Reference Guide

AudioCodes Voice Recognition Engine

AC Voca API Guide

for iOS and Android

Version 1.0

audiocodes

ACVoca
powered by AudioCodes

Table of Contents

This page is intentionally left blank.

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Customer Support

For customer support, please contact your support representative or the AudioCodes support team at support@acvoca.com.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

Document Revision Record

LTRT	Description
12980	Initial document release for Version 1.0.
12981	getLicenseKey, getLicenseStatus were added; init was updated.
12982	Permissions, getWaveForm sub-sections and iOS relevant information were added.
12983	getLicenseKey was updated.
12984	PTT Recognition was added.

Related Documentation

Document Name
AC Voca Release Notes
AC Voca Administration Guide
AC Voca API Guide for Windows

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our Web site at http://online.audiocodes.com/documentation-feedback.

1 Introduction

The AC Voca Software Developer Kit (SDK) for Smartphones is designed for software developers and integrators, allowing to enhance any mobile application's user interface with voice recognition capabilities to promote easy access for app functionality using natural voice commands.

The Voice-Engine SDK includes integration to the core voice-engine, voice audio drivers, customizable vocabulary and grammar, language related data and a complete set of APIs required for adding a cutting-edge voice recognition technology to any Android or iOS application.

Using an intuitive set of natural voice commands with high recognition accuracy levels, alongside the AC Voca pure offline capability (as the voice-engine doesn't require any Internet connectivity in order to operate in real-time), mobile users can enjoy an innovative, friendly and simple user-experience, driving their mobile usage by natural voice.

This page is intentionally left blank.

2 Getting Started

This section provides the necessary steps you need to take before working with the AC Voca SDK/API.

2.1 Deliverables

The following items will be provided by AudioCodes to enable you to use AC Voca SDK:

- **Library**
  - Android
    An AAR file that packs the SDK. In addition to the SDK, we provide an additional AAR file of utilities methods and utilities classes (Currently the SDK uses some methods and features within this Utilities library).
  - iOS
    The SDK is provided as a single framework.
- **License Key**
  This key is a string that represents your license to work with the SDK. You will have to pass the license key to the AC Voca SDK during initialization.

Note: Depending on the licensing model you might need to pre-request online, a license key per device using the SDK.

- **Compiled Grammar Files (.cmg)**
  AC Voca recognition is based on a closed grammar file. Depending on your project, you may need more than one .cmg file.
- **Language Definitions Files**
  Recognition is language-based. The AC Voca SDK needs to be fed with both the recognition dialect and all the language supporting files before you can start performing recognitions.

2.2 Setting up your Android Project to Work with the SDK

The following describes how to set up your android project to work with the SDK.

2.2.1 Linking your project with the SDK library

The procedure below describes how To link your project with the SDK library.

➢ **To link your project with the SDK library:**

1. Under the application module in your project, create a folder for the AAR files.
2. In the project *build.gradle* file (not in the app module), add that folder as a repository. For instance, if you named that folder in the first step ‘aar’, your *build.gradle* file should look like the following:

```
// Top-level build file where you can add
configuration options common to all sub-
projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath
'com.android.tools.build:gradle:2.2.3'
    }
}

allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'aar'
        }
    }
```

3. Add dependencies for the AAR files into the application module *build.gradle* file. In the dependencies section add the following:

```
compile(name: 'acvocautils, ext: 'aar')
compile(name: 'acnlpsdk, ext: 'aar')
```

2.2.2 Assets

The procedure below describes the procedures for preparing your Assets.

➢ **To prepare your assets:**

1. Copy all language pack and .cmg files to the **Assets** folder.
2. Before initializing the SDK, copy these files to the SDK files folder. You can get that folder by calling the SDKs method:
   ```
   ACVocaSDK.getInstance().getVoiceEngineFilesFolder()
   ```

Note: Copy the files only once. There is no need to re-copy the files every time the application starts. You only need to re-copy the files after an update to the files.

2.2.3 Permissions

The ACVocaSDK for Android requires the following permissions in order to work properly:

- android.permission.BLUETOOTH
- android.permission.RECORD_AUDIO
- android.permission.READ_PHONE_STATE
- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE

Note: Remember that from Android SDK Version 23, the application must request permissions at runtime.

2.3 Setting your iOS project to work with the SDK

The following describes how to set up your iOS project to work with the SDK.

2.3.1 Linking your project with the SDK library

1. Place the framework file under project folder.
2. Open your target and navigate to the **General** tab.
   Under ‘Embedded Binaries’ press the '+' button and select the ACVocaSDK framework file.
   If the ACVocaSDK.framework does not appear, locate it using the ‘add other’ button.

