



Speaker Verification Engine[®]
Application Program Interface

V2verify™
10832 Old Mill Rd. Ste 3
Omaha, NE 68154

February 11, 2019

V-0107-SVE-0300

Copyright Notice

The V2verify Speaker Verification Engine (SVE) is copyrighted by V2verify™. © 2005-2014

This document is the property of V2verify and distributed only to authorized individuals and organizations that have signed a non-disclosure agreement, which will be vigorously enforced. Additionally, certain aspects of the software and its documentation are trade secrets, and all software and documentation are copyrighted.

Trade secrets are limited in distribution and possession of this information by unauthorized individuals and organizations may result in severe civil and criminal penalties and will be prosecuted to the maximum extent possible under the law.

Copyright law and international treaties protect the software and its documentation referred to in this document. Unauthorized reproduction or distribution of any program, its documentation, or any portion of either of them, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under the law.

The specifications and information regarding the products in this guide are subject to change without notice. All statements, information and recommendations in this guide are believed to be accurate but are presented without warranty of any kind, expressed or implied. Users must take full responsibility for their application of any products. The software license and limited warranty for the accompanying product is set forth in the duly executed Master Software License Agreement (MSLA) and is incorporated herein by this reference. If unable to locate the Master Software License Agreement, contact a V2verify representative for a copy. Notwithstanding any other warranty herein, all document files are provided "AS IS" with all faults.

All other trademarks mentioned in this document or Web site are the property of their respective owners. The use of the word partner does not imply a partnership relationship between V2verify and any other company.

Disclaimer

This document has been prepared with all care to reflect the intended operation of the described software for the stated version. V2verify makes no warranties as to the operation or fitness for purpose of the software and the user will determine through evaluation and testing its suitability for their application. Where operation of the software described deviates from this document V2verify at its sole discretion will amend either the software or the documentation or both the software and documentation to make consistent. This document provides no obligation for V2verify to support any features described in future versions.

About Us

Established in 2005, V2verify™ provides the core Speaker Verification Engine (SVE) for some of the leading deployments of this technology. V2verify now has operations in North America, and Asia Pacific with plans for further expansion. Long term customers have proven the reliability and robustness of the V2verify SVE.

V2verify's Speaker Verification Engine provides significantly better performance than other available engines. Independent comparisons show performance in terms of Equal Error Rate (ERR) as being an order of magnitude better than the leading players. V2verify packages the SVE with an Application Programming Interface (API). The packaged engine is available to corporate and application developer customers as a superior authentication engine to those already on the market.

V2verify continues to build on its more than 5 years of intensive development. EER is considered the best measure of usability of a verification system. It is the measure of where the potential for false accepts and false rejects is balanced. A lower EER value means superior customer satisfaction whilst maintaining the required level of security.

V2verify's product focus is to provide enhanced security and privacy for the web on-line world and emerging 3G technologies (transactional & content management). V2verify aims to provide security, privacy and convenience for the individual and reduce costs for the business and call center operator. V2verify enhances privacy in offshore and virtual contact centers (on-shoring and home agents), but still allow businesses to utilize the cost benefit from operating in low cost jurisdictions. V2verify assists in driving automation of self-service requirements by providing a secure and convenient method of identifying customers and employees. V2verify also continues to work closely with leading research institutions to maintain the Speaker Verification Engine's leading-edge performance.



Table of Contents

1. Introduction	7
2. Response Messages	8
3. Cancel Method	9
3.1 Successful Result	9
4. Enrollment.....	10
4.1 Starting an Enrollment	11
4.1.1 Successful Result.....	11
4.1.2 Failure Result	12
4.2 Sending Voice Data to An Enrollment	12
4.2.1 Successful Result.....	13
4.2.2 Failure Result	13
4.3 Finalize an Enrollment	14
4.3.1 Successful Result.....	14
4.3.2 Failure Result	14
5. Verification.....	15
5.1 Starting a Verification	16
5.1.1 Successful Result.....	17
5.1.2 Failure Result	17
5.2 Sending Voice Data	18
5.2.1 Successful Result.....	19
5.2.2 Failure Result	19
5.2.3 Successful Result.....	20
5.2.4 Failure Result	20
5.3 Sending Live Voice Data	21
5.3.1 Successful Result.....	22
5.3.2 Failure Result	23
5.4 Finalize a Verification	24
5.4.1 Successful Result.....	24
5.4.2 Failure Result	25
6. Audio Formats	26

- 7. Behavioral Enhancement Pin27
 - 7.1 Submitting the Pin During Enrollment27
 - 7.2 Submitting the Pin During Verification27
- 8. Best Practices for Implementation28
 - 8.1 Natural Language Enrollment (On-Boarding during an existing call with an Agent (Patented Process))...28
 - 8.2 Smartphone App or IVR Enrollment.....28
 - 8.3 Natural Language Verification29
 - 8.4 Mobile App or IVR Verification29

