Oreka API

Revision 2.60-9487

Table of Contents

List of Tables

Chapter 1. Components

The main Oreka components accessible by API are:

- OrkWeb (Web UI frontend) - uses default port 8080.
- OrkTrack (Web UI backend) - uses default port 8080.
- OrkAudio (Audio recorder) - uses port 59140 for most commands, port 59120 for live audio streaming, and port 59150 for live event streaming.
- OrkRfb (Screen recorder) - uses port 59170 for commands.

Chapter 2. API Tables

Table 2.1. OrkTrack (Web UI backend) API table

Command	Description	Example
OrkTrack (Web UI backend)		
Recording Start	Message that a recording has started	http://localhost:8080//orktrack/command?type=tape&recid=20081116_190128_KUU&stage=start&captureport=KUU&ti
Recording Stop	Message that a recording has stopped	http://localhost:8080//orktrack/command?type=tape&recid=20081116_190128_KUU&stage=stop&captureport=KUU&ti

Command	Description	Example
Recording Ready	Message that a recording file is ready (was written)	http://localhost:8080//orktrack/command?type=tape&recid=20081116_190128_KUU&stage=ready&captureport=KUU&t
CTI metadata Start	CTI start message	Same as "Recording Start", with type=metadata instead of tape
CTI metadata Stop	CTI stop message	Same as "Recording Stop", with type=metadata instead of tape
Metadata evaluate	(Re-)evaluate CTI metadata for matching with recording entries	http://localhost:8080/orktrack/command?cmd=evalmetadata&ids=1,2,87&starti
Live user status	Get live status for list of users	http://localhost:8080/orktrack/command?cmd=livestatus&users=12,34,57,78,45
Query local party	Returns if a given local party is recordable and active	http://localhost:8080/orktrack/command?cmd=queryuser&localparty=5001
Poll external DB	Manually poll external database for named configured task	http://localhost:8080/orktrack/command?cmd=extdbsync&name=taskname
Provision a user, and associate it to a group	Add or update a user in the OrkWeb database (the loginstring is the search key)	http://localhost:8080/orktrack/command?cmd=adduser&firstname=John&lastna
Add more login strings to a user	Append multiple login strings to an existing user (an existing user loginstring is the search key)	http://localhost:8080/orktrack/command?cmd=addloginstring&findloginstring=
Import users	Add a list of users to OrkWeb database. File must be on the server. keepls: true=adds false=replaces login strings for existing user	http://localhost:8080/orktrack/command?cmd=addusers&keepls=false&recordal
Delete group	Delete a non-security group by specifying its name	http://localhost:8080/orktrack/command?cmd=deletegroup&groupname=Sales
Agent tracking	Used for Agent/Device mapping and Screen Recording agent login/logout	http://localhost:8080/orktrack/command?cmd=agentstate&state=login&agent=5
Keep recording	Record call for user id (called from Live Monitoring with Keep button)	http://localhost:8080/orktrack/command?cmd=keep&userid=1&elapsedtime=30
Discard recording	Discard call for user id (called from Live Monitoring with Keep button)	http://localhost:8080/orktrack/command?cmd=discard&userid=1&elapsedtime=
Reset recording	Reset the recording	http://localhost:8080/orktrack/command?cmd=neither&userid=1&elapsedtime=
On-Demand Record	Record given local party or orkuid (recording unique id)	http://localhost:8080/orktrack/command?cmd=ondemand&type=record&localpa
On-Demand Pause	Pause recording for given local party or orkuid (recording unique id)	http://localhost:8080/orktrack/command?cmd=ondemand&type=pause&localpa
On-Demand Resume	Resume recording for given local party or orkuid (recording unique id)	http://localhost:8080/orktrack/command?cmd=ondemand&type=resume&localp
On-Demand Stop	Stop recording for given local party or orkuid (recording unique id)	http://localhost:8080/orktrack/command?cmd=ondemand&type=stop&localpart
On-Demand Audio Stream (or listen)	Stream audio recording for given local party (or orkuid)	http://localhost:8080/orktrack/command?cmd=ondemand&type=streamaudio&l
Add a tag (based on another tag)	Add a tag to an existing or to a future recording based on another tag	http://localhost:8080/orktrack/command?cmd=addtag&findtagtype=findtagtype
Add a tag (based on a local party)	Add a tag to a live recording based on a local party. Useful for external applications	http://localhost:8080/orktrack/command?cmd=addtag&localparty=localparty_to
Add a tag (based on a recording uid)	Add a tag to a live or an existing recording based on the recording unique id (orkui). Useful mainly for OrkWeb. The tagentry parameter is useful to add multiple tags (tagtype/tagtext combinations) with one command: e.g. tagentry=CustomerNo:1234&tagentry=AgentID:3001.	http://localhost:8080/orktrack/command?cmd=addtag&orkuid=orkuid_to_find&
OrkTrack status	Ping orktrack to see if it is alive	http://localhost:8080/orktrack/command?cmd=ping
OrkTrack immediate cache update	Force OrkTrack to update its cache (for users, groups, programs, ...)	http://localhost:8080/orktrack/command?cmd=updatecache
OrkTrack cache update for programs Remote Party list	Force OrkTrack to update its program Remote Party lists	http://localhost:8080/orktrack/command?cmd=updatecache&cache=programsre

