Call Recorder Apresa API

Application Programming Interface

Apresa version 13.3.0

Table of contents

1 Introduction

The Apresa Call Recorder can be accessed externally using the Apresa API. The communication in the API is done using HTTP or HTTPS. To use the API, the application sends HTTP (POST/GET)

requests to the Apresa. The Apresa answers back with data, and in addition might perform a requested action.

The API is devided into two parts, the Client Interface and the Access Interface. Each is discussed separately in the following two chapters.

2 The Client Interface

The Apresa Client interface is accessed by sending HTTP POST or HTTP GET requests, using the following URL:

http://IP-address-Apresa/client.php

For example, if the Apresa has the IP address 192.168.32.2, then use:

http://192.168.32.2/client.php

2.1 General request and reply structure

The following parameters are required for each request:

Parameter	Value
`username`	User name defined in the User List on the Apresa server (Options → Users)
`password`	Password of the user account.

The reply is a list of values each on one line:

```
Param1: Value1
Param2: Value2
```

2.2 Information retrieval

For information retrieval, use HTTP GET requests.

2.2.1 List of active calls

Parameter	Value
`info`	`AllActiveCalls`

Returns the full list of active calls as far as permitted to the user. For example:

```
RequestedInfo: AllActiveCalls
Know: 6217()
List: [{id:6217,remoteid:"20130201_103244_001_07",startdate:1359711164,....
ServerTime: 1359711165
```

List: This is the list of calls. It is returned in a JSON-like format, but key names are currently not quoted. The following fields are used per call:

remoteid:	This is the ID to be used when performing actions
remote/local:	The remote/local telephone number or SIP identifier
lname/rname:	The name attached to the remote/local telephone number
direction:	0: incoming, 1: outgoing, 2: internal

remote2/local2: For VoIP, the IP address
linenr: For TDM/Analog recordings, the line number

2.2.2 Active call of user

Parameter	Value
`info`	`ActiveCalls`

Returns the (list of) active call(s) for the telephone(s) of the user. For example:

```
RequestedInfo: ActiveCalls
Know: 6218()
List: [{id:20130201_103504_001_08,oid:"101",on:"John",out:1,sd:1359711304,a:1}]
```

List: This is the list of calls. It is returned in a JSON-like format, but key names are currently not quoted. The following fields are used per call:

id: This is the ID to be used when performing actions
oid: The caller id (telephone number) of the other side
on: The name of the other side
a: 0: not recording, 1: recording

2.2.3 Recorded audio file

Parameter	Value
`info`	`AudioFile`
`id`	The id of the call

Outputs the recording as a wav file.

Parameter	Value
`info`	`RTCall`
`remid`	The id of the call

This function is for live monitoring of an active call. Monitoring must be enabled in the recording settings of the server. This function outputs: informational data about the recording (format and number of channels), followed by a zero byte, followed by the actual audio content, as it becomes available. Keep the connection open to receive all data. There will always be a small delay between the audio stream and the actual call.

2.3 Actions on a call

Use a HTTP POST request for actions. The following parameter always needs to be specified:

Parameter	Value
`id`	The id of the call
`act`	The action to perform. Possible values: start, stop, keep, silence, sound, and notes.

When successful, the reply includes:

```
Result: ActOk
```

If it fails, the reply will probably contain an error message, for example:

```
Error: Permission denied
```

The following is a description of possible actions:

The action `start`
Starts recording of a call. This action is only applicable if the call was configured to be recorded on demand, because otherwise it is recorded by default. The new recording will have a new id.

The action `stop`
Stops recording of a call that was started earlier.

The action `keep`
Notifies that this call must be stored. This action is applicable when the call was configured to be stored on demand.

The action `delete`
Stops the recording and deletes it.

The action `silence`
Silence the recording of a call, from this point in time.

The action `sound`
Stops the silencing, and from now on, record again the sound of the call.

Making notes

Parameter	Value
`act`	`notes` or `notes2`
`notes`	The text to be stored in the notes or secondary notes field

Changes the notes or secondary notes attached to a recording.

