ViGo® REST API Guide

Title: ViGo REST API Guide

Part number: VV/VIGO/DOC/186/B

VoiceVault Inc.

400 Continental Blvd

6th Floor

El Segundo

CA 90245

USA

(310) 426 2792

info@voicevault.com

Table of Contents

Introduction

The ViGo REST API is implemented as a REST-based XML service exposing the eight most common ViGo operations that are used to drive the user voice registration (enrollment) and 'login' (verification).

ViGo Overview

The ViGo biometric operations require the following elements to be in place:

- A *configuration* that defines the phrase to be used
- *Registered claimant identifiers* for each user of the system

Once the above elements are in place, the application can begin enrolling and verifying users.

Claimants

A *claimant* is the representation of a real world individual within the ViGo system.

The claimant is identified within the ViGo system by a unique blind identifier that stays with the voice model for that claimant for the life of the voice model. As a consequence of the blind identifier, no personal information about the claimant is known to or stored by the ViGo biometric system. The identifier is mapped to the real user in the customer application – outside of the ViGo system.

A claimant can enroll against one or more languages; these enrolments can then be used with any configuration relating to those languages (configurations are described in more detail below).

Access control to the voice biometric engine (i.e. which of the possible users are allowed to voice verify under a specific configuration) must be handled by the external application.

Claimant identifiers can be disabled and this can be used to revoke that identifier (they can also be be deleted). Disabling an identifier preserves the audit trail / history associated with the identifier. Identifiers must not be re-used or recycled for a different real-world user.

In order to obtain a valid claimant identifier an external application must register that claimant before it can be used. This registration operation creates the necessary information within the ViGo system to support the claimant identifier and then returns the identifier to the external application so that it can be mapped to information relating to an actual person.

Remember that when ViGo generates a claimant identifier for use by the external application, ViGo has no knowledge of whom the identifier will be assigned to, and nor will it ever need to.

Note: The REST API will reject an attempt to use a claimant identifier that has not previously been registered with the VoiceVault system.

Configurations

A ViGo *configuration* is a related group of settings that defines a single use case for the ViGo system. Each configuration is uniquely identified and this identifier is used by customer applications to call the VoiceVault biometric functions to obtain voice verification functionality.

Dialogue

A *dialogue* is the term used to refer to an on-going voice verification session through the ViGo REST API.

When a customer application needs to enroll or verify a claimant it must first instruct the API to start a dialogue.

The dialogue directs the customer application to gather speech from the claimant, and the speech is submitted into the dialogue. On an ongoing basis the dialogue provides a summary of the current status, which indicates whether (and what) further speech is required and whether the claimant is accepted or rejected.

Interacting with the API

The ViGo programming model is very simple - the same interaction model is used both for enrolment and verification. This greatly simplifies application development with the ViGo REST API.

For more information on the API logic flow, see the VoiceVault ViGo Getting Started Tutorial available on the VoiceVault website.

Enrolment

The process flow for enrolling a user with the ViGo system is shown in Figure 1.

The application must start by obtaining a claimant identifier for the user using **RegisterClaimant**. It is likely that each application installation (or device) will require a separate claimant identifier, but it is also possible for an application to allow a user to register and manage multiple identities.

Once an identifier has been obtained the application makes a call to start the dialogue (using **StartDialogue**) with the ViGo system, which responds by requesting that the user speak a 4-digit phrase.

The application must record the user speaking the required phrase and send the recording back to ViGo via the **SubmitPhrase** method, which in turn provides status on the progress of the dialogue and the next phrase (if any) to prompt the user for. The configuration settings determine the behavior of the **SubmitPhrase** method.

