VoiSentry

API guide

Revision 1.12V | 2019

VoiSentry

Proprietary information

The information contained in this document is the property of Aculab plc and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission. It should not be used for commercial purposes without prior agreement in writing.

All trademarks recognised and acknowledged.

Aculab plc endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission.

The development of Aculab's products and services is continuous and published information may not be up to date. It is important to check the current position with Aculab plc.

Copyright © Aculab plc. 2019 all rights reserved.

Rev	Date	By	Detail
1.12V	07/06/19	CRE	Formatting

Contents

1. The services

The services exposed by this API enable Speaker Verification, Speech Recognition and DTMF digit recognition.

1.1 Errors

The services will raise an HTTPError if something is wrong. If the function is used incorrectly, the status will be most likely 400 and a text description of the error, beginning with the words Bad API request, will be provided.

There are several other codes that might be returned:

405 - Bad API request: method [method] not allowed.

408 - Database timeout error or Database access error.

429 - Several possible descriptions:

- System database volume too full for enrolments or updates.
- Cluster enrolment quota exceeded.
- Tenant enrolment quota exceeded.
- Accesskey enrolment quota exceeded.

435 - Has three possible descriptions:

- audio file contains no data.
- sample rate is too low (below 8 kHz).
- audio file is not single-channel (mono).

445 - Insufficient speaker observations in the audio source (implies that the audio offered to the service did not contain enough speech for VoiSentry to run its operations).

500 - An exception has occurred; a description of the problem is given.

503 - Service temporarily disabled for this key or Service temporarily disabled for this tenant.

1.2 The result object

The services return a JSON encoded results or error object. This object will take the form:

```
{
'status' : STATUS,
'result' : RESULT_OBJECT
}
```

STATUS is a code that will be 200 if the service ran correctly. If an error occurred it will be one of the following:

400 - a bad API request

405 - the HTTP method is not allowed for this service

408 - a database error has occurred

429 - a database quota has been exceeded

435 - bad audio

445 - no enough speaker data in the audio

500 - an internal system failure

503 - the service is disabled for this key or tenant

If STATUS is 200, RESULT_OBJECT will be a dictionary and will always contain:

dataset - The dataset being worked on.

tenant - The current tenant.

service - The API service that was run.

transactionid - A unique ID for this transaction.

diskalert - True if the disk usage is greater than the alert level - take action.

nodestatus - 'A' active, 'B' blocked. If 'B', node may be going out of service, don't sent it more traffic.

Other relevant information is added to RESULT_OBJECT depending on the function that is called. If STATUS is not 200, RESULT_OBJECT will be an error string.

1.3 Exposed services

The following services are exposed by the API:

- ***ping*** - Ping the system, get some system information.
- ***enrol*** - Enrol one speaker.
- ***verify*** - Verify a speaker
- ***update*** - Update one speaker.
- ***query*** - Query one or more speakers. Retrieve some speaker statistics.
- ***reenrol*** - Re-enrol one speaker.
- ***delete*** - Delete one or more speakers from the system.
- ***check*** - Check whether one or more enrolment IDs exist.
- ***quotas*** - Retrieve the quota information for a user.
- ***recognise*** - Recognise some speech.
- ***recognise_verify*** - Verify a speaker and recognise the speech.

2. API versioning

Whenever a speaker model is created using the `enrol` service, the latest version of the API will be used to process the enrolment. The API version is stored in the speaker model so that future verifications and updates will use the same API version that was used during enrolment.

If a software update installs a later version of the API, new enrolments will use the later version but existing speaker models will continue to be verified and updated using the API version that enrolled them.

To upgrade a speaker model to the latest API version, use the `reenrol` service. This will replace the existing speaker model with a new one based on the set of audio files that is supplied to `reenrol`. It is not possible to upgrade an existing model without supplying audio data. It is, therefore, a requirement that the original set of audio files that were used to enrol the speaker is stored for any future re-enrolments (upgrades). If the audio files are not stored, and the speaker model is to be upgraded, it will be necessary for the speaker to provide fresh audio for the re-enrolment.

Over the lifetime of a speaker model it will be regularly updated with new audio, particularly on the occasion when a verification fails and the speaker is verified by other means. These audio recordings should also be stored so that they can be included in the re-enrolment and make it current.