2.3.2 Assets

The procedure below describes the procedures for preparing your Assets.

➢ **To prepare your assets:**

1. Copy all language pack and .cmg files to a folder linked to the xCode project.
2. Before initializing the SDK, copy these files to the SDK files folder. You can get that folder by calling the SDKs method:
   `ACVocaSDK.getInstance().getVoiceEngineFilesFolder()`

Note: Copy the files only once. There is no need to re-copy the files every time the application starts. You only need to re-copy the files after an update to the files.

2.3.3 Permissions

ACVocaSDK for iOS requires privacy permissions to microphone.

You should add the following key to your info.plist file:

- xCode Editor key name: `Privacy - Microphone Usage Description`
- xCode source key name: `NSMicrophoneUsageDescription`

2.4 Working with the ACVocaSDK

2.4.1 Initializing the ACVocaSDK

Once your workspace is ready, initialize the ACVocaSDK by following the steps as shown in the figures below.

Figure 2-1: ACVocaSDK Android Initialization

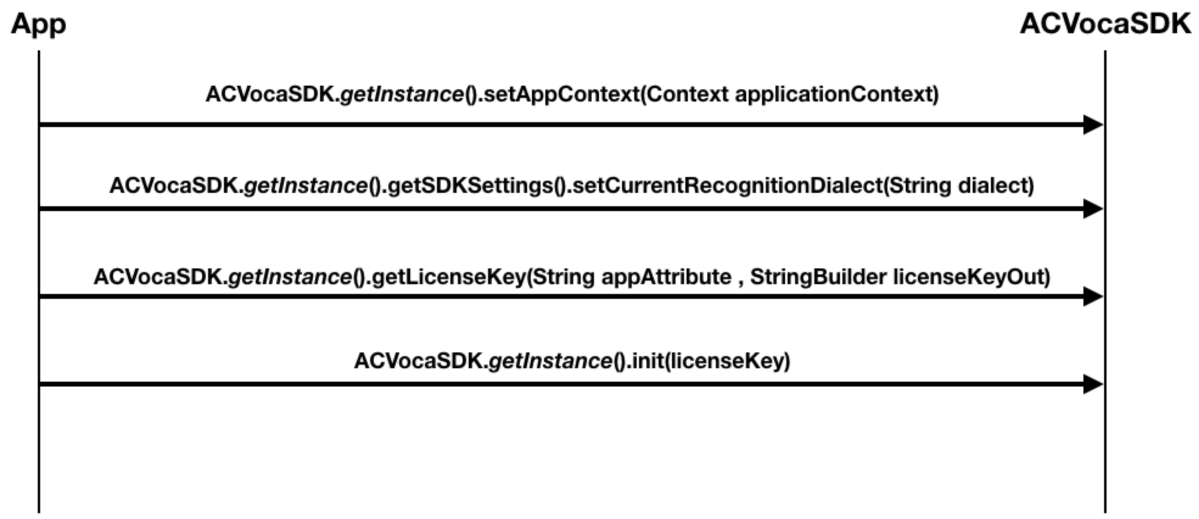
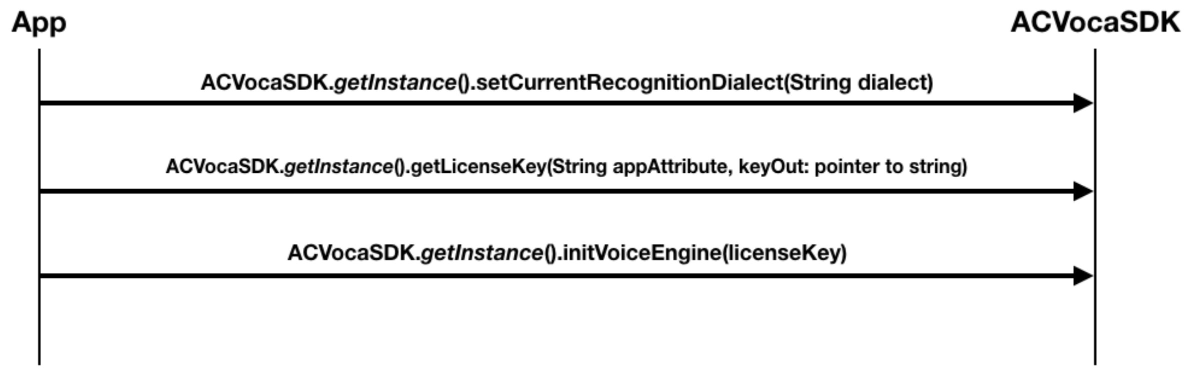