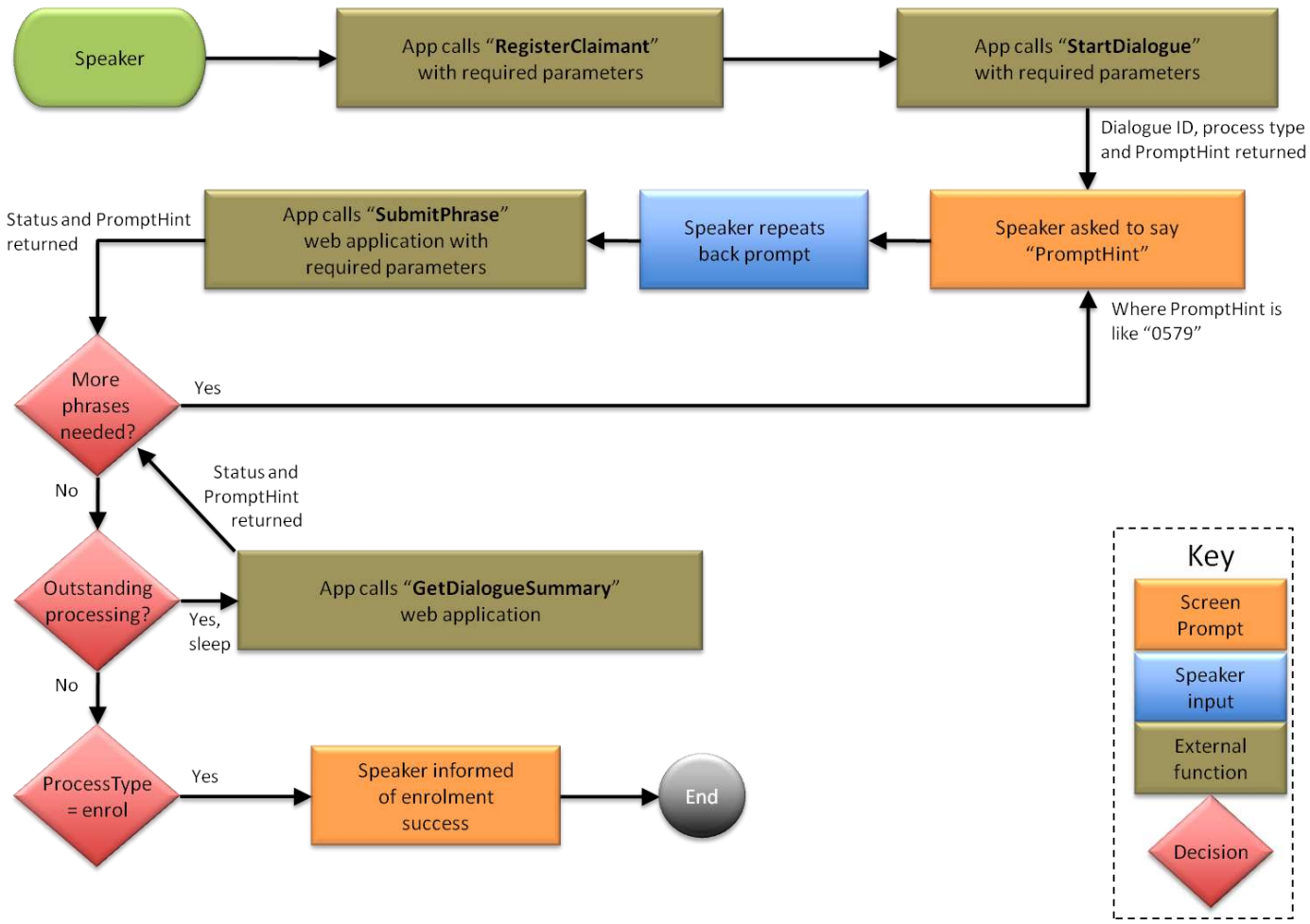

Figure 1 - Process flow for enrolling a user with ViGo

Once the application has provided ViGo with sufficient phrases (as defined by the configuration) the application enters a polling state, waiting for all of the user inputs to complete processing. At this point the system indicates back to the application either that additional phrases are required or that the process is complete.

Verification

The process flow for verifying a user with ViGo is shown in Figure 2.

In this case the application already knows the claimant identifier (as it was created and used at enrolment time).

Exactly as with enrolment, the application makes a call to start the dialogue.

The application then records the user speaking the required phrase and sends the recording back to ViGo via the **SubmitPhrase** method, which in turn provides status on the progress of the dialogue and the next phrase (if any) to speak.

Once the application has provided ViGo with sufficient phrases, the application enters a polling state, waiting for all of the inputs to complete processing. At this point the system indicates either that additional phrases are required or that the process is complete and that the user has been successfully verified or has been rejected. The application will decide how to proceed once it has been informed of the verification / rejection decision.

Note that the accept / reject threshold settings within the configuration determine if a user will be accepted or rejected. These threshold values are not modifiable by application developers.

For more information on the API logic flow, see the VoiceVault ViGo Getting Started Tutorial available on the VoiceVault website.