Table of Figures

Figure 1-1: Platform API Diagram 7

Figure 3-1: Session ID Parameter 9

Figure 4-1: Successful Enrollment Sequence 10

Figure 4-2: Rejected Enrollment Sequence..... 10

Figure 4-3: Enrollment Start Parameters 11

Figure 4-4: Enrollment Start Request Headers 11

Figure 4-5: Enrollment Start Response Headers 11

Figure 4-6: Enrollment Start Status Codes 12

Figure 4-7: Enrollment Process Session ID Parameter..... 12

Figure 4-8: Enrollment Process Data Parameter..... 12

Figure 4-9: Enrollment Sampling Codes 13

Figure 4-10: Enrollment Process Status Codes 13

Figure 4-11 Enrollment Terminate Session ID Parameter 14

Figure 4-12: Enrollment Finalize Status Codes..... 14

Figure 5-1: Successful Verification Process Flow 15

Figure 5-2: Failed Verification Example 15

Figure 5-3: Verification Start Parameters 16

Figure 5-4: Verification Start Request Headers 16

Figure 5-5: Verification Start Response Headers 16

Figure 5-6: Verification Start Error Codes 17

Figure 5-7: Verification Process Session ID Parameter 18

Figure 5-8: Verification Process Data Parameter 18

Figure 5-9: Verification Sampling Codes 19

Figure 5-10: Verification Process Status Codes..... 19

Figure 5-11: Verification Liveness Sampling Codes..... 20

Figure 5-12: Verification Liveness Process Status Codes 20

Figure 5-13: Verification Liveness Process Session ID Parameter 21

Figure 5-14: Verification Liveness Process Data Parameter 21

Figure 5-15: Verification Liveness Sampling Codes..... 22

Figure 5-16: Verification Liveness Process Status Codes 23

Figure 5-17: Verification Terminate Session ID Parameter..... 24

Figure 5-18: Verification Terminate Status Codes 25

Figure 7-1: Enrollment Start Header for Enhancement Pin 27

Figure 7-2: Verification Delete Header for Enhancement Pin 27

1. Introduction

Voice biometrics also known as Voice Authentication or Speaker Verification, verifies the identity of the speaker based on numerous vocal tract traits and style of speech. Voice biometrics is based on the fact that each individual's voice is unique. It is more effective and efficient than other biometric solutions as well as one of the latest non-intrusive technologies commercially available for identity verification. V2verify's Speaker Verification Engine (SVE) processes both Text Independent and Text Dependent utterances to provide flexibility across a range of security requirements. These utterances can be used to not only identify what the speaker is saying but it can also be used to identify the speaker by his/her vocal traits.

The Speaker Verification Engine (SVE) by itself when run does nothing except to wait for incoming audio. It is developed with a set of powerful API calls that allow for seamless integration into existing environments. In a live production environment, there is usually some other application that interacts with the SVE, such as an Interactive Voice Response (IVR) system that forwards incoming requests to the SVE for enrollment and verification.

This document covers the use of the Representational State Transfer (REST) Application Program Interface (API) for V2verify's Speaker Verification Engine (SVE); it does not address issues of engine deployment or tuning. Should greater control of the V2verify engine be required, such as changing the locations of stored data or the use of specific encryptions contact your nearest V2verify representative for additional support.

This interface supports enrollment and verification processes. It is up to the application developer to ensure that all other processing logic, security and procedures are developed appropriately. Storage of enrollment vectors is the responsibility of the application. The general structure is depicted in Figure 1 below.

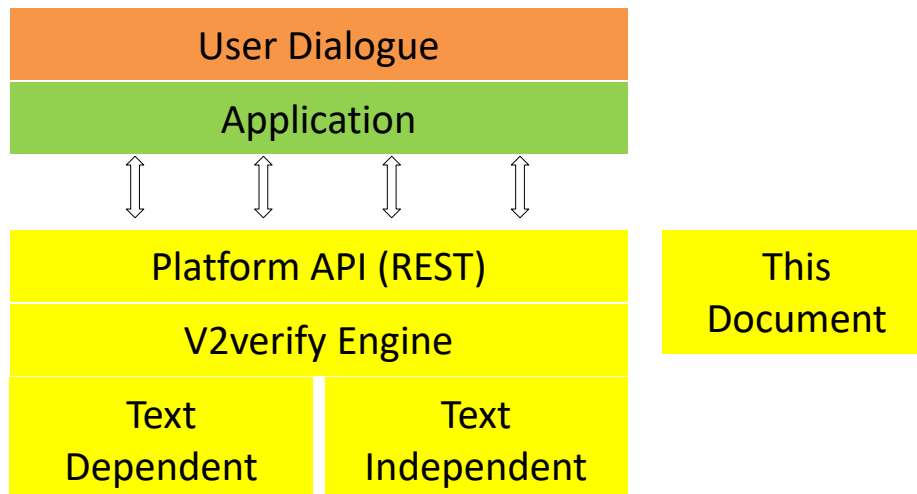


Figure 1-1: Platform API Diagram

2. Response Messages

The SVE REST API uses the JSON Language to send result responses. The API returns only two HTTP response codes: HTTP 200 OK and HTTP 500 INTERNAL SERVER ERROR. The HTTP 200 OK response message varies from API call to API call, while the HTTP 500 INTERNAL SERVER ERROR response message remains the same.