App
ACVocaSDK
ACVocaSDK.getInstance().setAppContext(Context applicationContext)
ACVocaSDK.getInstance().getSDKSettings().setCurrentRecognitionDialect(String dialect)
ACVocaSDK.getInstance().getLicenseKey(String appAttribute , StringBuilder licenseKeyOut)
ACVocaSDK.getInstance().init(licenseKey)

Figure 2-2: ACVocaSDK iOS Initialization


App
ACVocaSDK
ACVocaSDK.getInstance().setCurrentRecognitionDialect(String dialect)
ACVocaSDK.getInstance().getLicenseKey(String appAttribute, keyOut: pointer to string)
ACVocaSDK.getInstance().initVoiceEngine(licenseKey)

Please use the sample applications provided for code snippets.

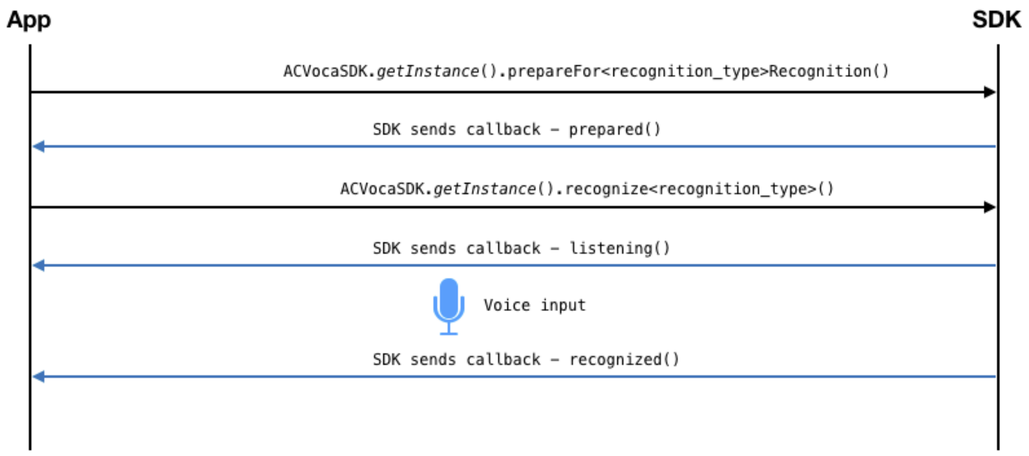
2.4.2 Voice Recognition Callbacks

The AC Voca SDK includes the following voice recognition callbacks:

- **Contacts recognitions:**
  - Contacts
  - Dynamic
  - Devices
- **Generic recognition**

Recognition is performed in several steps, as shown in the figure below:

Figure 2-3: Voice Recognition Callback Steps


App
SDK
ACVocaSDK.getInstance().prepareFor<recognition_type>Recognition()
SDK sends callback – prepared()
ACVocaSDK.getInstance().recognize<recognition_type>()
SDK sends callback – listening()
Voice input
SDK sends callback – recognized()

The voice recognition process is performed on a separate thread and will not block the User Interface (UI) thread. The application must be able to handle the recognition process callback methods.

In Android, voice recognition callbacks are returned on the voice engine thread. Any UI changes done using one of the call back methods must be done on the UI thread.

In iOS, returned callbacks are done on the mail thread.

It is recommended to handle callback methods from the SDK on each screen (activity/UIViewController). This can be achieved by either implementing the callback interface on the screen or to hold an instance member that implements that callback method. See Section 5.1 on page 455.1 for more details.

2.4.3 Uploading Recognition Data to the Server

The ACVocaSDK includes a built-in mechanism to upload recognition data to the AudioCodes backend.

Due to privacy concerns this feature is turned off by default.

To activate this feature, call ACVocaSDK.*getInstance*().getSDKSettings().setShouldUploadWavFilesToServer(boolean).

The application can also set the confidence threshold for upload (i.e., any recognition with a confidence level below that threshold, will be uploaded to the server).
To set the threshold, call
ACVocaSDK.*getInstance*().getSDKSettings().setUploadConfidenceThreshold(value).
The default upload threshold is 30%.

2.4.4 Recognition in PTT Mode

Working in Push-to-talk (PTT) mode varies from regular recognition in two ways:

- Handling the button press: Instead of just implementing button press, the application must also implement both press and release of the button.
- Calling a new API method which ends the recognition

Recognition callbacks flow has not changed. Therefore the application must still implement all relevant recognition callback methods.