Assigning categories

Parameter	Value
`act`	`categorize`
`category`	The category number

3 The Access Interface

The Access interface is accessed by sending HTTP POST requests, using the following URL:

http://IP-address-Apresa/access.php

For example, if the Apresa has the IP address 192.168.32.2, then use:

http://192.168.32.2/access.php

3.1 General request and reply structure

Parameter	Value
`auth`	A. API authorization code
`username / password`	B. Credentials of a user account in Apresa
`act`	The name of the action to be performed (for example: `startrec`)

Unless noted otherwise, the API authorization code (`auth`) needs to be sent as a parameter to get access. Some functions instead allow or require the use of a `username` and `password`, and this is noted in the description of those functions, and if this is done, then the `auth` parameter should not be sent. The authorization code must be configured in the Apresa web interface, as follows: In the Options, System settings, click "Advanced settings" in the upper-right corner, and then fill in the API authorization code.

Unless otherwise noted, the reply is one line of text. The line of text consists of a number, a space, and then optionally extra text. If the action was successful, the number is 0, otherwise it is an error code. For example:

```
0 Action completed successfully
```

3.2 Query availability

Parameter	Value
`act`	`ping`

No action is performed. This can be used to check if API access is working.

3.3 Recording starting and stopping

3.3.1 Start recording

Parameter	Value
`act`	`startrec`
`localnum / linenr`	local telephone number of the employee (for VoIP), or the line number in Apresa where the call is active (for TDM)
`data`	any data that must be saved
`datadest`	The destination of `data`. Possible values: • `notes`: The notes field • `notes2`: The secondary notes field • `extdata`: Data field that is included in recording data export. (default)
`data2`	any data that must be saved
`datadest2`	The destination of `data2`. See `datadest` above. The default is `notes2`.
`exp`	0 or 1 (optional)

A call can be specified in two ways, using the local telephone number, or the line number. If successful, recording will start for the specified call, and the data will be saved in the database for later retrieval. If export (`exp`) is enabled (1), and also enabled and configured on the server, then the "Recording data export" defined below, will be performed after the call is completed or when recording is stopped.

Defined error codes:
1: No call on this number (or not in allowed list)
2: Verification of local telephone number failed, if this is required. (For example: The employee has not yet entered his or her telephone number on the telephone for verification)
3: Recording already started

On the second line, it returns the ID of the call, for example:

```
0 Recording started
20130315_122438_o7932
```

3.3.2 Stop recording

Parameter	Value
`act`	`stoprec`
`localnum / remotenum / linenr / id / nci / email / usr`	specifies the call, based on phone number, line number (non-VoIP), the ID of the call, or the protocol call ID (nci), or the email address or username of a user account
`localnumdifmax / remotenumdifmax`	the number of extra characters allowed in the local or remote phone number when checking for a match (default is zero)

A call can be specified in multiple ways. If an active call is found, the recording if it was started on demand, will be stopped.

Defined error codes:
1: No call with this property
3: Recording is not started

3.3.3 Delete a recording

Parameter	Value
`act`	`delete`
`localnum / remotenum / linenr / id / nci / email / usr`	Specification of the call
`cache`	if set to 1, and no matching call is found, the request will be cached and retried on new calls, until a matching call is found, or the action expires
`expires`	when cached, the action will be disregarded after the specified number of seconds (optional parameter, otherwise the default expiration will be used)

If an active call with the specification is found, the recording is stopped and discarded. This action is currently available only for VoIP calls.

3.3.4 Delete and restart a recording

Parameter	Value
`Act`	`delrestart`
`localnum / remotenum / linenr / id / nci / email / usr`	Specification of the call

If an active call with the specification is found, the recording is stopped and discarded, but then a new recording is started for the remainder of the call. If recording on demand is on, then only a listing is created, but recording is not yet started. This action is currently available only for VoIP calls.

3.4 Store on demand

Parameter	Value
`act`	`keep`
`localnum / remotenum / linenr / id / nci / email / usr`	specifies the call, based on phone number, line number, the ID of the call, or the protocol call ID, or the email address or username of a user account
`localnumdifmax / remotenumdifmax`	the number of extra characters allowed in the local or remote phone number when checking for a match (default is zero)
`cache`	if set to 1, and no matching call is found, the request will be cached and retried on new calls, until a matching call is found, or a time-out is reached (2 hours).

This command instructs Apresa to keep the recording of the specified call. If the store on demand option is on, recordings are deleted at the end of the call, unless the keep command is given.