The following is an example of an HTTP 500 INTERNAL SERVER ERROR response message.

```
/* HTTP 500 - Json Error Response */
{
  /* Error Code (API error, not a catastrophic error) */
  "error":0,

  /* Description of the error code */
  "description":"Description of error"
}
```

It is worth noting that for some instances, the HTTP 200 OK response message can have an error block as well; this is considered a catastrophic error and the data inside needs to be transmitted to V2verify for inspection, along with the Developer-Key, Application-Key, Client-Id, and Vv-Session-Id from the message it came from.

3. Cancel Method

The Cancel Method is used to cancel an Enrollment or Verification session in the event an error occurred in the voice data phases, noted in sections 4.2 and 5.2.

DELETE - <http://v2ondemand.com:50202/1/sve/Cancel/{reason}>

HTTP Request Headers

Name	Type	Values	Description
Vv-Session-Id	string	Session Id returned by start enrollment/verification session(s)	Unique identifier of the enrollment/verification.

Figure 3-1: Session ID Parameter

{reason} – is a “-“ separated string of text of length char (64) supplied as the reason for cancelling the session.

URL Example: <http://v2ondemand.com:50202/1/sve/Cancel/testing-session-connection>

3.1 Successful Result

There is no JSON result response for a successful cancel.

4. Enrollment

The following calls are used to start process and terminate an enrollment.

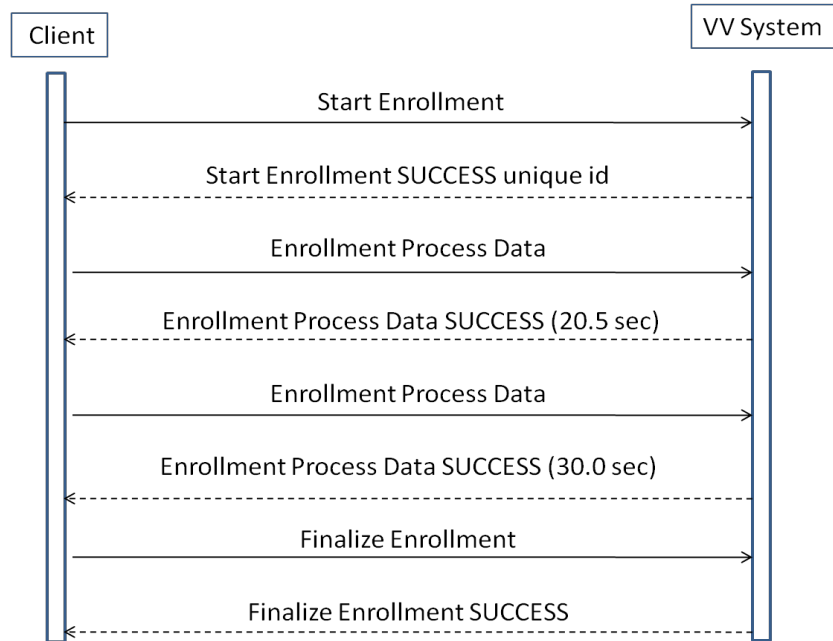


Figure 4-1: Successful Enrollment Sequence

We recommend having over 50 seconds of enrollment data for text independent profiles. In Figure 4-1 above, we have 20.5 + 30.0 = 50.05 seconds. The seconds returned is the amount of usable speech that the SVE feature extraction process found in the enrollment audio data file. It is important to note here that 50 seconds of audio data does not necessarily equal 50 seconds of enrollment speech. A general rule of thumb is to associate every 3 seconds of audio data with 1 second of enrollment speech data.

