Hound SDK Reference Guide

This document provides a reference for implementing the **Hound SDK** within an **iOS** application.

Version History

Version	Description	Date
1.1	• New audio architecture • Responses include parsed data models as well as raw dictionaries • Added Houndify class with drop-in Hound user interface • Updated end-of-speech algorithm • New error codes incorporated from lower level modules • Added helper method for dynamic responses • Renamed `resetConversationState` to `clearConversationState` • Renamed `enableSpeechActivationDetection` to `enableHotPhraseDetection`	16 November 2015
1.04	• Fix for iOS 9 speech speed • Fix for issues running on iPhone 6s	27 October 2015
1.03	• Added `enableSpeechActivationDetection` flag • Added `HoundVoiceSearchHotPhraseNotification` • Updated documentation with notifications	22 June 2015
1.02	• Removed `HoundCommon.h` • Added **raw** search mode • Added `enableEndOfSpeechDetection` flag • Updated documentation	02 June 2015
1.01	Initial Release	25 May 2015

What is Houndify?

Houndify is a **Developer Platform** to create smart, voice enabled, conversational interfaces to anything.

The **Houndify** API lets you **Houndify** your app, device, or anything else with an Internet connection. Once something has been **Houndified**, it becomes a **Houndified Client** and it can understand a wide variety of questions and commands in human language, either spoken or written.

System Requirements

The **Hound SDK** supports the following:

- iOS 8.0+
- XCode 7+
- iPhone 4s+, iPad 2+, iPod Touch (5 generation+) or Simulator

Package Contents

This section describes the contents of the **SDK** package.

- **Hound SDK Reference Guide**
  - `/Hound SDK Reference Guide.pdf`
- **Hound SDK Library**
  - `/HoundSDK/include/`
  - `/HoundSDK/libHoundSDK.a`
  - `/HoundSDK/HoundSDK.xcassets`
- **Hound SDK Test Application**
  - `/HoundSDK Test Application/HoundSDK Test Application.xcodeproj`
  - `/HoundifySDK Test Application/HoundifySDK Test Application.xcodeproj`

Note: Although the **SDK** is approximately 70 megabytes, it will not add this to the application file size. The **SDK** contains a number of different architectures and frameworks that get stripped out once the application is linked. The **SDK** also includes support for **bitcode**. Typically the **SDK** will only add a few megabytes to the size of an application.

Core Tasks

The **SDK** allows a developer to perform the following core tasks:

- **Voice Search** – Make voice queries against the Hound servers. Transcriptions and results will be returned in JSON format as well as a parsed object format. The SDK will control the audio session for the application.

    - **Related methods**:
        - `HoundVoiceSearch startListeningWithCompletionHandler`
        - `HoundVoiceSearch stopListeningWithCompletionHandler`
        - `HoundVoiceSearch startSearchWithRequestInfo`
        - `HoundVoiceSearch stopSearch`
        - `HoundVoiceSearch cancelSearch`
        - `HoundVoiceSearch stopSpeaking`

- **Voice Search with User Interface** – The **SDK** allows an application to perform voice searches against Hound servers. It comes with a built-in user interface that conforms to **Hound** user interface guidelines and simplifies development and integration.

    - **Related methods**:
        - `Houndify presentListeningViewControllerInViewController`
        - `Houndify dismissListeningViewControllerAnimated`

- **Raw Voice Search** – This is similar to voice search, however the application is responsible for managing the audio session and providing audio data to the SDK.

    - **Related methods**:
        - `HoundVoiceSearch setupRawModeWithInputSampleRate`
        - `HoundVoiceSearch startSearchWithRequestInfo`
        - `HoundVoiceSearch writeRawAudioData`
        - `HoundVoiceSearch stopSearch`
        - `HoundVoiceSearch cancelSearch`
        - `HoundVoiceSearch stopSpeaking`

- **Text Search** – Allows text-based queries against Hound servers. Results are delivered in the same format as a voice search.