When specifying a call using the email address or username of a user account, all the phones of the user are checked against both the local and remote ID, and if any exact match is found, the keep command is applied to the call. It allows multiple alternative telephone numbers or IDs to be checked at once. The difmax settings are ignored in this case.

3.5 Silencing

Parameter	Value
`act`	`silence` or `sound`
`localnum / remotenum / linenr / id / nci / email / usr`	specifies the call, based on phone number, line number, the ID of the call, or the protocol call ID

localnumdifmax / remotenumdifmax	the number of extra characters allowed in the local or remote phone number when checking for a match (default is zero)

These two actions can be used to silence the recording (silence), and to stop the silencing of the recording (sound). The call to be silenced can be identified in multiple ways.

Defined error codes:
1: No active call with this property
101: Missing parameter to identify call
102: The action could not be performed due to an error

3.6 Recording state monitoring

Parameter	Value
act	isactive
id	the ID of the call

This function checks if a call is being recorded and is still active. If the call, specified by the id, is active and recorded, it returns zero, otherwise it returns an error code.

Defined error codes:
1: No active call with this ID
3: Recording is not started

In addition, on the second line, it returns the duration. For example:
```
0 Call is active
34
```
In this example, the call is active for 34 seconds.

Parameter	Value
act	isready
id	the ID of the call

This function checks if a recording has finished and is added to the call listing. If so, it returns zero, otherwise it retuns an error code.
Authentication can be done with ‘auth’, or with ‘username’ and ‘password’.

Defined error codes:
1: No call found with this ID or no permission to list this call
2: Call is still active
3: Call has finished and the recording is converting

3.7 Retrieve call listing

Parameter	Value
act	calllistget
username / password	User name and password as defined in the User List on the Apresa server.

`count`	the number of records to return (if available), by default 10
`id`	Optionally specifies the unique identifier of the call to search for
`fd / ld`	Optional first and last search date (and time). Format: yyyy-mm-dd hh:mm:ss, the time part is optional, if missing 00:00:00 is assumed.
`groupinfo`	Set to 1 to output information about the groups and tenant to which this call belongs. This only works if the user is an administrator.

This function returns the latest calls (sorted on start date in reverse order), that match the search parameters. Only finished calls are returned, and in this context, a call is finished only when its audio conversion has also completed. This function is *not* accessed with an authorization code (so do not sent the `auth` parameter), but with the credentials of a user account, and the call list is filtered based on the list permission of this user.

The output is in JSON format, and consists of an array of the call records, with the following fields:

- **id:** this identifier can be used with other API calls (to get the audio recording)
- **start:** the start time and date of the recording (call) in the format: yyyy-mm-dd hh:mm:ss
- **duration:** length in time from beginning to end measured in seconds, the recording can possibly be shorter
- **local:** the telephone number or SIP ID of the local party, or if both parties are local/remote, the initiator of the call
- **remote:** the telephone number or SIP ID of the remote party, or if both parties are local/remote, the receiver of the call
- **lname:** the name associated with the detected local party
- **rname:** the name associated with the detected remote party
- **direction:** 0: incoming (remote calls local), 1: outgoing (local calls remote), 2: internal (local calls local), 3: external (remote calls remote)
- **tenant:** the name of the tenant if the call is assigned to a tenant (if groupinfo is 1)
- **groups:** the names of the groups (comma-separated) of the users that have a phone that matches with this call (if groupinfo is 1)
- **category:** the category number, or 0 if no category is assigned to the recording
- **catname:** name of the category
- **notes - notes5:** the notes fields
- **data1 – data5:** data fields
- **transcript:** this field is only present if a transcript is available and the user has permission

The fields local name, remote name, category, notes, and data fields are only included in the output if the columns are enabled in the display settings or the user is an administrator.

3.8 Retrieve audio recording

Parameter	Value
`act`	`audioget`

`username / password`	User name and password as defined in the User List on the Apresa server.
`id`	the id of the recording

Outputs the bytes of the audio recording file. The user account needs download permission for the requested call. The `auth` parameter should not be sent.

3.9 User and group management

After using these functions for user and group management, use the reconfig command to apply the new settings to the recording engine.