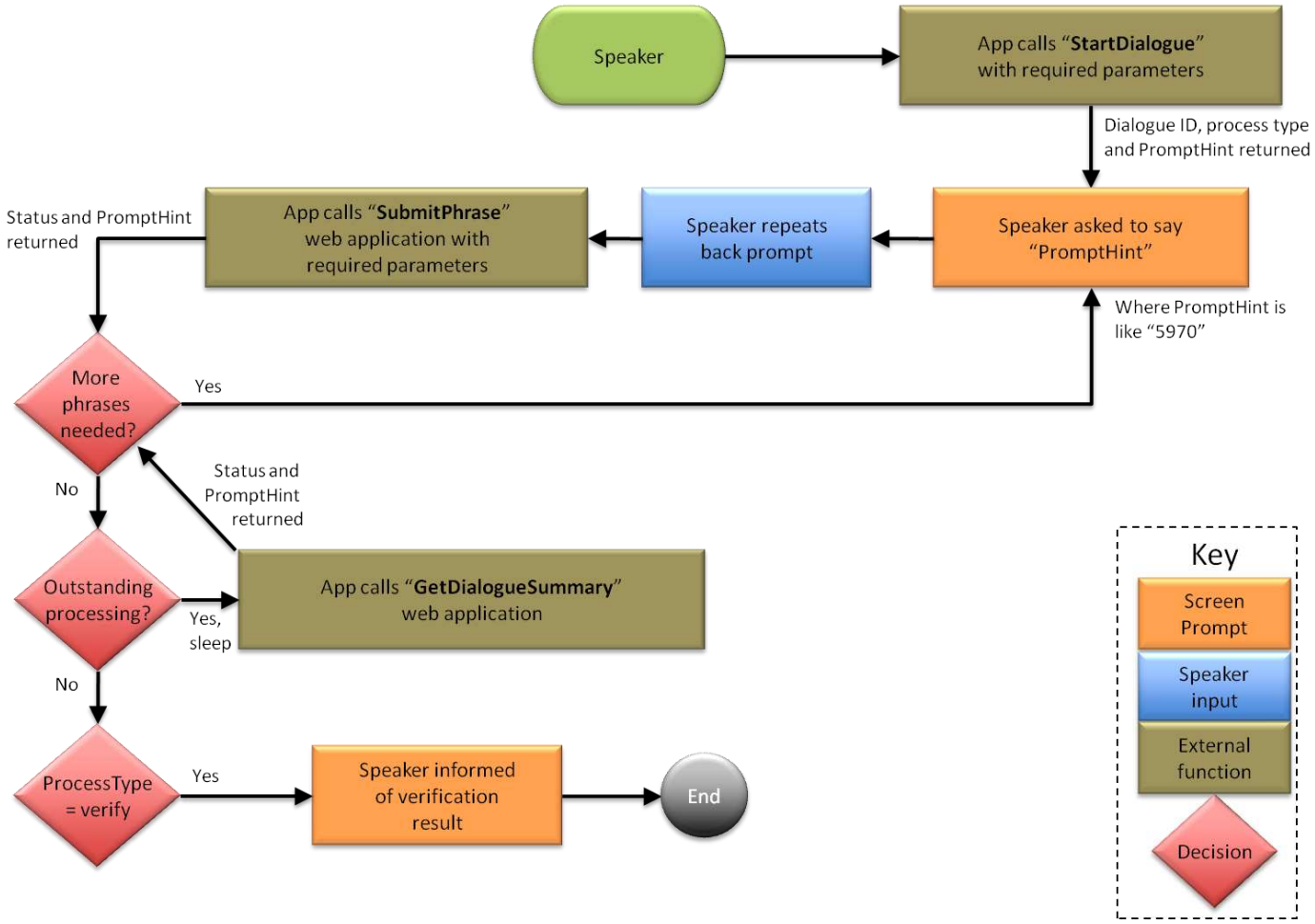

Figure 2 - Process flow for verifying a user with ViGo

Using the ViGo REST API

The REST API is called using simple HTTP operations, either GET or POST.

If a method name is prefixed with the word “Get” then the HTTP method to use is GET, otherwise the method to use is POST. There is only one GET method in the current version of the REST API.

The URL prefix for the REST API will be provided to you in a separate document.

As credentials are passed in plain text to the REST API, all communications must be over an SSL transport.

A typical HTML form to submit a request to the REST API will look similar to this:

```
<html>
  <head>
    <title> REST API Test </title>
  </head>
<body>
  <form method="POST"
     action="https://server/RestApi/RegisterClaimant.ashx">
    <label for="username">Username:</label>
   <input type="text" id="username" name="username" value=""/>
    <br/>
    <label for="password">Password:</label>
<input type="password" id="password" name="password" value=""/>
    <br/>
    <label for="organisation_unit">Org Unit Id:</label>
<input type="text" id="organisation_unit" name="organisation_unit"
   value=""/>
    <br/>
    <input type="submit" name="Submit" value="Submit"/>
```

```
  </form>
</body>
</html>
```