    - **Related methods**:
        - `HoundTextSearch searchWithQuery`
        - `HoundTextSearch cancelSearch`

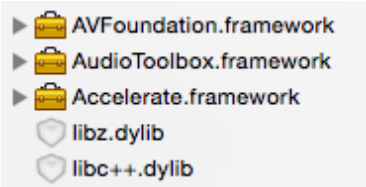
The **SDK** also includes speech support for voice search responses; this can be enabled or disabled by the developer through the SDK interface. See the **enableSpeech** flag of **HoundVoiceSearch**.

Please refer to https://houndify.com/ for documentation on the response formats.

Getting Started

Use to following steps to add the **Hound SDK** to your project.

1. Copy the **`HoundSDK`** directory into your project directory
2. Add the path to **`HoundSDK/include`** into the **Header Search Paths** in the project **Build Settings**
3. Add **`libHoundSDK.a`** to your project file
4. Add **`HoundSDK.xcassets`** to your project file
5. Add the following required system frameworks to your project:



6. Add the import statement where you intend to call the **Hound SDK**

```
#import <HoundSDK/HoundSDK.h>
```

7. Call **`setClientID`** and **`setClientKey`** on the `Hound` object with the provided authentication credentials in the **`didFinishLaunchingWithOptions`** method of the application delegate.

```
- (BOOL)application:(UIApplication*)application
      didFinishLaunchingWithOptions:(NSDictionary*)launchOptions
{
      [Hound setClientID:@"<INSERT YOUR CLIENT ID>"];
      [Hound setClientKey:@"<INSERT YOUR CLIENT KEY>"];

      ...
```

8. Start calling methods in the **Hound SDK**.

Class Reference

This section provides a description of the three (3) classes in the **Hound SDK**.

The classes are:

- **Hound Class** – High level common interfaces
- **HoundVoiceSearch Class** – Performs audio-based Hound queries
- **Houndify Class** – Simple support for voice searches with built-in user interface
- **HoundTextSearch Class** – Performs text-based Hound queries

These classes are described in further detail in the subsequent sections.

Hound Class

The **Hound** class allows high-level basic operations on the **Hound SDK**.

The **Hound** class has the following methods:

```
+ (NSString*)SDKVersion;

+ (void)setClientID:(NSString*)clientID;

+ (void)setClientKey:(NSString*)clientKey;
```

Definitions

```
+ (NSString*)SDKVersion;
```

This method returns the current version of the **SDK**. Used for debugging.

```
+ (void)setClientID:(NSString*)clientID;
+ (void)setClientKey:(NSString*)clientKey;
```

These methods allow the caller to set the authentication credentials used in **Hound** requests. These values will be used in both text and voice searches.

These values must be set prior to using the **SDK**.

Parameters	Description
clientID	The clientID provided by SoundHound
clientKey	The clientKey provided by SoundHound

```
+ (void)clearConversationState;
```

Clears the current conversation state. Subsequent queries will start anew and not following an existing conversation flow.

```
+ (void)handleDynamicResponse:(id)dynamicResponse
      andUpdateCommandResult:(HoundDataCommandResult*)commandResult;
```

This method updates the conversation state from a particular dynamic response. It also updates a command object with the properties of a dynamic response.

Parameters	Description
dynamicResponse	A dynamic response returned from a query. Dynamic responses are alternate responses the client may choose to show to the user. This object may be provided as **NSDictionary** or **HoundDataDynamicResponse**. Examples of dynamic responses include: • **CommandResult ClientActionSucceededResult** • **ComposeSMSCommand NoSMSAppResult** For a full description of commands, refer to: https://houndify.com/reference/CommandResult
commandResult	A command result object to be updated (optional).

HoundVoiceSearch Class

This class performs voice searches. It internally manages the audio session of the application, including recording audio from the microphone.

Voice searches support two (2) modes: **automatic** and **raw**.

In **automatic** mode, the **SDK** manages audio within the application. In **raw** mode, the caller is responsible for supplying audio data to the **SDK**.