Parameter	Value
`act`	`useradd, useredit`
`usr`	Username of the user account
`name`	The display name of the user
`groups`	Comma-separated list of groups that the new user will be a member of
`pwd`	Password for the new user account, when using local logon.
`logonmethod`	0 = local logon, 1=LDAP, 2=external identity provider
`enabled`	0 = user account not enabled, 1= user account enabled
`email`	Email address of the new user account
`phones`	Comma-separated list of telephone numbers (or IDs) belonging to the new user account

When editing a user account, the user account to edit is matched based on the specified username (usr). Besides the username, only send the properties that need change.

Parameter	Value
`act`	`userdel`
`usr`	Username of the user account to delete

Parameter	Value
`act`	`groupadd, groupedit`
`name`	The group name
`newname`	The new group name (optional) when editing a group
`tenant`	If set to 1 the group becomes a tenant, and if set 0, it becomes a regular group. If an existing tenant group is edited to become a regular group, any recordings are unassigned from the tenant (but not deleted) and its settings deleted.
`supergroup`	The name of the parent group

MaxDiskUsage	Maximum disk usage for a tenant (MB)

Parameter	Value
act	groupdel
name	The group name
DeleteData	If set to 1, all tenant data is deleted (including recordings and user accounts), if the group is a tenant. If this option is not set, and the group is a tenant, then the group deletion is cancelled.
DelFromBackup	If set to 1, and the group is a tenant, then the data from the tenant in the backup will also deleted – this can be a long process.
MaxWait	The maximum number of seconds to wait for the deletion of tenant data to complete. The default is 5. If the time-out is reached, an error is returned, but the deletion process itself will proceed as before.

Parameter	Value
act	groupmembers
name	The name of the group from which to retrieve information

The output is a list of users who are member of the group with their name and username in JSON format.

Parameter	Value
act	userlist

The output is a list in JSON format of users with their name, username, tenant name (if applicable), list of phones that belong to the user (only caller ID phones are listed, not IP address or digital line phones), and a list of groups of which the user is a member. This function can be used with the auth parameter for a full list, or by a user for a possibly filtered list (showing only the users in the same tenant). To use this function as a user, the user needs to be an administrator (full, level 2, or level 3), have the edit user accounts permission, or a tenant administrator with access to the audit trail.

Parameter	Value
act	grouplist
tenant	If set to 1, only tenant groups are listed.

The output is a list of group or tenant names in JSON format.

3.10 Retrieve user configuration

Parameter	Value
act	userconfget
usr	the username

This function checks whether various recording actions (recording on demand, etc.) are possible or enabled for the specified user. Example output:

```
0 OK
StoreOnDemand: 0
RecordingOnDemand: -1
SilenceOnDemand: 1
DeleteOnDemand: 0
User: tom
```

The value 0 means the function is disabled, 1 means enabled, and -1 means it is unknown. The function takes into account what has been the defined at the user level, and at the global level.

Defined error codes:
1: User not found

3.11 Recording filter

The telephone numbers in the recording filter for SIP calls can be configured using the API.

Parameter	Value
act	sipfilteradd
sipid	The SIP ID (or telephone number) to add to the SIP filter

Parameter	Value
act	sipfilterdel
sipid	The SIP ID (or telephone number) to remove from the SIP filter

Defined error codes:
1: Already in list (when adding), or not in list (when deleting)
2: SIP Filter not enabled. The SIP Filter must already be set to either positive or negative in the Recording options. This cannot be changed using this API.

Parameter	Value
act	sipfilterget

This action outputs first a line with 0 OK, then a line with the current SIP filter type (positive / negative), and then on a separate line each SIP ID in the filter.

3.12 Apply new configuration

Parameter	Value
act	reconfig

When changes are made with the user and group management functions in this API, or using direct database access, then use this function afterwards to recalculate and apply the new settings to the recording engine.