3. Working with thresholds

For a verification attempt to pass, the calculated confidence score must be greater than a threshold. In VoiSentry, each verification attempt will calculate a confidence score, and if this score is greater than 1 the verification should be deemed successful. In this case, 1 is the threshold and it will provide a good level of security.

Please note that the algorithm that calculates the confidence for a given speaker adapts with that speaker's model. Because of this, it cannot be said that a score of 1.2 for one speaker is better than another speaker's score of 1.1. In fact, for a given speaker, it is not possible to compare confidence scores over time (today's score of 1.2 for a given speaker is not necessarily better than yesterday's score of 1.1). Many times, a speaker will pass the verification attempt with a confidence score that is just above 1. This is normal and should not be considered a weak result.

If it becomes apparent that the error rates are not suitable for the application, a sensitivity value can be passed to the verify service to make it more or less sensitive. The sensitivity value can range from -10 to 10, by default it is 0. Setting a positive sensitivity, for example 1, will reduce the number of imposters that pass. Setting a negative sensitivity, for example -1, will reduce the number of real speakers that fail.

4. Working with enrolments

To enrol a new speaker, call the `enrol` service with at least ten seconds of speech. The speech can be divided over more than one audio file. Calling the `enrol` service with at least ten seconds of speech will, in most cases, create a good speaker model that is ready to use. However, to properly finalise the model, the enrolment process should include two verification attempts, each followed by a call to `update`. During this process, take note of the confidence that each verification attempt returns. If the confidence is below 1, another round of verify and update should be done. The verify plus update procedure will allow the algorithm to adjust to the speaker. As a result of this, you will notice the verification confidence score getting closer to 1.

5. Working with updates

There are a number of factors that can affect a speaker's verification confidence: updates to the model; a change of phone or environment; illness, and even simply being tired. Consequently, it is important to allow the speaker's model to adapt. A new model should be updated after each verification attempt until it has been updated two or three times (this can be part of the enrolment process); thereafter, it can be updated less frequently.

Note that that updates do not have to be done immediately after the verification session, they can be done off-line during quiet periods.

A speaker's model must only be updated with the audio from a successful verification; however, this should include verification by human interaction after a failed automatic attempt. It can happen that adverse conditions cause a speaker to fail to be verified because the confidence is too low. In these cases, a human verification process can allow the speaker to be passed, and, if the recorded speech is deemed to be of the speaker, the model can be updated. Care should be exercised in such cases because if the recording is not clean, or if the caller's identity is not correctly verified, updating the model will make it more difficult for the caller to be accepted in the future.

6. Ping

A simple ping interface.

The ping information returned is:

- ***clustername*** - A string. The name given to this cluster.
- ***serialno*** - A string. This node's serial number.
- ***userip*** - A string. The IP address from which the ping request originated.
- ***nodeactive4*** - A list. IPv4 addresses of active nodes with IPv4 addresses.
- ***nodeactive6*** - A list. IPv6 addresses of active nodes with IPv6 addresses.
- ***buildversion*** - A string. The VoiSentry build version.

The ping information is returned in a JSON object under the key `pinginfo`.

Required argument:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.

Usage example:

```
curl -k https://$TARGET:/api/ping?key=d50aae9d-424b-46b7-889b-c333059e90e4
```

Returns:

```
{
'dataset' : "dataset1", # A string, the dataset referenced by the current key.
'tenant' : "tenant1", # A string, the tenant referenced by the current key.
'service' : "ping", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level.
'nodestatus' : 'A', # 'A' active, 'B' blocked (the node may be going out of service).
'pinginfo' : {
'clustername' : "cluster", # A string, the cluster name.
'serialno' : "1499263460", # A string, the current node's serial number.
'userip' : "10.100.100.10", # A string, the user's IP address.
'nodeactive4' : ['10.11.12.13'.'10.22.12.14'], # A list of IPv4 addresses of active
nodes with IPv4 addresses.
'nodeactive6' : [], # A list of IPv6 addresses of active nodes with IPv6 addresses.
'buildversion': "1.17.8" # The VoiSentry build version.
}
}
```

7. Enrol

Create a speaker model and add it to the database.

Audio data can be supplied directly, or a URL to the audio can be given. Up to ten files of audio data can be supplied and each file is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If one or more URLs are given, enrol will not expect other audio sources.

The speaker model is based on the supplied audio. The audio must be a good representation of the speaker's normal speech. It is recommended that at least ten seconds of audio is provided.