A typical XML response will look like this:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<response_info xmlns="http://www.voicevault.com">
  <status_code>0</status_code>
  <message>success</message>
  <claimant_id>86e5c4a7-91e2-4854-8096-c400620977b1</claimant_id>
</response_info>
```

If an error occurs during processing the status code returned will be non-zero and the accompanying XML tag “message” will contain a human readable string that may assist with debugging the problem. In this error case none of the other method dependent tags will contain data. The third party application must be prepared to parse empty tags.

Biometric Operations

The operations exposed by the ViGo REST API are as follows:

1. **RegisterClaimant** – this operation is used to obtain a blinded identifier that can be used to associate a real world identity in your application with a biometric identity in the ViGo system.

2. **StartDialogue** – this operation is always the first to be called when initiating a biometric process, for either enrolment or verification.

3. **SubmitPhrase** – once a dialogue has been initiated, this operation is used to submit utterances for processing.

4. **GetDialogueSummary** – this method is used when the user does not wish to submit any further phrases for the current dialogue, and simply wishes to see the current processing status.

5. **AbortDialogue** - this method should be used when the caller hangs up prematurely. Note that if this method is **not** used when a caller hangs up during enrolment, the claimant identifier associated with that caller cannot immediately be used to begin a subsequent enrolment dialogue.

6. **AdaptClaimant** – this operation can be used to introduce new verification audio material into a claimants' enrollment model.

7. **DeleteClaimant** – this method can be used to ensure that all biometric information is removed for a particular claimant.

RegisterClaimant

This operation is used to register a new claimant within the ViGo system.

It creates a new claimant record in the database within the specified organization unit and returns a new claimant identifier GUID. Newly registered claimants are enabled by default.

You can find your **VIGO_APP_ID**, as well as your **VIGO_CREDENTIAL_ID** and **VIGO_CREDENTIAL_PWD** in the welcome email you received when you registered. This is also where you will find the **VIGO_SERVER_URL.**

The identifier returned should be used to reference a specific user in all future transactions and must remain associated with that user for the life of the application. Identifiers must not be re-used, or recycled, with a different real-world user. A user can, however, have multiple claimant identifiers associated with them and the application will manage the mapping of the identifier(s) to real-world data relating to the user.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https://VIGO_SERVER_URL/RegisterClaimant.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the*

	REST API. *Provided as part of your API access credentials.* *Must not be NULL.*
`organisation_unit`	**VIGO_APP_ID** *The organization unit within which the claimant is to be registered.* *Provided as part of your API access credentials.* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*
`claimant_id`	*The claimant identifier that has been registered in the system and can be used for future biometric operations.*

StartDialogue

This operation creates a new dialogue that is then used as a reference for all submitted phrases in the current enrolment or verification attempt.

The operation requires:

- The **CONFIGURATION_ID** of the configuration to be used for this enrolment or verification attempt
- The claimant against which to enroll or verify
- A user reference that can be used later for reporting purposes.
- The language of the passphrase (language is a required parameter even though it is ignored for verification modes other than passphrase).

The **CONFIGURATION_ID** for use with this operation will have been provided in the welcome email you received when you registered.

The claimant ID must have been previously created using the **RegisterClaimant** method.

The value for the **externalCallRef** parameter can be either left empty (null), or populated with a reference that can later be used.

The **Language** parameter is "EnglishUnitedStates" unless otherwise communicated to you (as will the case if you purchase additional phrases in other languages).

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /StartDialogue.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API*

	access credentials. *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`organisation_unit`	**VIGO_APP_ID** *The organization unit within which the claimant is to be registered.* *Provided as part of your API access credentials.* *Must not be NULL.*
`claimant_id`	*The claimant identifier, against which the dialogue should be started, previously returned by the RegisterClaimant operation.* *Must not be NULL.*
`external_ref`	*An arbitrary reference you create that will be used to tag the dialogue to assist you with future searching.*
`language`	*An enumerated type that is only used in passphrase mode (it is a required parameter but it is ignored in other modes).* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*
`dialogue_id`	*This GUID must be stored by the application, since it must be supplied as a parameter to all future operations for the current dialogue.*
`process_type`	*This is an enumeration that informs the user which type of dialogue has been created. This is entirely dependent on the status of the supplied claimant.* *If the claimant is not enrolled, the* ***ProcessType*** *will be ENROL, otherwise VERIFY.*
`prompt_hint`	*This, if set, tells the user which phrase should be submitted next for the current dialogue.*

Once a dialogue has been successfully created, as indicated by a **StatusCode** of "**0**" and a valid **dialogue_id**, the subsequent operations should be used to interact with it.

SubmitPhrase

See also **SubmitPhraseBase64.**

This operation is used to append a claimant's utterance to an existing dialogue.

It must be submitted using content type = "multipart/form-data" to allow the binary parameters to be correctly uploaded.

The operation requires:

- The dialogue ID that was returned by the call to **StartDialogue**
- The phrase that the user is believed to have spoken
- The sample format that the audio data is supplied in
- The binary audio data in a supported audio file format (see below)

The binary audio data array must contain a sound sample whose attributes depend on the **SampleFormat**:

Format	Details
U-Law	• 8KHz • 8-bit
A-Law	• 8KHz • 8-bit
Linear PCM	• 8KHz • 16-bit • BigEndian or LittleEndian • Mono

The **SampleFormat** enumeration value that must be supplied has the following members:

```
public enum SampleFormats

{

      Unknown,
```

```
        ALaw,
        MuLaw,
        LittleEndian,
        BigEndian
}
```

Note that if `SampleFormats.Unknown` is supplied, and the associated utterance data does not contain a RIFF header that can be successfully parsed, an error will be returned since the phrase will be unusable.

Method: POST

Encoding: multipart/form-data

URL: https:// VIGO_SERVER_URL /SubmitPhrase.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`dialogue_id`	*The dialogue ID that was returned*

	*by the call to **StartDialogue**.* *Must not be NULL.*
`prompt`	*The phrase that the user is believed to have spoken.* *See below for allowed characters.* *Must not be NULL.*
`format`	*The sample format that the audio data is supplied in, as described above.* *Must not be NULL.*
`utterance`	*The raw binary data of the audio recording, which should be submitted using content type = "**multipart/form-data**".* *Must not be NULL.*

The `prompt` parameter only accepts the following values:

- 0-9
- a-z (lower-case)
- A-Z (upper-case)
- Space (" ")
- ASCII characters C0-FF (accented alphabetic characters taken from the ISO-8859-1 extended ASCII character set.

In practice this means that even if a passphrase (for example) contains punctuation, that punctuation must not be passed in the `prompt` parameter.

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*
`dialogue_status`	*A value indicating whether the dialogue is in progress has been successful or has failed.* *See* DialogueStatus Enumeration *below for more details.*
`failure_reason`	*If the dialogue was not successful then this field provides more detail.* *See* FailureReason Enumeration *below for more details.*
`request_status`	*This value provides the status of this particular call to* ***SubmitPhrase****.* *See* RequestStatus Enumeration *below for details.*
`prompt_hint`	*If the dialogue has not completed then this value tells the user which phrase should be submitted next.*
`last_sqm_decision`	*The result of the speech quality measurement (SQM) decision on the previously submitted speech*

	sample (not the current one).
`verification_decision`	*The verification decision that has been made for this dialogue (if applicable).* *See appendix for possible values.*

Note that you will only receive the **WrongPhraseSubmitted** value in the **request_status** field when the **PromptedPhrase** attribute passed in **SubmitPhrase** does not match the expected **PromptHint** as returned to you by the previous **SubmitPhrase** or **GetDialogueSummary** call.

For example, if you pass "6379" as the **PromptedPhrase**, but the actual utterance passed is different (say “1234”), then it would be expected that this utterance would be rejected during the server side processing as an SQM failure ("MISSPEAK") rather than being rejected on the client side. In other words, the client has no means of knowing that the utterance data does not match the **PromptedPhrase** – the submission therefore needs to be processed and rejected by SQM on the server side. Note: If an utterance is so rejected then the system simply continues requesting additional utterances to replace it.

The following conditions control the process flow of an application:

- If dialogue_status is “**Started**” and request_status is “**OK**” then more phrases are needed
- If dialogue_status is “**Started**” and request_status is “**TooManyUnprocessedPhrases**” then processing is being carried out and the application should poll GetDialogueSummary
- If dialogue_status is “**Succeeded**” and process_type is “**Enrol**” then the enrolment completed successfully
- If dialogue_status is “**Succeeded**” and process_type is “**Verify**” then the verification attempt was **accepted**
- If dialogue_status is “**Failed**” and process_type is “**Verify**” and failure_reason is “**VerificationFailed**” then the verification attempt was rejected
- Any other combination of values is considered an error

SubmitPhraseBase64

See also **SubmitPhrase.**

This operation is exactly the same as **SubmitPhrase** but allows the upload of audio to be inline in a standard "application/x-www-form-urlencoded" body rather than as a "multipart/form-data" body. This is to support limited environments that do not have multipart MIME support. The disadvantage of this content type is that a Base64 encoded audio file is significantly larger than the raw audio, thus consumes more bandwidth, and the encoding consumes more CPU time on the client.

The operation requires:

- The dialogue ID that was returned by the call to **StartDialogue**
- The phrase that the user is believed to have spoken
- The sample format that the audio data is supplied in
- The binary audio data in a supported audio file format (see below), Base64 encoded

The Base64 encoded binary audio data array must contain a sound sample whose attributes depend on the **SampleFormat**:

Format	Details
U-Law	• 8KHz • 8-bit
A-Law	• 8KHz • 8-bit
Linear PCM	• 8KHz • 16-bit • BigEndian or LittleEndian • Mono

The **SampleFormat** enumeration value that must be supplied has the following members:

```
public enum SampleFormats

{
```

```
        Unknown,
        ALaw,
        MuLaw,
        LittleEndian,
        BigEndian
}
```

Note that if `SampleFormats.Unknown` is supplied, and the associated utterance data does not contain a RIFF header that can be successfully parsed, an error will be returned since the phrase will be unusable.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /SubmitPhraseBase64.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.*

	Provided as part of your API access credentials. *Must not be NULL.*
`dialogue_id`	*The dialogue ID that was returned by the call to* ***StartDialogue****.* *Must not be NULL.*
`prompt`	*The phrase that the user is believed to have spoken.* *See below for allowed characters.* *Must not be NULL.*
`format`	*The sample format that the audio data is supplied in, as described above.* *Must not be NULL.*
`utterance`	*The* ***Base64*** *encoded binary data of the audio recording.* *Must not be NULL.*

The `prompt` parameter only accepts the following values:

- 0-9
- a-z (lower-case)
- A-Z (upper-case)
- Space (“ “)
- ASCII characters C0-FF (accented alphabetic characters taken from the ISO-8859-1 extended ASCII character set.

In practice this means that even if a passphrase (for example) contains punctuation, that punctuation must not be passed in the `prompt` parameter.

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*
`dialogue_status`	*A value indicating whether the dialogue is in progress has been successful or has failed.* *See* DialogueStatus Enumeration *below for more details.*
`failure_reason`	*If the dialogue was not successful then this field provides more detail.* *See* FailureReason Enumeration *below for more details.*
`request_status`	*This value provides the status of this particular call to* ***SubmitPhrase****.* *See* RequestStatus Enumeration *below for details.*

`prompt_hint`	*If the dialogue has not completed then this value tells the user which phrase should be submitted next.*
`last_sqm_decision`	*The result of the speech quality measurement (SQM) decision on the previously submitted speech sample (not the current one).*
`verification_decision`	*The verification decision that has been made for this dialogue (if applicable).* *See Appendix for possible values.*

Note that you will only receive the **WrongPhraseSubmitted** value in the **request_status** field when the **PromptedPhrase** attribute passed in **SubmitPhraseBase64** does not match the expected **PromptHint** as returned to you by the previous **SubmitPhraseBase64** or **GetDialogueSummary** call.

For example, if you pass "At VoiceVault, I am verified as I speak" as the **PromptedPhrase**, but the actual utterance passed is different (say “VoiceVault knows me by the sound of my voice”), then it would be expected that this utterance would be rejected during the server side processing as an SQM failure ("MISSPEAK") rather than being rejected on the client side. In other words, the client has no means of knowing that the utterance data does not match the **PromptedPhrase** – the submission therefore needs to be processed and rejected by SQM on the server side. Note: If an utterance is so rejected then the system simply continues requesting additional utterances to replace it.

The following conditions control the process flow of an application:

- If dialogue_status is “**Started**” and request_status is “**OK**” then more phrases are needed
- If dialogue_status is “**Started**” and request_status is “**TooManyUnprocessedPhrases**” then processing is being carried out and the application should poll GetDialogueSummary

- If dialogue_status is "**Succeeded**" and process_type is "**Enrol**" then the enrolment completed successfully
- If dialogue_status is "**Succeeded**" and process_type is "**Verify**" then the verification attempt was **accepted**
- If dialogue_status is "**Failed**" and process_type is "**Verify**" and failure_reason is "**VerificationFailed**" then the verification attempt was rejected
- Any other combination of values is considered an error

GetDialogueSummary

This operation is used to retrieve the status of the current dialogue without the need to submit a phrase.

This operation is typically used when the call to **SubmitPhrase** returns a **request_status** value of `TooManyUnprocessedPhrases`.

In this scenario, it is impossible for the application to submit further phrases until the server has processed the pipeline, in which case the application must repeatedly call **GetDialogueSummary** until a valid status is received.

Method: GET

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /GetDialogueSummary.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.*

	Must not be NULL.
`dialogue_id`	*The dialogue ID that was returned by the call to **StartDialogue**.* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If **status_code** is non-zero then this field will indicate more precisely the cause of the error.*
`dialogue_status`	*A value indicating whether the dialogue is in progress has been successful or has failed.* *See* DialogueStatus Enumeration *below for more details.*
`failure_reason`	*If the dialogue was not successful then this field provides more detail.* *See* FailureReason Enumeration *below for more details.*
`request_status`	*This value provides the status of this particular call to*

	SubmitPhrase. *See* RequestStatus Enumeration *below for details.*
`prompt_hint`	*If the dialogue has not completed then this value tells the user which phrase should be submitted next.*
`last_sqm_decision`	*The result of the speech quality measurement (SQM) decision on the previously submitted speech sample (not the current one).*
`verification_decision`	*The verification decision that has been made for this dialogue (if applicable).* *See Appendix for possible values.*

The following conditions control the process flow of an application:

- If dialogue_status is "**Started**" and request_status is "**OK**" then more phrases are needed
- If dialogue_status is "**Started**" and request_status is "**TooManyUnprocessedPhrases**" then processing is being carried out and the application should poll GetDialogueSummary