The **HoundVoiceSearch** class has the following properties and methods:

```
@property(nonatomic, assign, readonly) HoundVoiceSearchState state;

@property(atomic, assign) BOOL enableHotPhraseDetection;
@property(atomic, assign) BOOL enableEndOfSpeechDetection;
@property(atomic, assign) BOOL enableSpeech;

+ (instancetype)instance;

// Setup raw mode

- (void)setupRawModeWithInputSampleRate:(double)inputSampleRate
    completionHandler:(HoundVoiceSearchErrorCallback)handler;

// Automatic search methods

- (void)startListeningWithCompletionHandler:(HoundVoiceSearchErrorCallback)handler;
- (void)stopListeningWithCompletionHandler:(HoundVoiceSearchErrorCallback)handler;

// Voice search

- (void)startSearchWithRequestInfo:(NSDictionary*)requestInfo
    endPointURL:(NSURL*)endPointURL
    responseHandler:(HoundVoiceSearchResponseCallback)responseHandler;

- (void)writeRawAudioData:(NSData*)data;

// General methods

- (void)stopSearch;
- (void)cancelSearch;

- (void)stopSpeaking;
```

Definitions

```
+ (instancetype)instance;
```

This method returns the singleton instance of the class. This is used for all voice searches.

```
@property(nonatomic, assign, readonly) HoundVoiceSearchState state;
```

The current state of voice search.

Possible values are:

- **HoundVoiceSearchStateNone** – Not listening for speech
- **HoundVoiceSearchStateReady** – Listening for speech
- **HoundVoiceSearchStateRecording** – Recording and transmitting audio to the server
- **HoundVoiceSearchStateSearching** – Waiting for a response from the server
- **HoundVoiceSearchStateSpeaking** – Speaking the response from the server

Note: The **SDK** must be in the **Ready** state to start an **automatic** search. **Raw** searches may be started in the **None** state.

```
@property(atomic, assign) BOOL enableHotPhraseDetection;
```

A flag indicating if the **SDK** should automatically detect the Hound hot phrase.

When this flag is enabled and the SDK is in listening mode, then when the user speaks “OK Hound”, the SDK will post the `HoundVoiceSearchHotPhraseNotification` to all listeners.

The application can intercept this notification and start a voice search.

The default is **NO**.

```
@property(atomic, assign) BOOL enableEndOfSpeechDetection;
```

A flag indicating if the **SDK** should automatically detect end of user speech.

If **NO**, then the search will stay active until terminated by the server (~10 seconds of silence). Otherwise, the search result is processed as soon as the user stops speaking.

The default is **YES**.

```
@property(atomic, assign) BOOL enableSpeech;
```

A flag indicating if the **SDK** should automatically speak the response from the server.

The default is **YES**.

```
- (void)setupRawModeWithInputSampleRate:(double)inputSampleRate
    completionHandler:(HoundVoiceSearchErrorCallback)handler;
```

This method places the **SDK** into raw search mode. This method is used when the application manages its own audio infrastructure.

If the application doesn't manage its own audio infrastructure, use `startListeningWithCompletionHandler` instead.

Parameters	Description
inputSampleRate	The sampling rate of the audio that will be passed to the SDK through **writeRawAudioData**
handler	This callback is invoked when the initialization is complete. The following values are returned: • **error** – An error object if the operation failed

---

```
- (void)startListeningWithCompletionHandler:
      (HoundVoiceSearchErrorCallback)handler;
```

This method places the **SDK** into listening mode. This must be successfully called before starting any **automatic** voice searches. This call is not used for **raw** searches.

The **SDK** will automatically prompt the user for microphone permissions if necessary. If the user declines microphone permissions then an error will be returned through the handler.

Note: The `AVAudioSession` for the application will be placed in the `AVAudioSessionCategoryPlayAndRecord` category and `AVAudioSessionModeDefault` mode.

Parameters	Description
handler	This callback is invoked when the initialization is complete. The following values are returned: • **error** – An error object if the operation failed

```
- (void)stopListeningWithCompletionHandler:
      (HoundVoiceSearchErrorCallback)handler;
```

This method stops the **SDK** from processing microphone input. The state transitions to **HoundVoiceSearchStateNone**.