3.13 Adding a recording

Parameter	Value
act	addrec
auth	A. authorization key
username	B. User name as defined in the User List on the Apresa server.
password	B. The password of the user account.
imei	Mobile IMEI (optional)
recording	audio file that is uploaded
local, remote	telephone numbers or IDs
simplifyid	By default IDs of the form “user@host” are simplified to “user”. If this option is set to zero, then this is not done.
localname, remotename	the name associated with the local and remote telephone number
notes, notes2, ..., notes5	the notes fields (optional)
data1,…, data5	optional data fields
category	0 = no category, 1 = the first category (optional)
direction	0 = incoming, 1 = outgoing, 2 = internal
startdate	format: yyyy-mm-dd hh:mm:ss
duration	in seconds
missed	1 = the call was not answered, no recording made (optional)
tenant	name of the tenant to whom the recording will be assigned (optional). However, if the upload is done by a tenant user, then this parameter will be ignored, and it will be assigned instead to the tenant this user belongs to.
recorderid, recordername	optionally specifies the recorder where the call was recorded. Otherwise the current Apresa server is seen as the recorder. Setting a custom recorder is available only when adding a recording using the global authorization code (not as user). The recorderid must be unique. The name can be freely chosen, and is used to create a recorder entry, if the recorderid is not known yet. If no name is specified, the id is used as name. The recorder can be used during search, and for user call access.

This actions adds a new recording to the database. The specified local telephone number (or the IMEI) must match one of the External Phones (Options menu). If the external phones list links an IMEI to a local phone number, then that phone number is used, instead of the local id specified using the API function. The number of external phones is determined by the number of external phone licenses (also called Mobile recording licenses).

This action can be performed using either A. the global authorization key (auth), or B. using the credentials of a user account (username and password). The user must have permission to add recordings. The local telephone number has to be one of the user phones.

On success, the second line will contain the id of the newly added recording. After this API function returns, the recording might be converted in the background, and will only appear in the call list after this has completed.

Note: Use HTTPS when uploading over a public network.

Defined error codes:
1: Not enough data specified
2: Error in file upload
4: Licensing error (Local ID or IMEI is unlicensed)
5: Error in handling of uploaded recording file
6: Local ID does not belong to user
7: Local ID of this IMEI does not belong to user
501: Login failed (when the username method is used)
502: The user does not have permission to add recordings

3.14 Adding video to a recording

Parameter	Value
`act`	`addvideo`
`id`	the ID of the call
`file`	the file containing the video or screen recording

This action can be used to add a screen or video recording. To allow playback in the web interface, it is recommended to use MP4 format. When adding multiple videos to one recording (for example multiple screens), upload these in a single ZIP file.

Authentication can be done with ‘auth’, or with ‘username’ and ‘password’.

3.15 Data editing and import

3.15.1 Editing active calls

Parameter	Value
`act`	`dataset`
`localnum / remotenum / linenr / id / nci / email / usr`	specifies the call, based on phone number, line number, the ID of the call, or the protocol call ID, or the email address or username of a user account
`localnumdifmax / remotenumdifmax`	the number of extra characters allowed in the local or remote phone number when checking for a match (default is zero)

`target`	what to change, can be: `local`, `notes`, `notes2`, ..., `notes5, data1 … data5`
`value`	the new value to write (the new Local ID, or the new notes)
`cache`	if set to 1, and no matching call is found, the request will be cached and retried on new calls, until a matching call is found, or the action expires
`expires`	when cached, the action will be disregarded after the specified number of seconds (optional parameter, otherwise the default expiration will be used)

This action can be used to change properties of an active call, or if caching is used, of a future call. The call to be changed can be identified in multiple ways. If multiple identifiers are specified, all need to match the call.

Defined error codes:
1: No active call with this property
101: Missing parameter to identify call
102: Invalid value for target

3.15.2 Editing calls based on ID

Parameter	Value
`act`	`editcall`
`id`	the ID of the call
`notes`, `notes2`, ..., `notes5`, `data1` … `data5`, `transcription`	the new value to write for the specified field

This action can be used to fill in one or more of the note fields of an active or finished call, or the transcription of a finished call.

Defined error codes:
1: No call with this ID
101: Parameter missing

Parameter	Value
`act`	`annotatecall`
`id`	the ID of the call
`count`	the number of annotations
`text1`, `text2, ...`	the text for the first, second, and other annotations

`time1, time2, ...`	the time in seconds measured from the start of the recording (may be a decimal number)

This action replaces (deletes) all existing annotations for the specified finished call, and adds to the call the new annotations at the specified time locations.