- If dialogue_status is "**Succeeded**" and process_type is "**Enrol**" then the enrolment completed successfully
- If dialogue_status is "**Succeeded**" and process_type is "**Verify**" then the verification attempt was **accepted**
- If dialogue_status is "**Failed**" and process_type is "**Verify**" and failure_reason is "**VerificationFailed**" then the verification attempt was rejected
- Any other combination of values is considered an error

AbortDialogue

This method should be used when the caller hangs up prematurely, or a dialogue needs to be terminated early for any reason.

It will abort the dialogue and all associated unprocessed exchanges. This is especially important for enrolment dialogues, as only one enrolment dialogue may be active at a time.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /AbortDialogue.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`dialogue_id`	*The dialogue ID that was used in*

	*the call to **SubmitPhrase**.* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If **status_code** is non-zero then this field will indicate more precisely the cause of the error.*
`request_status`	*This value provides the status of this particular call to **AbortDialogue**.* *See* RequestStatus Enumeration *below for details.*

AbortDialogue

This method should be used when the caller hangs up prematurely, or a dialogue needs to be terminated early for any reason.

It will abort the dialogue and all associated unprocessed exchanges. This is especially important for enrolment dialogues, as only one enrolment dialogue may be active at a time.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /AbortDialogue.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`dialogue_id`	*The dialogue ID that was used in*

	*the call to **SubmitPhrase**.* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If **status_code** is non-zero then this field will indicate more precisely the cause of the error.*
`request_status`	*This value provides the status of this particular call to **AbortDialogue**.* RequestStatus Enumeration *below for details.*

AdaptClaimant

This operation can be used to introduce new verification audio material into a claimant's enrollment model.

The material introduced must have been previously determined to be suitable for adaptation. For example, a verification dialogue that is either a strong "Accept" or "Reject" will usually not be suitable – the former because the model would not benefit from the material, and the latter because there is significant doubt over the identity of the speaker.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /AdaptClaiamant.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*

`claimant_id`	*The claimant for whom the adaptation will take place.* *Must not be NULL.*
`dialogue_id`	*The verification dialogue from which the adaptation audio material will be taken. If the dialogue consists of more than one phrase, the phrase that is most suitable for adaptation will be selected.* *Must not relate to an enrollment dialogue.* *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*

DeleteClaimant

This operation can be used to ensure that all biometric information stored in VoiceVault databases is removed for a particular claimant.

This includes any enrolled voice models, any replay detection metadata, and all speech that has been submitted in dialogues pertaining to the specified claimant identifier.

Method: POST

Encoding: application/x-www-form-urlencoded

URL: https:// VIGO_SERVER_URL /DeleteClaiamant.ashx

Parameters

Parameter	Description
`username`	**VIGO_CREDENTIAL_ID** *The username of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`password`	**VIGO_CREDENTIAL_PWD** *The password of the service account attempting to access the REST API.* *Provided as part of your API access credentials.* *Must not be NULL.*
`claimant_id`	*The claimant for whom the*

	biometric record will be removed. *Must not be NULL.*

Outputs

Parameter	Description
`status_code`	*The status of the REST request.* *Zero means success, otherwise an error has occurred and the message field will be relevant.* *See* StatusCode *section below for more details.*
`message`	*If* ***status_code*** *is non-zero then this field will indicate more precisely the cause of the error.*

RequestStatus Enumeration

The **RequestStatus** enumeration is returned for all **request_status** XML fields and has the following members:

	Member	Summary
0	OK	*The request completed successfully*
1	DialogueDoesNotExist	*Caller supplied an invalid dialogue ID*
2	ExchangeDoesNotExist	*Reserved for internal use*
3	ClaimantDoesNotExist	*Caller supplied an invalid claimant ID*
4	ConfigurationDoesNotExist	*Caller supplied an invalid configuration ID*
5	RequestProcessingError	*Reserved for internal use*
6	GeneralSystemError	*A system error occurred and has been logged*
7	NoSample	*Reserved for internal use*
8	InvalidProcessTypeRequest	*Reserved for internal use*
9	DuplicatePhrase	*Reserved for internal use*
10	TooManyUnprocessedPhrases	*The server is busy*

		processing previous phrases, so the current request should be retried later
11	`WrongPhraseSubmitted`	*Caller supplied a phrase that did not match the expected prompt*
12	`ClaimantIsNotEnabled`	*Claimant is currently disabled*
13	`NoSpecifiedClaimantRegistered`	*None of the claimants specified are registered*
14	`NoSpecifiedClaimantEnrolled`	*None of the specified claimants are enrolled*
15	`NoActiveSettings`	*No active settings for the specified configuration*
16	`DialogueAlreadyInProgress`	*Dialogue is already in progress for the specified claimant*
17	`AutoReEnrolment`	*Reserved for internal use*
18	`ConfigurationIsNotEnabled`	*The specified configuration is not enabled*
19	`SampleFormatMismatch`	*There is a mismatch between the audio format that was specified to that which was submitted.*

20	`SampleFormatUnsupported`	*The audio file was specified but does not contain enough information to help determine what the format actually is.* *For example, "Wave" could be specified and valid PCM audio files are submitted but without RIFF headers, so that we are unable to determine whether the file is big-endian or little-endian.*

DialogueStatus Enumeration