Searches cannot be started when the **SDK** is not listening.

Parameters	Description
handler	This callback is invoked when the listening is stopped. The following values are returned: • **error** – An error object if the operation failed

```
- (void)startSearchWithRequestInfo:(NSDictionary*)requestInfo
    endPointURL:(NSURL*)endPointURL
    responseHandler:(HoundVoiceSearchResponseCallback)responseHandler;
```

This method initiates a voice search. Audio is automatically recorded from the user and transmitted to the server.

Parameters	Description
requestInfo	A dictionary containing extra parameters for the search. The following keys are set by default if not supplied by the caller: • UserID, RequestID, TimeStamp, TimeZone, ClientID, ClientVersion, DeviceID, ConversationState, UnitPreference, PartialTranscriptsDesired, ObjectByteCountPrefix, SDK, SDKVersion **Note (1)**: The caller should populate the location keys in this dictionary. The **SDK** does not manage the user location. For a full description of parameters, refer to: https://houndify.com/reference/RequestInfo
endPointURL	The **URL** endpoint for voice search. For production, use the value: https://api.houndify.com/v1/audio
responseHandler	This callback is invoked during the search and may be called multiple times with different values. The following values are returned: • **error** – An error object if the operation failed • **responseType** – an enumeration indicating if it is a partial transcription or a full response • **response** – A parsed version of the response • **dictionary** – A dictionary containing the response The response object is one of two types: • **HoundDataPartialTranscript** – a partial text transcription • **HoundDataHoundServer** – a full response The callbacks stop when an error or full response is returned. For a complete reference of the format of the full response dictionary, refer to: https://houndify.com/reference/HoundServer

```
- (void)writeRawAudioData:(NSData*)data
```

This method allows the caller to supply raw audio data. This is used in conjunction with the **HoundVoiceSearchModeRaw** flag on `startSearchWithRequestInfo`.

The data must be 16 bit, Linear PCM audio data. 16 Khz is ideal for optimal performance.

This data is the same as returned by the **AudioUnitRender API** function in the **iOS AudioToolbox** framework.

```
- (void)stopSearch;
```

This stops the **SDK** from listening to the user's request, and transitions into the searching state.

The search may also be stopped internally when the **SDK** detects end of user speech if the **enableEndOfSpeechDetection** flag is **YES**.

```
- (void)cancelSearch;
```

The method cancels a search in progress. A cancel error is returned through the response handler.

```
- (void)stopSpeaking;
```

If the response is currently being spoken, this stops speech in progress.

Houndify Class

The **Houndify** class allows high-level basic operations on the **Houndify SDK**.

The **Houndify** class has the following methods:

```
+ (NSString*)SDKVersion;

+ (void)setClientID:(NSString*)clientID;

+ (void)setClientKey:(NSString*)clientKey;

+ (instancetype)instance;

- (void)presentListeningViewControllerInViewController:(UIViewController*)
  presentingViewController
  fromView:(UIView*)presentingView
  requestInfo:(NSDictionary*)requestInfo
  endPointURL:(NSURL*)endPointURL
  responseHandler:(HoundifyResponseCallback)responseHandler;

- (void)dismissListeningViewControllerAnimated:(BOOL)animated
  completionHandler:(HoundifyCompletionHandler)completionHandler;
```

Definitions

```
+ (instancetype)instance;
```

This method returns the singleton instance of the class. This is used for all voice searches.

```
- (void)presentListeningViewControllerInViewController:(UIViewController*)
    presentingViewController
    fromView:(UIView*)presentingView
    requestInfo:(NSDictionary*)requestInfo
    endPointURL:(NSURL*)endPointURL
    responseHandler:(HoundifyResponseCallback)responseHandler;
```

This method initiates a voice search and starts a full screen takeover with the Hound user interface.

The screen dims content behind it and shows a blue listening overlay at the bottom of the screen.

