ViGo® Getting Started Tutorial

ViGo Getting Started Tutorial (13 October 2015)

Part number: VV/VIGO/DOC/187/B

VoiceVault Inc.

400 Continental Blvd

6th Floor

El Segundo

CA 90245

USA

(310) 426 2792

info@voicevault.com

1. ViGo Concepts and Overview

The following is a short summary of these concepts and must be read in conjunction with the ViGo REST API documentation

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.

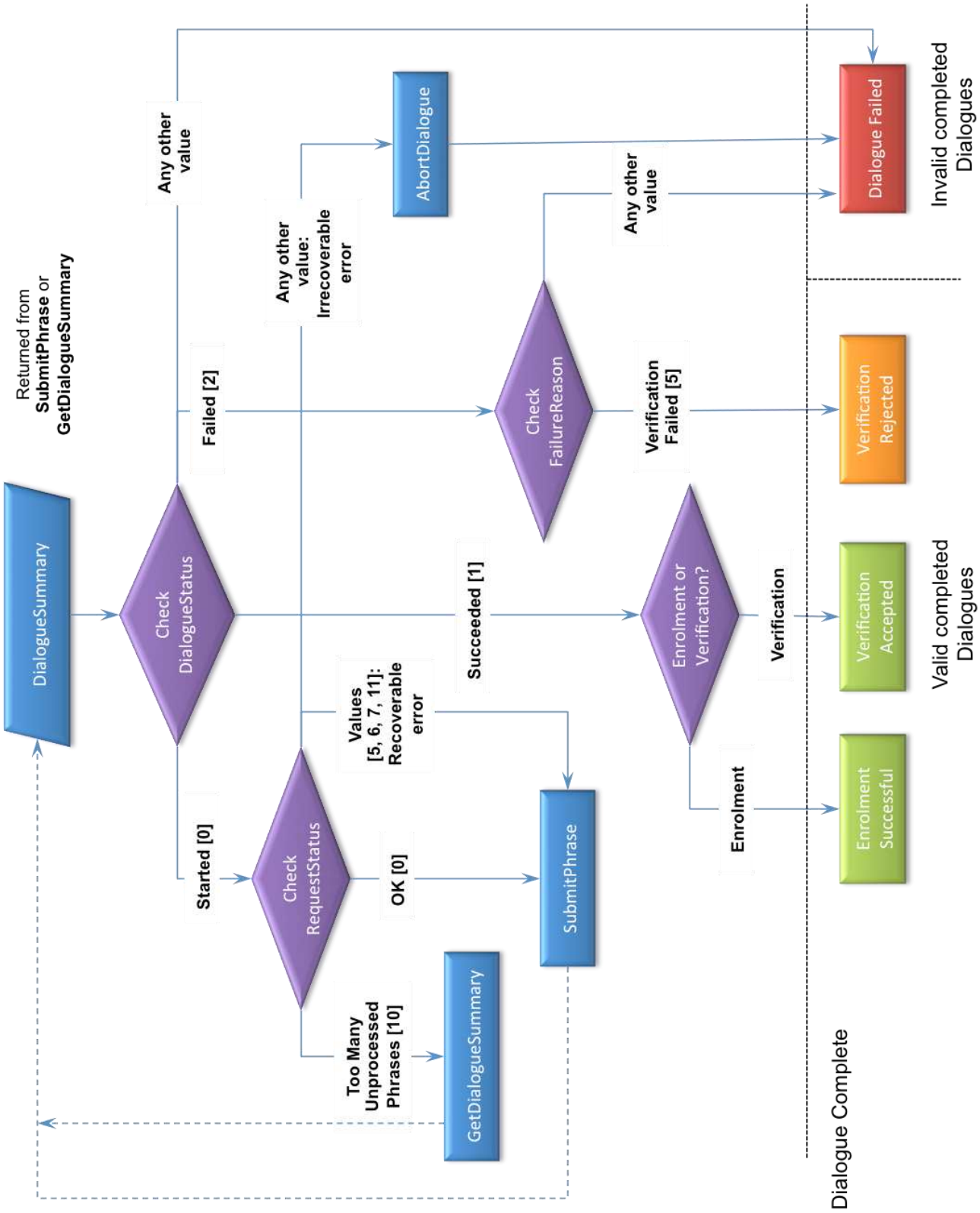
2. API Logic

The way that your application interacts with ViGo is encompassed by the logical use of the key API methods described in Section **Error! Reference source not found.** above:

- **StartDialogue**
- **SubmitPhrase**
- **GetDialogueSummary**
- **AbortDialogue**

Specifically, the values for **DialogueStatus** and **RequestStatus** define the conditional logic for what methods to call and when. This logic is defined visually as a flow chart and as pseudo code in the sections below.