Command	Description	Example
Reload logging configuration	Reloads logging.properties files	http://localhost:8080/orktrack/command?cmd=reloadloggingconfig

Table 2.2. OrkWeb (Web UI frontend) API table

Command	Description	Example
OrkWeb (Web UI front end)		
Remote login (or Single Sign-On)	Log in to OrkWeb by API	http://localhost:8080/orkweb/app?username=myname&password=mypwd&startdate=20120215_121500&enddate=now&userid=2&filterby=false&mainmenu
Direct access to pages	On-Demand page	http://localhost:8080/orkweb/app?page=LiveMonitoring&service=page&ondemand=true&localparty=5001
	Recording details page (by tag)	http://localhost:8080/orkweb/app?page=SegmentDetail&service=page&localparty=5001&tagtype=Monitored&tagtext=test

Table 2.3. OrkAudio (audio recorder) API table

Command	Description	Example
OrkAudio (audio recorder)		
Stream live events	Stream all live events, calls "start" and "stop" on all extensions. In text, easily parsable key-value format.	http://localhost:59150/command?type=streamevents
Start recording	Start recording for a specific local party	http://localhost:59140/command?type=record&party=5002&side=both
	Start recording for a specific nativecallid	http://localhost:59140/command?type=record&nativecallid=33333&side=both
	Start recording for a specific orkuid	http://localhost:59140/command?type=record&orkuid=20081113_031704_MNP&side=both
Pause recording	Pause recording for a specific local party	http://localhost:59140/command?type=pause&party=5002&side=both
	Pause recording for a specific nativecallid	http://localhost:59140/command?type=pause&nativecallid=33333&side=both
	Pause recording for a specific orkuid	http://localhost:59140/command?type=pause&orkuid=20081113_031704_MNP&side=both
Stop recording	Stop recording for a specific local party	http://localhost:59140/command?type=stop&party=5002&side=both
	Stop recording for a specific nativecallid	http://localhost:59140/command?type=stop&nativecallid=33333&side=both
	Stop recording for a specific orkuid	http://localhost:59140/command?type=stop&orkuid=20081113_031704_MNP&side=both

Command	Description	Example
Stream audio	Stream the audio data in real-time (e.g. Live Monitoring). The audio stream is 8 KHz, 16 bits per sample PCM audio data (128 KBit/s)	http://localhost:59120/?type=stream&localparty=5002&recid=20081113_031704_MNP&stereo=true
Report recordings	Process message.log file(s) and report to orktrack the messages in those files (all stage READY messages)	http://localhost:59140/command?type=reporttape&fromdate=20080605_080000&todate=20080605_090000
Import recordings	Import recordings from another recording system	http://localhost:59140/command?type=importtape&mediatype=A&url=/path/to/audio/file&timestamp=1352481737&duration=20&localparty=1234&remote
Load orkaudio license	Load new license.txt stored in the orkaudio configuration folder.	http://localhost:59140/command?type=loadlicensefile
OrkAudio status	Ping orkaudio process to see if it is alive	http://localhost:59140/command?type=ping