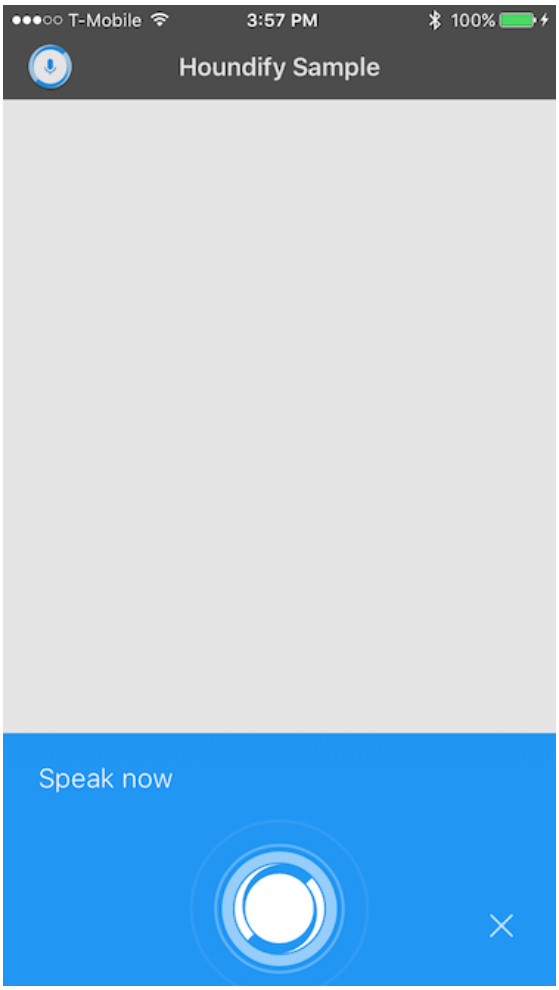


Figure: Houndify Full Screen Takeover from HoundifySDK Test Application

Parameters	Description
presentingViewController	The view controller that will present the **Hound** user interface. Typically this is the root view controller for the main window, such as a **UINavigationController** or **UITabBarController**.
presentingView	The view that launched the **Hound** search. This is typically a button. The **Houndify SDK** will center the launch animation on this view.
requestInfo	A dictionary containing extra parameters for the search. The following keys are set by default if not supplied by the caller: • UserID, RequestID, TimeStamp, TimeZone, ClientID, ClientVersion, DeviceID, ConversationState, UnitPreference, PartialTranscriptsDesired, ObjectByteCountPrefix, SDK, SDKVersion **Note (1)**: The caller should populate the location keys in this dictionary. The **SDK** does not manage the user location. For a full description of parameters, refer to: https://houndify.com/reference/RequestInfo
endPointURL	The **URL** endpoint for voice search. For production, use the value: https://api.houndify.com/v1/audio
responseHandler	This callback is invoked during the search and may be called multiple times with different values. The following values are returned: • **error** – An error object if the operation failed • **response** – A parsed version of the response • **dictionary** – A dictionary containing the response The response object is of type **HoundDataHoundServer**. For a complete reference of the format of the full response dictionary, refer to: https://houndify.com/reference/HoundServer

```
- (void)dismissListeningViewControllerAnimated:(BOOL)animated
    completionHandler:(HoundifyCompletionHandler)completionHandler;
```

This method dismisses the listening view controller and cancels a search if it is in progress.

This method must be called once a response is received to remove the **Houndify** user interface from the screen.

Parameters	Description
animated	A flag indicating if the dismiss should be animated or not.
completionHandler	This callback is invoked once the Hound user interface is dismissed. The callback has no parameters.

HoundTextSearch Class

This class performs text searches based on a query string.

The **HoundTextSearch** class has the following methods:

```
+ (instancetype)instance;

- (void)searchWithQuery:(NSString*)query
    requestInfo:(NSDictionary*)requestInfo
    endPointURL:(NSURL*)endPointURL
    completionHandler:(HoundTextSearchCallback)handler;

- (void)cancelSearch;
```

Definitions

```
+ (instancetype)instance;
```

This method returns the singleton instance of the class. This is used for all text searches.

```
- (void)searchWithQuery:(NSString*)query
    requestInfo:(NSDictionary*)requestInfo
    userID:(NSString*)userID
    endPointURL:(NSURL*)endPointURL
    completionHandler:(HoundTextSearchCallback)handler;
```

This is used to perform a text-based query.