The file names or URLs of the supplied audio sources are returned along with the enrolment ID and the API version that was used to create the speaker model. These are returned in a JSON object under the key `enrolment`.

The enrolment process should include one or two verification attempts, each followed by a call to `update`. This will allow VoiSentry to finalise the model. Take note of the confidence that each of these verification attempts returns. If the confidence is less than 1, another round of `verify` and `update` should be done.

Some of the supplied audio sources may fail the analysis. These sources are also returned under the `enrolment` key as failed sources. Failed sources will not be used when creating the model.

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The target speaker's unique enrolment ID. A string, chosen by the application writer. Only one `enrolid` can be supplied.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav.` More than one URL can be supplied.
- ***<audio sources>*** - One or more handles to audio sources. If one or more URLs are given, enrol will not look for these.

Optional argument:

- ***sviversion*** - A string. The speaker verification API version to use. Default is to use the latest version.

Usage example:

```
curl --request POST --form source1=@audio_a.wav --form source2=@audio_b.wav -k
"https://$TARGET:/api/enrol?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99997&svive
rsion=1"
```

Usage example:

```
curl -k "https://$TARGET:/api/enrol?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=999
97&sviversion=1&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.wav"
```

Returns:

```
{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "enrol", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'enrolment' : {
'enrolid' : "99997", # A string. The ID of the newly enrolled speaker.
'sources' : ["audio_a.wav", "audio_b.wav"], # A list of strings. The names or URLs of
the audio files.
'failed_sources' : [], A list of strings, or empty. The names or URLs of the audio files
that failed analysis.
'svi_version' : "1"
}
}
```

8. Verify

Run a verification test on a speaker model using new audio data.

Audio data can be supplied directly, or a URL to the audio can be given. Up to ten files of audio data can be supplied and each file is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If one or more URLs are given, verify will not expect other audio sources.

For the verification attempt to pass, the calculated confidence must be greater than 1. However, for increased security, the application writer can choose to use a value greater than 1. To make it less secure, it can be set below 1.

The verification results are returned in a JSON object under the key `verification`. The results to return are:

- ***enrolid*** - A string. The enrolment ID as supplied by the user.
- ***svi_version*** - A string, the API version used to verify this model
- ***date*** - A string. The verification date in the format `YYYY-MM-DD_HH:MM:SS`. For example, '`2017-09-25_11:32:11`'.
- ***confidence*** - A float, a confidence measure centred around 1. The bigger the value, the greater the confidence.
- ***sources*** - A list of strings. The audio file names or URLs

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The target speaker's unique enrolment ID. An alphanumeric string. Only one `enrolid` can be supplied.

Optional argument:

- ***sensitivity*** - A value between -10 and 10. This value adjusts the verification confidence measure. A negative value will increase the confidence and a positive value will decrease it.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav`. More than one URL can be supplied.
- ***<audio sources>*** - One or more handles to audio sources. If one or more URLs are given, verify will not look for these.

Usage example:

```
curl --request POST --form source1=@audio_a.wav --form source2=@audio_b.wav -k
"https://$TARGET:/api/verify?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99997"
```

Usage example:

```
curl -k "https://$TARGET:/api/verify?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99
997&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.wav"
```

Returns:

```
{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "verify", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'verification' : {
'enrolid' : "99997", # The enrolment ID supplied by the user.
'svi_version' : "1", # A string, the API version used to create this model
'sources' : ["audio_a.wav", "audio_b.wav"], A list of strings. The names or URLs of the
audio files.
'failed_sources' : [], A list of strings, or empty. The names or URLs of the audio files
that failed analysis.
'confidence' : 1.2, # A float, a confidence measure centred around 1.
'date' : "2017-09-25_11:32:11" # A string. The date and time as ``YYYY-MM-DD_HH:MM:SS``.
}
}
```

9. Update

Update an existing speaker model with new audio data.

Audio data can be supplied directly, or a URL to the audio can be given. Up to ten files of audio data can be supplied and each file is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If one or more URLs are given, update will not expect other audio sources. More than one audio file can be supplied.

It is important to allow the speaker's model to adapt over time and new models should be updated regularly. It is good practice to update the speaker's model after each successful verification attempt, using the audio that was used for the successful verification.

The file names or URLs of the supplied audio sources are returned along with the enrolment ID. These are returned in a JSON object under the key `updated`.

Some of the supplied audio sources may fail the analysis. These sources are also returned under the `updated` key.