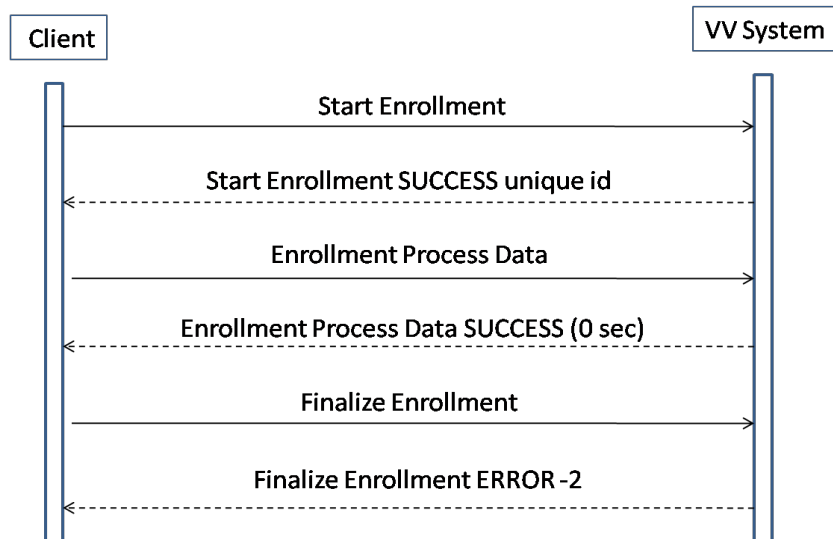


Figure 4-2: Rejected Enrollment Sequence

In this case the client should have called the Enrollment Process Data again, even with the same data. In the case of Text-Dependent processing, this would be the same enrollment phase, while additional data could be sent for Text Independent processing.

4.1 Starting an Enrollment

An enrollment session is initiated using a Developer-Key and an Application-Key to uniquely identify the developer's application and requires a unique client id and the sub-pop (gender) of the client enrolling.

POST - http://v2ondemand.com:50202/1/sve/Enrollment/{client_id}/{subpop}

URL Parameters:

Name	Type	Values	Description
client_id	String		Unique-ID any alphanumeric string
subpop	Char	M=MALE F=FEMALE	Supported Values

Figure 4-3: Enrollment Start Parameters

HTTP Request Headers:

Name	Type	Description
Developer-Key	String	The Developer-Key located on the Dashboard
Application-Key	String	The Application specific Key located on the Dashboard
Interaction-Id	String	(Optional) An external session id, used to correlate internal data with.

Figure 4-4: Enrollment Start Request Headers

HTTP Response Headers:

Name	Type	Description
VV-Session-Id	String	The session Id required to be sent with subsequent processing calls.

Figure 4-5: Enrollment Start Response Headers

4.1.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Enrollment Start Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error":{ ... },

  /* Profile information only present on enrollment start message */
  "profile.enroll":{

    /* Index number of the profile */
    "index":0,

    /* Kind of Profile: Independent - 1, Liveness - 2 */
    "kind":0,

    /* Type of Audio Codec: pcm_little_endian -or- alaw */
    "codec":"","

    /* Number of seconds of extracted speech required to train an enrollment. */
    "min_seconds_of_speech":0.0
  }
}

```

4.1.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error Description Code	Description
101	Session Exception
104	Developer has reached their limit
107	Unable to acquire a session.
202	Unable to start the feature extractor
302	Unable to start the enrollment processor
309	Invalid Application Key / Application Profile Not Found
310	Subpopulation parameter incorrect
311	Enrollment Limit Reached

Figure 4-6: Enrollment Start Status Codes

4.2 Sending Voice Data to An Enrollment

Sending voice data to an enrollment can be called multiple times. For Text-Independent Processing, we recommend gathering at least 50 seconds of buffered vectors.

POST - <http://v2ondemand.com:50202/1/sve/Enrollment>

HTTP Request Headers:

Name	Type	Values	Description
VV-Session-Id	string	Session Id returned by start enrollment session	unique identifier of the enrollment. Received in the start enrollment API.

Figure 4-7: Enrollment Process Session ID Parameter

Additional Notes:

- Uses File Upload Standard RFC 1867.
- "Form-based File Upload in HTML". An HTTP request submitted using the POST method with a content-type of "multipart/form-data".
- The audio data to be transmitted needs to be encoded by the same audio codec that was received with the *profile.enroll* JSON object from the start enrollment HTTP Data Stream. The audio data is **required** to be transmitting as a raw wav stream without any "RIFF" headers.

Stream Data:

Name	Type	Values	Description
data	binary	Max file size is 2 megabytes	Form data field name to store binary data is "data"

Figure 4-8: Enrollment Process Data Parameter

4.2.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Enrollment Process Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error":{
    ...
  },

  /* Present on enrollment process and end messages */
  "result.enroll":{

    /* The client_id used in the enrollment start message */
    "{client_id}":{

      /* Index number of the profile */
      "index":0,

      /* Number of seconds of speech extracted from all voice samples thus far */
      "seconds_extracted":0.0,

      /* Current processing call sampling error */
      "error":0
    }
  }
}

```

Sampling Error Code	Description
0	Success
303	Need More Voice
304	Enrollment Training Error (Only shows up on Finalizing an Enrollment)

Figure 4-9: Enrollment Sampling Codes

4.2.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error description code	Description
101	Session Exception (See Log)
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
104	Developer Limits Reached
311	Enrollment Limits Reached
315	Invalid Post Data, No Stream Found, or No voice found in stream
317	Invalid Post Data, Stream Found, Missing data field
318	Invalid Post Data, Unknown stream type, Missing form-data
701	Error occurred while updating database

Figure 4-10: Enrollment Process Status Codes

4.3 Finalize an Enrollment

Finalize an Enrollment, train and save data, destroy the session.