Upon releasing the recognition button, the application must call the endPTTRecognition() API method .
Calling this method will end the current recognition session and will return the results, based on the input during the time of when the button was pressed and held.

This page is intentionally left blank.

3 API Methods

This section describes additional Classes and Methods for the AC Voca SDK.

3.1 General

The following are general methods.

3.1.1 getInstance

Description

Gets the shared instance of the ACVocaSDK.

Syntax

```
public static ACVocaSDK getInstance()
```

Parameters

This command has no parameters.

Return Values

Returns the shared instance of the ACVocaSDK.

3.1.2 setAppContext

Description

Sets the application context.

Syntax

```
public void setAppContext(Context appContext)
```

Parameters

`appContext`	Defines the application context.

Return Values

This command has no return values.

3.1.3 getAppContext

Description

Gets the application context.

Syntax

```
public Context getAppContext()
```

Parameters

This command has no parameters.

Return Values

Returns the application context .

3.1.4 setAcnlpExternalFolderPath

Description

Sets the external folder that is to be used by ACVocaSDK. This is used mainly for logs.

Syntax

```
public void setAcnlpExternalFolderPath(String
acnlpExternalFolderPath)
```

Parameters

`acnlpExternalFolderPath`	The path of the external directory that has write permissions.

3.1.5 getVoiceEngineFilesFolder

Description

Gets the path of the files folder of the voice recognition engine.

Syntax

```
public String getVoiceEngineFilesFolder()
```

Parameters

This command has no parameters.

Return Values

Returns the absolute path to the application files directory.

3.2 License

3.2.1 getLicenseKey

Description

Defines a helper method to receive a license key online at runtime. This method connects to an external server and requires Internet connection. This method is synchronous.

Syntax

```
public int getLicenseKey(String appAttribute,
StringBuilder licenseKeyOut)
```

Parameters

`appAtribute`	A string that represents an application specific license attribute.
`licenseKeyOut`	An output parameter that the SDK will write the license key to.

Return Values

Returns license key retrieval error. The return value is 0 if the function succeeds.

Possible values are:

0	Ok
100	No more licenses
110	App package not recognized
200	General error

3.2.2 getLicenseStatus

Description

Returns the current license status. A valid return code will only be available after calling *init* with the current license key.

Syntax

```
public ACVocaSDKLicenseStatus getLicenseStatus()
```

Return Values

Returns the current status of the license key (valid only after init)

Possible values are:

- ACVocaLicenseStatusOK
- ACVocaLicenseStatusInvalid
- ACVocaLicenseStatusInvalidGraceActive
- ACVocaLicenseStatusNoLicense
- ACVocaLicenseStatusNoLicenseGraceActive

3.3 Initialization and Settings

The following are Initialization and Settings methods.

3.3.1 init

Description

Initializes the AC Voca SDK and ACNLP engine.

The *init* methos also validates the license key provided by the app, in case of license key validation error. Possible error codes are:

1000	No license key
2000	Invalid license key
3000	License key expired
4000	Wrong package identifier
5000	Wrong device identifier

Syntax

```
public int init(String licenseKey)
```

Parameters

`licenseKey`	Defines the license access key.

Return Values

Returns the engine *init* error code. The return value is 0 if the function succeeds.

3.3.2 initACMVE

Description

Initializes the AudioCodes Mobile Voice Engine (ACMVE).
Calling this method is optional. It should be called only if ACMVE functionality is needed.

Syntax

```
public int initACMVE()
```

Parameters

This command has no parameters.

Return Values

Returns the engine initialization error code. The return value is '0' if the function succeeds.

3.3.3 restartEngine

Description

Restarts the voice recognition engine.

Syntax

```
public void restartEngine()
```

Parameters

This command has no parameters.

3.3.4 stopVoiceEngine

Description

Stops the AC Voca voice recognition engine.

Syntax

```
public void stopVoiceEngine()
```

Parameters

This command has no parameters.

Return Values

There are no return values.

3.3.5 setSDKSettings

Description

To maintain your own ACNLP settings, you will need to set it for the ACNLP SDK to use. Be sure to reset it after a change, or maintain changes on the object that was set and save the changes back to you application when needed.

Syntax

```
public void setSDKSettings(ACVocaSDKSettings
vocaSDKSettings)
```

Parameters

`acnlpSettings`	Defines the SDK settings to set.

Return Values

This command has no return values.