Table 2.4. OrkRfb (screen recorder) API table

Command	Description	Example
OrkRfb (screen recorder)		
Start screen recording	Initiate screen recording for a given orkuid	http://localhost:59170/command?type=startrfb&hostname=deskto001&port=5900&loginstring=5001&username=john&password=mypassword&orkuid=20
Pause screen recording	Pause screen recording for a given orkuid	http://localhost:59170/command?type=pauserfb&hostname=deskto001&port=5900&loginstring=5001&username=john&password=mypassword&orkuid=2
Stop screen recording	Stop screen recording for a given orkuid	http://localhost:59170/command?type=stoprfb&hostname=deskto001&port=5900&loginstring=5001&username=john&password=mypassword&orkuid=20

Chapter 3. Recommendations

It is always better to use the API commands to OrkTrack rather than directly to the recorders, whereever possible. OrkTrack has built-in knowledge to identify which recorder a localparty or a recording ID (orkuid) belongs to, and will correctly play the proxy between the API consumer and producer.

This approach confers several advantages:

- API consumer does not need to know which recorder to send the command to in a distributed architecture.
- The recorders may not be directly accessible to the API consumer, in which case OrkTrack provides a point of access.

Chapter 4. Usage details

Table of Contents

This section provides details on each API command (expected parameters, type of response, ...)

Live User Status

The response is as follows:

class=livestatusresponse success=true returncode=NO_ERROR comment= userstatuses=inactive--NEITHER--68,localhost--59120!5001—5145551234--NEITHER--14--test1--IN--both--REC

The field of interest is userstatuses. It lists a series of statuses for the different users/local parties that are being inquired, comma-separated. In the example above, the assumption is that 2 local parties were being inquired for their statuses: 5000 and 5001. Below is an explanation of the different fields in the response.

Inactive

Format: inactive--status--elapsed--diskSpaceUsedInKb

Where:

- status is KEEP, DROP or NEITHER (based on the Live Monitoring page controls)
- elapsed is the elapsed time in seconds since the end of the last call (any value above 86400 simply means over 24 hours regardless of the specific value)
- diskSpaceUsedInKb is the user's current disk space usage. "null" is returned if not calculated or not applicable. This is returned only if the User Disk Space Quota feature is active

Example: inactive--NEITHER--68-3122

Active

Format: hostname--tcpport!localparty--remoteparty--status--elapsed--orkuid--direction--side--paused--tags--diskSpaceUsedInKb

Where:

- Hostname is the name/ip address of the orkaudio host
- tcpport is the port on which to access the app, here 8080
- status is KEEP, DROP or NEITHER (based on the Live Monitoring page controls)
- elapsed is the elapsed time in seconds since the beginning of the call
- orkuid is the unique Oreka identifier of the call
- direction is IN, OUT or UNKN
- side is both, local or remote
- paused is REC or PAUSED
- tags is a CSV list of tags to be added to the recording (e.g. ClientID-12345;Comment-rude to caller;...)
- diskSpaceUsedInKb is the user's current disk space usage. "null" is returned if not calculated or not applicable. This is returned only if the User Disk Space Quota feature is active.

Example: localhost--59120!5001—5145551234--NEITHER--14--20091026_000347_HRT--IN--both--REC--ClientID-12345;Comment-rude to caller--3122

User Provisioning

This command adds or updates a user into the OrkWeb database. The main key is the loginstring. If the login string exists, all data around it is updated. Otherwise, the user and login string are added. When an update occurs, only parameters with explicit values specified in the API command are updated in the database, all others remaining intact. The groupname and securitygroupname parameters allow association of the user to a logical and/or a security group.

Parameters

- firstname (semi-optional*): alphanumeric text representing the user first name.
- lastname (semi-optional*): alphanumeric text representing the user last name.
- loginstring (mandatory): alphanumeric text representing the user login string. This could be a phone number, an extension, a URI, ...
- recordable (optional): boolean (true/false, yes/no or 0/1). Default value is true. Designates whether a user should be recordable or not.
- external (optional): boolean (true/false, yes/no or 0/1). Default value is false. External users may be updated automatically by the user auto-provisioning feature too, while non-external ones are not.
- email (optional): alphanumeric text representing the user email, e.g. jsmith@abc.com.
- groupname (semi-optional*): regular (non-security) parent group to associate to the user. If the groupname does not exist, a new group will be created. If duplicates of the group name are found, this step is simply skipped.
- securitygroupname (semi-optional*): security group to associate to the user. If the securitygroupname does not exist, this step is simply skipped. Note: a user may belong to only one security group, hence, if the user already exists, this command will replace its security group.