DELETE - <http://v2ondemand.com:50202/1/sve/Enrollment>

HTTP Request Headers:

Name	Type	Values	Description
VV-Session-Id	string	Session Id as returned in start enrollment process	unique identifier of the enrollment. Received in the start enrollment process.

Figure 4-11 Enrollment Terminate Session ID Parameter

4.3.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Enrollment End Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error":{
    ...
  },
  /* Present on enrollment process and end messages */
  "result.enroll":{
    /* The client_id used in the enrollment start message */
    "{client_id}":{
      /* Index number of the profile */
      "index":0,
      /* Number of seconds of speech trained from all voice samples */
      "seconds_trained":0.0,
      /* Current processing call sampling error */
      "error":0
    }
  }
}
    
```

4.3.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error description code	Description
101	Session Exception (See Log)
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
104	Application Limits Reached
304	Enrollment Training Error
701	Error occurred while updating database

Figure 4-12: Enrollment Finalize Status Codes

5. Verification

The following calls are used to start, process, and terminate a verification:

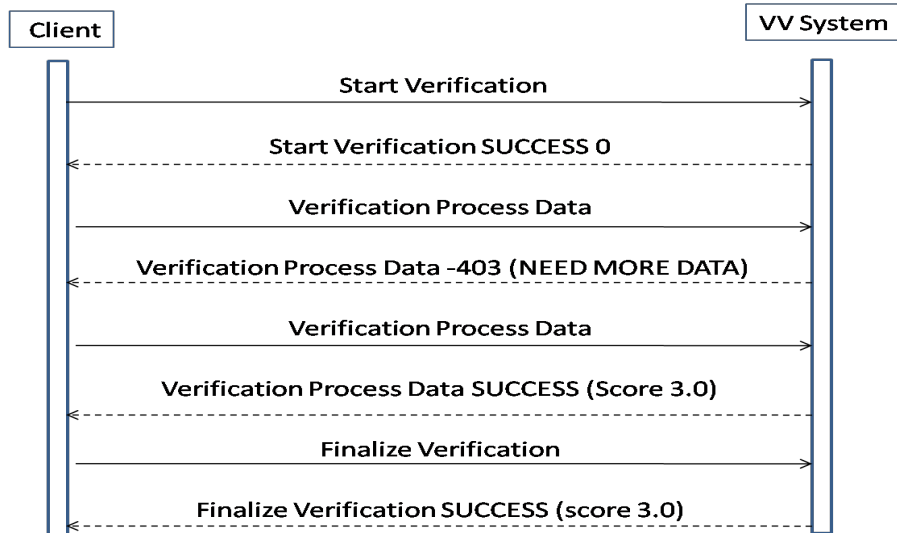


Figure 5-1: Successful Verification Process Flow

The general sequence for a verification is to start a session which may then be left open for as long as it is required. This supports being able to do continuous verifications during an interaction. A verification session can be finalized at any time.

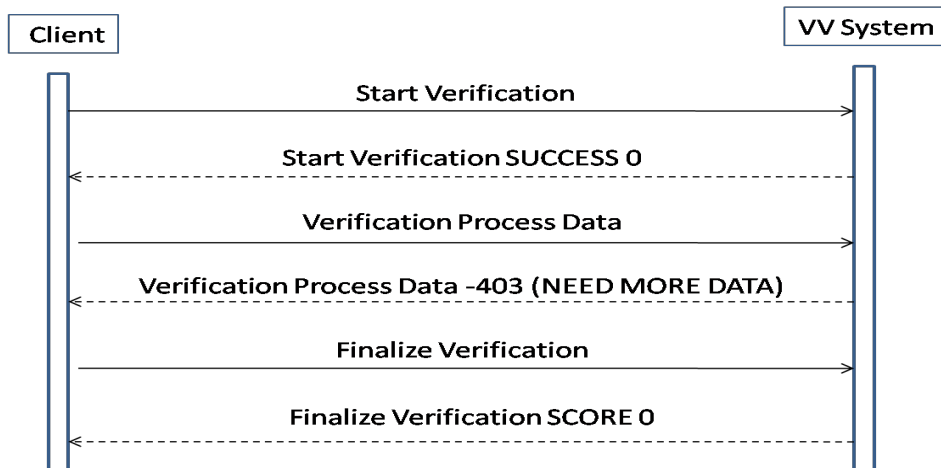


Figure 5-2: Failed Verification Example

Figure 5-2 above gives an example of what a failed verification looks like. In this example, the error returned indicates more data is required to produce a reliable result. In this case we would persist and submit additional data, or if trying to verify against a text dependent profile, we would resubmit the same data file.

5.1 Starting a Verification

Initiate a verification session using a pre-configured profile and the same unique client-id that was used in the enrollment session.

POST - <http://v2ondemand.com:50202/1/sve/Verification/{client id}>

URL Parameters:

Name	Type	Description
Client id	String	Client Id for the enrollment that this verifier is claiming – i.e. the enrollment that we wish to verify against.

Figure 5-3: Verification Start Parameters

HTTP Request Headers:

Name	Type	Description
Developer-Key	String	The Developer-Key located on the Dashboard
Application-Key	String	The Application specific Key located on the Dashboard
Interaction-Id	String	(Optional) An external session id, used to correlate internal data with.

Figure 5-4: Verification Start Request Headers

HTTP Response Headers:

Name	Type	Description
VV-Session-Id	String	The session Id required to be sent with subsequent processing calls.

Figure 5-5: Verification Start Response Headers

5.1.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Verification Start Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error":{
    ...
  },

  /* Profile information only present on verification start message */
  "profile.verify":{

    /* Index number of the profile */
    "index":0,

    /* Profile Kind: Independent - 1, Liveness - 2 */
    "kind":0,

    /* Profile Type: Single - 2 */
    "type":0,

    /* Type of Audio Codec: pcm_little_endian -or- alaw */
    "codec":"","

    /* Passing Threshold */
    "pass":0

    /* Failing Threshold */
    "fail":0
  }
}