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The target speaker's unique enrolment ID. An alphanumeric string. Only one enrolid can be supplied.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav`. More than one URL can be supplied.
- ***<audio sources>*** - One or more handles to audio sources. If one or more URLs are given, update will not look for these.

Usage example:

```
curl --request POST --form source1=@audio_a.wav --form source2=@audio_b.wav -k
"https://$TARGET:/api/update?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99997"
```

Usage example:

```
curl -k "https://$TARGET:/api/update?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99
997&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.wav"
```

Returns:

```
{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "update", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'updated' : {
'enrolid' : "99997", # A string. The ID of the newly enrolled speaker.
'sources' : ["audio_a.wav", "audio_b.wav"] # A list of strings. The names or URLs of the
audio files.
'failed_sources' : [], A list of strings, or empty. The names or URLs of the audio files
that failed analysis.
'svi_version' : "1"
}
}
```

10. Query

Return some verification statistics for one or more speakers.

The statistics to return are:

- ***deletion_date*** - A string. The date and time the speaker's model was deleted. The format is `YYYY-MM-DD_HH:MM:SS` or an empty string if not deleted.
- ***update_date*** - A string. The date and time the speaker's model was updated. The format is `YYYY-MM-DD_HH:MM:SS` or an empty string if not updated.
- ***enrolment_date*** - A string. The date and time the speaker first enrolled. The format is `YYYY-MM-DD_HH:MM:SS.` For example, '`2017-09-25_11:32:11`'.
- ***verification_attempts*** - The number of times the speaker has attempted to verify. An integer.
- ***verifications_passed*** - The number of successful verification attempts. An integer - see below.
- ***verifications_failed*** - The number of unsuccessful verification attempts. An integer - see below.
- l***ast_verification_date*** - A string. The date and time of the last verification attempt. The format is `YYYY-MM-DD_HH:MM:SS.` For example, '`2017-09-25_11:32:11`'.
- ***model_version*** - The current speaker model version number. An integer. This is incremented after each model update (not to be confused with re-enrolments).
- ***svi_version*** - The API version used to create this model. A string. To upgrade to a later version, run a re-enrolment.
- ***original_sources*** - The source names used for the original enrolment.

`verifications_passed` and `verifications_failed` are calculated using the threshold supplied to the verify service, or the default threshold if none was supplied. Since the verify service does return the confidence measure, the application may choose to pass or fail an individual verification based on a different criterion - ignoring the comparison with the supplied (or default) threshold.

However, the `verifications_passed` and `verifications_failed` counters are unable to reflect those decisions.

The statistics are returned as a JSON object containing the keys shown above. One such object is returned for each enrolment id supplied. The enrolment ids will be found under the key `statistics` in the returned object.

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The target speaker's enrolment ID. An alphanumeric string. More than one enrolment ID can be provided. For example '`99997`'.

Usage example:

```
curl -k "https://$TARGET:/api/query?key=73c1d0b1-9ffc-4696-8677-31349a716a15&enrolid=9999
7&enrolid=99998"
```

Returns:

```
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "query", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'statistics' : {
'99997' : {
'update_date' : "", # empty string if enrolment is not updated.
'deletion_date' : "", # empty string if enrolment is not deleted.
'enrolment_date' : "2017-09-25_11:32:11", # A string, ``YYYY-MM-DD_HH:MM:SS``.
'model_version' : 5, # An integer, incremented with each model update.
'svi_version' : "1",# A string, the API version used to create this model
'verification_attempts' : 10, # An integer, the total number of verification attempts.
'last_verification_date': "2017-10-25_11:32:11", # A string, ``YYYY-MM-DD_HH:MM:SS``.
'verifications_passed' : 5, # An integer, the number of verification passes.
'verifications_failed' : 5, # An integer, the number of verification failures.
'original_sources' : ['source1', 'source2'] # A list of strings, the original enrolment
sources
},
'99998' : { ... }, # As above
}
}
```

11. Reenrol

Use this command to remove an existing model and replace it with a new one.

Audio data can be supplied directly, or a URL to the audio can be given. Up to ten files of audio data can be supplied and each file is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If one or more URLs are given, reenrol will not expect other audio sources.

An update to the system may result in a new API version. Any existing speaker models will continue to use the API version that created them - this avoids any incompatibility issues. For a speaker model to use a later API version the speaker must be re-enrolled.