Parameters	Description
query	The query string for the search Example: “what time is it”
requestInfo	A dictionary containing extra parameters for the search. The following keys are set by default if not supplied by the caller: • UserID, RequestID, TimeStamp, TimeZone, ClientID, ClientVersion, DeviceID, ConversationState, UnitPreference, SDK, SDKVersion **Note (1)**: The caller should populate the location keys in this dictionary. The SDK does not manage the user location. For a full description of parameters, refer to: https://houndify.com/reference/RequestInfo
endPointURL	The **URL** endpoint for text search. For production, use the value: https://api.houndify.com/v1/text
completionHandler	This callback is invoked when the search is complete. The following values are returned: • **error** – An error object if the operation failed • **query** – The original query text passed in • **houndServer** – A parsed search result • **dictionary** – A dictionary containing the search result The **houndServer** object is of type **HoundDataHoundServer** For a complete reference of the format of the **houndServer** dictionary, refer to: https://houndify.com/reference/HoundServer

---

```
- (void)cancelSearch;
```

Cancels any text search in progress. A cancel error is returned in the response handler.

Notifications

This section lists all notifications sent by the SDK

Name	Description
HoundVoiceSearchStateChangeNotification	This notification is posted when the `HoundVoiceSearch` class changes state. The new state can be read from the `state` property.
HoundVoiceSearchAudioLevelNotification	This notification is posted in listening mode with the current audio level. The audio level is a number between 0 and 1 containing the current audio level from the microphone. This can be used for visualization purposes. The level value is stored as an `NSNumber` object in the `object` property of the notification. It can be read using: `[notification.object floatValue]`
HoundVoiceSearchHotPhraseNotification	This notification is posted by the **SDK** in listening mode, and the user speaks the activation phrase, "**OK Hound**". This can be used to trigger the start of a search. **Note**: The `enableHotPhraseDetection` property must be **YES** for detection occur.
HoundVoiceSearchFinalTranscriptionNotification	This notification broadcasts the final partial transcription. The string value is in: `notification.userInfo[@"finalTranscription"]`

Error Codes

This section lists the error codes returned by the **Hound SDK** for voice searches in the **HoundVoiceSearchErrorDomain**. These are provided to assist in debugging.

HTTP-level errors are returned in the **SHHTTPErrorDomain** where the error code is a non-**200 HTTP** status code.

These are in addition to any system errors **iOS** may return in domains such as:

- NSPOSIXErrorDomain
- kCFErrorDomainCFNetwork
- NSURLErrorDomain

Code	Description
`HoundVoiceSearchErrorCodeNone`	No error
`HoundVoiceSearchErrorCodeCancelled`	The user cancelled the search
`HoundVoiceSearchErrorCodeNotReady`	The audio system is not ready, ensure **startListeningWithCompletionHandler** has been called
`HoundVoiceSearchErrorCodeServerStatusError`	The server returned a high level protocol error
`HoundVoiceSearchErrorCodeNoResponseReceived`	No response was received from the server
`HoundVoiceSearchErrorCodeInvalidResponse`	An invalid response was received from the server
`HoundVoiceSearchErrorCodeAudioInterrupted`	The audio system was interrupted in the middle of a search
`HoundVoiceSearchErrorCodeParseFailed`	An error occurred parsing the response JSON
`HoundVoiceSearchErrorCodeAuthenticationFailed`	Failed to authenticate with Hound Ensure that the clientID and clientKey have been set properly with the credentials received from SoundHound.
`HoundVoiceSearchErrorCodeInternalError`	An generic internal error occurred
`HoundVoiceSearchErrorCodePermissionDenied`	The user has denied microphone permissions
`HoundVoiceSearchErrorCodeConnectionFailure`	Failed to connect to Hound servers
`HoundVoiceSearchErrorCodeConnectionTimeout`	Timed out waiting for a response from Hound servers