```

5.1.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error description code	Description
101	Session Exception
104	Developer has reached their limit
107	Unable to acquire a session
202	Unable to start the feature extractor
402	Unable to start the verification processor
408	Invalid Application Key / Application Profile Not Found
409	Invalid Parameter specified
410	No Model found for the specified client id
412	Verification limits have been reached

Figure 5-6: Verification Start Error Codes

5.2 Sending Voice Data

Sending Voice Data can be called multiple times even with the same data.

POST - <http://v2ondemand.com:50202/1/sve/Verification>

HTTP Request Headers:

Name	Type	Values	Description
VV-Session-Id	String		The Id received in the start verification result.

Figure 5-7: Verification Process Session ID Parameter

Additional Notes:

- Uses File Upload Standard RFC 1867.
- An HTTP request submitted using the POST method with a content-type of "multipart/form-data".
- The audio data to be transmitted needs to be encoded by the same audio codec that was received with the *profile.verify* JSON object from the start verification HTTP Data Stream. The audio data is **required** to be transmitting as a raw wav stream without any "RIFF" headers.

Streaming Data:

Name	Type	Values	Description
data	binary	Max file size is 2 megabytes	Form data field name to store binary data is "data"

Figure 5-8: Verification Process Data Parameter

5.2.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Verification Process Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error":{
    ...
  },

  /* Present on verification process and end messages */
  "result.verify":{

    /* The client_id used in the verification start message */
    "{client_id}":{

      /* Index number of the profile */
      "index":0,

      /* Current processing call sampling error */
      "error":0,

      /* Number of seconds of speech extracted from all voice samples thus far */
      "seconds_extracted":0.0,

      /* Score determined from all samples thus far */
      "score":0.0,

      /* Status of score. (P)assing, (A)mbiguous, (F)ailing */
      "status":"P"
    }
  }
}

```

Sampling Error Code	Description
0	Success
403	Need More Voice

Figure 5-9: Verification Sampling Codes

5.2.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error description code	Description
101	Session Exception (See Log)
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
104	Developer Limits Reached
412	Verification Limits Reached
416	Invalid Post Data, Unknown stream type, Missing form-data
417	Invalid Post Data, No Stream Found, or No voice found in stream
418	Invalid Post Data, Stream Found, Missing data field
701	Error occurred while updating database

Figure 5-10: Verification Process Status Codes

5.2.3 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Verification Process Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error": {
    ...
  },

  /* Present on verification process and end messages */
  "result.verify": {

    /* The client_id used in the verification start message */
    "client_id": {

      /* Index number of the profile */
      "index": 0,

      /* Current processing call sampling error */
      "error": 0,

      /* Number of seconds of speech extracted from all voice samples thus far */
      "seconds_extracted": 0.0,

      /* Score determined from all samples thus far */
      "score": 0.0,

      /* Status of score. (P)assing, (A)mbiguous, (F)ailing */
      "status": "P"
    }
  }
}

```

Sampling Error Code	Description
101	Session Exception (See Log)
403	Need More Voice

Figure 5-11: Verification Liveness Sampling Codes

5.2.4 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Sampling Error Code	Description
0	Success
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
104	Developer Limits Reached
412	Verification Limits Reached
416	Invalid Post Data, Unknown stream type, Missing form-data
417	Invalid Post Data, No Stream Found, or No voice found in stream
418	Invalid Post Data, Stream Found, Missing data field
420	Returns a Non-Live State
701	Error occurred while updating database

Figure 5-12: Verification Liveness Process Status Codes

5.3 Sending Live Voice Data

Sending live voice data, or more commonly referred to as a liveness test, is a proven method of ensuring the person verifying is alive, present at the time the voice is captured, and is not a recording.