3.3.6 getSDKSettings

Description

Gets the current SDK settings.

Syntax

```
public ACVocaSDKSettings getSDKSettings()
```

Parameters

This command has no parameters.

Return Values

Returns the current SDK settings object.

3.4 Address Book

The following are Address Book methods.

3.4.1 initEnterpriseLists

Description

Initializes the enterprise lists (grammar and address book) based on the current recognition dialect. The Compiled Grammar (.CMG) file should be copied to the SDK engine files folder prior to calling this method.

Syntax

```
public boolean initEnterpriseLists()
```

Parameters

This command has no parameters.

Return Values

- 'true' - if the enterprise files were found and able to start the initialization process.
- 'false' - if any of the enterprise files were not found.

3.4.2 initPrivateAddressBookForRecognition

Description

Initializes the private address book for recognition.

Syntax

```
public int initPrivateAddressBookForRecognition
(List<? extends VREBaseContactObject> addressBook,
   boolean forceUpdate)
```

Parameters

`addressBook`	Defines the personal address book to initialize.
`forceUpdate`	If this parameter is 'true', the Private Address Book is updated even if no change has taken place.

Return Values

Returns the private address book initialization error code. The return value is '0' if the function succeeds.

3.4.3 clearPrivateAddressBookCompilation

Description

Clears the current compiled private address book.

Syntax

```
public int clearPrivateAddressBookCompilation()
throws android.os.RemoteException
```

Parameters

This command has no parameters.

Return Values

This method returns '0' if the private address book is successfully removed from the voice engine.

Throws

```
android.os.RemoteException
```

3.4.4 isPrivateAddressBookCompilationInProgress

Description

Checks whether there is an active Private Address Book compilation in progress.

Syntax

```
public boolean
isPrivateAddressBookCompilationInProgress()
```

Parameters

This command has no parameters.

Return Values

The return value is 'true' if the voice recognition engine is currently compiling the Private Address Book for recognition.

3.4.5 isPrivateAddressBookExists

Description

Verifies if it holds a valid Private Address Book for recognition.

Syntax

```
public boolean isPrivateAddressBookExists()
```

Parameters

This command has no parameters.

Return Values

The return value is 'True' if the engine already has a Private Address Book.

3.5 Engine Status

The following are Engine Status methods.

3.5.1 isEngineInitialized

Description

Checks whether the voice recognition engine is initialized.

Syntax

```
public boolean isEngineInitialized()
```

Parameters

This command has no parameters.

Return Values

The return value is 'true' if the voice engine is initialized.

3.5.2 isIdle

Description

Checks whether the voice recognition engine is in idle state (i.e., no current recognition is in process).

Syntax

```
public boolean isIdle()
```

Parameters

This command has no parameters.

Return Values

The return value is 'true' if there is no recognition currently active.

3.5.3 isInRecognizeState

Description

Checks whether there is an active recognition in process.

Syntax

```
public boolean isInRecognizeState())
```

Parameters

This command has no parameters.

Return Values

The return value is 'true' if there is an active recognition in process.

3.6 Recognition

The following are Recognition methods.

3.6.1 prepareForContactsRecognition

Description

Prepares for contacts recognition. This is an asynchronous method.

Syntax

```
public int prepareForContactsRecognition()
       throws android.os.RemoteException
```

Parameters

This command has no parameters.

Return Values

The method returns 0 if the *prepareForRecognition* method started successfully. See the *prepared* callback method in Section 5.1.2 on page 45 for the asynchronous result on page 45.

Throws

```
android.os.RemoteException
```

3.6.2 prepareForDevicesRecognition

Description

Prepares for generic recognition. This is an asynchronous method.

Syntax

```
public int prepareForDevicesRecognition()
  throws android.os.RemoteException
```

Parameters

This command has no parameters.

Return Values

Returns '0' if the prepare for recognition started successfully.
See the *prepared* callback method in Section 5.1.2 on page 45, for asynchronous result.

Throws

```
android.os.RemoteException
```

3.6.3 prepareForDynamicRecognition

Description

Prepares for dynamic recognition. This is an asynchronous method.

Syntax

```
public int prepareForDynamicRecognition()
                    throws android.os.RemoteException
```

Parameters

This command has no parameters.

Return Values

Returns '0' if the prepare for recognition started successfully.
See prepared callback method in Section 5.1.2 on page 45 for the asynchronous result.

Throws

```
android.os.RemoteException
```

3.6.4 prepareForGenericRecognition

Description

Prepares for generic recognition. This is an asynchronous method.

