HTTP	

Event start XML

```
<?xml version="1.0" encoding="utf-8"?>
<ateas>
    <event>
        <id>1</id>
        <imageid>1</imageid>
        <level>1</level>
        <server>
            <id>1</id>
            <name>Server 1</name>
        </server>
        <camera>
            <id>1</id>
            <name>Camera 1</name>
        </camera>
        <source>
            <id>1</id>
        </source>
        <datetime>
            <utcstamp>128989433710312500</utcstamp>
            <localvalue>1.7.2023 9:05:51</localvalue>
        </datetime>
        <data></data>
        <dataex></dataex>
        <uuid></uuid>
        <videoobject>
            <rectangle>20 20 200 200</rectangle>
        </videoobject>
    </event>
</ateas>
```

Event stop XML

```
<?xml version="1.0" encoding="utf-8"?>
<ateas>
    <eventstop>
        <id>1</id>
        <datetime>
            <utcstamp>128989433710312500</utcstamp>
            <localvalue>1.7.2023 9:05:51</localvalue>
        </datetime>
        <data></data>
    </eventstop>
</ateas>
```

NOTE

Events are centralized and collected from all camera servers.

NOTE

The image number associates the event with a file name that might be uploaded to an FTP server.

NOTE

The source number uniquely identifies the event type, which can be observed using the test tool.

Source ID examples:

1 – camera motion detection
2 – device unavailable
3 – alarm input, data contains the input number
10 – 14 – vehicle LP detection, data contains the LP in decorative form
32 – video quality loss, data contains the required frame rate level
40 – server based motion detection

51 – 100 – custom events

110 – manual recording event

111 – 130 – Onvif events, data may contain additional information

131 – 150 – complex events

151 – 200 – custom events

201 – 250 – analytical events

NOTE

The UTC timestamp indicates the absolute time not affected by the time zone or daylight time changes. It is expressed as the number of 100-nanosecond intervals elapsed since 1.1.1601 UTC.

1.2.6. User notifications

Subscription

Protocol	API
TCP	automatic
HTTP	

User login XML

```
<?xml version="1.0" encoding="utf-8"?>
<ateas>
    <user>
        <id>10</id>
        <name>tester</name>
        <action>login</action>
        <datetime>
            <utcstamp>128989433710312500</utcstamp>
            <localvalue>22.7.2023 15:05:23</localvalue>
        </datetime>
    </user>
</ateas>
```

NOTE

During logout, the action parameter has the value of logout.

NOTE

The UTC timestamp indicates the absolute time not affected by the time zone or daylight time changes. It is expressed as the number of 100-nanosecond intervals elapsed since 1.1.1601 UTC.

1.3. ATEAS API of the camera server

1.3.1. External events

Receiving an external event

Protocol	API
TCP	[ATEAS EVENT (status) (deviceid) (code) (data)]
HTTP	

Parameters:

Parameter	Value	Default	Description
status	START, STOP		Start or stop
deviceid	1 – 999		Device number
code	(as configured)		Custom event name
data	(max. 200 characters)		Additional data

Příklad:

Protocol	API

TCP	[ATEAS EVENT START 1 TEMPERATURE 76]
HTTP	

NOTE

The event code must correspond with an existing name in the camera administration section.

NOTE

The event can be ended explicitly or by configuring a maximum event duration interval.

1.3.2. Metadata

Inserting metadata

Protocol	API
TCP	[ATEAS META (deviceid) (timestamp) (code) (data)]
HTTP	

Parameters:

Parameter	Value	Default	Description
deviceid	1 – 999		Device number
timestamp	0 – N		UTC timestamp
code	(as configured)		Custom event name
data	(max. 200 characters)		Metadata

Example:

Protocol	API

TCP	[ATEAS META 1 0 SCAN AB512459]
HTTP	

NOTE

The UTC timestamp indicates the absolute time not affected by the time zone or daylight time changes. It is expressed as the number of 100-nanosecond intervals elapsed since 1.1.1601 UTC.

NOTE

The timestamp may have the value of 0. In such a case the timestamp will be determined by the server. Using user defined timestamps helps for offline data uploads. Timestamps earlier than 30 days in the past or later than 1 minute in the future are not accepted, based on server time.

NOTE

The event code must correspond with an existing name in the camera administration section.

1.4. Parameterized application launch

ATEAS Security applications can be launched with additional parameters that are passed to the application executable while starting. In Windows these parameters can be added under the service settings. All existing parameters are described below.

1.4.1. Administration server

Parameter	Values	Meaning	Note
-ssl	password	Certificate password	Necessary to use when the PFX certificate is password protected.

1.4.2. Camera server

Parameter	Values	Meaning	Note

-ssl	password	Certificate password	Necessary to use when the PFX certificate is password protected.
-loglevel	0 - 1	Log level setting	Using a positive value activates logging of the record buffer level in the log subfolder.