POST - http://v2ondemand.com:50202/1/sve/Verification/Liveness/{language code}/{liveness_text}

NOTE: {liveness_text} currently only accepts 4 numbers, and they must each be separated by a single space. For example, if you were to send in the number set 1234, it should be sent as 1 2 3 4. More importantly the text should be URL escaped, which looks like 1%202%203%204. Any other character sequence not formatted as defined above, will return a 404 Not found error.

HTTP Request Headers:

Name	Type	Values	Description
Vv-Session-Id	String		The Id received in the start verification result.

Figure 5-13: Verification Liveness Process Session ID Parameter

Additional Notes:

- Uses File Upload Standard RFC 1867.
- An HTTP request submitted using the POST method with a content-type of "multipart/form-data".
- The audio data to be transmitted needs to be encoded by the same audio codec that was received with the *profile.verify* JSON object from the start verification HTTP Data Stream. The audio data is **required** to be transmitting as a raw wav stream without any "RIFF" headers.

Streaming Data:

Name	Type	Values	Description
data	binary	Max file size is 2 megabytes	Form data field name to store binary data is "data"

Figure 5-14: Verification Liveness Process Data Parameter

5.3.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Verification Process Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error": {
    ...
  },

  /* Present on verification process and end messages */
  "result.verify": {

    /* The client_id used in the verification start message */
    "{client_id}": {

      /* Index number of the profile */
      "index":0,

      /* Current processing call sampling error */
      "error":0,

      /* Number of seconds of speech extracted from all voice samples thus far */
      "seconds_extracted":0.0,

      /* Score determined from all samples thus far */
      "score":0.0,

      /* Status of score. (P)assing, (A)mbiguous, (F)ailing */
      "status":"P"

    }
  },

  /* Present on liveness verification process */
  "result.liveness": {

    /* Result of Liveness. Boolean: True or False */
    "is_alive": true
  }
}

```

Sampling Error Code	Description
101	Session Exception (See Log)
403	Need More Voice

Figure 5-15: Verification Liveness Sampling Codes

5.3.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Sampling Error Code	Description
0	Success
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
104	Developer Limits Reached
412	Verification Limits Reached
416	Invalid Post Data, Unknown stream type, Missing form-data
417	Invalid Post Data, No Stream Found, or No voice found in stream
418	Invalid Post Data, Stream Found, Missing data field
420	Returns a Non-Live State
701	Error occurred while updating database

Figure 5-16: Verification Liveness Process Status Codes

5.4 Finalize a Verification

Finalize the verification session and get the final score result.

DELETE - <http://v2ondemand.com:50202/1/sve/Verification>

HTTP Request Headers:

Name	Type	Values	Description
Vv-Session-Id	String		The Id received in the start verification result.

Figure 5-17: Verification Terminate Session ID Parameter

5.4.1 Successful Result

HTTP Status Code: 200 (Success)

```

/* HTTP 200 - JSON Verification End Response */
{
  /*
   * Only present if a catastrophic error occurs inside the API.
   * Data inside this block should be sent to ValidVoice for investigation.
   */
  "error": {
    ...
  },

  /* Present on verification process and end messages */
  "result.verify": {

    /* The client_id used in the verification start message */
    "{client_id}": {

      /* Index number of the profile */
      "index": 0,

      /* Current processing call sampling error */
      "error": 0,

      /* Number of seconds of speech extracted from all voice samples thus far */
      "seconds_extracted": 0.0,

      /* Score determined from all samples thus far */
      "score": 0.0,

      /* Status of score. (P)assing, (A)mbiguous, (F)ailing */
      "status": "P",

      /*
       * Has the client been authorized. Is status equal to P. true -or- false
       * NOTE: if liveness failed, but the voice was still recognized, the authorized will
       * still be false
       */
      "authorized": true
    }
  }
}

```


5.4.2 Failure Result

HTTP Status Code: 500 (Internal Server Error)

Error description code	Description
101	Session Exception (See Log)
102	Invalid Session (Session Timeout?)
103	Corrupt Session Id
701	Error occurred while updating database

Figure 5-18: Verification Terminate Status Codes

6. Audio Formats

The engine currently supports the following audio formats:

- Wave A-Law, 8000 Hz, 64 kbps, mono
- Wave PCM signed 16-bit, 8000 Hz, 128 kbps, mono

The expected audio format is returned in the Enrollment and Verification initialization response JSON messages located in the field **codec**. Please note, that failure to send in the audio format in any of the specified formats above may result in unintended results, such as for example, an unusually high amount of extracted speech. The extracted speech should always be less than the amount of audio sent to the engine.

7. Behavioral Enhancement Pin

As with other biometric solutions, a person's demeanor and physical environment can have either a positive or a negative outcome on the biometric result. V2verify has recognized this trait and has implemented a failsafe that can be used to both enhance a person's enrollment voiceprint and to guarantee that the person will always be authorized provided he/she enters the correct pin when prompted. Enhancing a person's enrollment voiceprint trains the engine to recognize the different physical environments or the different demeanors in which a person may use their voice for authorization.

7.1 Submitting the Pin During Enrollment

The pin should always be submitted at the starting phase of the enrollment (see section 4.1). The HTTP POST Method URL remains the same as defined in section 4.1, and the only addition to this API call is the mandatory addition of a header field (see below).

HTTP Request Headers:

Name	Type	Description
Vv-Override-Token	String	The "Training Pin" value entered by the user during registration.

Figure 7-1: Enrollment Start Header for Enhancement Pin

7.2 Submitting the Pin During Verification

The ability to use the behavioral enhancement pin during verification depends on server-side conditions. Please note, when an "overrideable" field is returned in the response message during a process interaction and its value is true, you should prompt users for their training-pin. You should then finalize the verification (see section 5.4) making sure to include the pin in the required header field as described below in figure 7-2. The "overrideable" field's visibility in the response message is also dependent on the server-side conditions becoming true.