To re-enrol a speaker, it is advised that the original set of audio files that were used to enrol the speaker is supplied. Over the lifetime of a speaker model it will be regularly updated with new audio, particularly on the occasion when a verification fails and the speaker is verified by other means. These audio recordings should also be stored so that the new model can be further updated and made current.

The file names or URLs of the supplied audio sources are returned along with the enrolment ID and the API version used. These are returned in a JSON object under the key `enrolment`.

Some of the supplied audio sources may fail the analysis. These sources are also returned under the `enrolment` key.

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The target speaker's unique enrolment ID. An alphanumeric string. Only one `enrolid` can be supplied.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav.` More than one URL can be supplied.
- ***<audio sources>*** - One or more handles to audio sources. If one or more URLs are given, reenrol will not look for these.

Optional argument:

- ***sviversion*** - A string. The speaker verification API version to use. Default is to use the latest version.

Usage example:

```
curl --request POST --form source1=@audio_a.wav --form source2=@audio_b.wav -k
"https://$TARGET:/api/reenrol?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99997&svi
version=1"
```

Usage example:

```
curl -k "https://$TARGET:/api/reenrol?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=9
9997&sviversion=1&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.wav"
```

Returns:

```
{
‘dataset’ : “dataset1”, # A string, the current dataset.
‘tenant’ : “tenant1”, # A string, the current tenant.
‘service’ : “enrol”, # A string, the service that was run.
‘transactionid’ : “a7e1b3f6a1cd11e78439000c29c5390d”, # A string, the unique transaction
ID.
‘diskalert’ : True / False, # A bool, True if disk usage has reached the alert level
‘nodestatus’ : ‘A’ # ‘A’ active, ‘B’ blocked (Node may be going out of service)
‘enrolment’ : {
‘enrolid’ : “99997”, # A string. The ID of the newly enrolled speaker.
‘sources’ : [“audio_a.wav”, “audio_b.wav”], # A list of strings. The names or URLs of
the audio files.
‘failed_sources’ : [], A list of strings, or empty. The names or URLs of the audio files
that failed analysis.
‘svi_version’ : “1”
}
}
```

12. Delete

Remove one or more speaker enrolments from the database.

Supply one or more enrolment IDs. The list of enrolment IDs that were removed is returned.

Enrolment IDs that are not found, or are already deleted, will be omitted from the returned list.

Deleted enrolments are not physically removed, just marked as deleted. The removed enrolment IDs can be used for a different speaker, in which case the existing data is replaced.

Required arguments:

- ***key*** - The user access key. A string. For example, ‘`73c1d0b1-9ffc-4696-8677-31349a716a15`’.
- ***enrolid*** - The target speaker’s enrolment ID. An alphanumeric string. More than one enrolment ID can be provided.

Usage example:

```
curl -k “https://$TARGET:/api/delete?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99
997&enrolid=99998”
```

Returns:

```
{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "delete", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'enrolids' : ["99997", "99998"] # A list of strings. The enrolment IDs that were
deleted.
}
```

13. Check

Supply one or more enrolment IDs. Matching IDs currently enrolled in the system are returned. IDs that are not found (or have been deleted) are omitted from the list.

Required arguments:

- ***key*** - The user access key. A string. For example, '`73c1d0b1-9ffc-4696-8677-31349a716a15`'.
- ***enrolid*** - The enrolment ID to check. An alphanumeric string. More than one enrolment ID can be provided.

Usage example:

```
curl -k "https://$TARGET:/api/check?key=73c1d0b1-9ffc-4696-8677-31349a716a15&enrolid=9999
7&enrolid=99998"
```

Returns:

```
{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "check", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'enrolids' : ["99997", "99998"] # A list of strings. Matching IDs enrolled in the
system.
}
```

14. Quotas

Retrieve the user's quotas and current counts.

The quotas and counts to return are:

- ***maxenrols*** - The maximum number of enrolments allowed by the licence agreement for this tenant. 0 means unlimited.
- ***maxverifs*** - The maximum number of verifications per day allowed by the licence agreement for this tenant. 0 means unlimited.
- ***currenrols*** - The current enrolment count.
- ***currverifs*** - The current number of verifications for this day.

The quotas and counts are returned in a JSON object. The object key in the results object is `quotas`.

Required argument:

- ***key*** - The user access key.