Defined error codes:
1: No call with this ID
101: Parameter missing

3.15.3 Import data based on matching call information

Parameter	Value
`act`	`dataimport`
`csvdata`	Uploaded file with CSV data
`timedifmax`	Maximum difference of the start time and duration, measured in seconds, default: 5
`numdifmax`	Number of allowed extra unmatched characters in local and remote number, default: 2
`datadest`	The destination. Possible values: • `notes`, `notes2`, ..., `notes5`: One of the notes fields (`notes2` is the default) • `data1`,…, `data5`: Data fields • `extdata`: Data field that is included in recording data export.
`verbose`	Set to 1 to let it output diagnostic information

This function is used to import additional data into the Notes field of existing recordings. For each line in the imported field, a corresponding call is searched, and if it matches sufficiently, the data is imported for this call. The CSV file is expected to have the following format: Fields are comma separated, and optionally quoted using double quotes. A CSV line should contain 5 fields:

1. start time
2. stop time
3. remote telephone number
4. local telephone number
5. custom data

The start and stop time have the following format: `yyyy-mm-dd hh:mm:ss.mmm` or `yyyy-mm-dd hh:mm:ss`. If the specified stop time is empty, then only the start time is used for time matching. The specified time and duration must not differ too much from the time as recorded by Apresa (the maximum is `timedifmax`). Therefore, it is important that both computers have synchronized time (using NTP). If the remote or local telephone number is not specified, it is not checked. If it is specified, then the specified telephone number in CSV must be found inside the telephone number that is recorded by Apresa, or the other way around. The extra characters found must not exceed `numdifmax`. If several acceptable matches are found, then the best match is chosen, that has the least difference. The custom data is stored in the Apresa database, in the field that was selected (using `datadest`), for example in the Notes field.

Defined error codes:
1: No csvdata file was uploaded
2: The upload failed

If the import completed, Apresa reports the number of records that were imported, and how many failed. For example:

```
0 Imported OK:372 fail:2
```

If verbose is set to 1, additional lines will be outputted after this line, containing diagnostic information about the matching process.

3.15.4 Import notes based on local phone number per day

Parameter	Value
`act`	`importnotes`
`csvdata`	Uploaded file with CSV data
`date`	Only call records of this day will be checked. Format: yyyy-mm-dd
`target`	The destination. Possible values: • `notes`: The notes field • `notes2`: The second notes field (default)

Recorded calls by specified agents on the specified day are marked with a specified text in the notes field. The CSV file is expected to have the following format: Fields are comma separated, and optionally quoted using double quotes. A CSV line should contain 2 fields:

1. value to write in the notes field
2. local telephone number (or ID) to check for

The first line is expected to be a header, and is ignored. If special non-ASCII characters are used, then the CSV file should be in UTF-8 encoding.

If the import completed, Apresa reports the number of records that were set of all the records of the specified day.

```
0 OK. Set 5 of 11
```

3.15.5 Translate phone numbers

Parameter	Value
`act`	`phonemap`
`map`	List of phone number mappings.

This API function is for converting phone numbers, or more generally, for converting Local and Remote IDs. If a certain ID is detected, it is replaced by another ID. The call will be handled as if the new ID had been detected, so the new ID will be used for the recording filter, for user access filtering, and it will be displayed in the call listing in the web interface. This can be useful in the case that the detected Local ID would otherwise have a changing dynamic mapping to a user (free-seating).

The defined `map` is a list of phone mappings with the following format. Each entry is separated by a

comma or by a newline. Each entry has the format source=destination. If the source phone number is detected by the recording engine, it will be replaced with the destination phone number. Instead of a phone number, it can be a SIP identifier of the format user@host or phonenumber@host (for SIP the full SIP ID needs to be specified), or another type of ID depending on the recorded protocol.

This API call defines the new current phone mapping, replacing all previous phone mappings of this kind. The mapping will stay in effect until a new mapping is defined. The phone mapping will effect only new calls, and will not be applied to any finished call. Currently the setting applies only to VoIP calls.