```

/* HTTP 200 - JSON Verification Process Response */
{
  "result.verify":{
    /* The client_id used in the verification start message */
    "{client_id}":{
      ...
      /* Can the verification be overridden? */
      "overrideable": true/false,
      ...
    }
  }
}

```

HTTP Request Headers:

Name	Type	Description
Vv-Override-Token	String	The "Training Pin" value entered by the user during registration.

Figure 7-2: Verification Delete Header for Enhancement Pin

8. Best Practices for Implementation

V2verify recommends that users who are developing their own interfaces should consider the following methods to implement the on-boarding (Enrollment) and User Authentication processes. Please note that different on-boarding methods can be used based on how the process is implemented.

8.1 Natural Language Enrollment (On-Boarding during an existing call with an Agent (Patented Process))

When implementing a natural language enrollment, the inbound phone channel is monitored, and the live voice used for the process. The benefit to both organization and customer is that the customer/enrollee does not need to utter any specific words or sentence, thus reducing friction, and startup cost. The natural language speech is then sent to v2ondemand which is our voice biometric engine solution (www.v2ondemand.com) system for enrollment.

The engine will return the results of the enrollment, indicating whether it accepted the enrollment, whether it rejected the enrollment and a possible reason why, or whether it requires more speech based on the minimum seconds of speech parameter configured for this specific application instance.

V2verify recommends a bare “minimum seconds of speech” enrollment value of **40** seconds. Please note, seconds of speech here indicates trained speech, that is, the seconds of speech the engine was able to extrapolate from the raw audio it received. The number of seconds of raw audio captured is generally more than the number of seconds the engine extrapolates.

8.2 Smartphone App or IVR Enrollment

The application should be developed to guide the user through the enrollment process step by step. When enrolling with this method, it is always good to use a set of questions during the enrollment process. These questions should be asked up to 3 times each to ensure that there is enough variance in the captured voice. A recommended question set, and sequence of questions are as follows:

- Please say your full name
- Please say your full address
- Please repeat your full name
- Please repeat your full address
- Please repeat your full name one last time
- Please repeat your full address one last time
- Please count from zero to nine
- Additional Questions if needed:
 - Please say the days of the week
 - Please say the months of the year
 - Please say the four seasons of the year.

The engine will return the results of the enrollment, indicating whether it accepted the enrollment, whether it rejected the enrollment and a possible reason why, or whether it requires more speech based on the minimum seconds of speech parameter configured for this specific application instance.

V2verify recommends a bare “minimum seconds of speech” enrollment value of **12** seconds. Please note, seconds of speech here indicates trained speech, that is, the seconds of speech the engine was able to extrapolate from the raw audio it received. The number of seconds of raw audio captured is generally more than the number of seconds the engine extrapolates.

8.3 Natural Language Verification

When implementing a natural language verification, the inbound phone channel is monitored, and the live voice used for the process. The user does not need to utter any specific words or sentence, thus reducing friction and overall continuous cost. The natural language speech is then sent to v2ondemand which is our voice biometric engine cloud solution (www.v2ondemand.com) system for enrollment.

The system will return the results of the verification. It will either return a verified, not verified or ambiguous result or it may ask for more speech. The application should capture and send more speech to www.v2ondemand.com, if requested.

8.4 Mobile App or IVR Verification

The application should be developed to guide the user through the verification process step by step. The process should resemble the outline below:

- Ask the user “Please say your full name”
- Either display a set of 4 numbers (pin pattern) and ask the user to repeat them on mobile apps, or say a set of 4 numbers and ask the user to repeat them on an IVR for a liveness test.
- Additional Questions, if required:
 - Ask the user “Please say your full address”
 - Ask the user “Please say your complete phone number”
 - Ask the user “What day of the week is it today”

The system will return the results of the verification. It will either return a verified, not verified or ambiguous result or it may ask for more speech. The application should capture and send more speech to www.v2ondemand.com, if requested.