Usage example:

```
curl -k "https://$TARGET:/api/quotas?key=2fed35b8-985e-44c4-9a25-19cbfa480ce7"

Returns:

{
'dataset' : "dataset1", # A string, the current dataset.
'tenant' : "tenant1", # A string, the current tenant.
'service' : "quotas", # A string, the service that was run.
'transactionid' : "a7e1b3f6a1cd11e78439000c29c5390d", # A string, the unique transaction
ID.
'diskalert' : True / False, # A bool, True if disk usage has reached the alert level
'nodestatus' : 'A' # 'A' active, 'B' blocked (Node may be going out of service)
'quotas' : {
'maxenrols' : 100, # An integer, the maximum number of enrolments per day.
'maxverifs' : 1000, # An integer, the maximum number of verifications per day.
'currenrols' : 50, # An integer, the number of enrolments performed, so far, this day.
'currverifs' : 500 # An integer, the number of verifications performed, so far, this day.
}
}
```

15. Recognise

Run a speech recognition task on some audio data. Speech recognition can be used to recognise the digits zero to nine or the words yes and no. Five languages are supported, German, English, French, Spanish and Italian.

Audio data can be supplied directly, or a URL to the audio can be given. Only one set of audio data can be supplied and it is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If a URL is given, recognise will not expect other audio sources.

DTMF digit detection can be run on the audio together with speech recognition.

If words are recognised, they are returned in a list of strings, for example, [‘one’, ‘two’, ‘three’, ‘four’]. If DTMF digits are recognised, they are returned as a string, for example, ‘1234’.

The arguments are described below.

Required arguments:

- ***key*** - The user access key. A string. For example, ‘`73c1d0b1-9ffc-4696-8677-31349a716a15`’.
- ***type*** - The speech recognition grammar type to use. A string. Only one grammar type can be specified. See below for the list of grammar options.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav`. Only one can be supplied.
- ***<audio sources>*** - An audio source handle. If a URL is given, recognise will not look for this. Only one can be supplied.

Optional argument:

- ***enrolid*** - The target speaker’s enrolment ID. An alphanumeric string. Only one `enrolid` can be supplied.
- ***checkdtmf*** - A character. If set to ‘`y`’, DTMF digit detection will be attempted on the audio signal. DTMF detection will be attempted before speech recognition is run. If DTMF digits are detected, speech recognition is not attempted.

Recognition grammars:

- ***Deu1digit*** - German, recognise a single digit.
- ***Deu2digits*** - German, recognise two digits.
- ***Deu3digits*** - German, recognise three digits.
- ***Deu4digits*** - German, recognise four digits.
- ***DeuNdigits*** - German, recognise any number of digits.
- ***DeuYesNo*** - German, recognise the words `ja` or `nein`.
- ***Eng1digit*** - English, recognise a single digit.
- ***Eng2digits*** - English, recognise two digits.
- ***Eng3digits*** - English, recognise three digits.
- ***Eng4digits*** - English, recognise four digits.
- ***EngNdigits*** - English, recognise any number of digits.
- ***EngYesNo*** - English, recognise the words `yes` or `no`.
- ***Fra1digit*** - French, recognise a single digit.
- ***Fra2digits*** - French, recognise two digits.
- ***Fra3digits*** - French, recognise three digits.
- ***Fra4digits*** - French, recognise four digits.
- ***FraNdigits*** - French, recognise any number of digits.
- ***FraYesNo*** - French, recognise the words `oui` or `non`.
- ***Ita1digit*** - Italian, recognise a single digit.
- ***Ita2digits*** - Italian, recognise two digits.
- ***Ita3digits***' - Italian, recognise three digits.
- ***Ita4digits***' - Italian, recognise four digits.
- ***ItaNdigits***' - Italian, recognise any number of digits.
- ***ItaYesNo***' - Italian, recognise the words `si` or `no`.
- ***Spa1digit*** - Spanish, recognise a single digit.
- ***Spa2digits***' - Spanish, recognise two digits.
- ***Spa3digits***' - Spanish, recognise three digits.
- ***Spa4digits***' - Spanish, recognise four digits.
- ***SpaNdigits***' - Spanish, recognise any number if digits.
- ***SpaYesNo***' - Spanish, recognise the words `si` or `no`.