Syntax

```
public int prepareForGenericRecognition(String
                  compiledGrammerFileName,
                  AcNlpRecAndPrepParam
                  recognitionParams) throws
                  android.os.RemoteException
```

Parameters

`compiledGrammerFileName`	Defines the filename of the compiled grammar that this recogntion will be based on (including the file extension).
`recognitionParams`	Defines the set of recognition parameters (passes 'null' if you l use default values).

Return Values

The method returns '0' if the prepare for recognition started successfully.
See prepared callback method in Section 5.1.2 on page 45 for the asynchronous result.

Throws

```
android.os.RemoteException
```

3.6.5 recognizeContacts

Description

Starts a recognition contacts session. This is an asynchronous method.

Syntax

```
public int recognizeContacts(String cookie)
                      throws android.os.RemoteException
```

Parameters

`cookie`	Defines a cookie string to be returned with recognition results.

Return Values

Returns '0' if the recognition started successfully.
See listening and recognizedContacts callback methods for the asynchronous result.

Throws

```
android.os.RemoteException
```

Notes

See also:

prepareForContactsRecognition()

3.6.6 recognizeDevices

Description

Starts a recognition contacts session. This is an asynchronous method.

Syntax

```
public int recognizeDevices(String cookie)
                     throws android.os.RemoteException
```

Parameters

`cookie`	Defines a cookie string to be returned with recognition results.

Return Values

This method returns '0' if the recognition started successfully.
See listening and recognizedDevices callback methods for asynchronous results.

Throws

```
android.os.RemoteException
```

Notes

See also: `prepareForDevicesRecognition` ()

3.6.7 recognizeDynamic

Description

Starts a recognition dynamic session. This is an asynchronous method.

Syntax

```
public int recognizeDynamic(String cookie)
                     throws android.os.RemoteException
```

Parameters

`cookie`	Defines a cookie string to be returned with recognition results.

Return Values

Returns '0' if the recognition started successfully.
See listening and recognizedDynamic callback methods for asynchronous results.

Throws

```
android.os.RemoteException
```

Notes

See also:

`prepareForDevicesRecognition` ()

3.6.8 recognizeGeneric

Description

Starts a generic recognition session. This is an asynchronous method.

Syntax

```
public int recognizeGeneric(String cookie)
                               throws
android.os.RemoteException
```

Parameters

`cookie`	Defines a cookie string to be returned with recognition results.

Return Values

This method returns '0' if the recognition started successfully.
See listening and recognizedGeneric callback methods for asynchronous results.

Throws

android.os.RemoteException

Notes

See also:

`prepareForGenericRecognition`(`String`, `AcNlpRecAndPrepParam`)

3.6.9 cancelRecognition

Description

Cancels the current active recognition session.

Syntax

```
public int cancelRecognition()
```

Parameters

This command has no parameters.

Return Values

This command has no return values.

3.6.10 endPTTRecognition

Description

Ends a current recognition in PTT mode.

Syntax

```
public int endPTTRecognition()
```

Parameters

This command has no parameters.

Return Values

This command has no return values.

3.7 Delegates

The following are Delegates methods.

3.7.1 setRecognitionCallback

Description

Sets a callback listener object for recognition actions.

Syntax

```
public void
setRecognitionCallback(IAcNlpRecognitionCallback
        recognitionCallback)
```

Parameters

`recognitionCallback`	Defines the callback listener for recognition actions.

Return Values

This command has no return values.

3.7.2 removeRecognitionCallbackListener

Description

Remove the current recognition callback listener. Before removing the current listener, verify that it's the same listener as passed.

Syntax

```
public boolean removeRecognitionCallbackListener
  (IAcNlpRecognitionCallback recognitionCallback)
```

Parameters

`recognitionCallback`	Defines the callback listener for recognition actions.

Return Values

This command returns a 'true' value if the callback listener was successfully removed.

3.7.3 setGeneralCallbackListener

Description

Sets a new general callback listener.

Syntax

```
public void
setGeneralCallbackListener(IAcnlpGeneralCallback
generalCallbackListener)
```

Parameters

`generalCallbackListener`	Defines the callback listener for recognition actions.

Return Values

This command has no return values.

3.7.4 removeGeneralCallBackListener

Description

Removes the current general callback listener. Before removing the current listener, verify that it's the same listener as the one that has been passed.

Syntax

```
public boolean removeRecognitionCallbackListener
  (IAcNlpRecognitionCallback recognitionCallback)
```

Parameters

`recognitionCallback`	Defines the callback listener for recognition actions.