* Semi-optional parameters may be required to complement the mandatory ones: either firstname and lastname or one of the groupname or securitygroupname pair should must be present.

Expected results

Successful response example:

```
class=userresponse success=true comment= returncode=NO_ERROR
```

Unsuccessful response example:

```
class=userresponse success=false comment=Error:%sERROR_SAVING_USER returncode=ERROR_SAVING_USER
```

Availability

OrkWeb 1.9-3228 for the API

OrkWeb 1.10-3518 for the groupname and securitygroupname parameters

Add multilpe login strings to an existing user

This command adds a login string to an existing user with at least one other login string. The main key is the findloginstring. The user associated to findloginstring will have a newloginstring associated to it.

Parameters

- findloginstring: login string used as a search key to find the user to append the newloginstring to.
- newloginstring: login string to append to the user.

Expected results

Successful response example:

```
class=userresponse success=true comment= returncode=NO_ERROR
```

Unsuccessful response example:

```
class=userresponse success=false comment=Error:%sERROR_NO_USER_FOUND_FOR_LOGINSTRING returncode=ERROR_NO_U
```

Availability

OrkWeb 1.10-3557

Delete a group

This command deletes a non-security group specified by its name. It only succeeds if the group name exists as a non-security group, and does not have duplicates.

Parameter

- groupname: exact name of group.

Expected results

Successful response example:

```
class=groupresponse success=true comment= returncode=NO_ERROR
```

Unsuccessful response example:

```
class=groupresponse success=false comment=Error:%sERROR_NO_USER_FOUND_FOR_LOGINSTRING returncode=ERROR_NO_
```

Availability

OrkWeb 1.10-3587

Agent Tracking

Login usage examples:

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=login&agent=5000&device=2000&agentgroup=6000

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=login&agent=5000&workstation=HomePC

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=login&agent=5000&device=2000&agentgroup=6000&workstation=HomePC

Logout usage examples:

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=logout&agent=5000&device=2000

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=logout&device=2000

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=logout&agent=5000&workstation=HomePC

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=logout&workstation=HomePC

http://orktrack-server:8080/orktrack/command?cmd=agentstate&state=logout&workstation=HomePC&rfbport=5900

Notes:

- This API expects at least an agent state and either a device or a workstation parameter. For logins, an agent parameter is also required.
- It is possible to log an agent into a device and/or a workstation (for screen recordings).
- It is **NOT** possible to log an agent into multiple devices. A login to a new device logs the agent out from the previous device they were logged into (as of 2.11-6721)
- It is **NOT** possible to log an agent into multiple workstations. A login to a new device logs the agent out from the previous workstation they were logged into (as of 2.11-6721)
- It is possible to log an agent into a device that is associated to a workstation in OrkWeb (for screen recordings).
- The workstation parameter specified **must**already exist in OrkWeb.
- The workstation parameter may be a hostname or an IP address, as long as it matches the hostname field configured for the workstation in OrkWeb.
- The rfbport parameter allows different agents to login to the same workstation on different ports.

Stream live events

This API call is different than a standard API call in the sense that it opens a socket for an indefinite period of time. As long as the socket is open, the client gets live call lifecycle events as they happen in an easily parseable key-value format until the client tears down the socket. The server never attempts to tear down the socket. Fields values of the reported events are escaped using standard URL escaping rules .

Possible event types:

- type=tape: these events are the call lifecycle events, start (call starts), stop (call stops) and ready (recording is ready for consumption after successful transcoding to the final storage format)
- type=addtag: these events add more metadata to an existing recording session via either 1. an orkuid unique identifier (that can be found as recid in the tape events) or 2. a localparty. In this case, the most recent session reported under this localparty will receive the tag. These tags are then searchable from the OrkWeb user interface.