Conditional Logic Flow



Conditional Logic Pseudo code

This pseudo code provides an example of the conditional logic for processing a **DialogueSummary** object returned from a call to the Biometric API **SubmitPhrase**() or **GetDialogueSummary**() methods.

Note the following:

- **AbortDialogue** should only be called when the **RequestStatus** shows that there is an irrecoverable error. This might be for example if the Configuration ID that has been passed doesn't exist – in this scenario, there is no way that the dialogue can proceed.
- An irrecoverable error is different to the **Error** state of the **DialogueStatus** as it is essentially business logic error that indicates an incorrect configuration of the application, as opposed to an unexpected exception. As such, the irrecoverable error state will be of most use at development time as a correctly configured and operational system should not, in theory, encounter an irrecoverable error.
- The decision to call **SubmitPhrase** (to prompt the user for a spoken phrase) should be based exactly on the logic described. If **PromptHint** option is enabled, and if the logic dictates that SubmitPhrase should be called, the appropriate **PromptHint** will always be present.
- The VoiceVault Fusion API is asynchronous. What this means in practice is that when you call **SubmitPhrase** you provide us with some audio that we put on a work queue and return control to you immediately, returning the current state of the dialogue at the precise instant in time. The information we return may or may not include the result of the audio you just submitted, indeed it only will include that result if we manage to process the audio instantaneously following submission. This means that your application can continue to gather additional speech from the user (if so required and indicated by the information returned to you) while we are processing your previous submission.
- In practice this means that you can execute a hard loop getting the user to speak the prompt hint and calling **SubmitPhrase** until we return **TooManyUnprocessedPhrases** (or a termination state). When **TooManyUnprocessedPhrases** is returned, you must proceed to a loop calling **GetDialogueSummary** until either a termination state is reached or you need to return to the **SubmitPhrase** loop.
- There is no need to try to wait for the completion of processing of your previous submission between **SubmitPhrase** calls, and no need to call **GetDialogueSummary** between **SubmitPhrase** calls (unless so indicated by the return status **TooManyUnprocessedPhrases**).
- It is important to build a timeout into the polling loop on **GetDialogueSummary** and an appropriate value would be around five seconds. Note however that if the timeout is triggered there is no way tot

recover the dialogue and **SubmitPhrase** should not be called again – in fact as the dialogue will have been aborted, the appropriate call would be to **AbortDialogue**. This is because the application is in a state where it has not completed processing, which needs to happen before the dialogue can proceed.

```
if (Dialogue_Status = 0)
    if (Request_Status = 0)
        //All OK, continue
        SubmitPhrase()
    else if (Request_Status = 10)
        //Wait, and poll
        GetDialogueSummary()
    else if (Request_Status = 5 or 6 or 7 or 11)
        //Recoverable error
        SubmitPhrase()
    else
        //Irrecoverable error, dialogue cannot continue
        AbortDialogue()
    end if
else if (Dialogue_Status = 1)
    if (Process_Type = 1)
        Dialogue Complete… Enrolment succeeded (valid result)
    else
        Dialogue Complete… Verification succeeded (valid result)
    end if
```

```
else if (Dialogue_Status = 2)
        if (Failure_Reason = 5)
                Dialogue Complete… Verification failed (valid result)
        else
                //Dialogue error, dialogue terminated
                Dialogue Complete… Dialogue failed (invalid result)
        end if
else
        //Dialogue error or otherwise terminated, cannot continue
        Dialogue Complete… Dialogue failed (invalid result)
end if
```

The following enumerations are used in this context:

Dialogue_Status

0 = Started

1 = Succeeded

2 = Failed

3 = Error

4 = Aborted

5 = Abandoned

Request_Status

0 = OK

1 = DialogueDoesNotExist

2 = ExchangeDoesNotExist

3 = ClaimantDoesNotExist

4 = ConfigurationDoesNotExist

5 = RequestProcessingError

6 = GeneralSystemError

7 = NoSample

8 = InvalidProcessTypeRequest

9 = DuplicatePhrase

10 = TooManyUnprocessedPhrases

11 = WrongPhraseSubmitted

12 = ClaimantIsNotEnabled

13 = NoSpecifiedClaimantRegistered

14 = NoSpecifiedClaimantEnrolled

15 = NoActiveSettings

16 = DialogueAlreadyInProgress

17 = AutoReEnrolment

18 = ConfigurationIsNotEnabled

19 = SampleFormatMismatch

20 = SampleFormatUnsupported

ProcessType

0 = Unknown

1 = Enrol

2 = Verify

3 = Migration

4 = Adapt

FailureReason

0 = NotSet

1 = MaxTotalSqmFailuresExceeded

2 = MaxPhraseSqmFailuresExceeded

3 = MaxTotalVerifyFailuresExceeded

4 = MaxTotalSystemErrorsExceeded

5 = VerificationFailed

6 = Abandoned

7 = RecordingDetected

8 = PassphraseInvalid