Return Values

This command returns a 'true' value if the callback listener was successfully removed.

Notes

See also: `IAcnlpGeneralCallback`

3.8 Play File

The following are Play File methods.

3.8.1 playFile

Description

Plays a sound file.

Syntax

```
Public int playFile(java.lang.String path,
                    com.audiocodes.android.media.auxiliar
                    y.EnumsAC.FileType fileType,
                    com.audiocodes.android.media.auxiliar
                    y.EnumsAC.MixOption mixOption, int
                    playOption)
```

Parameters

`path`	Defines the path of the sound file.
`fileType`	Defines the enumerator holding the type of file to send.
`mixOption`	Defines the enumerator describing how to handle mixed voices from the microphone and the audio being played. There are four options: ✓ [0] - Mute Voice ✓ [1] – Mix With Voice ✓ [2] – Whisper Voice ✓ [3] – Whisper File

`playOption`	Defines the play method: ✓ [-1] - The raw sound file plays in a loop ✓ [0] - The raw sound file is now played ✓ " > 0" - Defines number of playing cycles of the sound file

Return Values

The return value is 0 if the function succeeds.

3.8.2 stopPlayFile

Description

Stops the PlayFile process.

Syntax

```
public void stopPlayFile()
```

Parameters

This command has no parameters.

Return Values

This command has no return values.

4 Additional Classes and Methods

The following are additional classes and methods.

4.1 AcnlpGenericRecognitionResult

Description

This class represents a generic voice recognition result.

Syntax

```
public class AcnlpGenericRecognitionResult
```

4.1.1 getWarnings

Description

Returns recognition warnings from AC Voca SDK engine.
Possible values are:

1	Low Signal-to-noise ratio (SNR)
2	No speech
4	Too loud
8	Too soft
16	Too long
32	Too early
64	Too short
128	No match

Syntax

```
public int getWarnings()
```

4.1.2 getResults

Description

Returns a list of results for the Automatic Speech Recognition (ASR) generic task.

Syntax

```
public
java.util.List<AcnlpGenericRecognitionResult.GenericRes
ult> getResults()
```

Notes

See also: `AcnlpGenericRecognitionResult.GenericResult`

4.1.3 getWaveForm

Description

Returns an absolute path to the voice recording of the recognition.

Syntax

```
public int getWaveFormPath()
```

4.1.4 getOperation

Description

Returns a recognition operation.

Syntax

```
public int getOperation()
```

4.2 AcnlpGenericRecognitionResult.GenericResult

Description

This is an inner class that represents a generic voice recognition result.

Syntax

```
com.audiocodes.android.media.acnlp.data.AcnlpGenericRec
ognitionResult.GenericResult
```

4.2.1 getConfidence

Description

Returns the confidence of the recognized result.

Syntax

```
getConfidence()
```

4.2.2 getItems

Description

Gets the items recognized for this result.

Syntax

```
getItems()
```

4.2.3 getConcatenatedTranscript

Description

This method gets a concatenated transcript of all items in the result.

Syntax

```
getConcatenatedTranscript()
```

4.3 AcnlpGenericRecognitionResult.GenericResult.GenericItem

Description

This class describes a generic recognition item within a generic recognition result.

Syntax

```
public class
AcnlpGenericRecognitionResult.GenericResult.GenericItem
```

4.3.1 getConfidence

Description

This method returns the confidence of the recognized result.

Syntax

```
getConfidence()
```

4.3.2 getTranscript

Description

This method gets a concatenated transcript of all items in the result.

Syntax

```
getTranscript()
```

5 Callback Delegates

5.1 Interface IAcNlpRecognitionCallback

Description

This interface handles recognition callback events.

Syntax

```
public interface IAcNlpRecognitionCallback
```

5.1.1 listening

Description

This callback method is used when the voice recognition engine is listening for voice input. It is called as a result of calling the *recognize* method.

Syntax

```
public interface IAcNlpRecognitionCallback listening()
```

5.1.2 prepared

Description

This callback method is used when the voice recognition engine is ready to start a recognition session. It is called as a result of calling the *prepare* method.

Syntax

```
void prepared(boolean success,
              IAcNlpRecognitionCallback.UICallbackError
errorCode)
```

Parameters

`success`	Set to 'true' if the engine has been prepared successfully.
`errorCode`	Can be any of the IAcNlpRecognitionCallback.UICallbackError errors.

5.1.3 recognizedContacts

Description

This callback method is used when the voice recognition engine has recognized contacts.