Example usage:

http://orkaudio-server:59150/command?type=streamevents

Example output:

```
type=tape&recid=20140127_114145_AAAB&stage=start&captureport=AAAB&timestamp=1390840905&filename=2014%2F01%2F27%2F11%

type=addtag&localparty=&orkuid=20140127_114145_AAAB&tagtype=dtmfdigit&tagtext=0&offsetmsec=20006&

type=addtag&localparty=&orkuid=20140127_114145_AAAB&tagtype=dtmfdigit&tagtext=0&offsetmsec=27008&

type=tape&recid=20140127_114215_AAAC&stage=start&captureport=AAAC&timestamp=1390840935&filename=2014%2F01%2F27%2F11%

type=tape&recid=20140127_114145_AAAB&stage=stop&captureport=AAAB&timestamp=1390840905&filename=2014%2F01%2F27%2F11%2

type=tape&recid=20140127_114215_AAAC&stage=stop&captureport=AAAC&timestamp=1390840935&filename=2014%2F01%2F27%2F11%2

type=tape&recid=20140127_114145_AAAB&stage=ready&captureport=AAAB&timestamp=1390840905&filename=2014%2F01%2F27%2F11%
```

Stream audio

This command streams the audio data in real-time (e.g. Live Monitoring). The audio stream is 8 KHz, 16 bits per sample PCM audio data (128 Kbit/s).

For testing or troubleshooting, it's possible to redirect the resulting binary stream to a file and then import it in audio editor such as Audacity by using the raw import feature (file/import/raw data, pick the file, select signed 16 bit PCM, Little-endian, 1 channel if mono, 2 channels if stereo, 8000 Hz sample rate). Getting the raw binary date into a file can be done on the command-line like this:

```
        wget "http://orkaudio-server:59120/?type=stream&localparty=5002"
```

For production systems however, it is safer to use the "recid" returned by the streamevents API command, so that if the given extension handles several calls concurrently or close to each other, there is a guarantee that the right call is actually streamed, e.g.

```
        wget "http://orkaudio-server:59120/?type=stream&recid=20081113_031704_MNP"
```

When doing voice analytics, it might be a better idea to stream in stereo, i.e. have the audio for each side of the call be streamed in a separate audio channel. This typically helps achieving better results. A pre-requisite for this is to configure orkaudio for stereo recording. If already the case, then here is how to get a stereo stream:

```
        wget "http://orkaudio-server:59120/?type=stream&recid=20081113_031704_MNP&stereo=true"
```

The resulting stream will be 16 bit samples, 256 Kbit/s, interlaced like in a stereo wav file. I.e. one sample for the left channel, one sample for the right channel, one sample for the left channel and so on.

Import Recordings

The importtape API command allows you to import recordings made by another recording system. It can be useful compared to the standard tape message API especially when the external recordings need to be encrypted or converted to an audio format that is accepted by Oreka.

Here are the parameters details:

- mediatype: use A for audio, S for screen capture
- timestamp: must be the the timestamp of the beginning of the call in seconds since Unix EPOCH time
- duration : in seconds
- direction: can be "in" or "out" or left empty
- url: so far only a local file is accepted
- localparty: the URL encoded phone number of the local caller
- remoteparty: the URL encoded phone number of the remote caller

- nativecallid: this is an optional URL encoded unique ID that can be supplied in order to find the recording more easily in the user interface.

To trigger the importtape command to convert the media file using an external tool using CLI (e.g. convert mp3 audio into wav format), you would need to modify the config.xml as follows:

```
<TapeProcessors>BatchProcessing, CommandProcessing, Reporting</TapeProcessors>
<CommandProcessingCommand>ffmpeg -i [IN] -acodec libgsm_ms -ab 13000 [OUT]</CommandProcessingCommand>
```

The CommandProcessingCommand must be added under the top node in that file and may be modified as needed. The [IN] and [OUT] representing the input and output files must appear literally (no need to specify filenames).

Load orkaudio license

This API allows you to load a new orkaudio license WITHOUT restarting orkaudio. It assumes that you have applied the new valid orkaudio license file, license.txt, to the orkaudio installation folder (typically /etc/orkaudio on Linux, C:\Program Files\OrkAudio in Windows).

Here are some response examples for both success and failure:

```
<response>
  <success>true</success>
  <comment>
    License info: company name:abcd valid:true valid from:Tue 2016-01-05 00:00:00 EST valid to:Sat 2016-01
  </comment>
</response>

<response>
  <success>false</success>
  <comment>
    License info: company name: valid:false valid from:Wed 1969-12-31 19:00:00 EST valid to:Wed 1969-12-31
  </comment>
</response>
```