3.15.6 Establish phone number to notes mapping

Parameter	Value
`act`	`idmap`
`csvdata`	Uploaded file with CSV data
`target`	The destination. Possible values: • `notes`: The notes field • `notes2`: The secondary notes field (default)
`replace`	Possible values: • `0`: edit existing mappings or add if new • `1`: completely replace the existing mappings (default)

This function establishes a mapping between a local phone number (or ID) and a note. When a call is made with the specified local phone number, the notes field is filled with the specified data. This does not apply to existing calls, but only to new calls. It will continue to be applied until a new mapping is specified. The CSV file is expected to have the following format: Fields are comma separated, and optionally quoted using double quotes. A CSV line should contain 2 fields:

1. value to write in the notes field
2. local telephone number (or ID) to check for

The first line is expected to be a header, and is ignored. If special non-ASCII characters are used, then the CSV file should be in UTF-8 encoding.

3.16 Recording data export

If enabled and configured (Options, System settings, Export recordings, Configure), the Call Recorder Apresa will export audio and call data to a network share directory. The audio file will be written first. The audio file has a filename generated by the Apresa, for example:

2013342834827.wav

The information file will be in .xml format, and will be written after the .wav file is copied. The XML file has the same file name as the audio file, except with a different extension, for example:

2013342834827.xml

The XML has the following structure:

Field	***Description***
startdate	start date and time of the recording (standard format: yyyy-mm-dd hh:mm:ss)

startms	milliseconds part of the start time (0-999)
duration	duration in seconds of the recording
durms	milliseconds part of the duration
local	the telephone number (or id) of the local call participant, or, if both participants were local, the call initiator
local_name	the name of this call participant (or empty when unknown)
remote	the telephone number or id of the remote call participant, or, if both participants were local, the call receiver
remote_name	the name of this call participant
caller	the telephone number or id of the initiator of the call
caller_name	the name of the initiator of the call
receiver	the telephone number or id of the receiver of the call
receiver_name	the name of the receiver of the call
groups	a comma separated list of group names of the users that have a matching phone
direction	0=incoming, 1=outgoing, 2=internal, (3=external)
category	category number (could be set by user)
notes	primary notes field (could be set by user)
notes2 - notes5	secondary notes fields (could be set by user)
data	a copy of the data received with `startrec` action
id	a unique identifier of the recording (used by other API calls)
recording	filename of the recording
location	custom directory filled in by user, plus the filename of the recording
screenrec	filename of the screen recording (if available)
screenloc	custom directory filled in by user, plus the filename of the screen recording
errormsg	currently empty and unused

Example:

```
<callinfo>
     <startdate>2012-12-31 15:23:11</startdate>
     <duration>130</duration>
     <local>305</local>
     <local_name>John</local_name>
     <remote_name>Mike Hike</remote_name>
     <remote>+35401997252</remote>
     <direction>1</direction>
     <data>docnr=1203,seqnr=333</data>
</callinfo>
```

3.17 SNMP custom events

To enable SNMP traps, go to the system settings under the alarm tab. Here SNMP can be enabled

and a trap receiver can be configured.

Parameter	Value
act	sendtrap
quantity	The quantity for which the alarm is generated. Can be any string. Optional (OID 1.3.6.1.4.1.47036.1.1.2).
value	The value of the quantity. Can be any string. Optional. (OID 1.3.6.1.4.1.47036.1.1.3).
health	The health status of the value. This can have the following possible values: - ignored - okay - info - alarm - alarmTooHigh - alarmTooLow Optional (OID: 1.3.6.1.4.1.47036.1.1.4)
message	A custom message that can be send with the trap. Can be any string. Optional (OID: 1.3.6.1.4.1.47036.0.2.1

This action will generate a custom SNMP trap of the type apresaAlarmNotification (OID 1.3.6.1.4.1.47036.0.0.1) and send it to the configured trap receiver. Several values can be bound to this trap.

Parameter	Value
act	sendokay

This action will generate an SNMP trap of the type apresaAllOkayNotification (OID 1.3.6.1.4.1.47036.0.0.2

3.18 Problem solving

Problem: The Apresa web server answers with an HTML page containing the error message "417 – Expectation Failed".
Solution: Remove the "Expect" header from the HTTP request. When using libcurl, this can be done as follows:

```
struct curl_slist *headerList = NULL;
headerList = curl_slist_append(headerList, "Expect:");
curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headerList );
...
curl_slist_free_all(headerList);
```

Problem: The Apresa answers "Missing parameter auth".
Solution: Make sure the *auth* parameter is sent in the HTTP POST request, and use as Content-Type: `application/x-www-form-urlencoded` (normally) or `multipart/form-data` (when uploading files).