It is called as a result of calling the *recognize* method.

Syntax

```
void
recognizedContacts(IAcNlpRecognitionCallback.UICallback
Error errorCode,AcNlpRecognitionResult
acNlpRecognitionResult,
        String cookie)
```

Parameters

`errorCode`	This can be any of the IAcNlpRecognitionCallback.UICallbackError errors. If the voice engine does not find any results, the error code value will be 'RECOGNIZE_NO_RESULTS'.
`acNlpRecognitionResult`	Defines the recognition result object.
`cookie`	Defines the cookie set at the start of the recognition. If none is set, the value is null.

Notes

See also:

- `IAcNlpRecognitionCallback.UICallbackError`
- `AcNlpRecognitionResult`

5.1.4 recognizedDevices

Description

This callback method is used when the voice recognition engine has recognized devices.

It is called as a result of calling the *recognize* method.

Syntax

```
void
recognizedDevices(IAcNlpRecognitionCallback.UICallbackE
rror errorCode, String device,
           String cookie)
```

Parameters

`errorCode`	This can be any of the IAcNlpRecognitionCallback.UICallbackError errors. If the voice engine does not find any results, the error code value will be 'RECOGNIZE_NO_RESULTS'.
`device`	Defines the string representing the device type.
`cookie`	Defines the cookie set at the start of recognition. If none is set, the value is null.

5.1.4.1 recognizedDynamic

Description

This callback method is used when the voice recognition engine has recognized dynamic results. It is called as a result of calling the *recognize* method.

Syntax

```
void
recognizedDynamic(IAcNlpRecognitionCallback.UICallbackE
rror errorCode,   AcNlpRecognitionResult
acNlpRecognitionResult,
             String cookie)
```

Parameters

`errorCode`	This can be any of the IAcNlpRecognitionCallback.UICallbackError errors. If the voice engine does not find any results, the error code value will be 'RECOGNIZE_NO_RESULTS'.
`acNlpRecognitionResult`	Defines the recognition results object.

`cookie`	Defines the cookie set at the start of recognition. If none is set, the value is null.

Notes

See also:

- `IAcNlpRecognitionCallback.UICallbackError`
- `AcNlpRecognitionResult`

5.1.4.2 recognizedGeneric

Description

This callback method is used when the voice recognition engine has recognized generic results. It is called as a result of calling the *recognize* method.

Syntax

```
void
recognizedGeneric(IAcNlpRecognitionCallback.UICallbackE
rror errorCode,    AcnlpGenericRecognitionResult
                   acnlpGenericRecognitionResult,
              String cookie)
```

Parameters

`errorCode`	This can be any of the IAcNlpRecognitionCallback.UICallback Error errors. If the voice engine does not find any results, the error code value will be 'RECOGNIZE_NO_RESULTS'.
`acNlpGenericRecognitionRe sult`	Defines the generic recognition results object.
`Cookie`	Defines the cookie set at the start of recognition. If none is set, the value is null.

Notes

See also:

- `IAcNlpRecognitionCallback.UICallbackError`
- `AcnlpGenericRecognitionResult`

A ACNLP Engine Error Codes

The table below lists the Error Codes from the ACNLP engine. See Section 3.4.2 on page 26.

Table A-1: ACNLP Engine Error Codes

Type	Code	Description
ACNLP_NO_ERR	0	No error
ACNLP_WRONG_USAGE	0x00010000	Method used not according to requirements
ACNLP_PARAM_INAVLID	0x00020000	Parameter is out of range
ACNLP_AUDIO_WRITE_OVERRUN	0x00030000	Audio internal overrun problem
ACNLP_FILE_ERR	0x00040000	POSIX read errors
ACNLP_STOPPED	0x00080000	Reserved
ACNLP_ERR_BUSY	0x00100000	API called within another API, where applicable
ACNLP_ERR_INIT_FAIL	0x00200000	The *init* failed.
ACNLP_ERR_ALREADY_EXISTS	0x00400000	The error already exists
ACNLP_ERR_MORE_DATA	0x00800000	String pointer to API call did not provide sufficient space for returned value
ACNLP_AD_ERR_FLAG	0x01000000	Bitwise high error indicating audio driver error

International Headquarters
1 Hayarden Street,
Airport City
Lod 7019900, Israel
Tel: +972-3-976-4000
Fax: +972-3-976-4040

AudioCodes Inc.
27 World's Fair Drive,
Somerset, NJ 08873
Tel: +1-732-469-0880
Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide
Website: https://www.audiocodes.com

Document #: LTRT-12984