The following enumerations are returned from calls to **SubmitPhrase** and **GetDialogueSummary** and are used to determine what operation should be called next in the process flow.

The **DialogueStatus** enumeration is returned for all **dialogue_status** XML fields and has the following members:

	Member	Summary
0	Started	*The dialogue is still in progress*
1	Succeeded	*The dialogue has successfully completed and, for verification dialogues, the claimant has been successfully verified*
2	Failed	*The dialogue has failed. This could be due, for example, to a rejected verification attempt, or a recording being detected – see the **FailureReason** for further details*
3	Error	*An error has caused the dialogue to terminate and has been logged*
4	Aborted	*The dialogue has been purposely terminated prematurely. This could be, for example, if a caller has hung up the phone.*
5	Abandoned	*A dialogue is set to this status when it has been "Started" for a configurable period of time, without any activity, and is deemed to be no longer in use.*

FailureReason Enumeration

The **FailureReason** enumeration is returned for all **failure_reason** XML fields and has the following members (the default being **NotSet**):

Member	Summary
`NotSet`	*The dialogue was successful or is not complete*
`MaxTotalSqmFailuresExceeded`	*The threshold for the total number of permissible SQM errors within a dialogue has been exceeded*
`MaxPhraseSqmFailuresExceeded`	*The threshold for SQM errors on a single phrase has been exceeded*
`MaxTotalSystemErrorsExceeded`	*The threshold for the total number of permissible system errors within a dialogue has been exceeded*
`VerificationFailed`	*Indicates that a verification dialogue has been rejected*
`RecordingDetected`	*The threshold to determine whether a submitted utterance within this dialogue is a recording of a previous one has been exceeded*
`PassphraseInvalid`	*The phrase text contains words that are*

	not in the language lexicon. *In preactice this means that the words must be added to the language lexicon before the phrase can be used.*

StatusCode Enumeration

The **StatusCode** enumeration is returned for all REST API XML fields and has the following members:

Member	Summary
`NoError`	*The method call completed successfully and all documented outputs will be present in the XML response.*
`UnknownError`	*A non-recoverable server error occurred. Further details will be included in the message field.*
`InvalidParameter`	*The method was called with one or more invalid parameters.* *The first parameter that failed validation will be returned in the message field.*
`SoapFault`	*An error occurred delegating on to the underlying and richer ViGo SOAP API.* *This error could be server side, or it could be the result of inputs supplied by the caller. The message field will provide more detail.*
`RequestStatusError`	*The **RequestStatus** enumeration resulted in a terminal (for the current dialogue) error.* *The message field will indicate which RequestStatus value was returned.* *See* RequestStatus Enumeration *for more details.*

For all status codes other than zero the XML response will include a “message” field that gives additional human readable information about the error.

SQMDecision Enumeration

The following enumerations are returned from calls to **SubmitPhrase** and **GetDialogueSummary** and are used to determine the speech quality measurement (SQM) for the *previously* submitted phrase (NOT the one that has *just* been submitted).

Value	Member	Summary
0	NotSet	*Speech sample has not undergone SQM processing (possibly due to an error).*
1	OK	*Speech sample has passed the speech quality measaurement test/*
2	Quiet	*Speech is too quiet or no speech is heard. Could result from there being not enough signal to detect the prompted for words.*
3	Misspeak	*The spoken phrase does not contain the words that were prompted for.*
4	Noisy	*The signal to noise ratio is too low. The spoken words are indistinguishable over other, background, noise.*
5	BeginCutoff	*Data is missing from the start of the submitted speech sample. Could result from the speaker beginning to say the required phrase before the application recording process has started.*
6	EndCutoff	*Data is missing from the end of the submitted speech sample. Could be because the*

		speaker is not ebign given enough time to respond to the prompt.
7	Short	*The words in the speech sample are too close together – likely because the speaker spoke too fast.*
8	Isolated	*The words in the speech sample are too far apart together – likely because the speaker spoke too slowly and the words were not spoken so as to be a sinlge phrase.*
9	Recording	*The speech sample has been detected as being identical to a previously submitted sample.*
10	Alignment Failure	*There is a high chance that the speech sample is a recording of a previously submitted sample, but the associated dialogue will be allowed to continue.*
11	Passphrase Too Short	*The passphrase contains less than the minimum required number of phonemes.* *This message comes form the speech recognizer and is returned whenever there is an issue with the phrase (wrong words spoken; phrase cutoff; user not speaking clearly; not enough phonemes could be extracted; etc.).* *If appropriate, the user could be prompted to repeat the phrase.*

VerificationDecision Enumeration

Member	Summary
`NotApplicable`	*This is not a verification dialogue*
`NotSet`	*This is a verification dialogue that is still in progress, and no audio material has yet been processed by the verification engine*
`Uncertain`	*The dialogue is in progress, at least one phrase has been processed by the verification engine, but more audio material is required in order to make a final verification decision*
`Accept`	*The dialogue is complete and verification was successful with a high confidence level*
`Reject`	*The dialogue is complete and verification was unsuccessful with a high confidence level*
`StrongAccept`	*The dialogue is complete and verification was successful with an extremely high confidence level*
`SrongReject`	*The dialogue is complete and verification was unsuccessful with an extremely high confidence level*