The recognition result is returned as a JSON object under the key `recognition`. The recognition object will contain:

- ***words*** - A list of strings, e.g., [‘`one`’, ‘`two`’, ‘`three`’, ‘`four`’]. This key will be omitted if DTMF digits were detected.
- ***dtmf*** - A string of digits, e.g., ‘`1234`’. DTMF detection is run if checkdtmf is set to ‘`y`’. If DTMF digits are found, this will replace the ‘`words`’ key.
- ***type*** - A string. The recognition type that was requested, e.g., ‘`4digits`’.
- ***source*** - The file name or URL of the supplied audio source.
- ***date*** - A string. The recognition date and time in the format `YYYY-MM-DD_HH:MM:SS.` For example, ‘`2017-09-25_11:32:11`’.

If neither words nor `dtmf` were recognised, the `recognition` key will be omitted from the result.

Usage example:

```
curl --request POST --form source1=@audio_a.wav -k “https://$TARGET:/api/
recognise?key=d50aae9d-424b-46b7-889b-c333059e90e4&type=Eng4digits&checkdtmf=y”
```

Usage example:

```
curl -k “https://$TARGET:/api/recognise?key=d50aae9d-424b-46b7-889b-c333059e90e4&type=En
g4digits&checkdtmf=y&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.wav”
```

If `checkdtmf` is set to ‘`y`’, this function will detect any DTMF digits present in the audio recording. If DTMF digits are found, ASR will not be run on the recording.

Returns:

```
{
‘dataset’ : “dataset1”, # A string, the current dataset.
‘tenant’ : “tenant1”, # A string, the current tenant.
‘service’ : “recognise”, # A string, the service that was run.
‘transactionid’ : “a7e1b3f6a1cd11e78439000c29c5390d”, # A string, the unique transaction
ID.
‘diskalert’ : True / False, # A bool, True if disk usage has reached the alert level
‘nodestatus’ : ‘A’ # ‘A’ active, ‘B’ blocked (Node may be going out of service)
‘recognition’ : {
‘source’ : “audio_a”, # The name or URL of the audio file that was supplied.
‘words’ : [“one”, “two”, “three”, “four”], # A list of strings. Omitted if DTMF is
detected.
‘dtmf’ : “1234”, # If checkdtmf was set, DTMF digit detection will be run.
‘type’ : “4digits”, # A string. The grammar type that was used.
‘date’ : “2017-09-25_11:32:11” # A string, the date and time as ``YYYY-MM-DD_HH:MM:SS``.
}
}
```

16. Recognise_verify

Run a speech recognition and verification task on some audio data. Speech recognition can be used to recognise the digits zero to nine or the words yes and no. Five languages are supported, German, English, French, Spanish and Italian.

Audio data can be supplied directly, or a URL to the audio can be given. Only one set of audio data can be supplied and it is limited to five megabytes. The audio data must be mono WAV files. The preferred sampling rate is 8kHz. If a URL is given, recognise_verify will not expect other audio sources.

The speech recognition results and the verification results are returned, as they are for the `recognise` and `verify` services. Unlike the `recognise` service, DTMF digit recognition cannot be set for this service.

The words that are recognised are returned in a list of strings, for example, [‘`one`’, ‘`two`’, ‘`three`’, ‘`four`’].

The arguments are described below.

Required arguments:

- ***key*** - The user access key. A string. For example, ‘`73c1d0b1-9ffc-4696-8677-31349a716a15`’.
- ***enrolid*** - The target speaker’s enrolment ID. An alphanumeric string. Only one ID can be supplied.
- ***type*** - The speech recognition grammar type to use. A string. Only one grammar type can be specified. See below for the list of grammar options.

Audio sources:

- ***url*** - A string. The address from which to download audio data. For example `http://my.wav.files.com:8001/get_wav?filename=audio_a.wav`. Only one can be supplied.
- ***<audio sources>*** - An audio source handle. If a URL is given, recognise_verify will not look for this. Only one can be supplied.

Recognition grammars:

- ***Deu1digit*** - German, recognise a single digit.
- ***Deu2digits*** - German, recognise two digits.
- ***Deu3digits*** - German, recognise three digits.
- ***Deu4digits*** - German, recognise four digits.
- ***DeuNdigits*** - German, recognise any number of digits.
- ***DeuYesNo*** - German, recognise the words `ja` or `nein`.
- ***Eng1digit*** - English, recognise a single digit.
- ***Eng2digits*** - English, recognise two digits.
- ***Eng3digits*** - English, recognise three digits.
- ***Eng4digits*** - English, recognise four digits.
- ***EngNdigits*** - English, recognise any number of digits.
- ***EngYesNo*** - English, recognise the words `yes` or `no`.
- ***Fra1digit*** - French, recognise a single digit.
- ***Fra2digits*** - French, recognise two digits.
- ***Fra3digits*** - French, recognise three digits.
- ***Fra4digits*** - French, recognise four digits.
- ***FraNdigits*** - French, recognise any number of digits.
- ***FraYesNo*** - French, recognise the words `oui` or `non`.
- ***Ita1digit*** - Italian, recognise a single digit.
- ***Ita2digits*** - Italian, recognise two digits.
- ***Ita3digits***' - Italian, recognise three digits.
- ***Ita4digits***' - Italian, recognise four digits.
- ***ItaNdigits***' - Italian, recognise any number of digits.
- ***ItaYesNo***' - Italian, recognise the words `si` or `no`.
- ***Spa1digit*** - Spanish, recognise a single digit.
- ***Spa2digits***' - Spanish, recognise two digits.
- ***Spa3digits***' - Spanish, recognise three digits.
- ***Spa4digits***' - Spanish, recognise four digits.
- ***SpaNdigits***' - Spanish, recognise any number if digits.
- ***SpaYesNo***' - Spanish, recognise the words `si` or `no`.

The recognition result is returned as a JSON object under the key `recognition`. The recognition object will contain:

- ***words*** - A list of strings, e.g., [‘`one`’, ‘`two`’, ‘`three`’, ‘`four`’].
- ***type*** - A string. The recognition type that was requested, e.g., ‘`4digits`’.
- ***source*** - A string. The file name of the supplied audio source.
- ***date*** - A string. The recognition date and time in the format `YYYY-MM-DD_HH:MM:SS.` For example, ‘`2017-09-25_11:32:11`’.

If no words were recognised, the recognition key will be omitted from the result.

The verification results are returned in a JSON object under the key verification. The results to return are:

- ***enrolid*** - A string. The enrolment ID as supplied by the user.
- ***svi_version*** - A string, the API version used to verify this model
- ***confidence*** - A float, a confidence measure centred around 1. The bigger the value, the greater the confidence.
- ***source*** - A string. The file name or URL of the supplied audio source.
- ***date*** - A string. The verification date in the format `YYYY-MM-DD_HH:MM:SS.` For example, ‘`2017-09-25_11:32:11`’.

Usage example:

```
curl --request POST --form source1=@audio_a.wav -k “https://$TARGET:/api/recognise_
verify?key=d50aae9d-424b-46b7-889b-c333059e90e4&enrolid=99997&type=Eng4digits”
```

Usage example:

```
curl -k “https://$TARGET:/api/recognise_verify?key=d50aae9d-424b-46b7-889b-c333059e90e4
&enrolid=99997&type=Eng4digit&url=http://my.wav.files.com:8001/get_wav?filename=audio_a.
wav”
```

Returns:

```
{
‘dataset’ : “dataset1”, # A string, the current dataset.
‘tenant’ : “tenant1”, # A string, the current tenant.
‘service’ : “recognise_verify”, # A string, the service that was run.
‘transactionid’ : “a7e1b3f6a1cd11e78439000c29c5390d”, # A string, the unique transaction
ID.
‘diskalert’ : True / False, # A bool, True if disk usage has reached the alert level
‘nodestatus’ : ‘A’ # ‘A’ active, ‘B’ blocked (Node may be going out of service)
‘recognition’ : {
‘source’ : “audio_a”, # The name of the audio file that was supplied.
‘words’ : [“one”, “two”, “three”, “four”], # A list of strings.
‘type’ : “4digits”, # A string. The grammar type that was used.
‘date’ : “2017-09-25_11:32:11” # A string, the date and time ast ``YYYY-MM-DD_
HH:MM:SS``.
},
‘verification’ : {
‘enrolid’ : “99997”, # The enrolment ID supplied by the user.
‘svi_version’ : “1”, # A string, the API version used to create this model
‘source’ : “audio_a.wav”, # A string. The name or URL of the supplied audio file.
‘failed_source’ : A string (can be empty). The name or URL of the audio file that failed
analysis.
‘confidence’ : 1.2, # A float, a confidence measure centred around 1.
‘date’ : “2017-09-25_11:32:11” # A string. The date and time as ``YYYY-MM-DD_HH:MM:SS``.
}
}
```

Follow VoiSentry: Twitter | Blog | LinkedIn

+44 (0) 1908 27 38 38 | www.voisentry.com

VoiSentry